Phantom SDK Reference Manual

Version 13.8.804.6 (March 2023)

100 Dey Rd Wayne, NJ 07470, USA | Tel +1 973 696 4500
Revision 13.8.804.6 © 2023 Vision Research - AMETEK Materials Analysis Division

Copyright © 1992-2023 Vision Research Inc. All Rights Reserved.

Notice

The information contained in this document file includes data that is proprietary of Vision Research Inc. and shall not be duplicated, used, or disclosed – in whole or in part – without the prior written permission of Vision Research Inc. The data subject to this restriction is contained in all pages of this file.

THIS PUBLICATION AND THE INFORMATION HEREIN ARE FURNISHED AS IS, ARE FURNISHED FOR INFORMATIONAL USE ONLY, ARE SUBJECT TO CHANGE WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY VISION RESEARCH INC. VISION RESEARCH INC ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR INACCURACIES THAT MAY APPEAR IN THE INFORMATIONAL CONTENT CONTAINED IN THIS GUIDE, MAKES NO WARRANTY OF ANY KIND (EXPRESS, IMPLIED, OR STATUTORY) WITH RESPECT TO THIS PUBLICATION, AND EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSES, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

Licenses and Trademarks

Windows is a trademark of Microsoft Corporation.

Software release

Some new fields of the SETUP structure may be added in new software releases.

This document is based on software release 804.6

(PhCon.dll, PhFile.dll, PhInt.dll version 13.8.804.6, PCC version 3.8.804.6).

Table of Contents

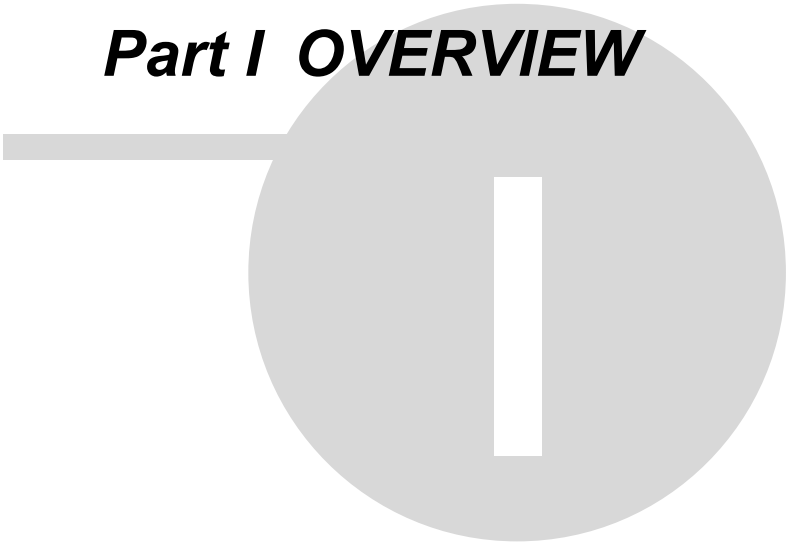
© 2023 Vision Research - AMETEK Materials Analysis Division

© 2023 Vision Research - AMETEK Materials Analysis Division

© 2023 Vision Research - AMETEK Materials Analysis Division

© 2023 Vision Research - AMETEK Materials Analysis Division

Part I OVERVIEW



© 2023 Vision Research - AMETEK Materials Analysis Division

1. Overview

1.1. About This Document

This document provides information for software developers who are writing applications using Phantom SDK for controlling Vision Research high-speed cameras and who need to:

- configure cameras and set acquisition parameters
- send record/trigger commands to a camera
- obtain live or recorded images from a camera
- save images in different video file formats

Chapter 2 introduces the Phantom camera general model thoroughly describing each component element, interactions between these elements and possible control operations. It is not intended to be a presentation of Vision Research cameras and their features, but a conceptual description of the software perspective over these cameras' universe. You can skip this chapter if you already are familiar with notions such as *camera, cine, cinehandle* or *Frame Buffer Memory.*

Chapter 3 describes Phantom SDK data structures, chapter 4 lists user defined function types, while chapter 5 details Phantom SDK functions. Please note that some of the data structures and functions available in Phantom SDK headers may not be documented here. They are deprecated and while best effort is put to maintain correct functionality, their use is not recommended.

1.2. Phantom SDK Structure

The Phantom SDK includes:

- this reference manual
- 64-bit binary .dll files
- public header .h files for C/C++
- public header .cs files for C#
- 64-bit static library .lib files
- Demo C/C++
    - HelloPhantomC - a C/C++ hello world demo application
    - PhSimpleDemoCPP - a C/C++ simple GUI demo application
    - PhDemoCPP - a C/C++ GUI demo application
- Demo C#
    - PhSimpleDemoCS - a C# simple GUI demo application
    - PhDemoCS - a C# GUI demo application
- Demo Matlab
    - HelloPhantom - a Matlab hello world demo application
    - SimplePhMatlabDemo - a Matlab simple GUI demo application
    - PhDemoMathlab - a Matlab GUI demo application
    - SimpleCineFileReader - a Matlab read cine file demo application
- Demo LabView
    - SimpleVideoAnalyzer - a LabView demo application
    - Automated Save - a LabView demo to save Cine files

© 2023 Vision Research - AMETEK Materials Analysis Division

Current SDK version is available for Microsoft 64 bit operating systems versions. All code has been written and tested under Microsoft Visual Studio 2012. Phantom SDK functions can also be used from Matlab or LabView applications as depicted by the two included demos.

There are six main dynamic link libraries: *PhCon*, *PhFile*, *PhInt*, *PhSig*, *PhSigV and PhRange*. Each of them deals with different camera control components. However, they are interdependent and must therefore be used together.

PhCon is the control library for Phantom high-speed cameras. It allows setting camera parameters, recording a movie in the camera *Frame Buffer Memory* (FBM) or transferring images to the client's computer through camera's Ethernet connection or high speed 1394 bus.

PhFile is the library responsible for files and movies management. It offers support for reading, saving, converting recordings no matter if stored on camera or inside a file.

PhInt is the color interpolation and image processing library for Phantom cameras. It implements various algorithms for demosaicing raw sensor information in order to output RGB color images.

PhSig and *PhSigV* deal with signal acquisition using a separate and optional device: The Signal Acquisition Module (SAM) produced by Vision Research. This module allows acquiring analog and binary signals, in parallel and synchronized with the image acquisition. It is controlled from Vision Research PCC application therefore these dlls are not described in this document.

PhRange deal with signal acquisition on the range data input. This module allows acquiring the range data signals, in parallel and synchronized with the image acquisition.

1.3. Backward Compatibility

The current version of the libraries is a superset of previous versions, made public starting in May 2001. It maintains compatibility with old third party applications written for a previous SDK version. Such an application can be updated simply by replacing old binary Phantom files with the latest version. The update enables the application to get access to new cameras previous unavailable.
However, in order to fully take advantage of new cameras and new features, a more complex update is required involving, among other steps, a full application recompiling.

1.4. System Requirements

It is recommended that the computer you use have a Pentium class microprocessor running 1.7 GHz or higher. The Phantom SDK can be used on Microsoft operating systems only:

- Windows 10 64 bit
- Windows 8/8.1 64 bit
- Windows 7 64 bit

A more powerful computer and fast disk drive will deliver faster display and playback speeds as well as shorter save and download times for your application. Also, image processing functions are configured to take advantage of multicore processors.

All recent Vision Research cameras offer a 1Gbit/s Ethernet connection for both camera control and image data transfer; therefore, a corresponding Ethernet card would be required for your computer. For Cine Stations (readers for our CineMags) and some cameras an optional 10Gbit/s Ethernet connection is available enabling even faster transfer rates.

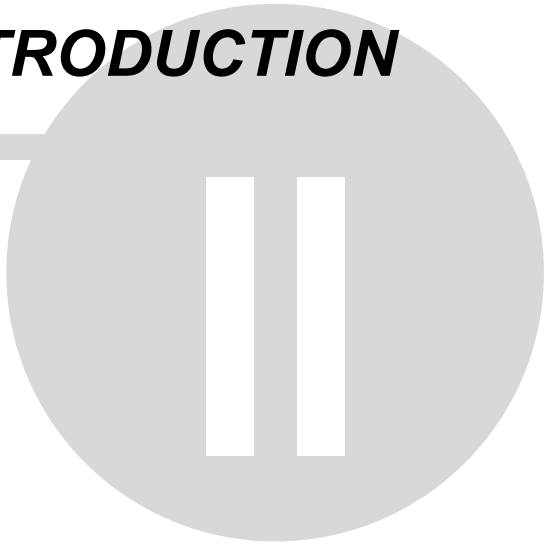
© 2023 Vision Research - AMETEK Materials Analysis Division

1.5. Supported Phantom Cameras

All Vision Research high speed video cameras produced until March, 2023 except multi-head cameras and I3 external memories are supported by this SDK version. However, please note that the software has not been tested on older cameras, such as firewire cameras.

© 2023 Vision Research - AMETEK Materials Analysis Division

Part II INTRODUCTION



© 2023 Vision Research - AMETEK Materials Analysis Division

2. Introduction

The Phantom SDK provides a way to programmatically access and control Phantom cameras and recorded video movies. This programming model consists of *entities*, *relations* between entities and *actions*.

2.1. Entities

There are two main types of entities the Phantom SDK is working with: *camera* entities and *cine* entities.

2.1.1. The Camera

A *camera* is a generalization of an actual Phantom camera. A *camera* allows a unitary, simplified approach no matter if working with the latest generation Vision Research camera or with other older models.

A camera is responsible for recording and storing images. It performs these actions following explicit requests and conforming to pre-configured parameters. There are four main types of parameters:

- camera options – used for general camera configuration (for example: what video output standard to use, should the camera store acquired images on the CineMag if one is present, what should the camera do after a recording finishes)
- acquisition parameters – controlling a movie acquisition (for example: frame rate, exposure time, image resolution)
- image processing parameters – various image adjustment parameters (for example: brightness, saturation, hue)
- other parameters – parameters not necessarily in one of the above categories (for example: lens information)

The first two types of parameters are represented by the `CAMERAOPTIONS` and `ACQUIPARAMS` structures described in following chapters.

A high speed video camera temporarily stores recorded movies in the internal video memory, called *Frame Buffer Memory* (*FBM*). The frame buffer memory can be split into multiple partitions to quickly record successive events in different movies. The memory of each partition is managed to work like a circular buffer: during a long recording, even if memory gets filled, the acquisition will continue by overwriting the oldest images from that partition.

Each partition has three main buffers allocated for acquisition parameters, image processing parameters and other parameters. The first one allows configuring recording parameters. As soon as the acquisition ends, these parameters become read-only.

Image processing parameters are not affecting recorded image data. They are stored as meta-data and used when images are displayed on video output or on PC applications. Unlike the acquisition parameters, they cannot be preconfigured. They are inherited from one special camera structure when the recording of the cine ends. This camera structure is holding a set of global image processing parameters used for processing the images of the movie being currently recorded. As soon as the acquisition ends, values from the camera data structure are copied into the image processing buffer of the partition the movie was recorded on. From this point forward, the image processing parameters of the recorded movie can be changed as needed by directly accessing its own image processing buffer.

© 2023 Vision Research - AMETEK Materials Analysis Division

Besides the main partitions where recorded movies are stored, there is a hidden video memory partition used for preview purposes. If all partitions contain finished recordings, this is where the camera will store one live image. In this case, the camera uses a smaller acquisition frame rate thus reducing power consumption. This working mode is called *pretrigger mode*. A camera automatically switches to *pretrigger mode* when all main partitions become full. A camera can also be set in *pretrigger mode* even if the FBM is empty; in this mode one can preview images for a certain set of acquisition parameters and, in the same time, reduce power consumption.

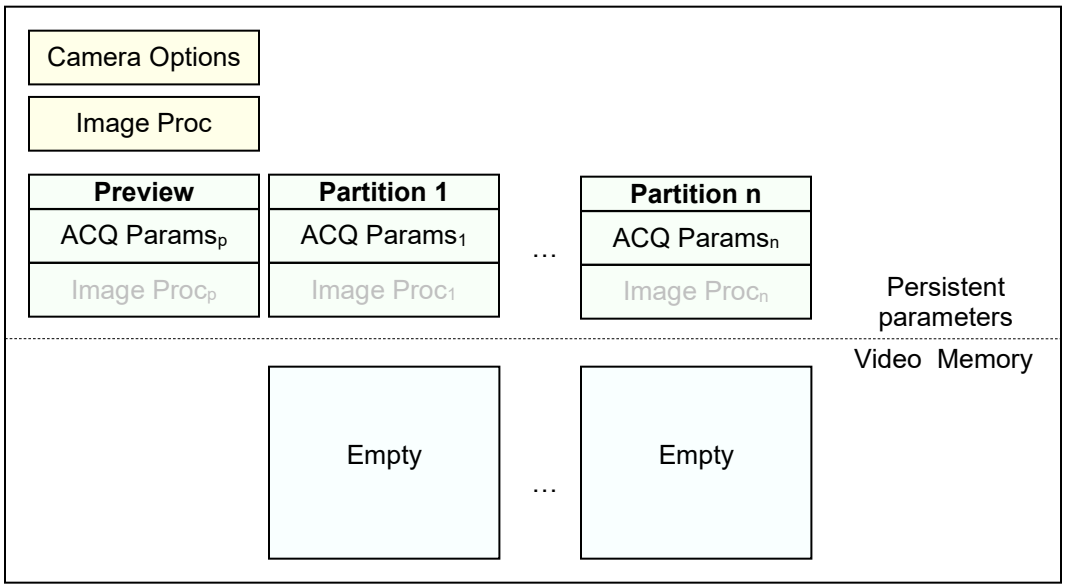


Figure 2-1. Camera data organization before starting a new acquisition

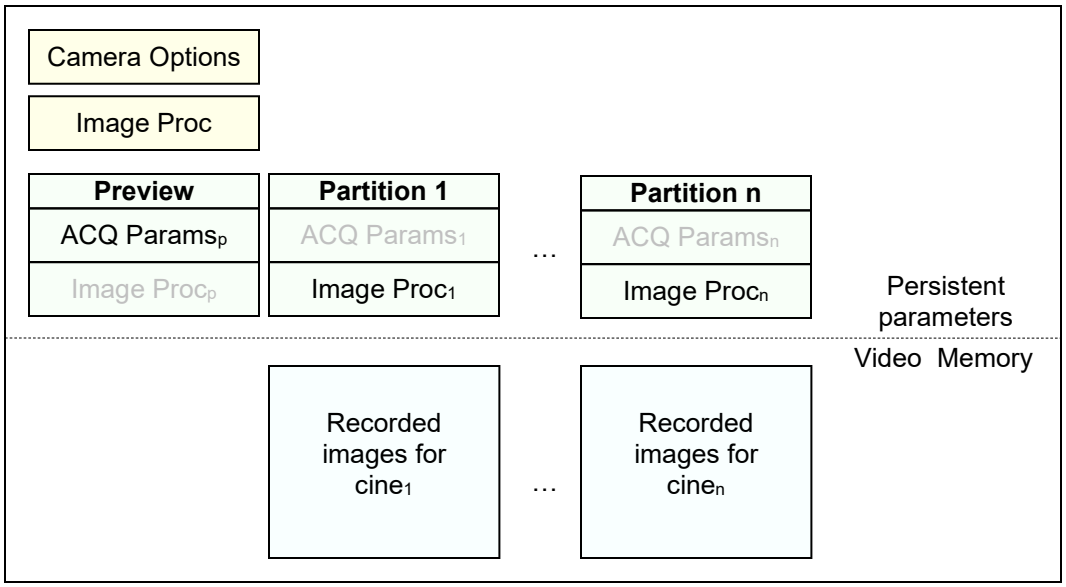


Figure 2-2. Camera data organization after finishing the acquisition

2.1.2. The Cine

A *cine* is a generalization of a Phantom camera movie. It is a collection of one or more stored images accompanied by a header describing the cine. The header contains information such as image resolution, acquisition frame rate or image processing parameters. A *cine* allows a unitary, simplified approach no

© 2023 Vision Research - AMETEK Materials Analysis Division

matter if working with a movie from the latest generation Vision Research camera or from older camera models. It also enables similar handling over different types of movie storage (the movie can reside on the camera, on special storage devices, like the CineMag, or it can be stored in a file).

While most of the operations are performed in the same manner for all types of cines, a cine storage type imposes some distinctions over what actions are allowed or not. For example, only a recorded cine stored inside the RAM of the camera can be saved to a CineMag; a file stored cine movie cannot be saved to a CineMag. Thus, the two main types of cines are: camera *cines* and file *cines*.

A camera *cine* refers to a cine for which images and most of the header data are stored on camera. Such a cine can be a recorded movie stored either in camera's RAM or in special storage devices like the CineMag.

A cine stored in the RAM of the camera is volatile; it will be lost if the camera is turned off or unpowered. A cine stored in the camera flash (like the CineMag) is nonvolatile; it will not be lost if the camera is turned off.

A file *cine* generally refers to a cine for which images and header data are stored in file. It can be one of Vision Research proprietary formats (*.cine*, *.cci*) or any other video formats the SDK offers reading support for. Check Supported File Formats for more details.

Phantom SDK offers a way to perform basic cine operations in the same way no matter the cine type, by introducing another special element: the *CINEHANDLE*. A *CINEHANDLE* is a handle to a cine. It facilitates access to cine header or image data, and it enables simple control for actions such as saving or converting. Most of the cine operations allowed by the Phantom SDK require a *CINEHANDLE* object to identify the cine.

Remember to always use a *CINEHANDLE* object for reading images from a cine!

Please note that the two notions, the cine – as an abstract generalization of a Vision Research high speed camera movie – and the *CINEHANDLE* – as a programming element used for referring to a cine – may overlap in this document. For example, a *CINEHANDLE* input parameter of a function may be named the input cine.

2.1.3. Image Numbering

The images are numbered in a growing order using 32 bit signed values. The images before trigger have negative numbers. The images after the trigger have zero or positive numbers.

2.2. Relations

2.2.1. The Camera Pool

All Phantom cameras the Phantom SDK established connections with at a given time are referred to as the *camera pool*. If a camera is not in the *camera pool* than the SDK is not aware of its presence, cannot communicate with it, cannot read images or set any of its acquisition parameters.

The *camera pool* is a buffer to keep information about connected cameras. Inside the camera pool each camera is identified based on a numerical index, called the *camera number*. The first index is `0`. The *camera number* is used a lot in Phantom SDK when controlling a camera or reading/writing its parameters.

Building a camera pool can be done by setting the software to discover all cameras that are available on the network or by entering a list of the IP addresses of the cameras we want to connect to.

In the first case a "discovery protocol" is used. The computer sends a broadcast message on the network and the cameras answer to it providing their IP address.

© 2023 Vision Research - AMETEK Materials Analysis Division

An important feature for a camera control system is to be able to discover new added cameras and to support the disconnecting and reconnecting of the cameras.

Before version 770, applications based on Phantom SDK did that by discovering changes in camera pool (calling `PhCheckCameraPool`) and updating camera pool (calling `PhNotifyDeviceChangeCB`). Last call can be done only when there is no pending operation on cameras.

The enumeration is now fully controlled inside the SDK. Application call `PhConfigPoolUpdate` to enable the automated refresh of the camera pool.

New cameras that appear during the application run could be added at the end of the camera pool, so the existing cameras will not change number and the communication to them will not be affected at all.

The application will notice a new camera is available just by the increase of the number returned by `PhGetCameraCount`.

If the communication to an existing camera is broken (a timeout occurred) the software will see camera in a special “Offline” status. If the camera gets back online later, the connection will be restored, and camera will be “revived” in the same position in the pool.

Existing cameras that go Offline will be found by the Timeout result at the commands sent to the camera. Both new cameras on network and the Offline revived cameras will be found by the network discovery protocol based on broadcast and UDP responses.

2.2.2. Cameras and Cines

As their name implies, camera cines belong to a camera. If a camera is identified based on a camera number, a camera cine is identified based on:

- the number of the camera it belongs to
- a *cine number*

The cine number represents the partition number the cine is stored on or recorded on. The first partition number is `1`. For example, if a camera has three partitions, corresponding cines numbers will be: `1`, `2`, and `3`.

Special values are used for cines stored in cameras NVMEM (flash or CineMag). They are indexed beginning with the value returned by the function `PhFirstFlashCine`.

While a camera cine as previously defined is strictly bound to one of the partitions of a camera, there is a special type of camera cine that does not seem to follow the rules. However, it is extremely useful and necessary. This camera cine is called *live cine*. The term *current cine* may also be used.

The *live cine* is the cine the camera is currently recording images into; thus, it is not necessarily associated with the same partition number at different moments in time. Each camera has exactly one *live cine*.

An application will often need to display images as currently recorded by the camera. When requesting such images, one cannot precisely tell from which partition number or at which resolution these will be taken. This happens because from the moment the image request is sent and the moment the camera processes it, the current cine number and acquisition parameters may change. Here is where the *live cine* becomes most useful: an application can use a *live cine* entity for reading images like any other cine letting the SDK and the camera to solve any details about the actual cine the camera is recording images into.

As previously mentioned, for basic cine access and control one should use a *CINEHANDLE* object. The same stands true for camera cines no matter if they are stored cines or live cines. While the camera and cine number physically identify a camera cine and also allow low level control, a CINEHANDLE object will greatly simplify operations such as reading images, reading time information, setting image processing

© 2023 Vision Research - AMETEK Materials Analysis Division

parameters or saving to file. Except for the creation step of a CINEHANDLE object, all of the above actions are performed in the same way for either a live cine, a camera recorded cine or a file stored cine.

Please note that while a cine refers to a set of recorded images plus a header, the term *cine number* may sometimes be used in this documentation to identify the partition where a new cine will be recorded.

2.3. Actions

2.3.1. Camera and Cine Control

Phantom SDK functions also allow controlling camera and cine objects. Most camera operations directly refer to camera capabilities and features and are not discussed in detail in this document. Typical camera actions include but are not limited to:

- operations for preparing a new recording session
- recording control operations
- playing back recorded movies
- saving recorded movies to the computer hard disk
- saving recorded movies to camera flash memory (like the CineMag)
- configuring video output parameters
- playing back cine images on video output or recording this output to tape devices

Main cine control operations are related to cine image access, image processing and cine saving operations. A camera cine stored either in camera RAM or on special storage devices (like the CineMag) can be saved to a file using our proprietary cine file format or any other supported file types (see Supported File Formats chapter). Phantom SDK allows image related operations for file cines as well. Similar to a camera cine, a file cine can be saved to various cine file formats. Saving a file cine to another file is also referred to by the term *conversion*.

2.3.2. Cine Use Case

As previously defined, a cine is a collection of images plus a header describing the images and also containing different metadata. An important part of this header is represented by the image processing parameters. Their values are stored separately so that the original sensor data (raw pixels) remains untouched allowing to "undo" any processing.

An application may need to display images from a cine, perform different analysis and measurement operations, save one or more images of this cine to various file formats. For each of these actions, images of the input cine are read, processed and further displayed or written to a file. However, none of these operations will affect the input raw cine. The input raw cine remains a collection of images accompanied by a header. For example, requesting images from a color camera CINE in order to be displayed will involve a demosaicing process and will produce images with 3 samples per pixel. Nevertheless, the source cine remains a cine containing images with 1 sample per pixel. The cine is not affected by any operations performed on its images after being read.

In order to indicate more intuitively all processing that must be done for images of the input cine, the user will specify a *Use Case* for that cine. It is a simple way of saying what a user wants to do with a cine, with its images.
One may want to get cine images in order to later display them or to save them in one of the supported file formats.

It is easier to specify that a certain set of camera cine images are to be displayed (hence color interpolated) or to be saved to a raw format (no color interpolation) than to explicitly call each processing one by one.

© 2023 Vision Research - AMETEK Materials Analysis Division

Phantom SDK introduced use cases as a way of simplifying image processing related configuration. Together with options that allow customizing use cases, it offers a very flexible and easy to use solution for either basic or advanced cine image processing.

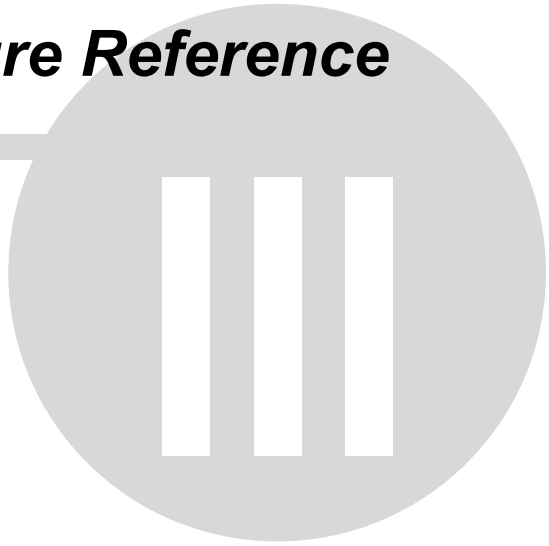
2.3.3. Putting It All Together - Creating a New Application

Any application built upon Phantom SDK should follow these basic steps:

- Register the application as a Phantom SDK client
  - ✓ `PhRegisterClientEx`
- Obtain a `CINEHANDLE` for the live cine
  - ✓ `PhGetCineLive`
- Prepare for a new recording session by configuring the camera and setting-up acquisition parameters for each partition
  - ✓ `PhSetCameraOption`
  - ✓ `PhSetSingleCineParams/PhSetCineParams`
  - ✓ `PhSet`
- In the same time, display live images
  - ✓ `PhGetCineImage`
- Start the recording
  - ✓ `PhRecordCine`
- End the recording
  - ✓ `PhSendSoftareTrigger`
- Obtain a `CINEHANDLE` for the newly registered cine
  - ✓ `PhNewCineFromCamera`
- Display cine images
  - ✓ `PhGetCineImage`
- Adjust recorded cine image parameters
  - ✓ `PhSetCineInfo`
- Read recorded cine acquisition parameters
  - ✓ `PhGetCineParams`
  - ✓ `PhGetCineInfo`
  - ✓ `PhGet`
- Play images from the recorded cine
  - ✓ `PhVideoPlay`
- Finalize
  - ✓ `PhDestroyCine`
- Unregister
  - ✓ `PhUnregisterClientEx`

© 2023 Vision Research - AMETEK Materials Analysis Division

Part III Data Structure Reference



© 2023 Vision Research - AMETEK Materials Analysis Division

3. Data Structure Reference

3.1. Time Related Data Structures

3.1.1. TIME64

```
typedef UINT32 FRACTIONS, *PFRACTIONS;

typedef struct tagTIME64
{
    FRACTIONS fractions;
    UINT32    seconds;
}TIME64, *PTIME64;
```

Header: PhInt.h, PhCon.h

fractions

Fractions of a second.
Stored here multiplied by 2^{32} and rounded to integer.
Least significant 2 bits store information about IRIG synchronization and the camera Event input:

Bit0 Value	Meaning
0	IRIG synchronized
1	Not synchronized

Bit1 Value	Meaning
0	Event input = 0 (short to ground)
1	Event input = 1 (open)

seconds

Seconds from January 1, 1970.
Compatible with the C library routines.
The maximum year allowed by the unsigned representation is 2106.

TIME64 is a compact format for storing very precise and absolute time. The internal representation is fixed point 32.32, the unit is second and the origin is January 1, 1970. The resolution of this representation is below 1 nanosecond and the allowed interval of years is from 1970 to 2106.
The least significant bit of *fractions* stores the information whether the IRIG is locked to an external signal. The second least significant bit (bit 1) stores the status of Event, a general-purpose binary input.
The type of the *seconds* field was changed to `UINT32` (from `time_t`) to enhance the compatibility with Visual Studio 2005 and above while maintaining its size to 32 bits.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.1.2. TC

```
typedef struct tagTC
{
      unsigned char framesU:4;
      unsigned char framesT:2;
      unsigned char dropFrameFlag:1;
      unsigned char colorFrameFlag:1;
      unsigned char secondsU:4;
      unsigned char secondsT:3;
      unsigned char flag1:1;
      unsigned char minutesU:4;
      unsigned char minutesT:3;
      unsigned char flag2:1;
      unsigned char hoursU:4;
      unsigned char hoursT:2;
      unsigned char flag3:1;
      unsigned char flag4:1;
      UINT          userBitData;
}TC, *PTC;
```

Header: PhInt.h, PhCon.h

framesU
Units of frames.
4 bits are used for storing this value.

framesT
Tens of frames.
2 bits are used for storing this value.

dropFrameFlag
Dropframe flag.
Currently not used. It is always 0.

colorFrameFlag
Colorframe flag.
Currently not used. It is always 0.

secondsU
Units of seconds.
4 bits are used for storing this value.

secondsT
Tens of seconds.
3 bits are used for storing this value.

flag1
Currently not used. It is always 0.

minutesU
Units of minutes.
4 bits are used for storing this value.

minutesT
Tens of minutes.
3 bits are used for storing this value.

© 2023 Vision Research - AMETEK Materials Analysis Division

flag2
Currently not used. It is always 0.

hoursU
Units of hours.
4 bits are used for storing this value.

hoursT
Tens of hours.
2 bits are used for storing this value.

flag3
Currently not used. It is always 0.

flag4
Currently not used. It is always 0.

userBitData
32 bits for storing user data.
Currently not used.

TC is used for representing time code according to SMPTE 12M-1999 standard. Time data, flag bits and user bits are all represented in **TC** precisely following the standard data packing specifications.
A time code is a 64bit code assigned to each video frame consisting of hours, minutes, seconds, a frame number, all counting for 32 bits, plus another 32 bits for storing user data. The time code is usually limited to one day duration. The frame number goes from 0 to (playback rate – 1).
The hours, minutes, seconds, frames are represented using BCD. From the four bits reserved for each digit some may never be used. The tens of hours, for example, cannot be larger than 2. Such unnecessary bits are assigned special uses (flags).
Note that the maximum playback rate is 40 (the maximum tens of frames is 3).

© 2023 Vision Research - AMETEK Materials Analysis Division

3.1.3. TCU

```
typedef struct tagTCU
{
      UINT frames;
      UINT seconds;
      UINT minutes;
      UINT hours;
      BOOL dropFrameFlag;
      BOOL colorFrameFlag;
      BOOL flag1;
      BOOL flag2;
      BOOL flag3;
      BOOL flag4;
      UINT userBitData;
}TCU, *PTCU;
```

Header: PhInt.h, PhCon.h

frames
Frame index.

seconds
Number of seconds.

minutes
Number of minutes.

hours
Number of hours.

dropFrameFlag
Dropframe flag.
Currently not used. It is always 0.

colorFrameFlag
Colorframe flag.
Currently not used. It is always 0.

flag1
Currently not used. It is always 0.

flag2
Currently not used. It is always 0.

flag3
Currently not used. It is always 0.

flag4
Currently not used. It is always 0.

userBitData
32 bits for storing user data.
Currently not used.

TCU is used for representing SMPTE time code. However, unlike `TC`, it does not follow the standard specifications regarding data packing. The *frames* index, the *hours* value, the *minutes* value, the *seconds* value are each represented by a decimal coded value.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.2. Acquisition Parameters

3.2.1. ACQUIPARAMS

```
typedef struct tagACQUIPARAMS
{
    UINT   ImWidth;
    UINT   ImHeight;
    UINT   FrameRate;
    UINT   Exposure;
    UINT   EDRExposure;
    UINT   ImDelay;
    UINT   PTFrames;
    UINT   ImCount;
    UINT   SyncImaging;
    UINT   AutoExposure;
    UINT   AutoExpLevel;
    UINT   AutoExpSpeed;
    RECT   AutoExpRect;
    BOOL   Recorded;
    TIME64 TriggerTime;
    UINT   RecordedCount;
    INT    FirstIm;
    UINT   FRPSteps;
    INT    FRPImgNr[16];
    UINT   FRPRate[16];
    UINT   FRPExp[16];
    UINT   Decimation;
    UINT   BitDepth;
    UINT   CamGainRed;
    UINT   CamGainGreen;
    UINT   CamGainBlue;
    UINT   CamGain;
    BOOL   ShutterOff;
    UINT   CFA;
    Char   CineName[256];
    Char   Description[4096];
    UINT   FRPShape[16];
    double dFrameRate
}ACQUIPARAMS, *PACQUIPARAMS;
```

Header: PhCon.h

Image Format

ImWidth
Image width.

ImHeight
Image height.

BitDepth
Acquisition bit depth.
Read-only field.

CFA
Color Filter Array of the camera the cine was recorded on.

© 2023 Vision Research - AMETEK Materials Analysis Division

CamGainRed
Red gain attached to a cine saved in the magazine. Normally it tells the White balance at recording time.

CamGainGreen
Green gain attached to a cine saved in the magazine.

CamGainBlue
Blue gain attached to a cine saved in the magazine.

CamGain
Global gain.

Main Acquisition Parameters

FrameRate
Frame rate in frames per second.
It is the requested frame rate represented as an integer. The camera will realize the closest possible frame rate, depending on its internal clock frequency. This parameter is normally constant during recording.
However, it is possible to dynamically change the frame rate by using the frame rate profile. See *FRPSteps*, *FRPImgNr*, *FRPRate* fields.
The actual frame rate can be obtained from the difference of two successive image times. To read a cine image time use PhGetCineAuxData with the `GAD_TIME` selector.

dFrameRate
High precision acquisition frame rate, replace uint32_t FrameRate

Exposure
Exposure duration in nanoseconds.
It is the requested value; the camera will do the closest possible value.
If the autoexposure mode is enabled, the exposure can change dynamically during the recording. See *AutoExposure*, *AutoExpLevel*, *AutoExpRect* fields.
To get the actual exposure duration for any acquired cine image use PhGetCineAuxData with the `GAD_EXPOSURE` selector.

PTFrames
Count of frames to be recorded after trigger.
It can be any non-zero, positive integer, up to 4 billion.
A value greater than *ImCount* has the effect of an adjustable delay; the frame buffer is a circular buffer and thus only the last *ImCount* images will be stored here.

SyncImaging
Defines the source of image acquisition pulses. Possible values are:

Value	Meaning
`SYNC_INTERNAL`	Repetitive acquisition at the specified *FrameRate* frequency
`SYNC_EXTERNAL`	It starts the exposure of each frame when an external signal rises
`SYNC_LOCKTOIRIG`	It allows the synchronization of multiple cameras using the common IRIG time signal

Cine Status Information

Recorded
If this field is `TRUE`, the cine was recorded and is available for playback. *TriggerTime*, *RecordedCount*, *ImCount*, *FirstIm* fields contain valid and final values. They are now read only fields.

© 2023 Vision Research - AMETEK Materials Analysis Division

CineName
Cine name.

Description
Cine description on maximum 4095 chars.

TriggerTime
Exact trigger time for a recorded cine. Its value is somewhere between the time of image `-1` and the time of image `0`.
Read-only field.

ImCount
For an empty partition, this field represents the maximum number of images that can be stored using current acquisition parameters.
For a partition holding a recorded cine, this field represents the number of cine images actually stored on this partition.
Read only field.

RecordedCount
Number of recorded frames.
ImCount may be smaller if the cine was partially saved to flash memory.
Read-only field.

FirstIm
First image number of this cine; it may be different from *PTFrames* - *ImCount* if the cine was partially saved to Non-Volatile memory.
Read-only field.

Advanced Acquisition Parameters

EDRExposure
Extreme dynamic range exposure duration (nanoseconds).

ImDelay
Delay in nanoseconds between the sync/irig and the start of the image exposure.
It is not available in all camera models.

AutoExposure
Control of the exposure duration based on the light conditions.

Value	Meaning
`0`	Disabled
`1`	Enabled, lock at trigger
`3`	Enabled, active after trigger

If auto exposure mode is enabled, the camera continuously adjusts the exposure duration to get the best images, even when light conditions change. In this case, the exposure the camera chooses is never greater than the *Exposure* value.

AutoExpLevel
Autoexposure level to be realized by the automatic exposure system.
Range of possible values is [0...255].

AutoExpSpeed
Speed for autoexposure control.
Currently not used.

AutoExpRect
Rectangle representing the region of interest where the autoexposure measures the average pixel level.

© 2023 Vision Research - AMETEK Materials Analysis Division

ShutterOff

This BOOL setting puts the shutter off to force the maximum exposure of the camera. It is useful for PIV where the camera is expected to have the shortest interval between successive exposures.

Frame Rate Profile

FRPSteps

Number of frame rate profile steps:

Value	Meaning
0	No frame rate profile
any other value	Number of points where the frame rate changes

The frame rate can be changed only after the trigger moment and the change can be a step to another value that will remain constant (flat) or can be continuously variable from a value to another value (ramp).

FRPImgNr

Image number where to change the acquisition frame rate.
The frame rate can be changed only after the trigger moment.
The acquisition parameters used before *FRPImgNr[0]* are the pretrigger parameters as stored in *FrameRate* and *Exposure* fields.
Allocated for 16 points.

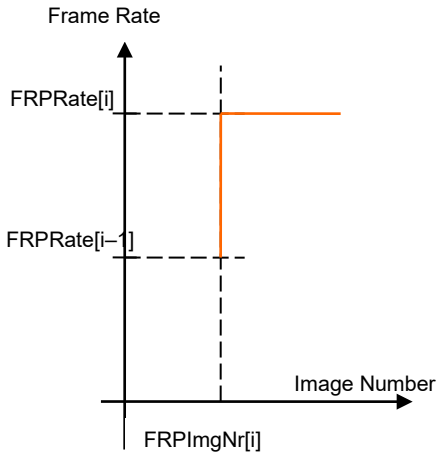
FRPRate

New value for frame rate (fps).
At image number *FRPImgNr[i]* the frame rate becomes *FRPRate[i].*
Allocated for 16 points.

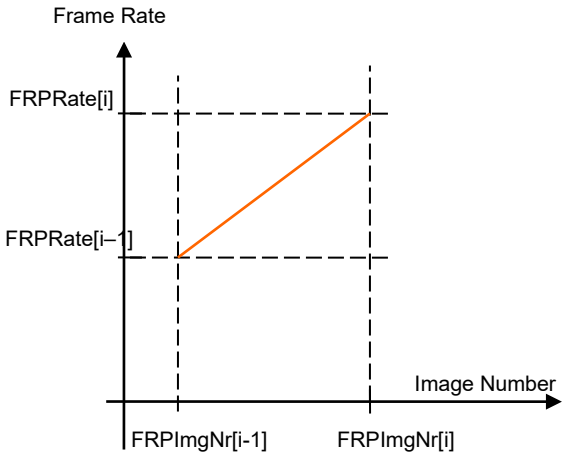
FRPShape

Frame rate evolution shape during recording.
Read-only parameter.
At image number *FRPImgNr[i]* the frame rate becomes *FRPRate[i]* following a shape described by *FRPShape[i].*
All the steps share the same shape: ramp in cameras that support it and flat in the older cameras.

Value	Meaning
0	flat
1	ramp



Flat Shape



Ramp Shape

© 2023 Vision Research - AMETEK Materials Analysis Division

FRPExp

New value for exposure (nanoseconds).
Currently not implemented in cameras - the exposure cannot be changed during the profile. All frame rates specified in the profile have to be smaller than the reciprocal of the exposure.

Other

Decimation

Factor by which to reduce the frame rate when sending images from camera. Currently used for i3 external memories.

ACQUIPARAMS structure gathers most of the camera parameters related to the acquisition process from basic acquisition parameters such as exposure time to more advanced options like frame rate profile. It is used when configuring a camera for a new acquisition, as well as when reading the parameters, a cine was recorded with. To configure a camera for a new recording, use `PhSetCineParams` or `PhSetSingleCineParams` functions depending on whether the camera is set to work on multi cine or single cine mode. To read the acquisition parameters from a stored camera cine use `PhGetCineParams` function.
The acquisition parameters as depicted by the ACQUIPARAMS structure are also stored in the nonvolatile memory of the camera. This way, the last settings remain active even after the camera is powered off and restarted.

Newer acquisition parameters are controlled by calling `PhSet`/`PhGet` functions with a corresponding selector (see **Error! Reference source not found.**). These are also the functions future Phantom SDK versions will develop for acquisition parameters control.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.3. Cine Related Data Structures

3.3.1. IMRANGE

```
typedef struct tagIMRANGE
{
    INT First;
    UINT Cnt;
}IMRANGE, *PIMRANGE;
```

Header: PhCon.h

First
: First image number.

Cnt
: Count of images.

IMRANGE defines a range of images by specifying the first image and the image count.

3.3.2. CINESTATUS

```
typedef struct tagCINESTATUS
{
    BOOL Stored;
    BOOL Active;
    BOOL Triggered;
}CINESTATUS, *PCINESTATUS;
```

Header: PhCon.h

Stored
: Recording of this cine finished.

Active
: Acquisition is currently taking place in this cine.

Triggered
: The cine has received a trigger.

CINESTATUS structure provides information about whether a recording finished, or the trigger signal was received. Use `PhGetCineStatus` to obtain a **CINESTATUS** array with status information for all the cines in a certain camera.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.4. Camera Related Data Structures

3.4.1. CAMERAOPTIONS

```
typedef struct tagCAMERAOPTIONS
{
    UINT OSD;
    UINT VideoSystem;
    BOOL ModulatedIRIG;
    BOOL RisingEdge;
    UINT FilterTime;
    BOOL MemGate;
    BOOL StartInRec;
    BOOL Color;
    BOOL TestImage;
    RGBQUAD OSDColor[8];
    BOOL OSDOpaque;
    Int  OSDLeft;
    int  OSDTop;
    int  OSDBottom;
    int  ImageX;
    int  ImageY;
    BOOL AutoSaveNVM;
    BOOL AutoSaveFile;
    BOOL AutoPlay;
    BOOL AutoCapture;
    CHAR FileName[MAX_PATH];
    IMRANGE SaveRng;
    BOOL LongReady;
    BOOL RealTimeOutput;
    UINT RangeData;
    UINT SourceCamSer;
    UINT SliceNr;
    UINT SliceCnt;
    BOOL FRPi3Trig;
    UINT UT;
    int  AutoSaveFormat;
    UINT SourceCamVer;
    UINT RAMBitDepth;
    int  VideoTone;
    int  VideoZoom;
    int  FormatWidth;
    int  FormatHeight;
    UINT AutoPlayCnt;
    UINT OSDDisable;
    BOOL RecToMag;
    BOOL IrigOut;
    Int  FormatXOffset;
    Int  FormatYOffset;
}CAMERAOPTIONS, *PCAMERAOPTIONS;
```

Header: PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

On Screen Display Options

OSD

If `TRUE`, it enables the camera On Screen Display.

Color

If `TRUE`, it enables OSD color in PAL/NTSC modes.

OSDColor

Array of 8 `RGBQUAD` elements describing various OSD colors. They have the following meaning:

Array Element	Meaning
OSDColor[0]	Background color for OSD fields.
OSDColor[1]	Logo color in waiting for trigger mode.
OSDColor[2]	Logo color in trigger mode.
OSDColor[3]	Logo color in cine stored mode.
OSDColor[4]	Logo color in preview mode.
OSDColor[5]	Foreground color for camera identification fields.
OSDColor[6]	Foreground color for acquisition parameter fields.
OSDColor[7]	Foreground color for status fields.

This parameter may not be available in newer cameras.

OSDOpaque

OSD text is opaque or transparent.

OSDLeft

Extra border on the left of the OSD text.

OSDTop

Extra border above the OSD text.

OSDBottom

Extra border below the OSD text.

UT

Time format as displayed by the OSD:

Value	Meaning
0	Local time.
1	Universal Time (GMT).

OSDDisable

Selective disable of the analog and digital OSD:

Value	Meaning
`0`	OSD on.
`1`	OSD turned off on monitor.
`2`	OSD turned off on recorder output.
`3`	OSD turned off on both the monitor and the recorded output.

Production Area Rectangle

FormatWidth

Width for the rectangle to overlap on the video image.

FormatHeight

Height for the rectangle to overlap on the video image.

© 2023 Vision Research - AMETEK Materials Analysis Division

FormatXOffset
x offset for the rectangle to overlap on the video image.

FormatYOffset
y offset for the rectangle to overlap on the video image.

Video Out Control

VideoSystem
Video output standard used.
Possible values are:

0	NTSC
1	PAL
4	720P60
5	720P50
12	720P59
20	1080P30
21	1080P25
28	1080P29
36	1080P24
44	1080P23
52	1080PSF30
53	1080PSF25
60	1080PSF29
68	1080I30
69	1080I25
76	1080I29
84	1080PSF24
92	1080PSF23

TestImage
If `TRUE`, a color bar test image is displayed on the connected monitor.

ImageX
Image X coordinate on screen.

ImageY
Image Y coordinate on screen.

AutoPlayCnt
Repeat count for the auto playback to video.

Value	Meaning
0	If *AutoCapture* is 0, the video play back loops forever. If *AutoCapture* is non zero, the video is played back only once and afterwards the acquisition is restarted.
Non zero value	Loop counter for the video play back.

VideoZoom
Zoom of the video image:

Value	Meaning
`0`	Fit.
`1`	Zoom 1.

© 2023 Vision Research - AMETEK Materials Analysis Division

Global Camera Options

RisingEdge

Trigger polarity.

Value	Meaning
TRUE	Camera triggers on the rising edge of the trigger signal.
FALSE	Camera triggers on the falling edge of the trigger signal.

ModulatedIRIG

If TRUE, IRIG input accepts modulated signal.

FilterTime

A time constant related to the filter the camera uses for ignoring spurious trigger signals.
Range of possible values: [0..255].

MemGate

Specifies the function for one of the capture connector pins.
For this pin, a TTL supplied input signal can be used to toggle the Memory Gate or as a Pre-trigger input signal.
When the Memory Gate feature is enabled, an input signal, when activated, prevents the storing of the acquired image data by disabling the write-access to memory.
A pre-trigger input, on the other hand, allows the user to decide when to place the camera into the Ready-State or Capture Mode.

Value	Meaning
TRUE	Memgate function
FALSE	Pre-trigger function

StartInRec

Specifies the default operational state a camera should be placed into when starting up.

Value	Meaning
TRUE	Start in recording mode. The camera is recording images and is waiting for trigger.
FALSE	Start in preview mode. The camera is waiting for pretrigger to start the recording. This mode saves power.

IrigOut

If TRUE, change the Strobe pin to Irig Out at Miro cameras.

LongReady

Value	Meaning
TRUE	Ready signal is 1 from the start to the end of recording.
FALSE	Ready signal is 1 from the start of recording until the cine is triggered.

RangeData

Range data size per image in bytes.

Value	Meaning
0	No range data.
Any other value	The range data size in bytes per image.

RecToMag

If TRUE, acquired images are recorded directly into the Cine Mag.

RealTimeOutput

If TRUE, it enables the real time output.

© 2023 Vision Research - AMETEK Materials Analysis Division

Camera Autosave

AutoSaveNVM
If `TRUE`, a camera having an internal flash storage will save the just-recorded cine to this storage.

AutoSaveFile
If `TRUE`, the camera will save the just-recorded cine to a file specified by the *FileName* field.

AutoPlay
If `TRUE`, the camera will play back the recorded cine on video immediately after the recording ends and after the autosave operations, if any, are finished.

AutoCapture
If `TRUE`, the camera will restart the acquisition immediately after the current recording ends and after the autosave operations, if any, are finished.

FileName
File path where to save a cine if *AutoSaveFile* is enabled.

SaveRng
Image range for the above operations.

AutoSaveFormat
Frame format used for file/flash cine save operations:

Value	Meaning
8	Calibrated 8bit format.
16	Calibrated 16bit format.
266	Calibrated 10bit packed format. Only for CineMag.

Image Format

RAMBitDepth
Bit depth used by the camera when acquiring image data.

VideoTone
It selects a predefined tone curve for video.

Image Cube

SourceCamSer
Source camera serial.

SliceNr
Image Cube Slice number (`0, 1, 2 ...`).

SliceCnt
Total count of Image Cube slices connected to a certain camera.

FRPi3Trig
Frame rate profile start at Image Cube trigger.

SourceCamVer
Source camera version (external memory slice).

CAMERAOPTIONS structure is used for general camera configuration: video output control, parameters adjustment for the OSD, camera behavior when restarting or when an acquisition ends.
To read current camera configuration, use PhGetCameraOptions function.
To change any of the **CAMERAOPTIONS** fields, read the current settings, update the fields you want to change and write the updated structure back to the camera by calling `PhSetCameraOptions` function.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.4.2. CAMERAID

```
typedef struct tagCAMERAID
{
    UINT IP;
    UINT Serial;
    char Name[256];
    char Model[256];
} CAMERAID, *PCAMERAID;
```

Header: PhCon.h

IP
Integer value representing a camera IP address.
Each of the four IP address values is stored on one unsigned byte.

Serial
Camera serial number.

Name
Camera name.

Model
Camera model.

CAMERAID is a simple structure storing basic information about a camera. It is currently used by `PhGetAllIpCameras` function.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.4.3. PULSEPARAM

```
typedef struct tagPULSEPRAAM
{
    UINT   Size;
    BOOL   Invert;
    BOOL   Falling;
    double Delay;
    double Width;
    double Filter;
}PULSEPARAM, *PPULSEPARAM;
```

Header: PhCon.h

Size
Size of the structure, for versioning in the future if needed

Invert
Invert the signal at the output of the pulse processor

Falling
Selects falling edge mode for the pulse processor. This mode is only relevant if width is also specified. When the falling token is present together with width, the pulse processor will generate a negative pulse, triggered from the negative edge of the input signal

Delay
Delay the output pulse by the specified time. clocks. If the width token is not present, both edges of the signal are delayed by the same amount. If the width token is present, the delay is measured from the rising edge of the input signal unless the falling token is present, in which case the delay is measured from the falling edge of the input. The delay time is specified in seconds and is internally converted and rounded to pixel clock units. The maximum delay time is at least 10 seconds

Width
When a width token is present, a defined-length pulse is generated, which starts after the specified delay after the active edge of the input signal. The length of the pulse is specified in seconds and internally converted to pixel clock units. The maximum pulse width is at least 10 seconds. However, if the period of the input signal is lower than the width, the latter is dynamically clamped to the signal period. The minimum pulse width is one pixel clock

Filter
When a filter token is present, the input of the pulse processor is filtered through an edge filter of the specified time. In order for the output of the filter to be asserted, the input signal must be continuously asserted for the specified duration. In order for the output of the filter to be deasserted, the input signal must be continuously deasserted for the same duration. The edges of the input are thus delayed by the specified filter time (for a “clean” input pulse). Filtering is applied before and independently of the delay and duration. The filter time is specified in seconds and is internally converted and rounded to pixel clock units. The maximum filter time is 1 second

PULSEPARAM is a simple structure for control of programmable I/O operation. It is currently used by `PhGetAllIpCameras1` function.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.5. Save Related Data Structures

3.5.1. SAVECINEINFO

```
typedef struct tagSAVECINEINFO
{
    CINEHANDLE CH;
    UINT progressPercent;
}SAVECINEINFO, *PSAVECINEINFO;
```

Header: PhFile.h

CH
Cine handle.

progressPercent
Current progress for a write cine to file operation.

SAVECINEINFO is used to store information about a write cine file in progress operation.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.6. Image Headers

Two data structures are used by the Phantom SDK to describe an image: **BITMAPINFOHEADER** and **IH**.

BITMAPINFOHEADER is the Windows data structure containing size and color image information. However, the Phantom SDK extends some fields meaning in order for images with more than 8bits per color component to be supported.

IH is the Phantom SDK specific image header, built as an extension of the **BITMAPINFOHEADER** structure.

Even if a **IH** structure is an extension of a **BITMAPINFOHEADER** structure, be careful to never use a cast for converting from one type to the other. Where one or more images are described by a **IH** structure while a **BITMAPINFOHEADER** one is needed, call `PhIHtoBITMAPINFOHEADER` function.

3.6.1. BITMAPINFOHEADER

```
typedef struct tagBITMAPINFOHEADER
{
    DWORD    biSize;
    LONG     biWidth;
    LONG     biHeight;
    WORD     biPlanes;
    WORD     biBitCount;
    DWORD    biCompression;
    DWORD    biSizeImage;
    LONG     biXPelsPerMeter;
    LONG     biYPelsPerMeter;
    DWORD    biClrUsed;
    DWORD    biClrImportant;
}BITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

Header: PhInt.h

biSize
The number of bytes required by this structure.

biWidth
Image width in pixels.

biHeight
Image height in pixels.
Phantom SDK specific:
This field is always positive. For `BI_RGB` formats, images are always stored bottom up. For `BI_PACKED` formats, images are always stored top-down.

biPlanes
It specifies the number of planes for the target device.
This value must be set to 1.

biBitCount
It specifies the number of bits-per-pixel used for storing a pixel value.
Phantom SDK specific:
Possible *biBitCount* values are limited to 8 and 16 for monochrome images, 24 and 48 for RGB color images.

© 2023 Vision Research - AMETEK Materials Analysis Division

The meaning of a 16 bit image is different from Windows: it is a 16 bit per pixel gray image. Each pixel value is stored on 16 bits even if the real bit depth produced by the camera is 14, 12 or 10 bits. 48 value of this field corresponds to a color image having 16 bits per color component. Color palette images (8 bpp) are not supported; in the Phantom environment they are converted to 24 bpp after the file load or after the copy from clipboard. The palette is not written in the cine file but a gray palette is needed to render the monochrome 8bpp DIBs in Windows.

biCompression

It specifies the type of compression for a compressed bottom-up bitmap (top-down DIBs cannot be compressed).

Phantom specific:

Only `BI_RGB = 0` and `BI_PACKED = 256` are supported.

When `BI_PACKED` is used, image data is packed without using any padding bits. For example, for 10bit image data, 4 sample values are stored in 5 bytes. This packing reduces used memory or file size by a factor of 10/16. A `BI_PACKED` bitmap is stored in top-down row order both in a packed cine file and in the camera memory.

biSizeImage

Image size in bytes.

biXPelsPerMeter

It specifies the horizontal resolution, in pixels-per-meter, of the target device for the bitmap. An application can use this value to select a bitmap from a resource group that best matches the characteristics of the current device.

Phantom specific:

biXPelsPerMeter, biYPelsPerMeter are resolutions computed at the camera sensor level. To get the scene resolution, you have to multiply these values by the distance from the camera to the scene divided by the focal length of the lenses.

biYPelsPerMeter

Vertical resolution in pixels per meter.

biClrUsed

Specifies the number of color indexes in the color table that are actually used by the bitmap. If this value is zero, the bitmap uses the maximum number of colors corresponding to the value of the *biBitCount* member for the compression mode specified by *biCompression*.

biClrImportant

It specifies the number of color indexes that are required for displaying the bitmap. If this value is zero, all colors are required.

Phantom specific:

If *biBitCount* is 16 or 48 *biClrImportant* should be the maximum sample value + 1, which is the maximum number of sample levels. *biClrImportant* has to be 16384, 4096 or 1024, for images having a bit depth of 14, 12 or 10 bits per sample.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.6.2. IH

```
typedef struct tagIH
{
    DWORD   biSize;
    LONG    biWidth;
    LONG    biHeight;
    WORD    biPlanes;
    WORD    biBitCount;
    DWORD   biCompression;
    DWORD   biSizeImage;
    LONG    biXPelsPerMeter;
    LONG    biYPelsPerMeter;
    DWORD   biClrUsed;
    DWORD   biClrImportant;
    int     BlackLevel;
    int     WhiteLevel;
}IH, *PIH;
```

Header: PhInt.h

biSize

biWidth

biHeight

biPlanes

biBitCount

biCompression

biSizeImage

biXPelsPerMeter

biYPelsPerMeter

biClrUsed

biClrImportant

All of the above fields have the same meaning as their corresponding fields in the **BITMAPINFOHEADER** structure.

BlackLevel

Black level for the image this **IH** describes.

WhiteLevel

White level for the image this **IH** describes.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.7. Image Processing Related Data Structures

3.7.1. WBGAIN

```
typedef struct tagWBGAIN
{
    float R;
    float B;
}WBGAIN, *PWBGAIN;
```

Header: PhInt.h, PhCon.h

R

White balance coefficient representing gain correction for red channel.
The gain coefficient for green channel is considered to be `1.0`.

B

White balance coefficient representing gain correction for blue channel.

WBGAIN is a data structure use for storing white balance gains. It is the basic data type used when referring to an image white balance.

3.7.2. IMFILTER

```
typedef struct tagIMFILTER
{
    int dim;
    int shifts;
    int bias;
    int Coef[5*5];
}IMFILTER, *PIMFILTER;
```

Header: PhInt.h

dim

Square convolution kernel size. It can be `3` for a `3x3` kernel or `5` for a `5x5` kernel.

shifts

Right shifts of *Coef* (`8` shifts mean divide by `256`). The shift is applied after the multiply and add operations of the convolution kernel.

bias

Bias to add after convolution.

Coef

Filter coefficients represented as integer values.
The actual value of the coefficients is *Coef[i,j]* shifted right *shifts* times. For example if *Coef[i,j]* is `1` and *shifts* is `4` the actual float value of the coefficient is `1/16 = 0.0625`. *Coef* can be used both for a `3x3 or a 5x5` filter.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.7.3. REDUCE16TO8PARAMS

```
typedef struct tagREDUCE16TO8PARAMS
{
    float fGain16_8;
}REDUCE16TO8PARAMS, *PREDUCE16TO8PARAMS;
```

Header: PhInt.h

fGain16_8
Gain coefficient to be applied when converting to 8 bits per sample.

When converting images to 8 bits per sample, the software can take the most significant 8 bits, the least significant 8 bits or anything intermediate. To provide a fine adjustment the conversion to 8 bits will be done by multiplying the input pixel value by the factor:
fGain16_8 * $(2^8 / 2^{bitdepth})$.
If *fGain16_8* is `1.0`, the maximum value of the input pixel is converted to `255` (this is the maximum value for 8 bits per sample data).

3.7.4. TONEDESC

```
typedef struct tagToneDesc
{
      Int   controlPointsCounter;
      float controlPoints[32 * 2];
      char  label[256];
}TONEDESC, *PTONEDESC;
```

Header: PhFile.h

controlPointsCounter
Number of control points describing the tone.

controlPoints
Tone control points array: x_0, y_0, …, $x_{controlPointsCounter - 1}$, $y_{controlPointsCounter - 1}$,
All values are in `[0..1]` range.

label
Optional string label describing this tone.

TONEDESC is the basic data type used for describing or referring to a tone.

© 2023 Vision Research - AMETEK Materials Analysis Division

3.7.5. CMDESC

```
typedef struct tagCmDesc
{
      float matrix[9];
      char  label[256];
}CMDESC, *PCMDESC;
```

Header: PhFile.h

matrix
A 3x3 RGB color matrix.

label
Optional color matrix label.

CMDESC is the basic data type used for describing or referring to a color matrix.

© 2023 Vision Research - AMETEK Materials Analysis Division

Part IV User Defined Functions

© 2023 Vision Research - AMETEK Materials Analysis Division

4. User Defined Functions

4.1. PROGRESSCALLBACK

```
typedef BOOL (WINAPI *PROGRESSCALLBACK)(UINT CN, UINT Percent);
```

Parameters

CN [in]
Camera number.

Percent [in]
Percentage of work done.

Return Value

Value	Meaning
TRUE	Continue
FALSE	Abort

Remarks

PROGRESSCALLBACK depicts a callback function type a user can decide to implement and further provide as an input parameter for different functions that take a long time to complete.

The time consuming operation usually refers to a camera. That is why the camera number is one of the callback function parameters.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

4.2. PROGRESSCB

```
typedef BOOL (WINAPI* PROGRESSCB)(UINT Percent, CINEHANDLE hC);
```

Parameters

Percent [in]
Percentage of work done.

hC [in]
Handle to the cine for which some work is in progress.

Return Value

Value	Meaning
TRUE	Continue saving
FALSE	Abort current save operation

Remarks

PROGRESSCB depicts a callback function type a user can decide to implement and further provide as an input parameter for different functions that take a long time to complete.

The time consuming operation usually refers to a cine. That is why the cine handle is one of the callback function parameters.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

4.3. DEMOSAICINGFUNCPTR

```
typedef void (*DEMOSAICINGFUNCPTR)(PBITMAPINFOHEADER pBMIH,
                                   PBYTE pPixel,
                                   UINT CFA,
                                   UINT Algorithm
);
```

Parameters

pBMIH [in, out]
Input: header describing input image data.
Output: header describing output image data.

pPixel [in, out]
Input: Image data buffer to be color interpolated.
Output: Buffer to store color interpolated image.

CFA [in]
Color filter array of the camera sensor the image was recorded with. Phantom SDK will call your function with one of the following values for the CFA parameter:

Value	Meaning
CFA_BAYER	GB RG
CFA_BAYERFLIP	RG GB
CFA_VRI	GBRG RGGB
CFA_VRIV6	BGGR GRBG

Algorithm [in]
Desired algorithm to be run for demosaicing the raw input image.

Return Value

None

Remarks

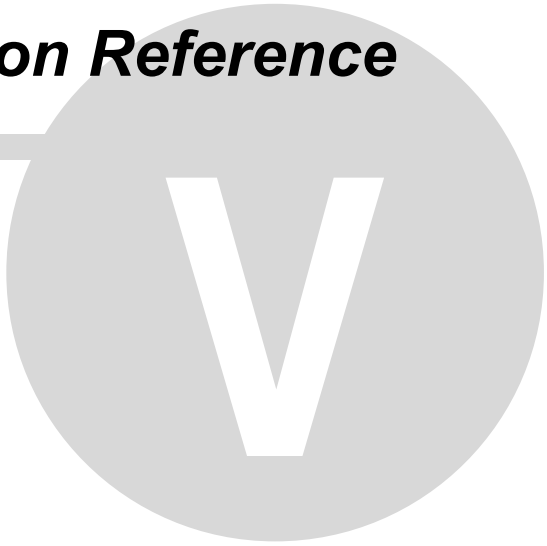
This type of function is needed if an application prefers using its own demosaicing algorithms, different from the ones Phantom SDK implements. By calling PhSetCineInfo together with GCI_DEMOSAICINGFUNCPTR selector and a pointer to your own color interpolation function, the next image request will let the specified function take care of color interpolation.
Using PhSetCineInfo/PhGetCineInfo together with GCI_INTALGO selector allows an application to control which of its interpolation algorithms will be run at a given time.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

Part V Function Reference



© 2023 Vision Research - AMETEK Materials Analysis Division

5. Function Reference

5.1. Initialize/Finalize

5.1.1. PhRegisterClientEx

```
HRESULT PhRegisterClientEx(HWND hWnd,
                           char *pStgPath,
                           PROGRESSCALLBACK pfnCallback,
                           UINT PhConHeaderVer
);
```

Parameters

hWnd [in, optional]
Main application window used to receive notifications from the old Phantom Firewire cameras. It can be NULL.

pStgPath [in, optional]
Path where to store camera stg files.
It can be NULL. In this situation, a default path will be used.

pfnCallback [in, optional]
Pointer to a function that will be called periodically to indicate progress while searching for connected cameras.
It can be NULL.

PhConHeaderVer [in]
Value indicating the PhCon version the caller is using. It provides information about the version of the Phantom SDK data structures the caller application is built with.
Use the constant PHCONHEADERVERSION from PhCon.h for this parameter.
It is useful for backward compatibility.

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

This function registers an application as a client of the Phantom dlls. Call PhRegisterClientEx before any other function (except for PhSetDllsOption) whether working with Phantom cameras or with cine files.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.1.2. PhUnregisterClient

```
HRESULT PhUnregisterClient(HWND hWnd);
```

Parameters

hWnd [in]
Use the same value for hWnd as in PhRegisterClientEx(). It can be NULL

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

The function unregisters an application previously registered as a client of the Phantom dlls. Call this function to ensure proper resource deallocation when the Phantom dlls are not needed anymore.

Required Header

PhCon.h

5.2. Create & Destroy Cine Objects

5.2.1. PhGetCineLive

```
HRESULT PhGetCineLive(INT CN, PCINEHANDLE phC);
```

Parameters

CN [in]
Camera number the requested live cine belongs to.

phC [out]
Pointer where to store live cine handle.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Returns the cinehandle associated with the live cine for the given camera. The live cine main use is reading live images from a certain camera. However, it can be used like any other cine as well; for example, for setting images processing parameters.
Please note that the live cinehandle is automatically created and automatically destroyed. `PhGetCineLive` is used just for obtaining this already allocated and initialized cinehandle. `PhDestroyCine` must not be called for the live cine.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.2.2. PhNewCineFromCamera

```
HRESULT PhNewCineFromCamera(INT CN, INT CineNr, PCINEHANDLE phC);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.
RAM stored cines are indexed beginning with `1`.
Flash cines are indexed beginning with value returned by the function `PhFirstFlashCine`.

phC [out]
Pointer where to store camera cine handle.

Return Value

The function returns `ERR_Ok` in case of success or `ERR_NOCINE` if there is no triggered cine in the camera. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function creates a cine handle that corresponds to a recorded camera cine. Call `PhDestroyCine` after finishing working with this cine.

Required Header

PhFile.h

5.2.3. PhNewCineFromFile

```
HRESULT PhNewCineFromFile(PSTR szFN, PCINEHANDLE phC);
```

Parameters

szFN [in]
Cine file name.

phC [out]
Pointer where to store file cine handle.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function creates a cine handle that corresponds to a cine file.
Call `PhDestroyCine` after finishing working with this cine.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.2.4. PhDuplicateCine

```
HRESULT PhDuplicateCine(PCINEHANDLE phCDest, CINEHANDLE hCSrc);
```

Parameters

phCDest [out]
Pointer or to store a new cine handle created by duplicating *hCSrc.*

hCSrc [in]
Source cine to be duplicated.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It creates a duplicate of the input cine.
It cannot be used for live cines.
Use this function when needing to run in parallel different operations for the same cine (e.g. reading images for display and reading images for saving to file) if cine parameters have changed from the moment the cine was created. Since some parameters are stored in internal buffers only, a `PhNewCineFromCamera` or `PhNewCineFromFile` call will also create a new cine, but it will not be aware of the parameters previously changed. This can happen especially for file cines where an image processing parameter change is stored only in internal buffers and written to file just when the cine is saved.
Call `PhDestroyCine` after finishing working with the duplicate cine.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.2.5. PhGetOpenCineName

```
BOOL PhGetOpenCineName(PSTR szFileName, UINT Options);
```

Parameters

szFileName [out]
String where to store selected file name.

Options [in]
Available options are:

Value	Meaning
0	The user can select only one file name.
OFN_MULTISELECT	The user can select one or more file names.

Return Value

Value	Meaning
TRUE	The operation has succeeded. The user has ended the dialog with OK.
FALSE	The operation has been canceled. The user has ended the dialog with CANCEL.

Remarks

This function displays a dialog box allowing the user to select an existing cine file. The file name is returned in the parameter.

Required Header

PhFile.h

5.2.6. PhDestroyCine

```
HRESULT PhDestroyCine(CINEHANDLE hC);
```

Parameters

hC [in]
Cine handle.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function should be called when finishing working with a specific cine. It frees any allocated resources the cine needed.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.3. Read Cine Images

5.3.1. PhGetCineImage

```
HRESULT PhGetCineImage(CINEHANDLE hC,
                       PIMRANGE pRng,
                       PBYTE pPixel,
                       UINT BufferSize,
                       PIH pIH
);
```

Parameters

hC [in]
Cine to read images from.

pRng [in, optional]
Requested image range.
For live cines, this parameter is `NULL`.

pPixel [out]
Buffer where to store requested image data.

BufferSize [in]
Buffer size in bytes.
If the input buffer is too small for reading and processing requested images the operation is aborted and `ERR_ImgBufferTooSmall` is returned.

pIH [out]
Image header describing buffer image data.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Reads and processes images from the input cine and stores them in *pPixel* buffer. The function can be used for any type of cine: live cine, camera cine or file cine.
Requested images are read from the source cine and then processed according to current use case and corresponding parameters. See PhSetUseCase, PhGetUseCase for more details.
The image header, *pIH*, fully describes the final image format.
Use *pRng* to specify what images you are interested in. For live cines, this parameter can be `NULL`, in which case the function will return one live image.
The pixel buffer, *pPixel*, has to be allocated by the caller. It is used for storing images all throughout the processing pipeline. For this reason, it must be large enough to accommodate all the images no matter the processing phase. To obtain the maximum memory size necessary for processing one image call `PhGetCineInfo` with the `GCI_MAXIMGSIZE` selector.
No more than one `PhGetCineImage` call per `CINEHANDLE` object must exist at any given time.
Use `PhIHtoBITMAPINFOHEADER` function if the image is used in a context where a `BITMAPINFOHEADER` is requested.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.3.2. PhSetUseCase

```
HRESULT PhSetUseCase(CINEHANDLE hC, int CineUseCaseID);
```

Parameters

hC [in]
Cine to set the use case for.

CineUseCaseID [in]
Desired use case. Possible values are:

Value	Meaning
UC_VIEW	Images requested from this cine are going to be displayed/played.
UC_SAVE	Images requested from this cine are going to be written to file.

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

A cine use case tells how this cine will be used and it mainly refers to cine images use. One can request cine images in order to save them to file or it can request cine images in order to display them in a video player application. Different processing steps and different optimizations apply in each of these situations. The use case represents a simple way of hiding all of these details.
By default, each cine has its use case set to UC_VIEW.
Please note that a cine (a CINEHANDLE object) cannot have more than one use case set at a time. When needing to display and save images from the same cine in parallel create one CINEHANDLE for each thread and set a corresponding use case for each of them. Check PhNewCineFromCamera, PhNewCineFromFile and PhDuplicateCine functions.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.3.3. PhGetUseCase

```
HRESULT PhGetUseCase(CINEHANDLE hC, int* pCineUseCaseID);
```

Parameters

hC [in]
Cine to get the use case from.

pCineUseCaseID [out]
Pointer where to store the use case.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Returns the current use case for the given cine.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4. Save Cine Images to File

5.4.1. PhGetSaveCineName

```
BOOL PhGetSaveCineName(CINEHANDLE hC);
```

Parameters

hC [in, out]
Cine to be saved.

Return Value

Value	Meaning
`TRUE`	The operation has succeeded. The user has ended the dialog with OK.
`FALSE`	The operation has been canceled. The user has ended the dialog with CANCEL.

Remarks

This function displays a browse dialog window allowing a user to specify various cine saving options: the user can select destination file name and file path, destination format and format options or the image range to be saved. All information about the user choices is stored inside the cine.
Cine saving options are also accessible by using PhSetCineInfo or PhGetCineInfo functions with corresponding selectors as described in **Error! Reference source not found.**.
Check also File Naming Convention chapter for special file name saving options.
PhWriteCineFile or `PhWriteCineFileAsync` can further be called for saving cine images to file according to previously configured options.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4.2. PhGetSaveCinePath

```
BOOL PhGetSaveCinePath(CINEHANDLE hC);
```

Parameters

hC [in, out]
Cine to be saved.

Return Value

Value	Meaning
TRUE	The operation has succeeded. The user has ended the dialog with OK.
FALSE	The operation has been canceled. The user has ended the dialog with CANCEL.

Remarks

This function displays a browse dialog window allowing a user to specify various options for saving a group of one or more cines: destination path, destination file naming parameters, destination format and format options. All information about the user choices is stored inside the input cine.
The caller has to further take care the same saving options are set for all the others cines in the group. This can be accomplished by calling PhGetCineInfo and PhSetCineInfo functions with corresponding selectors as described in **Error! Reference source not found.**. PhWriteCineFile or `PhWriteCineFileAsync` can then be called for each cine in the input cine group.
Check also File Naming Convention chapter for special file name saving options.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4.3. PhWriteCineFile

```
HRESULT PhWriteCineFile(CINEHANDLE hC, PROGRESSCB pfnCallback);
```

Parameters

hC [in]
Cine to be saved.

pfnCallback [in, optional]
Pointer to the progress callback function which shows the amount of work done. It can be `NULL`.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function saves the input cine to file. The progress callback function will provide progress updates periodically.
Before calling this function, make sure cine saving options are properly configured by calling either `PhGetSaveCineName` or PhSetCineInfo with corresponding selectors as described in **Error! Reference source not found.**.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4.4. PhWriteCineFileAsync

```
HRESULT PhWriteCineFileAsync(CINEHANDLE hC, PROGRESSCB pfnCallback);
```

Parameters

hC [in]
Cine to be saved.

pfnCallback [in, optional]
Pointer to the progress callback function which shows the amount of work done.
It can be `NULL`.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Asynchronous function used for saving a cine to file.
Asynchronous equivalent of `PhWriteCineFile`.
It starts a thread responsible for writing the file and immediately returns.
When the write operation finishes either because an error was encountered or because all data has been written to file, the callback function is called with the percent parameter set to 100.
To differentiate between the two possible situations, the caller should call `PhGetCineInfo` with `GCI_WRITEERR` code selector to check if any error was encountered.
At most one save operation per cine can be active at any given time.
Call `PhStopWriteCineFileAsync` to abort an asynchronous save operation.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4.5. PhStopWriteCineFileAsync

```
HRESULT PhStopWriteCineFileAsync(CINEHANDLE hC);
```

Parameters

hC [in]
Cine handle.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Stops an asynchronous save operation for the specified cine.

Required Header

PhFile.h

5.4.6. PhGetWriteCineFileProgress

```
HRESULT PhGetWriteCineFileProgress(CINEHANDLE hC,
                                   PUINT pProgressPercent
);
```

Parameters

hC [in]
Cine handle.

pProgressPercent [out]
Pointer where to store current progress percent.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Returns current save progress for the input cine.
It can be called for either an asynchronous or a synchronous write cine operation. However, please note that for a synchronous save operation the thread the save operation has been launched on is expected to be different from the thread this function is called from.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.4.7. PhGetWriteCineFileProgressAll

```
HRESULT PhGetWriteCineFileProgressAll(PSAVECINEINFO pSaveCineInfo,
                                      PUINT pCount
);
```

Parameters

pSaveCineInfo [out]
Preallocated array of `SAVECINEINFO` elements to be filled with information about all currently active write cine file operations.

pCount [in, out]
Input: *pSaveCineInfo* size.
Output: number of filled elements in *pSaveCineInfo* array.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Returns progress information about all synchronous and asynchronous currently active write cine file operations.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5. Cine Parameters

5.5.1. PhSetCineInfo

```
HRESULT PhSetCineInfo(CINEHANDLE hC, UINT Selector, PVOID pVal);
```

Parameters

hC [in]
The cine the information is set for.

Selector [in]
Numerical value identifying the parameter to be set.

pVal [in, out]
Input: pointer containing the value to be set.
Output: pointer containing accepted value.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It is the main function used for setting cine parameters such as image processing parameters or saving options. It requires as input data a cine handle, a numerical value indicating what parameter is to be modified and the actual value to be set.
Please note that the last parameter must be passed by reference! The function will return back to the caller the value actually accepted for the specified cine parameter.
Check **Error! Reference source not found.** for a complete list of selectors `PhSetCineInfo` accepts.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.2. PhGetCineInfo

```
HRESULT PhGetCineInfo(CINEHANDLE hC, UINT Selector, PVOID pVal);
```

Parameters

hC [in]
The cine the information is read from.

Selector [in]
Numerical value identifying the parameter to be read.

pVal [out]
Pointer to the buffer in which the value for the requested property is to be returned.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function reads the cine parameter specified by Selector. Check **Error! Reference source not found.** for a complete list of selectors `PhGetCineInfo` accepts.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.3. PhCineGet

```
HRESULT PhCineGet(UINT CN, INT CineNr, UINT Selector, PVOID pVal);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

Selector [in]
Numerical selector identifying the parameter to be read.

pVal [out]
Pointer where to store requested information.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function gets requested information for the specified camera cine. The cine is identified based on a cine number and the camera number it belongs to.
Check **Error! Reference source not found.** for possible selector values.

Important! This is a low level function. Use it only if neither `PhGetCineInfo` nor `PhGetCineParams` can provide needed information. There is no interaction between this function and `CINEHANDLE` objects!

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.4. PhCineSet

```
HRESULT PhCineSet(UINT CN, INT CineNr, UINT Selector, PVOID pVal);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

Selector [in]
Numerical selector identifying the parameter to be set.

pVal [in, out]
Input: value to be set.
Output: accepted value for the cine parameter.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Sets requested information for the specified camera cine. The cine is identified based on a cine number and the camera number it belongs to.
Please note that the last parameter must be passed by reference. The function will return back to the caller accepted value for the specified cine parameter.
Check **Error! Reference source not found.** for possible selector values.

Important! This is a low level function. Use it only if neither `PhSetCineInfo` nor `PhSetCineParams`/`PhSetSingleCineParams` can provide needed functionality. There is no interaction between this function and `CINEHANDLE` objects!

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.5. PhGetCineAuxData

```
HRESULT PhGetCineAuxData(CINEHANDLE hC,
                         INT ImageNumber,
                         UINT SelectionCode,
                         UINT DataSize,
                         PVOID pData
);
```

Parameters

hC [in]
Cine handle.

ImageNumber [in]
The number of the cine image.

SelectionCode [in]
Possible values are:

Value	Meaning
GAD_TIME	Acquisition time.
GAD_EXPOSURE	Exposure duration.
GAD_RANGE1	Range data.
GAD_BINSIG	Binary acquired signals.
GAD_ANASIG	Analog acquired signals.
GAD_SMPTETC	SMPTE time code
GAD_SMPTETCU	SMPTE time code unpacked

DataSize [in]
pData buffer size. It depends on the requested auxiliary data:

Requested Data	Size
GAD_TIME	sizeof(TIME64)
GAD_EXPOSURE	sizeof(UINT)
GAD_RANGE1	128 bits
GAD_BINSIG	(binary channel count + 7) / 8 x (acquired samples per image)
GAD_ANASIG	2 x (acquired samples per image) x (analog channel count)
GAD_SMPTETC	sizeof(TC)
GAD_SMPTETCU	sizeof(TCU)

pData [out]
Pointer where to store requested data.

Return Value

The function returns ERR_Ok in case of success, ERR_BadCine if cine handle was not initialized or ERR_AuxDataNotFound if auxiliary data was not found. To get the text of the error message call PhGetErrorMessage.

Remarks

This function returns in *pData* requested auxiliary data for a cine.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.6. PhMeasureCineWB

```
HRESULT PhMeasureCineWB(CINEHANDLE hC,
                        PPOINT pP,
                        int SquareSide,
                        PIMRANGE pRng,
                        PWBGAIN pWB,
                        PUINT pSatCnt
);
```

Parameters

hC [in]
Input cine. It can be a camera or a file cine.

pP [in]
The point around which white balance is to be computed.

SquareSide [in]
Side length of the square area used to compute white balance.

pRng [in]
Image range to analyze when computing white balance coefficients.

pWB [out, optional]
Address where to store corrective gains for red and blue.
This parameter can be `NULL` if not needed.

pSatCnt [out, optional]
Address where to store saturated pixel count.
This parameter can be `NULL` if not needed.

Return Value

The function returns `ERR_Ok` in case of success or `ERR_BadParam` if the input cine is not a raw cine. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function reads images from the input cine as specified by the *pRng* parameter and:

- if *pWB* is not `NULL`, it computes white balance gains
- if *pSatCnt* is not `NULL`, it computes average saturated pixel count

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.7. PhPrintTime

```
HRESULT PhPrintTime(CINEHANDLE hC,
                    INT ImNo,
                    UINT Options,
                    PSTR szTime
);
```

Parameters

hC [in]
The cine for which time information is requested.

ImNo [in]
The cine image number the time is requested for.

Options [in, optional]
Numerical value specifying time format options. Possible values are:

Value	Meaning
PPT_FULL	Full Time
PPT_DATE_ONLY	Date only
PPT_TIME_ONLY	Time only
PPT_FRAC_ONLY	Fractions of second only
PPT_ATTRIB_ONLY	Attributes only

szTime [out]
Char buffer to be filled with time data according to the specified format.

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

Prints the time for the given cine image according to current cine time format and specified format options. To control the time format, use PhSetCineInfo/PhGetCineInfo and the GCI_TIMEFORMAT selector.
Time format options are supported only for TF_LT and TF_UT time formats. For TF_SMPTE format, the time is always printed in full.
Here it is how the buffer can look like:

Time Format	Time Format Options	szTime
TF_UT	PPT_FULL	Fri May 20 2011 08:09:07.344 109.00 utc
TF_UT	PPT_DATE_ONLY	Fri May 20 2011
TF_UT	PPT_TIME_ONLY	08:09:07
TF_UT	PPT_FRAC_ONLY	.344 109.00
TF_UT	PPT_ATTRIB_ONLY	utc
TF_LT	PPT_FULL	Fri May 20 2011 11:07:23.344 109.00
TF_LT	PPT_DATE_ONLY	Fri May 20 2011
TF_LT	PPT_TIME_ONLY	11:07:23
TF_LT	PPT_FRAC_ONLY	.344 109.00
TF_LT	PPT_ATTRIB_ONLY	
TF_SMPTE	-	08:29:33:09 TC

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.5.8. PhGetCineTriggerTime

```
HRESULT PhGetCineTriggerTime(CINEHANDLE hC, PTIME64 pTriggerTime);
```

Parameters

hC [in]
The cine the trigger time is requested for.

pTriggerTime [out]
Buffer where to store the trigger time.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Gets the moment of the trigger for the input cine as a TIME64 value.

Required Header

PhFile.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6. Acquisition Parameters

5.6.1. PhGetCineParams

```
HRESULT PhGetCineParams(UINT CN,
                        INT CineNr,
                        PACQUIPARAMS pParams,
                        PBITMAPINFO pBMI
);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

pParams [out]
Pointer to an `ACQUIPARAMS` structure to be filled by this function.

pBMI [out]
Pointer to a `BITMAPINFO` structure to be filled by this function.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function is responsible for reading the acquisition parameters for a cine identified based on a cine number and a camera number for the camera the cine belongs to. It can be used both before and after the acquisition ends.
The function also fills an image header. Please allocate space for the `BITMAPINFOHEADER` and for the gray palette, `256 * sizeof(RGBQUAD)`.
Please note that this header can be different from the image header returned by `PhGetCineImage` depending on the image request parameters.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.2. PhSetCineParams

```
HRESULT PhSetCineParams(UINT CN, INT CineNr, PACQUIPARAMS pParams);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

pParams [in, out]
Input: buffer containing parameters to be set.
Output: buffer containing accepted acquisition parameters.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It configures the camera according to the acquisition parameters specified by *pParams* parameter. On return, this buffer contains validated and accepted values.
If the camera memory is a single partition one, use `PhSetSingleCineParams` function instead.

Required Header

PhCon.h

5.6.3. PhSetSingleCineParams

```
HRESULT PhSetSingleCineParams(UINT CN, PACQUIPARAMS pParams);
```

Parameters

CN [in]
Camera number.

pParams [in, output]
Input: buffer containing parameters to be set.
Output: buffer containing accepted acquisition parameters.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It configures the camera according to the acquisition parameters specified by *pParams* parameter. On return, this buffer contains validated and accepted values.
This function is to be used for single partition mode only.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.4. PhGetBitDepths

```
HRESULT PhGetBitDepths(UINT CN, PUINT pCnt, PUINT pBitDepths);
```

Parameters

CN [in]
Camera number.

pCnt [out]
Count of values in *pBitDepths*.

pBitDepths [out]
Array to be filled with possible recording bit depths for this camera.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It provides a list of the possible pixel bit depths when recording in the camera FBM. Example: the Phantom v9.1 camera can record at 8, 10, 12, 14, 16 bits per pixel. Larger bit depths may provide better image or better sensitivity but reduce the number of images that can be recorded in the camera memory.
The bit depths are stored in the array in increasing order.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.5. PhGetResolutions

```
HRESULT PhGetResolutions(UINT CN,
                         PPOINT pRes,
                         PUINT pCnt,
                         PBOOL pCAR,
                         PUINT pADCBits
);
```

Parameters

CN [in]
Camera number.

pRes [out, optional]
Array of `POINT` elements to be filled with resolutions available for this camera.
It can be `NULL` if this information is not needed, but in this case *pCnt* must be `NULL` as well.

pCnt [in, out, optional]
Input: Pointer containing *pRes* number of allocated elements.
Output: Pointer containing *pRes* number of filled elements.
It can be `NULL` if this information is not needed, but in this case *pRes* must be `NULL` as well.

pCAR [in, out]
At input, if *pCAR* is NULL the result in *pRes* will be a list of usual resolutions even if the camera supports Continuous Adjustable Resolution (CAR).
At output, if *pCAR* is not `NULL`:
**pCAR* will be `TRUE` if the camera supports CAR.
In this case the results in *pRes* are:

Value	Meaning
`pRes[0]`	Maximum Resolution (compatible with the old style).
`pRes[1]`	Minimum Resolution.
`pRes[2]`	Increments.

Available `(X, Y)` resolutions can further be generated following these guidelines:

```
X = MinX + i * IncX;     i = 0...M
Y = MinY + j * IncY;     j = 0...N
where:
      MaxX = pRes[0].x
      MaxY = pRes[0].y
      MinX = pRes[1].x
      MinY = pRes[1].y
      IncX = pRes[2].x
      IncY = pRes[2].y
      M = (MaxX - MinX) / IncX
      N = (MaxY - MinY) / IncY
```

pADCBits [out, optional]
Pointer where to store the maximum bit depth available for this camera. This is actually the analog to digital converters resolution.
It can be `NULL` if this information is not needed.

© 2023 Vision Research - AMETEK Materials Analysis Division

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It gets a list of image dimensions available for this camera, as well as the maximum pixel depth. If one resolution is requested and several are available, the maximum camera resolution is returned in *pRes*.

Required Header

PhCon.h

5.6.6. PhGetFrameRateRange

```
HRESULT PhGetFrameRateRange(UINT CN,
                            POINT Res,
                            UINT *pMinFrameRate,
                            UINT *pMaxFrameRate
);
```

Parameters

CN [in]
Camera number.

Res [in]
Image reference dimension.

pMinFrameRate [out]
Pointer where to store the minimum frame rate.

pMaxFrameRate [out]
Pointer where to store the maximum frame rate.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Given a reference resolution for a camera, this routine finds the minimum and the maximum available frame rates.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.7. PhGetExactFrameRate

```
HRESULT PhGetExactFrameRate(UINT CamVer,
                            UINT SyncMode,
                            UINT RequestedFrameRate,
                            double *pExactFrameRate
);
```

Parameters

CamVer [in]
Camera version.

SyncMode [in]
Source of image acquisition pulses.
It has the same value as the *SyncImaging* field of an `ACQUIPARAMS` structure containing current acquisition parameters.

RequestedFrameRate [in]
Requested frame rate.

pExactFrameRate [out]
The exact frame rate the camera can achieve, close to the requested one.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Given a requested frame rate, this function computes the exact frame rate the camera is able to accomplish. The period corresponding to this frame rate has to be an exact multiple of the internal clock period of the camera.
This routine does not check if the resulted rate is in the interval (minimum frame rate, maximum frame rate). Call `PhGetFrameRateRange` to validate returned frame rate.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.8. PhGetExposureRange

```
HRESULT PhGetExposureRange(UINT CN,
                           UINT FrameRate,
                           UINT *pMinExposure,
                           UINT *pMaxExposure
);
```

Parameters

CN [in]
Camera number.

FrameRate [in]
Frame rate.

pMinExposure [out]
Pointer where to store minimum exposure (nanoseconds).

pMaxExposure [out]
Pointer where to store maximum exposure(nanoseconds).

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function computes the limits of the exposure parameter for the specified frame rate.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.6.9. PhGetExposureRangeDouble

```
HRESULT PhGetExposureRangeDouble(UINT CN,
                                 double FrameRate,
                                 UINT *pMinExposure,
                                 UINT *pMaxExposure
);
```

Parameters

CN [in]
Camera number.

FrameRate [in]
Frame rate.

pMinExposure [out]
Pointer where to store minimum exposure (nanoseconds).

pMaxExposure [out]
Pointer where to store maximum exposure(nanoseconds).

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function computes the limits of the exposure parameter for the specified frame rate.

Required Header

PhCon.h

5.6.10. PhParamsChanged

```
HRESULT PhParamsChanged(UINT CN, PBOOL pChanged);
```

Parameters

CN [in]
Camera number.

pChanged [in]
If `TRUE`, changes have been made to the camera's acquisition parameters.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to determine whether another computer, remote control, or on-camera control unit changed the acquisition or image processing parameters of the camera. If *pChanged* is TRUE, changes have been made and the new settings must be read from camera.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7. Camera Pool

5.7.1. PhGetCameraCount

```
HRESULT PhGetCameraCount(UINT *pCnt);
```

Parameters

pCnt [out]
Pointer where to store camera count.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It gets the count of Phantom cameras connected to the computer.
If there are no connected cameras (or if you access a camera number above or equal to this value) the library switches to a simulated mode that allows you to test or to use the software without a camera connected.

Required Header

PhCon.h

5.7.2. PhConfigPoolUpdate

```
HRESULT PhConfigCameraPool(UINT Period);
```

Parameters

Period [in]
Interval between camera enumerations in milliseconds.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

PhConfigPoolUpdate() will enable automatic search for new or revived cameras. It will create a separate thread for enumeration of the cameras in Phantom SDK. Period will define the interval between two broadcasts.
If Period is 0, the function will stop the automatic enumeration (if enabled) and will do a single re-enumeration.
The periodic or single enumerations mentioned above will do both identifying there is a change in the camera pool and will activate that change (revive an Offline camera or add a new camera to the end of the pool).

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.3. PhGetAllIpCameras

```
HRESULT PhGetAllIpCameras(PUINT pCnt, PCAMERAID pCam);
```

Parameters

pCnt [out]
Address where to store discovered camera count.

pCam [out]
Buffer to be filled with `CAMERAID` elements describing discovered cameras.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It provides an option for getting information about connected cameras and can be called in parallel with other camera operations. It does not affect or interact with current camera pool.

Required Header

PhCon.h

5.7.4. PhOffline

```
BOOL PhOffline(UINT CN);
```

Parameters

CN [in]
Camera number

Return Value

The function returns TRUE when the connection to the camera was lost and FALSE if the Ethernet camera work normally.

Remarks

When an attempted access to a camera returns a timeout error, the
camera is not removed from the pool, but its status is changed to “Offline”.
If the connection to the camera is working again and PhConfigPoolUpdate() activated the enumeration thread, the camera is revived and can be accessed.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.5. PhMakeAllIpVisible

```
HRESULT PhMakeAllIpVisible();
```

Parameters

None

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Empties both the *Visible* and the *Ignored* camera lists. This way, any camera will become visible.

Required Header

PhCon.h

5.7.6. PhGetVisibleIp

```
HRESULT PhGetVisibleIp(PUINT pCnt, PUINT pIpAddress);
```

Parameters

pCnt [out]
Count of visible camera IP addresses.
Limited to `MAXCAMERACNT` (see PhCon.h).
pIpAddress [out]
Array of visible camera IP addresses.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to obtain an array with IP addresses for all the cameras in the visible or “Connect Only To” list. The visible list contains the cameras you wish to be connected to, no matter if they are or are not on the network of your computer at that moment.
If the visible list contains at least one camera, the computer connects only to cameras in the visible list and does not connect to cameras in the ignored list.
If you include a camera in both the visible and ignored lists, the visible list will have priority so the camera will be connected to your computer.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.7. PhAddVisibleIp

```
HRESULT PhAddVisibleIp(UINT IpAddress);
```

Parameters

IpAddress [in]
Camera IP address.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It adds a camera IP address to the visible list. The visible list cannot have more than `MAXCAMERACNT` (see PhCon.h) elements.

Required Header

PhCon.h

5.7.8. PhRemoveVisibleIp

```
HRESULT PhRemoveVisibleIp(UINT IpAddress);
```

Parameters

IpAddress [in]
Camera IP address.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It removes a camera IP address from the visible list.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.9. PhDisableVisible

```
HRESULT PhDisableVisible(BOOL bDisabled);
```

Parameters

bDisabled [in]
TRUE/FALSE enable or disable visible list.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Enable or disable the visible list. It allows the list to remain persistent when you want to switch temporary to “See all cameras” mode.

Required Header

PhCon.h

5.7.10. PhIsVisibleDisabled

```
HRESULT PhIsVisibleDisabled(void);
```

Parameters

None

Return Value

The function returns the current status of the Visible list.

Remarks

The visible list can be active or disabled.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.11. PhFillIDInfo

```
HRESULT PhFillIDInfo(PCAMERAID pCam);
```

Parameters

pCam [in, out]
Camera identification information including IP address, name, serial, model ….
IP is “in” parameter for this function,
all other are “out”.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

You have to fill the IP address of the camera pCam->IP (it has to be net order i.e. big endian here) and the function will fill the Serial, Name, Model, CFA, CamVer. This information is provided even if the camera is absent now but was seen in the past at least once. It can be used to populate your list of cameras with more friendly details about the camera.

Required Header

PhCon.h

5.7.12. PhGetIgnoredIp

```
HRESULT PhGetIgnoredIp(PUINT pCnt, PUINT pIpAddress);
```

Parameters

pCnt [out]
Count of ignored camera IP addresses.
Limited to `MAXCAMERACNT` (see PhCon.h).
pIpAddress [out]
Array of ignored camera IP addresses.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to obtain an array with IP addresses for all cameras in the ignored, or “Do Not Connect To” list. The ignored list contains the cameras which you do not wish to be connected to, no matter if they are or are not on the network of your computer at that moment.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.13. PhAddIgnoredIp

```
HRESULT PhAddIgnoredIp(UINT IpAddress);
```

Parameters

IpAddress [in]
Camera IP address.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It adds a camera IP address to the ignored list. The ignored list cannot have more than `MAXCAMERACNT` (see PhCon.h) elements.

Required Header

PhCon.h

5.7.14. PhRemoveIgnoredIp

```
HRESULT PhRemoveIgnoredIp(UINT IpAddress);
```

Parameters

IpAddress [in]
Camera IP address.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It removes a camera IP address from the ignored list.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.7.15. PhAddSimulatedCamera

```
HRESULT PhAddSimulatedCamera(UINT CamVer, UINT Serial);
```

Parameters

CamVer [in]
Simulated camera version.
Serial [in]
Desired serial for the simulated camera.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It adds a new simulated camera to the camera pool. `PhGetCameraCount` will count these cameras too.
You may later load a stg file for the camera having the specified serial and simulate the behavior and the calibrations of that camera.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8. Camera Parameters

5.8.1. PhGet

```
HRESULT PhGet(UINT CN, UINT Selector, PVOID pVal);
```

Parameters

CN [in]
Camera number.
Selector [in]
Selector value indicating the information to be retrieved.
pVal [out]
Pointer to the buffer in which the value for the requested property is to be returned.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function gets information about the given camera based on a selection code. For a complete list of currently available selectors, go to **Error! Reference source not found.**.

Required Header

PhCon.h

5.8.2. PhSet

```
HRESULT PhSet(UINT CN, UINT Selector, PVOID pVal);
```

Parameters

CN [in]
Camera number.

Selector [in]
Selector value indicating information to be set.

pVal [out]
Pointer to the buffer containing data to be set.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function performs a set operation for the given camera.
For a complete list of currently available selectors, go to **Error! Reference source not found.**.
Please note that the last parameter must be passed by reference. The function will return back to the called the value actually accepted for the specified parameter.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.3. PhGet1

```
HRESULT PhGet1(UINT CN, UINT Selector, UINT Selector1, PVOID pVal);
```

Parameters

CN [in]
Camera number.

Selector [in]
Selector value indicating the information to be retrieved.

Selector1 [in]
Programmable port number or other information, depending on Selector
Range of possible values is [1..N] where <N> is the value returned from `PhGet` using `gsPortCount` selector

pVal [out]
Pointer to the buffer in which the value for the requested property is to be returned.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function gets information about the given camera based on a selection code. For a complete list of currently available selectors, go to 9.5.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.4. PhSet1

```
HRESULT PhSet1(UINT CN, UINT Selector, UINT Selector1, PVOID pVal);
```

Parameters

CN [in]
Camera number.

Selector [in]
Selector value indicating information to be set.

Selector1 [in]
Programmable port number or other information, depending on Selector
Range of possible values is [1..N] where <N> is the value returned from `PhGet` using `gsPortCount` selector

pVal [out]
Pointer to the buffer containing data to be set.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function performs a set operation for the given camera.
For a complete list of currently available selectors, go to 9.5**Error! Reference source not found.**.
Please note that the last parameter must be passed by reference. The function will return back to the called the value actually accepted for the specified parameter.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.5. PhGet2

```
HRESULT PhGet2(UINT CN,
               UINT Selector,
               UINT Selector1,
               UINT Selector2
               PVOID pVal
);
```

Parameters

CN [in]
Camera number.

Selector [in]
Selector value indicating the information to be retrieved.

Name	Meaning	Type	Get	Set
gsSigName	Signal name, given the port number and signal number	Char[MAXSTDSTRSZ]	✓	✖

Selector1 [in]
Programmable port number
Range of possible values is [1..N] where <N> is the value returned from PhGet using gsPortCount selector

Selector2 [in]
Programmable port signal index
Range of possible values is [0..M-1] where <M> is the value returned from PhGet using gsSigCount selector

pVal [out]
Pointer to the buffer in which the value for the requested property is to be returned.

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

This function gets information about the given camera based on a selection code.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.6. PhGetCameraOptions

```
HRESULT PhGetCameraOptions(UINT CN, PCAMERAOPTIONS pOptions);
```

Parameters

CN [in]
Camera number.

pCAMERAOPTIONS[out]
Buffer to be filled with current camera options.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function reads general camera configuration parameters and stores them in the `CAMERAOPTIONS` structure the caller provided.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.7. PhSetCameraOptions

```
HRESULT PhSetCameraOptions(UINT CN, PCAMERAOPTIONS pOptions);
```

Parameters

CN [in]
Camera number.

pCAMERAOPTIONS [in]
Pointer to the camera options.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It configures the camera according to parameters specified by the `CAMERAOPTIONS` structure provided.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.8. PhGetVersion

```
HRESULT PhGetVersion(UINT CN, UINT VerSel, PUINT pVersion);
```

Parameters

CN [in]
Camera number.

VerSel [in]
Code to select the desired information.

pVersion [out]
Buffer where to store requested information.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It reads version information for the input camera as indicated by the *VerSel* parameter. *VerSel* parameter can have one of the following values:

Value	Meaning
`GV_CAMERA`	Camera hardware version.
`GV_FIRMWARE`	Camera firmware version.
`GV_FPGA`	Camera FPGA version.
`GV_PHCON`	PhCon.dll version.
`GV_CFA`	Sensor CFA.
`GV_KERNEL`	Camera kernel version.
`GV_MAGAZINE`	Camera magazine version.
`GV_FIRMWAREPACK`	Camera firmware packed version. (Ph16 Cameras)
`GV_HEAD_FIRMWARE`	Miro N5 head firmware version.
`GV_HEAD_FPGA`	Miro N5 FPGA.
`GV_HEAD_FIRMWAREPACK`	Miro N5 firmware packed version.
`GV_VSHUTTER_FIRMWARE`	V Shutter firmware version.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.9. PhGetCameraID

```
HRESULT PhGetCameraID(UINT CN, UINT *pSerial, char *pCameraName);
```

Parameters

CN [in]
Camera number.

pSerial [out, optional]
Pointer where to store the serial number of the camera.
It can be `NULL`.

pCameraName [out, optional]
Buffer where to store camera name.
Maximum 15 characters.
It can be `NULL`.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It reads camera name and camera factory serial number.
The serial number is unique for each camera while the camera name can be changed by the user.

Required Header

PhCon.h

5.8.10. PhSetCameraName

```
HRESULT PhSetCameraName(UINT CN, char *pCameraName);
```

Parameters

CN[in]
Camera number.

pCameraName [in]
Buffer containing camera name on maximum 15 characters.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It allows the user to change the camera's name.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.11. PhFindCameraNumber

```
HRESULT PhFindCameraNumber(UINT Serial, UINT *pCN);
```

Parameters

Serial [in]
Factory serial number.

pCN [out]
Pointer where to store camera number.

Return Value

The function returns `ERR_Ok` in case of success or `ERR_SerialNotFound` if it does not find the camera with the specified serial.
To get the text of the error message call `PhGetErrorMessage`.

Remarks

It scans all cameras inside the camera pool to find the one with the specified serial number.

Required Header

PhCon.h

5.8.12. PhGetCameraModel

```
HRESULT PhGetCameraModel(UINT CamVer, PSTR pModel);
```

Parameters

CamVer [in]
Camera version.

pModel [out]
Buffer where to store a text description of the camera model.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to obtain a text description corresponding to an integer camera model value. For example, when *CamVer* is `120` the result in `pModel` will be `"Phantom v12"`.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.13. PhGetCameraTime

```
HRESULT PhGetCameraTime(UINT CN, PTIME64 pTime64);
```

Parameters

CN [in]
Camera number.

pTime64 [out]
Buffer where to put the current time in the `TIME64` format.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It reads the camera clock.
The fractions of second are important for the information stored in bit 0 and bit 1: the presence of IRIG synchronization signal and the Event binary input.
Use the standard `ctime` function to convert it to a string of characters.

Required Header

PhCon.h

5.8.14. PhSetCameraTime

```
HRESULT PhSetCameraTime(UINT CN, UINT32 Time);
```

Parameters

CN [in]
Camera number.

Time [in]
Seconds from January 1, 1970 0:0:0.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It sets the camera clock (for example from the computer clock).
If IRIG signal is present, this operation will fail.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.15. PhMaxCineCnt

```
HRESULT PhMaxCineCnt(UINT CN);
```

Parameters

CN [in]
Camera number.

Return Value

The function returns the maximum number of cines that can be in the camera.

Remarks

Phantom cameras have a preview cine numbered as 0 and a number of recordable cines numbered from 1 upward. Depending on camera the maximum count of recordable cines can be 63, 511. This function replace the constant MAXCINECNT from previous versions of the SDK.

Required Header

PhCon.h

5.8.16. PhFirstFlashCine

```
INT PhFirstFlashCine(UINT CN);
```

Parameters

CN [in]
Camera number.

Return Value

The function returns the offset for the numbers of the cines available in the camera flash (magazine, SSD).

Remarks

The function replaces the previous constant FIRST_FLASH_CINE and has to be used to compute the cine numbers to access the recordings from flash.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.17. PhGetPartitions

```
HRESULT PhGetPartitions(UINT CN, PUINT pCount, PUINT pPartitionSize);
```

Parameters

CN [in]
Camera number.

pCount [in, out]
Input: Pointer containing *pPartitionSize* number of allocated elements.
Output: Pointer containing *pPartitionSize* number of elements filled by this function

pPartitionSize [out]
Array to be filled with memory size for each camera partition.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function fills the input array with memory size in MB for each camera partition.

Required Header

PhCon.h

5.8.18. PhSetPartitions

```
HRESULT PhSetPartitions(UINT CN, UINT Count, PUINT pWeights);
```

Parameters

CN [in]
Camera number.

Count [in]
Partition count.

pWeights [in]
Array of weights of the memory size to be allocated to each cine. To get the percentage of each partition, it multiplies its weight by `100` and divides the result by the sum of all weights.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function is used for partitioning a camera memory. The caller must specify desired partition count as well as a weight for each partition size. All recorded cines are deleted when this function is called.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.19. PhGetCineStatus

```
HRESULT PhGetCineStatus(UINT CN, PCINESTATUS pStatus);
```

Parameters

CN [in]
Camera number.

pStatus [out]
Array of `CINESTATUS` elements to be filled with information about each cine status. The buffer must be allocated for the maximum cine count ij the camera obtained using PhMaxCineCnt().

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It checks the status for all cines in this camera (finished, active, or triggered).

Required Header

PhCon.h

5.8.20. PhGetCineCount

```
HRESULT PhGetCineCount(UINT CN, PUINT pRAMCount, PUINT pNVMCount);
```

Parameters

CN [in]
Camera number.

pRAMCount [out]
Pointer where to store recorded cine count.

pNVMCount [out]
Pointer where to store NVM stored cine count.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It counts the RAM and NVM cines available for playback from the specified camera.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.8.21. PhMemorySize

```
HRESULT PhMemorySize(UINT CN, PUINT pDRAMSize, PUINT pNVMSize);
```

Parameters

CN [in]
Camera number.

pDRAMSize [out]
Pointer where to store the DRAM memory size in MB.

pNVMSize [out]
Pointer where to store the NonVolatile memory size in MB.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It provides the size of the camera DRAM and NonVolatile Memory.

Required Header

PhCon.h

5.9. Camera Control

5.9.1. PhRecordCine

```
HRESULT PhRecordCine(UINT CN);
```

Parameters

CN [in]
Camera number.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It starts a new recording in the first partition of the camera.
Any previously recorded cine will be overwritten. Please note that this operation may result in unwanted data loss if the initial cine is not saved.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.9.2. PhRecordSpecificCine

```
HRESULT PhRecordSpecificCine(UINT CN, int CineNr);
```

Parameters

CN [in]
Camera number.
CineNr [in]
Cine number for which to start the recording.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It starts a new recording in the specified partition of a multicine camera. It deletes the existing cine if there is one recorded.

Required Header

PhCon.h

5.9.3. PhSendSoftwareTrigger

```
HRESULT PhSendSoftwareTrigger(UINT CN);
```

Parameters

CN [in]
Camera number.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It sends a software trigger to the camera. This is a handy way to finish a recording.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.9.4. PhDeleteCine

```
HRESULT PhDeleteCine(UINT CN, INT CineNr);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It deletes the specified camera cine. A new recording may use the parameters and the space of that cine.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.9.5. PhVideoPlay

```
HRESULT PhVideoPlay(UINT CN,
                    INT CineNr,
                    PIMRANGE pRng,
                    INT PlaySpeed
);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number for a recorded camera cine.

pRng [in, optional]
Range of cine images to be played.
If this is `NULL`, the function changes only the playback speed to the value specified in the *PlaySpeed* parameter.

PlaySpeed [in]
Playback speed in milihertz.
Negative values means play backward:

Value	Meaning
`1000`	1 frame per second
`< 0`	Play backward.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It plays back a clip from the specified cine to the camera video out.

Parameter Values	Meaning
`CineNr < 1`	The video will show preview images.
`pRng->Cnt == 0` or `PlaySpeed == 0`	Continuously play `pRng->First` image (Pause Mode). Using this feature, one can implement Step Forward / Step Backward.
`pRng->Cnt != 0`	Playback the specified range of images to the video output forward (if `PlaySpeed` is positive) or backward (if `PlaySpeed` is negative).

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.9.6. PhGetVideoFrNr

```
HRESULT PhGetVideoFrNr(UINT CN, PINT pCrtIm);
```

Parameters

CN [in]
Camera number.

pCrtIm [out]
Pointer where to store current image number being played back to the video output.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It gets the image number that is played back to the video out at the current moment. It can be used to update a browse slider or a numeric display in a video player.

Required Header

PhCon.h

5.9.7. PhGetCameraErrMsg

```
HRESULT PhGetCameraErrMsg(UINT CN, char *pErrText);
```

Parameters

CN [in]
Camera number.

pErrText [out]
Error message.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It provides the last error message from the camera firmware. After calling the function, the error buffer of this camera is emptied.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.10. Camera Calibration

5.10.1. PhBlackReference

```
HRESULT PhBlackReference(UINT CN, PROGRESSCALLBACK pfnCallback);
```

Parameters

CN [in]
Camera number.

pfnCallback [in]
Progress indicator function.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function is responsible for controlling a new black reference operation.
Images are acquired in the darkness and averaged in order to determine pixel offset values. These offsets are stored in internal tables, in the stg file and, depending on the camera version, inside the camera as well.
During a black reference operation some acquisition parameters may be changed, but they will be restored to their initial values after the calibration finishes.
Because images must be acquired in the dark, camera lenses must be covered while running a black calibration. This is not a problem for cameras having a mechanical shutter: PhBlackReference will take care of controlling the shutter as needed. Call `PhGet` function with the `gsHasMechanicalShutter` selector to find out if a camera has a mechanical shutter or not.
After the black reference operation finished, call `PhWriteStgFlash` for the new offset values to be sent to camera. This way the same calibration values will be used for video output.
Better results can be obtained by calling `PhBlackReferenceCI` which does the calibration without changing the resolution or other acquisition parameters.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.10.2. PhBlackReferenceCI

```
HRESULT PhBlackReferenceCI(UINT CN, PROGRESSCALLBACK pfnCallback);
```

Parameters

CN [in]
Camera number.

pfnCallback [in]
Progress indicator function.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function is responsible for controlling a new black reference operation at the current acquisition parameters, also names *Current Session Reference* or *CSR*.
Images are acquired in the darkness and averaged in order to determine pixel offset values. These offsets are to be used for the current acquisition parameters only. Any acquisition parameter change may require repeating the black reference operation.
Because images must be acquired in the dark, camera lenses must be covered while running a black calibration. This is not a problem for cameras having a mechanical shutter: PhBlackReferenceCI will take care of controlling the shutter as needed. Call `PhGet` function with the `gsHasMechanicalShutter` selector to find out if a camera has a mechanical shutter or not.
Newer cameras are able to perform themselves this operation. Offsets are stored inside the camera and automatically applied to image pixels. This way, the new calibration will immediately take effect on the video output.
However, for the new offsets to take effect on older cameras' video output as well, they must be explicitly written to camera by calling `PhWriteStgFlash` function.
To distinguish between the two types of cameras call `PhGet` function with the `gsSupportsOffGainCorrections` selector.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.10.3. PhWriteStgFlash

```
HRESULT PhWriteStgFlash(UINT CN, PROGRESSCALLBACK pfnCallback);
```

Parameters

CN [in]
Camera number.

pfnCallback [in]
Pointer to the progress indicator function. This operation may take longer time for high resolution cameras.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to send current calibration information to the camera flash in order to be used for the video output.
This operation is necessary:

- after a CSR using `PhBlackReferenceCI` if the current camera does not implement offset gain corrections internally
  (It returns `FALSE` on calling `PhGet` function with the `gsSupportsOffGainCorrections` selector)
- after performing a black calibration using `PhBlackReference`

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11. NVM

5.11.1. PhNVMGetStatus

```
HRESULT PhNVMGetStatus(UINT CN,
                       PUINT pCineCnt,
                       PUINT32 pTime,
                       PUINT pFreeSp
);
```

Parameters

CN [in]
Camera number.

pCineCnt [in, out]
Input: *pTime* number of elements.
Output: Count of cines stored in NV memory.

pTime [out, optional]
Array with the trigger time in seconds for each cine.
It can be NULL.

pFreeSp [out]
Free space in NVM in MB.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function reads the number of NVM stored cines and, optionally, NVM free space. Also optionally, *pTime* array can be filled with each cine trigger time represented in seconds.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11.2. PhNVMErase

```
HRESULT PhNVMErase(UINT CN, PROGRESSCALLBACK pfnCallback);
```

Parameters

CN [in]
Camera number.

pfnCallback [in]
Callback function to see the progress.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It erases all the cines from camera NVM.

Required Header

PhCon.h

5.11.3. PhNVMEraseCine

```
HRESULT PhNVMEraseCine(UINT CN, int CineNr);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

.Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It erases selected cine from removable SSD camera NVM.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11.4. PhNVMContRec

```
HRESULT PhNVMContRec(UINT CN,
                     PBOOL pSave,
                     PUINT pOldValue,
                     PUINT pEnable
);
```

Parameters

CN [in]
Camera number.

pSave [out]
If TRUE, this camera is saving the cine to the NVM or is automatically playing back to video so the images cannot be downloaded – that happens at Firewire cameras.

pOldValue [out, optional]
Buffer where to store previous status information. It is a combination of the following flags (see PhCon.h):

Value	Meaning
NVCR_APV	Automatic playback to video.
NVCR_CONT_REC	Continuous recording to NVM mode.
NVCR_REC_ONCE	Recording once mode.

pEnable [in, optional]
Pointer where a numerical value is stored. This numerical value controls the continuous recording mode. It is a combination between values described for *pOldValue*.

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

It sets the continuous recording mode for the nonvolatile memory. When a cine is available it is saved in the NV memory and the recording of a new cine is automatically started. This function tests if a cine save to NVM operation is in progress in a Firewire camera using the pSave.
An easier way of controlling these automatic actions is to configure *AutoSaveNVM*, *AutoSaveFile*, *AutoPlay* and *AutoCapture* fields of the CAMERAOPTIONS structure and further call PhSetCameraOptions function.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11.5. PhNVMGetSaveRange

```
HRESULT PhNVMGetSaveRange(UINT CN, PIMRANGE pRng);
```

Parameters

CN [in]
Camera number.

pRng [out]
Image range.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It gets the current image range configured for a NVM Save or NVM continuous recording.

Required Header

PhCon.h

5.11.6. PhNVMSetSaveRange

```
HRESULT PhNVMSetSaveRange(UINT CN, PIMRANGE pRng);
```

Parameters

CN [in]
Camera number.

pRng [in]
Image range.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It sets the image range to be used for a NVM save or NVM continuous recording. It does not affect the range used by `PhNVMSaveClip`.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11.7. PhNVMSave

```
HRESULT PhNVMSave(UINT CN, INT CineNr, PROGRESSCALLBACK pfnCallback);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

pfnCallback [in]
Callback function to see the progress.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Saves the specified DRAM recorded cine to camera non–volatile memory.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.11.8. PhNVMSaveClip

```
HRESULT PhNVMSaveClip(UINT CN,
                      int CineNr,
                      PIMRANGE pRng,
                      UINT Options,
                      PROGRESSCALLBACK pfnCallback
);
```

Parameters

CN [in]
Camera number.

CineNr [in]
Cine number.

pRng [in]
Image range to be saved.
`NULL` pointer means save full cine.

Options [in]
Use `GI_BPP16` to save the cine on 16 bits. If absent, the bit depth will be 8 bits.

pfnCallback [in]
Callback function to see the progress.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Saves the specified DRAM recorded cine to camera non-volatile memory. This saving does not depend on the range set by `PhNVMSetSaveRange`.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.12. DLLs Options

5.12.1. PhSetDllsOption

```
HRESULT PhSetDllsOption(UINT optionSelector, PVOID pValue);
```

Parameters

optionSelector [in]
Value indicating the option to be set. Possible values are:

Value	Meaning
DO_IGNORECAMERAS	Ignore all cameras when a new application registers as a DLL client.

pValue [out]
Buffer containing the value to be set. This value can have one of the following types:

Selector Value	Type
DO_IGNORECAMERAS	INT

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

Sets various user options regarding Phantom dlls use. This function must be called before PhRegisterClientEx.

Required Header

PhCon.h

5.12.2. PhGetErrorMessage

```
HRESULT PhGetErrorMessage(int ErrNo, char *pErrText);
```

Parameters

ErrNo [in]
Error code returned by Phantom functions.

pErrText [out]
A text that describes the error.
Allocate this buffer for MAXERRMESS (see PhCon.h) number of characters.

Return Value

The function returns ERR_Ok in case of success or ERR_UnknownErrorCode if the error code is unknown.

Remarks

It provides an ASCII string with the description of each error code.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.12.3. PhSetDllsLogOption

```
HRESULT PhSetDllsLogOption(UINT SelectCode, UINT Value);
```

Parameters

SelectCode [in]

It selects the module to set the log option for.
To select a certain module or option use the constants defined in PhCon.h:

Value	Meaning
SDLO_PHANTOM	Phantom application logging
SDLO_PHCON	PhCon module logging
SDLO_PHINT	PhInt module logging
SDLO_PHFILE	PhFile module logging
SDLO_PHSIG	PhSig module logging
SDLO_PHSIGV	PhSigV module logging
SDLO_TORAM	Logging to RAM. Faster, but lost if application crashes.

Value [in]

Value	Meaning
0xFFFFFFFF	Enable all log dumps from a module
0	Disable all log dumps

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

This function allows enabling and disabling logging for each module. For example, to enable logging for the PhCon module, use the SDLO_PHCON selector with *Value* set to 0xFFFFFFFF. To stop logging use the same selector with *Value* set to 0.
These settings are also accessible in Help/About/Logging from Phantom or in Manager/Preferences from PCC.
These settings are stored in Windows registry and are unique for all applications using the Phantom SDK.
For a minimum impact on application performance, set SDLO_TORAM to 1. However, please note that if the application crashes or fails to unregister, log information may be lost. To debug crashes set SDLO_TORAM to 0.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.12.4. PhGetDllsLogOption

```
HRESULT PhGetDllsLogOption(UINT SelectCode, PUINT pValue);
```

Parameters

SelectCode
It selects the module to get log options for. See `PhSetDllsLogOption` for possible values.

pValue
Pointer where to store requested log option value.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Call this function to find the current logging settings. These settings are used by all applications build on Phantom libraries and are persistent.

Required Header

PhCon.h

5.13. Other - Image Handling & Processing

5.13.1. PhIHtoBITMAPINFOHEADER

```
HRESULT PhIHtoBITMAPINFOHEADER(PIH pIH, PBITMAPINFOHEADER pBMIH);
```

Parameters

pIH [in]
Buffer containing input image header to be converted.

pBMIH [out]
Buffer where to store the BITMAPINFOHEADER equivalent of the input image header.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function converts a `IH` structure to a `BITMAPINFOHEADER` structure. The two input pointers may be identical in which case the conversion is done in place. Use this function whenever a `BITMAPINFOHEADER` structure is needed for describing one or more images for which only a `IH` structure is available. Please note that even if `IH` is an extension of `BITMAPINFOHEADER` structure it is not safe to use a cast for converting from one type to the other.

Required Header

PhInt.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.13.2. PhProcessImage

```
HRESULT PhProcessImage(PBYTE pPixelSrc,
                       PBYTE pPixelDest,
                       PIH pIH,
                       UINT ProcID,
                       PVOID pProcParams
);
```

Parameters

pPixelSrc [in]
Input pixel data to be processed.

pPixelDest [out]
Output buffer containing processed data.

pIH [in, out]
Input: image header describing input data.
Output: image header describing output data.

ProcID [in]
Numerical value indicating the processing operation to be done. Possible values are:

Value	Meaning
`IMG_PROC_REDUCE16TO8`	Reduce image bit depth to 8 bits per sample

pProcParams [in]
Pointer to image processing parameters. The image processing parameters can be of one of the following data types:

procID	Data type
`IMG_PROC_REDUCE16TO8`	`REDUCE16TO8PARAMS`

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

The function processes the input cine as indicating by the numerical selector *ProcID* and according to *pProcParams* parameters.
pPixelSrc and *pPixelDest* may be identical, in which case, the operation is done in place.
Use `PhIHtoBITMAPINFOHEADER` function if the image is used in a context where a `BITMAPINFOHEADER` is requested.

Required Header

PhInt.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.13.3. PhInterpolateColor

```
void PhInterpolateColor(PBITMAPINFOHEADER pBMIH,
                        PBYTE pPixel,
                        UINT CFA,
                        UINT Algorithm
);
```

Parameters

pBMIH [in, out]
Input: image header describing input data.
Output: image header describing output data.

pPixel [in, out]
Input: Raw image data.
Output: Color interpolated image data.

CFA
Color filter array for the sensor on the camera this image was acquired with. Possible values are:

Value	Meaning
CFA_BAYER	GB RG
CFA_BAYERFLIP	RG GB
CFA_VRI	GBRG RGGB
CFA_VRIV6	BGGR GRBG

To obtain a camera CFA type call PhGetVersion with the GV_CFA selector. To obtain the CFA for a cine identified based on a CINEHANDLE object, call PhGetCineInfo with the GCI_CFA selector.

Algorithm
Numerical value identifying the algorithm to be used for demosaicing the input image. Different speed/quality combinations are available (see PhInt.h):
FAST_ALGORITHM
BEST_ALGORITHM
NO_DEMOSAICING

Return Value

None

Remarks

Builds a color RGB image from an uninterpolated input image containing raw sensor pixels. Both 8 bits per sample and 16 bits per sample input images can be processed by this function. After demosaicing, the *pBMIH* is updated correspondingly.
Since the color interpolation is done in place, the input *pPixel* buffer must be three times larger than the actual input raw data.
When interpolating an image from a multi-head Phantom v6 camera, this function is able to recognize the gray heads and to interpolate only the color quarters. This is done using the most significant bits of the CFA selected using the TL_GRAY, TR_GRAY, BL_GRAY, BR_GRAY

© 2023 Vision Research - AMETEK Materials Analysis Division

masks. The pixels from the gray heads will have their gray level copied three times in the result for the Blue, Green and Red components.

Required Header

PhInt.h

5.13.4. PhTriplicateGray

```
EXIMPROC void PhTriplicateGray(PBITMAPINFOHEADER pBMIH,
                               PBYTE pPixel,
);
```

Parameters

pBMIH
DIB header.

pPixel
Pixel array.

Return Value

None

Remarks

Build a color RGB image from a monochrome bitmap by copying the same value in the three color component. The input image can be 8 or 16 bit per pixel, the result will be 24 or 48 bpp. The `biBitCount` and `biSizeImage` fields of the header are updated.
The `pPixel` array must be allocated for the resulting color image that is three times larger than the original array. This function is needed in LabVIEW.

Required header

PhInt.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.13.5. PhWBAdjustUnint

```
void PhWBAdjustUnint(PBITMAPINFOHEADER pBMIH,
                     PBYTE pPixel,
                     UINT CFA,
                     PWBGAIN pWBg
);
```

Parameters

pBMIH [in]
Image header describing input data.

pPixel [in, out]
Input: Raw image data.
Output: White balance corrected raw image data.

CFA
Color filter array for the sensor on the camera this image was acquired with. Possible values are:

Value	Meaning
CFA_BAYER	GB RG
CFA_BAYERFLIP	RG GB
CFA_VRI	GBRG RGGB
CFA_VRIV6	BGGR GRBG

To obtain a camera CFA type call PhGetVersion with the GV_CFA selector. To obtain the CFA for a cine identified based on a CINEHANDLE object, call PhGetCineInfo with the GCI_CFA selector.

pWBg
Gain corrections on red and blue channels used to adjust the white balance.

Return Value

None

Remarks

PhWBAdjustUnint applies white balance gain corrections for a raw image. For better results, white balance correction must be run before color interpolation.

Required Header

PhInt.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.13.6. PhComputeWB

```
HRESULT PhComputeWB(PBITMAPINFOHEADER pBMIH,
                    PBYTE pPixel,
                    PPOINT pP,
                    int SquareSide,
                    UINT CFA,
                    PWBGAIN pWB,
                    PUINT pSatCnt
);
```

Parameters

pBMIH [in]
Image header.

pPixel [in]
Image pixels.

pP [in]
The point around which white balance is to be computed.

SquareSide [in]
Side length of the square area used to compute white balance.

CFA [in]
Color filter array for the sensor on the camera this image was acquired with.

pWB [out, optional]
Address where to store corrective gains for red and blue.
This parameter can be `NULL` if not needed.

pSatCnt [out, optional]
Address where to store saturated pixel count.
This parameter can be `NULL` if not needed.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

It computes the compensating `WBGAIN` based on the pixels color in the specified square. For the same square, it can also return saturated pixel count.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.13.7. PhImHisto

```
void PhImHisto(PBITMAPINFOHEADER pBMIH,
               PBYTE pPixel,
               UINT HistoSize,
               UINT Color,
               PUINT pHisto,
               PUINT pAvgR,
               PUINT pAvgG,
               PUINT pAvgB
);
```

Parameters

pBMIH [in]
Image header.

pPixel [in]
Input image data.

HistoSize [in]
Number of elements for the input histogram array.
Can be less than the number of possible levels for the selected color; in this case `[0, MaxPix]` is mapped to `[0, HistoSize – 1]`.

Color [in]
Compute the histogram on this selected color:

Value	Meaning
0	RGB converted to gray.
1	Red.
2	Green.
3	Blue.

pHisto [out]
Array of integers counting the occurrence of each level or set of levels (if *HistoSize* < maximum value of the pixel).

pAvgR [out, optional]
Average Red channel value for the entire image.
This parameter can be `NULL` if not needed.

pAvgG [out, optional]
Average Green channel value for the entire image.
This parameter can be `NULL` if not needed.

pAvgB
Average Blue channel value for the entire image.
This parameter can be `NULL` if not needed.

Return Value

none

Remarks

It computes the histogram for the input image.

Required Header

PhInt.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.14. LabView Specific Functions

5.14.1. PhLVRegisterClientEx

```
HRESULT PhLVRegisterClientEx(char *pStgPath,
                             PROGRESSCALLBACK pfnCallback,
                             UINT PhConHeaderVersion
);
```

Parameters

pStgPath [in]
The path for the stg files of cameras.
It may be your exe path or your StartIn path.

pfnCallback [in, optional]
Pointer to a caller function that will be called periodically to indicate the progress during the search for the connected cameras.
This pointer can be `NULL` if a progress indicator is not needed or it can be an integer from `1` to `15` if `PhLVProgress` use instead of a callback is preferred.

PhConHeaderVersion [in]
PhCon library version. Pass the value of `PHCONHEADERVERSION` symbol defined in PhCon.h.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

This function registers a LabView application as a client of the Phantom dlls. Call PhLVRegisterClientEx before any other function (except for `PhSetDllsOption`) whether working with Phantom camera or with cine files.

Required Header

PhCon.h

5.14.2. PhLVUnregisterClient

```
HRESULT PhLVUnregisterClient();
```

Parameters

None.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

The function unregisters a LabView application previously registered as a client of the Phantom dlls. Call this function to ensure proper resource deallocation when the Phantom dlls are not needed anymore.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.14.3. PhLVProgress

```
HRESULT PhLVProgress(UINT CallID,
                     PUINT pPercent,
                     BOOL Continue
);
```

Parameters

CallID [in]
The call identifier (a number from 1 to 15), which was passed as a *PROGRESSCALLBACK* parameter in a time consuming function.

pPercent [out]
The percentage of work done for the time consuming routine call, identified by *CallID.*

Continue [in]
Specifies if the function must continue.

Value	Meaning
TRUE	Continue
FALSE	Stop the function call, and force it to return

Return Value

The function returns ERR_Ok in case of success or an error code, otherwise. To get the text of the error message call PhGetErrorMessage.

Remarks

This function is used as a progress indicator for time-consuming functions. It is designed to work with environments which do not support function pointers (like LabView).
The time-consuming function call and the PhLVProgress call must be in different threads.
In LabView the time-consuming function call node must be set to Reentrant.
All API calls that have a PROGRESSCALLBACK function pointer parameter accept a *CallID* in it.
CallID is a UINT and it must be in the range [1..15].
For example, the functions can be imagined as having the following headers:

© 2023 Vision Research - AMETEK Materials Analysis Division

```
// Phcon.dll
HRESULT PhBlackReference(UINT CN, UINT CallID);
HRESULT PhNVMErase(UINT CN,  UINT CallID);
HRESULT PhNVMRestore(UINT CN, UINT CineNo,  UINT CallID);
HRESULT PhNVMSave(UINT CN, INT Cine, UINT CallID);
HRESULT PhNVMSaveClip(UINT CN, int Cine, PIMRANGE pRng, UINT Options,
UINT CallID);

// Phfile.dll
HRESULT PhWriteCineFile(CINEHANDLE hC, UINT CallID);
```

After calling one of these functions, you can call `PhLVProgress` from another thread to see the amount of work done or to stop the operation.

Example:

```
// Thread 1
PhWriteCineFile(hC, 1); //time consuming function
// Thread 2
UINT Percent;
//...
while(true)
   {
   //see the progress of PhWriteCineFile
   PhLVProgress(1, &Percent, TRUE);
   Sleep(1000);
   }
```

Important:
To make this function work, the application must register itself by calling `PhLVRegisterClientEx`.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.15. Nucleus Specific Functions

5.15.1. PhUpgradeFirmware

```
HRESULT PhUpgradeFirmware(UINT IpAddress,
                          UINT CamVer,
                          UINT Selector,
                          char *pFilename);
```

Parameters

IpAddress [in]
Camera Ip Address

CamVer [In]
Camera hardware version

Selector [in]
Selector value indicating the firmware to be upgraded.

pFilename [in]
Pointer to complete file name and path. The file must be a .Phfw.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to upgrade a camera's firmware, FPGA, (Field Programmable Gate Array), Flash FPGA, Cinemag, and Kernel firmware. For a complete list of currently available selectors, go to **Error! Reference source not found.Error! Reference source not found.**.

Note:

1) The external third-party applications "plink.exe" and "pscp.exe" are required for this function. They should be in the same folder as the application and PhCon.Dll.

2) After a camera upgrade, an application program restart should be performed to resynchronize any application cache settings with the updated camera.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

5.15.2. PhiLoad

```
HRESULT PhiLoad(INT CN);
```

Parameters

CN [in]
Camera number.

Return Value

The function returns `ERR_Ok` in case of success or an error code, otherwise. To get the text of the error message call `PhGetErrorMessage`.

Remarks

Use this function to perform a iLoad (factory reset) operation on the camera**Error! Reference source not found.**.

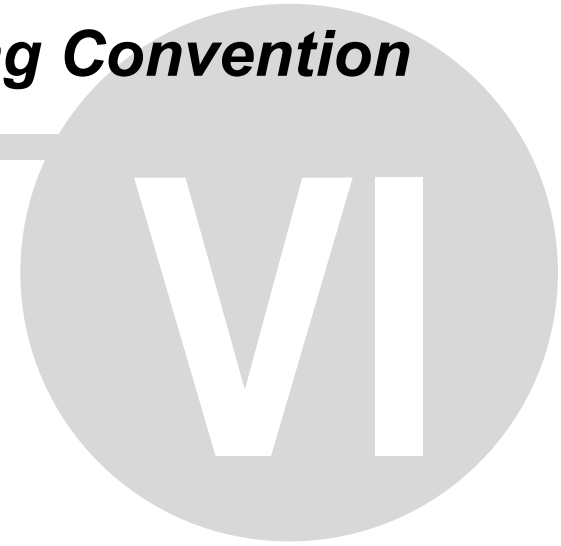
Note:
1) A iload operation can take some time to complete (up to a minute), and this time is different for each camera model.

Required Header

PhCon.h

© 2023 Vision Research - AMETEK Materials Analysis Division

Part VI File Naming Convention



© 2023 Vision Research - AMETEK Materials Analysis Division

6. File Naming Convention

The "language" for specifying file names

When naming files saved from the Phantom software, you can use a few special seldom used characters that are legal for filenames. Phantom software recognizes these wildcard characters and substitutes them with different characters in order to better support batch processing.

@ **followed by a single digit (1 to 8)**
Used to insert the cine file number to the path and/or file names. The digit indicates the number of digit counter places the numbering scheme should use. The @ character can be used within both path names and file names. The cine number counter is initialized to zero each time the software is started and incremented by 1 each time a cine is saved to a drive. Use it when running Continuous Recording.

\+ **followed by a single digit (1 to 8)**
Used to insert an image number in the image file name, starting from 1. The digit number specifies how many digit counter places will be used for the image number.

! **followed by a single digit (1 to 8)**
Used to insert an image number in the file name, starting from the first frame's number. The digit number specifies how many digit counter places will be used for the image number. Remember to consider the minus sign in the image number, too.

$ **followed by a single digit (1 to 8)**
Used to denote a session number, if desired, in the path and/or file names. The digit number indicates how many digit counter places the numbering scheme should use.

~ **followed by a single digit (8 only)**
Used to insert the current date and time within the file names only. The digit number "8" specifies the number of digit places used to report date in the file.

In Continuous Recording character **~** can also be used, like the following:

- a **~** character followed by anything else but 8 will add the cine full trigger time string, including day of week, year, month, ... second, fractions.
- the **~**8 characters after the root file name will add 8 characters that represent part of the cine trigger time string: first two characters from the month name, the day, the hour and the minute.

6.1. Converting Cines into Multiple Images

You can convert one or several cines into multiple images using this "language", according to your needs. The destination image files can be stored in different ways.

6.1.1. Converting Single Cine into Multiple Images

- When you are converting only one cine, the image files can be all saved in the same folder where the source cine is placed. In this case, you have to mention just the desired name of the file, followed by a **+** or a **!** and a number. The file names will count from 1 or, respectively, from the first frame's number.

- If you want the image files to be saved in a new folder or a series of new folders, you have to mention the name of the new folders followed by a \ and the name of the image files (described

© 2023 Vision Research - AMETEK Materials Analysis Division

above) and the Phantom software will create them as subfolders within the one in which the source cine is placed.

- The name of the image files described above can also be preceded by a full path name, containing one or several new folders. Phantom software will create the full path before saving the image files.

6.1.2. Converting Multiple Cines into Multiple Images

The same rules apply in converting several cines, too, but in this case the Phantom software creates new folders for each cine, named like the source cines (except the file extension), each one containing its own destination image files. These new folders are subfolders in the one in which the source cines are placed or they are placed in new folders according to the complete name given by you.

The names of your new folders can also contain a counter of cines – by using the **@** character – or a counter of sessions – by using the **$** character – or the date-and-time string – by using the **~** character.

6.2. Examples

1. When converting a cine into multiple images – let's say `cine1.cin` – containing a range of frames from -10 to 2, you can specify the generic file name like this:

   ```
   imag!4
   ```

The images will be saved in the same folder with the one in which `cine1.cin` is, in files named:

```
imag-010
imag-009
imag-008
…
imag0000
imag0001
imag0002
```

2. Besides the file names, you can indicate a new directory or the full path name, using the wildcards described above. For example, when converting several cines into multiple images – `cine1.cin, avi1.avi` – containing a range of frames from –10 to 2, you can specify the following Files in the Multiple Convert Destination dialog box:

   ```
   c:\cines\imag!4
   ```

   The Phantom software will create `cines` folder if it doesn't exist, it will insert new folders named the same like the source files (without extension) – `cine1` and `avi1` in this example – and the images will be saved like this:

   ```
   c:\cines\cine1\imag-010
   c:\cines\cine1\imag-009
   …
   c:\cines\cine1\imag0001
   c:\cines\cine1\imag0002

   c:\cines\avi1\imag-010
   c:\cines\avi1\imag-009
   …
   c:\cines\avi1\imag0001
   c:\cines\avi1\imag0002
   ```

3. You can also introduce a counter of cines, using the @ sign. For the same conversion as the one mentioned above, if you specify:

© 2023 Vision Research - AMETEK Materials Analysis Division

```
c:\cine@2\imag!4
```

you will get the following:

```
c:\cine00\cine1\imag-010
c:\cine00\cine1\imag-009
…
c:\cine00\cine1\imag0001
c:\cine00\cine1\imag0002

c:\cine01\avi1\imag-010
c:\cine01\avi1\imag-009
…
c:\cine01\avi1\imag0001
c:\cine01\avi1\imag0002
```

4. If you prefer a counting of images starting from 1, here is an example for converting the file `cine1.cin` with the frame range from –10 to 2:

```
c:\cines\imag+4
```

and you will get:

```
c:\cines\imag0001
c:\cines\imag0002
…
c:\cines\imag00012
c:\cines\imag00013
```

5. Continuous recording file naming can be simply defined like the following:

```
cont@4
```

and the cines will be saved into:

```
cont0001
cont0002
cont0003
…
```

If you wish to add the trigger time string to the file name, you can define it:

```
cont ~
```

and you will get:

```
cont Mon Dec 06 2010 16 51 19.274 658
cont Mon Dec 06 2010 16 51 26.009 567
…
```

Phantom software adds WeekDay Month Day Year Hour Minute Second Fractions of second to the file name root.

If you define the name like this:

```
cont ~8
```

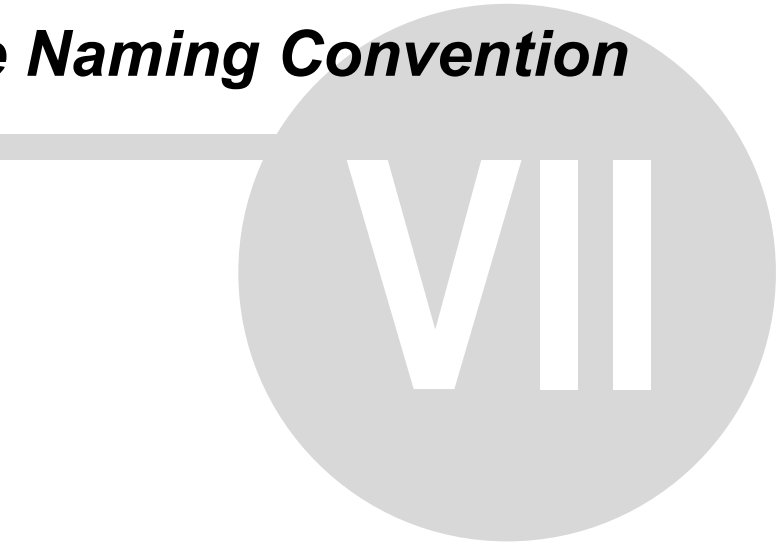
the files will be named:

```
cont DE061710
cont DE061711
…
```

containing the first two characters of the month name, the day, the hour and the minute.

© 2023 Vision Research - AMETEK Materials Analysis Division

Part VII File Naming Convention



© 2023 Vision Research - AMETEK Materials Analysis Division

7. Supported File Formats

Supported cine formats:

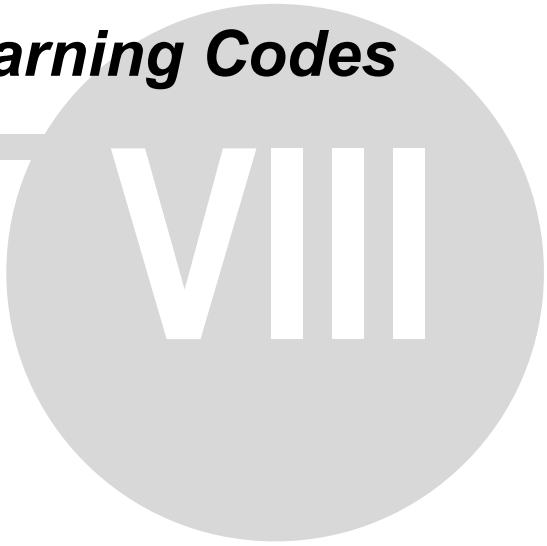
FileType	Code	Value	Extension	Write	Read
Cine Raw	MIFILE_RAWCINE	0	.cine	✓	✓
Cine	MIFILE_CINE	1	.cine	✓	✓
JPEG Cine	MIFILE_JPEGCINE	2	.cci	✓	×
AVI	MIFILE_AVI	3	.avi	✓	✓
Multipage TIFF	MIFILE_TIFCINE	4	.tif	✓	×
MPEG	MIFILE_MPEG	5	.mpg	✓	×
QTIME (uncompressed)	MIFILE_QTIME	8	.mov	✓	×
MP4	MIFILE_MP4	9	.mp4	✓	×

Supported image formats:

FileType	Code	Value	Extension	Write	Read
TIFF 8/24 bpp	SIFILE_TIF8	-8	.tif	✓	✓
TIFF 12/36 bpp	SIFILE_TIF12	-9	.tif	✓	✓
TIFF 16/48 bpp	SIFILE_TIF16	-10	.tif	✓	✓
RAW	SIFILE_RAW	-25	.raw	✓	×
DNG	SIFILE_DNG	-26	.dng	✓	×
DPX	SIFILE_DPX	-27	.dpx	✓	×

© 2023 Vision Research - AMETEK Materials Analysis Division

Part VIII Error and Warning Codes



© 2023 Vision Research - AMETEK Materials Analysis Division

8. Error and Warning Codes

8.1. Error Codes

Value	Label	Meaning
-200	ERR_NULLPointer	A NULL pointer was provided for an input parameter or for an output parameter that must be non-NULL.
-201	ERR_MemoryAllocation	Memory allocation failed.
-202	ERR_NoWindow	The hWnd parameter is not a correct window handle.
-203	ERR_CannotRegisterClient	Cannot register client.
-204	ERR_CannotUnregisterClient	Cannot unregister client.
-205	ERR_AsyncRead	1394 communication error.
-206	ERR_AsyncWrite	1394 communication error.
-207	ERR_IsochCIPHeader	1394 isochronous transfer error (Common Isochronous Packet header error).
-208	ERR_IsochDBCContinuity	1394 isochronous transfer error (Data Block Continuity error).
-209	ERR_IsochNoHeader	1394 isochronous transfer error (Sony Vaio).
-210	ERR_IsochAllocateResources	1394 isochronous transfer error: resource allocation.
-211	ERR_IsochAttachBuffers	1394 isochronous transfer error: attach buffers.
-212	ERR_IsochFreeResources	1394 isochronous transfer error: free resources.
-213	ERR_IsochGetResult	1394 isochronous transfer error: get result.
-214	ERR_CannotReadTheSerialNumber	The serial number of the camera cannot be read. The serial number is an important element in the management of the 1394 bus so this is a serious error.
-215	ERR_SerialNumberOutOfRange	The serial read from the camera is incorrect and should be restored.
-216	ERR_UnknownCameraVersion	The connected hardware is not supported by this version of PhCon.
-217	ERR_GetImageTimeOut	The image from camera was not available in the allowed time.
-218	ERR_ImageNoOutOfRange	A recorded image has the maximum number PTFrames-1 and the minimum number PTFrames-ImCount.
-220	ERR_CannotReadStgHeader	Stg file read error or incorrect content.
-221	ERR_ReadStg	Stg file read error or incorrect content.
-222	ERR_StgContents	Stg file read error or incorrect content.
-223	ERR_ReadStgOffsets	Stg file read error or incorrect content.
-224	ERR_ReadStgGains	Stg file read error or incorrect content.
-225	ERR_NotAStgFile	Stg file read error or incorrect content.
-226	ERR_StgSetupCheckSum	Stg file read error or incorrect content.
-227	ERR_StgSetup	Stg file read error or incorrect content.
-228	ERR_StgHardAdjCheckSum	Stg file read error or incorrect content.
-229	ERR_StgHardAdj	Stg file read error or incorrect content.
-230	ERR_StgDifferentSerials	The Stg file name is different from the serial stored inside.

© 2023 Vision Research - AMETEK Materials Analysis Division

Value	Label	Meaning
-231	ERR_WriteStg	Write to Stg file error.
-232	ERR_NoCine	A recorded image, block or a time buffer was requested but this camera didn't record a cine.
-233	ERR_CannotOpenDevice	1394 isochronous transfer error: cannot open the device.
-234	ERR_TimeBufferSize	The time buffer reported by the camera has an unexpected size.
-236	ERR_CannotWriteCineParams	Cannot write Cine Params to the single cine camera (v4-v6).
-250	ERR_NVMError	Nonvolatile memory error.
-251	ERR_NoNVM	Nonvolatile memory not installed.
-253	ERR_FlashEraseTimeout	Flash erase timeout.
-254	ERR_FlashWriteTimeout	Flash write timeout.
-255	ERR_FlashContents	Flash content is bad.
-256	ERR_FlashOffsetsCheckSum	Flash offsets checksum bad.
-257	ERR_FlashGainsCheckSum	Flash gains checksum bad.
-258	ERR_TooManyCameras	Too many cameras connected.
-259	ERR_NoResponseFromCamera	No response from Ethernet camera.
-260	ERR_MessageFromCamera	Camera produced an error message.
-261	ERR_BadImgResponse	Bad response to the image request.
-262	ERR_AllPixelsBad	Calibration: all sensor pixels seem to be bad.
-263	ERR_BadTimeResponse	Bad response to the time request.
-264	ERR_GetTimeTimeOut	Time out while getting the image time buffer.
-270	ERR_InBlockTooBig	BASE64 transfer: input block too big.
-271	ERR_OutBufferTooSmall	BASE64 transfer: buffer too small.
-272	ERR_BlockNotValid	BASE64 transfer: Block not valid.
-273	ERR_DataAfterPadding	BASE64 transfer: Data after padding.
-274	ERR_InvalidSlash	BASE64 transfer: Invalid slash.
-275	ERR_UnknownChar	BASE64 transfer: Unknown character.
-276	ERR_MalformedLine	BASE64 transfer: Malformed line.
-277	ERR_EndMarkerNotFound	BASE64 transfer: End marker not found.
-280	ERR_NoTimeData	No time recorded.
-281	ERR_NoExposureData	No exposure recorded.
-282	ERR_NoRangeData	No range data recorded.
-283	ERR_NotIncreasingTime	The image time does not increase from one image to the next.
-284	ERR_BadTriggerTime	The trigger time is bad.
-285	ERR_TimeOut	Timeout at the communication to the camera.
-286	ERR_NullWeightsSum	The sum of all weights for partitions size cannot be 0.
-287	ERR_BadCount	Count of partitions cannot be 0 or > maximum partition count.
-288	ERR_CannotChangeRecordedCine	Cannot change the parameters of a recorded cine.
-289	ERR_BadSliceCount	Count of visible i3 slices is different from the stored value.

© 2023 Vision Research - AMETEK Materials Analysis Division

Value	Label	Meaning
-290	ERR_NotAvailable	The device you requested information about is not present in the system.
-291	ERR_BadImageInterval	The interval between images is not constant and equal to 1/FrameRate.
-292	ERR_BadCameraNumber	The camera number cannot be above the camera count.
-293	ERR_BadCineNumber	The cine number parameter is incorrect.
-294	ERR_BadSyncObject	A mutex, semaphore or other sync object cannot be created.
-295	ERR_IcmpEchoError	Icmp (ping) echo error.
-296	ERR_MlairReadFirstPacket	Error reading first packet from file.
-297	ERR_MlairReadPacket	Error reading packet from file.
-298	ERR_MlairIncorrectOrder	Incorrect ordered packets detected.
-299	ERR_MlairStartRecorder	Error starting MLAIR recorder process.
-300	ERR_MlairStopRecorder	Error stopping MLAIR recorder process.
-301	ERR_MlairOpenFile	Error opening input data file.
-302	ERR_CmdMutexTimeOut	Command mutex time out.
-303	ERR_CmdMutexAbandoned	Mutex abandoned.
-304	ERR_UnsupportedConversion	Unsupported data conversion.
-305	ERR_TenGigLostPacket	10 Giga lost packet.
-306	ERR_TooManyImgReq	Too many simultaneous image requests.
-307	ERR_BadImRange	Bad image range.
-308	ERR_ImgBufferTooSmall	Image buffer too small.
-309	ERR_ImgSize0	Image size is 0.
-310	ERR_IppError	Ipp library error.
-2000	ERR_BadCine	Cine handle has not been initialized.
-2002	ERR_UnsupportedFormat	File format unsupported.
-2003	ERR_InsufficientAlloc	The allocated size of the data is too small.
-2004	ERR_NotInRange	Image number is not in the cine image range.
-2005	ERR_SaveAvi	Error in save AVI process.
-2006	ERR_Encoder	The selected encoder cannot be added.
-2007	ERR_UnsupportedTiffFormat	Tiff file format unsupported.
-2009	ERR_SplitQuartersUnsupported	Split Quarters option not available on selected cine format (AVI and multipage Tiff).
-2010	ERR_NotACineFile	The file is not a cine.
-2012	ERR_FileOpen	Cannot open file.
-2013	ERR_FileRead	Cannot read file.
-2014	ERR_FileWrite	Cannot write file.
-2015	ERR_FileSeek	Cannot set file pointer.
-2016	ERR_DecompressImage	Cannot decompress image.
-2017	ERR_CineVerNewer	Cine file version is newer than the supported cine version.
-2018	ERR_NotSupported	File format not supported.

© 2023 Vision Research - AMETEK Materials Analysis Division

Value	Label	Meaning
-2021	ERR_CannotBuildGraph	Cannot build filter graph.
-2023	ERR_AuxDataNotFound	Auxiliary Data not found.
-2024	ERR_NotEnoughMemory	Not enough memory to perform the operation.
-2025	ERR_MpegReaderNotFound	MPEG Reader Interface not found.
-2028	ERR_FunctionNotFound	Function not found in a dynamic link library.
-2029	ERR_CineInUse	Cine is already in use for a similar operation.
-2030	ERR_SaveCineBufferFull	Too many write cine operation are already in progress.
-2031	ERR_NoCineSaveInProgress	No save/write operation is currently in progress for this cine.

© 2023 Vision Research - AMETEK Materials Analysis Division

8.2. Warning Codes

Value	Label	Meaning
0	ERR_Ok	No error, the function completed successfully.
100	ERR_SimulatedCamera	There are no cameras connected or the accessed camera number is above the available cameras. This is intended for testing or using the software without cameras connected.
101	ERR_UnknownErrorCode	The function PhGetErrorMessage cannot provide a message for the specified code.
102	ERR_BadResolution	The resolution specified was incorrect and was adjusted to the maximum camera resolution.
103	ERR_BadFrameRate	The frame rate specified was incorrect and was adjusted to the nearest acceptable value (minimum frame rate if it was less than minimum and maximum frame rate if it was greater than maximum).
104	ERR_BadPostTriggerFrames	Post-trigger frames cannot be 0. It is forced to 1.
105	ERR_BadExposure	The exposure duration specified was incorrect and was adjusted to the nearest acceptable value (minimum exposure if it was less than minimum, and maximum exposure if it was greater than maximum).
106	ERR_BadEDRExposure	EDRExposure cannot be larger than Exposure. It is forced to the value of Exposure.
107	ERR_BufferTooSmall	The buffer provided is too small to fit all the camera resolutions.
108	ERR_CannotSetTime	The camera is synchronized to the IRIG signal and the clock cannot be set manually.
109	ERR_SerialNotFound	PhFindCameraNumber cannot find a camera having the specified serial.
110	ERR_CannotOpenStgFile	The serial tag file (containing settings and calibration) does not exist at the path specified in PhRegisterClient function.
111	ERR_UserInterrupt	User interrupt.
112	ERR_NoSimulatedImageFile	No simulated image file.
113	ERR_SimulatedImageNot24bpp	Simulated image file has to have 24 bpp.
114	ERR_BadParam	Function called with improper argument.
115	ERR_FlashCalibrationNewer	The calibration from camera flash is newer than the stg file.
117	ERR_ConnectedHeadsChanged	The heads connected to this multihead camera changed.
118	ERR_NoHead	Selected head is not installed.
119	ERR_NVMNotInstalled	The Non Volatile Memory is not installed in this camera.
120	ERR_HeadNotAvailable	A cine was stored in a v6.2 camera flash and the recording head is not available at this moment. The images will not be corrected.
121	ERR_FunctionNotAvailable	This function is not available or not implemented in this camera or firmware version.
122	ERR_Ph1394dllNotFound	Phantom Firewire driver was not installed - the Firewire cameras will not be visible.
123	ERR_oldtNotFound	Dll's for the signal acquisition (oldaapi32.dll, olmem32.dll) are not available. Signal acquisition board will not be visible.
124	ERR_BadFRPSteps	The FRPSteps field value is too large.
125	ERR_BadFRPImgNr	FRPImgNr value is not correct. It has to be between 0 and PTFrames-1.

© 2023 Vision Research - AMETEK Materials Analysis Division

Value	Label	Meaning
126	ERR_BadAutoExpLevel	The autoexposure level is not correct. It has to be in the interval [0, 255], regardless of the recording bit depth.
127	ERR_BadAutoExpRect	The autoexposure rectangle is not correct.
128	ERR_BadDecimation	The Decimation factor is not in the interval [1, 16].
129	ERR_BadCineParams	The cine parameters read from camera have incorrect values.
130	ERR_IcmpNotAvailable	Icmp library (or function from this dll) not available.
131	ERR_CorrectResetLine	Special correction for the reset line was requested.
132	ERR_CSRDoneInCamera	Current session Reference was done in camera.
133	ERR_ParamsChanged	Camera parameters were changed from another connection.
134	ERR_ParamReadOnly	Parameter is read-only.
135	ERR_ParamWriteOnly	Parameter is write-only.
136	ERR_ParamNotSupported	Parameter not supported.
137	ERR_IppWarning	Ipp library warning.
140	ERR_BadSensorMode	Invalid sensor mode
141	ERR_CameraOffline	Camera is offline
142	ERR_InvalidFile	Invalid File
2000	ERR_8BitDestinationCine	The destination cine format does not support 16 bit images.The result will be on 8 bit.
2001	ERR_VoidRange	Your selected range does not intersect with the cine image range. The save range contains 0 images.
2002	ERR_ReducedRange	Your selected range intersects partially with the cine image range. The save range is reduced.
2003	ERR_Seeking	This file does not support frame seeking.
2004	ERR_UserAbort	You interrupted this operation.
2005	ERR_FreeLibrary	Cannot free a dynamic loaded library.
2006	ERR_LeadToolsNotFound	LeadTools dlls not found. CCI cine format and some image formats are not available.
2007	ERR_EVSNotFound	EVS library for the MXF file format not found. MXF format will not be available.
2008	ERR_BadResampleParam	Bad resample parameter.
2009	ERR_BadCropParam	Bad crop parameter.

© 2023 Vision Research - AMETEK Materials Analysis Division

Part IX Appendices

© 2023 Vision Research - AMETEK Materials Analysis Division

9. Appendices

9.1. Appendix 1 – PhGetCineInfo/PhSetCineInfo Selectors

Name	Meaning	Type	Get	Set
General Cine Parameters				
GCI_ISFILECINE	TRUE if this is a file cine.	BOOL	✓	×
GCI_FROMFILETYPE	File type (see Supported File Formats).	INT	✓	×
GCI_COMPRESSION	Compression type. Possible values are: 0 – Gray cine 1 – JPEG compressed cine (*.cci) 2 – Uninterpolated cine (RAW)	UINT	✓	×
GCI_IMAGECOUNT	Saved image count.	UINT	✓	×
GCI_TOTALIMAGECOUNT	Total recorded image count.	UINT	✓	×
GCI_FIRSTIMAGENO	First image number for the selected/saved range.	INT	✓	×
GCI_FIRSTMOVIEIMAGE	First recorded image number.	INT	✓	×
GCI_MAXIMGSIZE	Maximum size in bytes required for reading and processing one image of this cine.	UINT	✓	×
GCI_CINEDESCRIPTION	Cine description	char[4096]	✓	×
Acquisition Parameters				
GCI_CAMERAVERSION	An integer describing camera version.	UINT	✓	×
GCI_CAMERASERIAL	The serial of the camera that acquired the cine.	UINT	✓	×
GCI_DFRAMERATE	Cine frame rate.	Double	✓	×
GCI_EXPOSURENS	Image exposure in nanoseconds.	UINT	✓	×
GCI_CFA	CFA of the camera this cine was recorded on.	UINT	✓	×
GCI_UVSENSOR	Sensor supports UV light	BOOL	✓	
GCI_SENSORMODE	Sensor configuration	UINT	✓	×
GCI_IMWIDTH	Current image width. Can be different from image width at acquisition time if a crop or border operation has been applied for the images of this cine.	UINT	✓	×

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_IMHEIGHT	Current image height. Can be different from image height at acquisition time if a crop or border operation has been applied for the images of this cine.	UINT	✓	×
GCI_IMWIDTHACQ	Image width at acquisition time.	UINT	✓	×
GCI_IMHEIGHTACQ	Image height at acquisition time.	UINT	✓	×
GCI_LENSDESCRIPTION	Null terminated string containing various lens information such as producer, model or focal range.	char[MAXSTDSTRSZ]	✓	×
GCI_LENSAPERTURE	Aperture f number.	FLOAT	✓	×
GCI_LENSFOCALLENGTH	Lens focal length (zoom factor).	FLOAT	✓	×
GCI_RECORDINGTIMEZONE	The time zone active during the recording of the cine.	INT	✓	×
GCI_TRIGTC	Trigger frame SMPTE time code and user bits represented as a TC value.	TC	✓	✓
GCI_TRIGTCU	Trigger frame SMPTE time code and user bits represented as a TCU value.	TCU	✓	✓
GCI_PBRATE	Video playback rate (fps) active when the cine was captured.	FLOAT	✓	✓
GCI_TCRATE	Playback rate (fps) used for generating SMPTE time code.	FLOAT	✓	✓
GCI_TRIGTIMESEC	The seconds of the trigger time.	UINT	✓	×
GCI_TRIGTIMEFR	The fractions of the trigger time.	UINT	✓	×
GCI_POSTTRIGGER	Post-trigger frames.	UINT	✓	×
GCI_TRIGFRAME	Information about sync image mode: 0 – internal 1 – external 2 – locktoirig.	UINT	✓	×
GCI_FRAMEDELAYNS	Frame delay in nanoseconds.	UINT	✓	×
GCI_EDREXPOSURENS	EDR exposure in nanoseconds.	UINT	✓	×
GCI_AUTOEXPOSURE	Cine auto-exposure.	UINT	✓	×
GCI_AUTOEXPLEVEL	Level for autoexposure control.	UINT	✓	×
GCI_AUTOEXPTOP	Top coordinate of the autoexposure rectangle.	UINT	✓	×
GCI_AUTOEXPLEFT	Left coordinate of the autoexposure rectangle.	UINT	✓	×
GCI_AUTOEXPBOTTOM	Bottom coordinate of the autoexposure rectangle.	UINT	✓	×
GCI_AUTOEXPRIGHT	Right coordinate of the autoexposure rectangle.	UINT	✓	×
GCI_FRPSTEPS	Number of frame rate profile steps.	UINT	✓	×
GCI_FRPIMGNOARRAY	Array containing image numbers where to change the acquisition frame rate.	INT[16]	✓	×

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_FRPRATEARRAY	Array containing new values for the acquisition frame rate.	UINT[16]	✓	✗
GCI_FRPSHAPEARRAY	Array containing frame rate evolution shape for each frame rate profile step. Possible values are: 0 – flat 1 – ramp	UINT[16]	✓	✗
GCI_RECBPP	Recording bit depth for the cine	UINT	✓	
Image Format				
GCI_REALBPP	Pixel color depth for the cine.	UINT	✓	✗
GCI_IS16BPPCINE	TRUE if this is a 16 bpp cine.	BOOL	✓	✗
GCI_ISCOLORCINE	TRUE if this is a color cine.	BOOL	✓	✗
Image Processing Parameters				
GCI_WB	White balance gains for raw cines.	WBGAIN	✓	✓
GCI_WBVIEW	White balance gains applied on color interpolated cines.	WBGAIN	✓	✓
GCI_WBISMETA	If TRUE, white balance gains are stored as metadata.	BOOL	✓	✗
GCI_BRIGHT	Image offset. Range of possible values is [-1.0, 1.0]. 0.0 is neutral. 1.0 means adding the maximum value for the current bit depth.	FLOAT	✓	✓
GCI_CONTRAST	Gain image processing. Range of possible values is [0.1, 10.0]. 1.0 is neutral.	FLOAT	✓	✓
GCI_GAMMA	Per-component gamma Range of possible values is [0.1, 10.0]. 1.0 is neutral.	FLOAT	✓	✓
GCI_GAMMAR	Difference between red channel gamma and overall gamma 0.0 is neutral.	FLOAT	✓	✓
GCI_GAMMAB	Difference between blue channel gamma and overall gamma. 0.0 is neutral.	FLOAT	✓	✓
GCI_SATURATION	Saturation Range of possible values is [0.0, 2.0]. 1.0 is neutral.	FLOAT	✓	✓
GCI_HUE	Degrees and fractions of degree to rotate the color hue of every image pixel. Range of possible values is [-180.0, 180.0]. 0.0 is neutral.	FLOAT	✓	✓
GCI_FLIPH	Flip the image horizontally.	BOOL	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_FLIPV	Flip the image vertically.	BOOL	✓	✓
GCI_ROTATE	Rotate information: 0 No rotation +90 Counterclockwise -90 Clockwise	INT	✓	✓
GCI_FILTERCODE	Image filter code. See **Error! Reference source not found.** for possible values.	INT	✓	✓
GCI_IMFILTER	User defined convolution filter.	IMFILTER	✓	✓
GCI_INTALGO	Demosaicing algorithm selector. Phantom SDK offers the following options (see PhInt.h): FAST_ALGORITHM BEST_ALGORITHM NO_DEMOSAICING	UINT	✓	✓
GCI_DEMOSAICINGFUNCPTR	Pointer to current demosaicing responsible function.	DEMOSAICINGFUNCPTR	✓	✓
GCI_TONE	Tone description.	TONEDESC	✓	✓
GCI_RESAMPLEACTIVE	If TRUE, resampling is active.	BOOL	✓	✓
GCI_RESAMPLEWIDTH	Desired resample width.	UINT	✓	✓
GCI_RESAMPLEHEIGHT	Desired resample height.	UINT	✓	✓
GCI_CROPACTIVE	If TRUE, crop is active.	BOOL	✓	✓
GCI_CROPRECTANGLE	Rectangle to crop from the input image.	RECT	✓	✓
GCI_GAIN16_8	Sensitivity value used when converting cine images to 8 bits per sample: pixelOut = pixelIn * fGain16_8 * (2^8 / $2^{bitdepth}$)	FLOAT	✓	✓
GCI_PEDESTALR	Red channel offset to be applied after gamma. Range of possible values is [-1.0, 1.0]. 0.0 is neutral. 1.0 means adjusting by the maximum value for the current bit depth.	FLOAT	✓	✓
GCI_PEDESTALG	Green channel offset to be applied after gamma. Range of possible values is [-1.0, 1.0]. 0.0 is neutral. 1.0 means adjusting by the maximum value for the current bit depth.	FLOAT	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_PEDESTALB	Blue channel offset to be applied after gamma. Range of possible values is [-1.0, 1.0]. 0.0 is neutral. 1.0 means adjusting by the maximum value for the current bit depth.	FLOAT	✓	✓
GCI_ENABLEMATRICES	TRUE if the user matrix is currently active.	BOOL	✓	✓
GCI_USERMATRIX	User color matrix description.	CMDESC	✓	✓
GCI_FLARE	Offset to be applied before white balance. Range of possible values is [-1.0, 1.0]. 0.0 is neutral. 1.0 means adjusting by the maximum value for the current bit depth.	FLOAT	✓	✓
GCI_CHROMA	Chrominance adjustment applied after gamma. Range of possible values is [0.0, 2.0]. 1.0 is neutral.	FLOAT	✓	✓
Cine Use Case Parameters				
GCI_VFLIPVIEWACTIVE	For UC_VIEW use cases, cine images are vertically flipped if this parameter is TRUE.	BOOL	✓	✓
GCI_UNCALIBRATEDIMAGE	If TRUE, no calibration correction is applied for this cine images. Camera calibration or CSR results will thus be ignored. Bad pixel correction will also be disabled.	BOOL	✓	✓
GCI_NOPROCESSING	If TRUE, no image processing is applied for this cine images except for calibration corrections and bad pixel repairing.	BOOL	✓	✓
GCI_BADPIXELREPAIR	If TRUE, bad pixel correction is enabled.	BOOL	✓	✓
Cine Save Options				
GCI_SAVERANGE	Image range to be saved.	IMRANGE	✓	✓
GCI_SAVEFILENAME	Destination file name.	char[MAX_PATH]	✓	✓
GCI_SAVEFILETYPE	Destination file type. See Supported File Formats for possible values.	INT	✓	✓
GCI_SAVE16BIT	If TRUE, save on 16 bits when more than 8 bits are available.	BOOL	✓	✓
GCI_SAVEPACKED	If >0 and if source cine is packed, destination format is also packed. 0 – Not Packed 1 – Packed 10 2 – Packed 12L	UINT	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_SAVEXML	If TRUE, an XML file will be created containing destination cine header.	BOOL	✓	✓
GCI_SAVESTAMPTIME	Image time stamp options. Possible values are: 0 – no time stamp 1 – stamp the absolute time 3 – stamp the time from trigger	INT	✓	✓
GCI_SAVEDECIMATION	Decimation value for image save. Value must be an integer value	FLOAT	✓	✓
GCI_SAVEAVIFRAMERATE	Preferred avi playback frame rate	UINT	✓	✓
GCI_SAVEAVICOMPRESSORLIST	String array containing available avi compressors.	char[MAXCOMPRCNT] [MAXSTDSTRSZ]	✓	✗
GCI_SAVEAVICOMPRESSORINDEX	Preferred avi compressor index.	UINT	✓	✓
GCI_SAVEDPXLSB	If TRUE, preferred DPX byte order is least significant byte first (little endian). Otherwise, big endian byte order is used.	BOOL	✓	✓
GCI_SAVEDPXTO10BPS	If TRUE, images are converted to 10 bits per sample.	BOOL	✓	✓
GCI_SAVEDPXDATAPACKING	Preferred DPX data packing. Possible values are: 0 – Filled to 32-bit method A 1 – Filled to 32-bit method B 2 – Packed into 32-bit words	UINT	✓	✓
GCI_SAVEDPX10BITLOG	If TRUE, images are converted to 10bit log format.	BOOL	✓	✓
GCI_SAVEDPXEXPORTLOGLUT	If TRUE, look-up tables for linear to log and log to linear are saved in separate files.		✓	
GCI_SAVEDPX10BITLOGREFWHITE	White reference for 10bit log format.	UINT	✓	✓
GCI_SAVEDPX10BITLOGREFBLACK	Black reference for 10bit log format.	UINT	✓	✓
GCI_SAVEDPX10BITLOGGAMMA	Gamma for 10bit log format.	FLOAT	✓	✓
GCI_SAVEDPX10BITLOGFILMGAMMA	Film gamma for 10bit log format.	FLOAT	✓	✓
GCI_SAVEQTPLAYBACKRATE	Preferred playback frame rate for MOV QuickTime files.	UINT	✓	✓
GCI_SAVECCIQUALITY	CCI preferred quality in percents. Range of possible values is [1..100].	UINT	✓	✓
Other Parameters				
GCI_CINENAME	Cine name.	char[MAXSTDSTRSZ]	✓	✓
GCI_WRITEERR	Last error encountered during a save operation from this cine.	INT	✓	✗

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
GCI_TIMEFORMAT	Preferred time format for this cine. Possible values are (see PhFile.h): TF_LT – Local Time TF_UT – Universal Time TF_SMPTE – SMPTE TimeCode	UINT		

© 2023 Vision Research - AMETEK Materials Analysis Division

9.2. Appendix 2 – Image Filtering

Predefined filter codes and kernels (see PhInt.h):

Value	Code	Kernel
1	PREWITT_3x3_V	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
2	PREWITT_3x3_H	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
3	SOBEL_3x3_V	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
4	SOBEL_3x3_H	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
5	LAPLACIAN_3x3	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
6	LAPLACIAN_5x5	$\begin{bmatrix} -1 & -3 & -4 & -3 & -1 \\ -3 & 0 & 6 & 0 & -3 \\ -4 & 6 & 20 & 6 & -4 \\ -3 & 0 & 6 & 0 & -3 \\ -1 & -3 & -4 & -3 & -1 \end{bmatrix}$
7	GAUSSIAN_3x3	$\frac{1}{16} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
8	GAUSSIAN_5x5	$\frac{1}{571} * \begin{bmatrix} 2 & 7 & 12 & 7 & 2 \\ 7 & 31 & 52 & 31 & 7 \\ 12 & 52 & 127 & 52 & 12 \\ 7 & 31 & 52 & 31 & 7 \\ 2 & 7 & 12 & 7 & 2 \end{bmatrix}$

© 2023 Vision Research - AMETEK Materials Analysis Division

9	HIPASS_3x3	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
10	HIPASS_5x5	$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$
11	SHARPEN_3x3	$\frac{1}{8} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 16 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
-1	User filter	

© 2023 Vision Research - AMETEK Materials Analysis Division

9.3. Appendix 3 – PhGet/PhSet Selectors

Name	Meaning	Type	Get	Set
Camera Info				
gsModel	Camera model	char[MAXSTDSTRSZ]	✓	✗
gsUvSensor	UV Sensor	BOOL	✓	
gsIPAddress	Camera current Ethernet address	char[MAXIPSTRSZ]	✓	✗
Video out Parameters				
gsVideoPlayCine	Cine number currently played on video out	INT	✓	✗
gsVideoPlaySpeed	Video out play speed in mHz.	INT	✓	✗
gsVideoOutputConfig	Video output type. Possible values are: 0 – Single-feed mode 1 – Single-feed mode with dual-link 444 2 – Dual-feed mode 3 – Dual-feed mode with 444	INT	✓	✓
Temperatures				
gsSensorTemperature	Sensor temperature	UINT	✓	✗
gsCameraTemperature	Camera temperature	INT	✓	✗
Lens Parameters				
gsLensFocus	Move the focus ring by the given number of incremental units relative to current focus position.	INT	✗	✓
gsLensAperture	fstop value	double	✓	✓
gsLensApertureRange	Focal length range	double[2]	✓	✗
gsLensDescription	Lens description	char[MAXSTDSTRSZ]	✓	✗
gsLensFocusInProgress	Focus in progress	BOOL	✓	✗
gsLensFocusAtLimit	Focus adjustment has hit a mechanical limit	BOOL	✓	✗
Camera Capabilities				
gsSupportsInternalBlackRef	Tells if the camera knows to perform a black calibration	BOOL	✓	✗
gsSupportsImageTrig	Image based auto trigger capability	BOOL	✓	✗
gsSupportsCardFlash	Tells if the camera supports a compact flash	BOOL	✓	✗
gsSupportsMagazine	Tells if the camera supports a CineMag	BOOL	✓	✗
gsSupportsHQMode	Tells if HQ mode is supported	BOOL	✓	✗

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
gsSupportsGenlock	Tells if video genlock is supported	BOOL	✓	×
gsSupportsEDR	Tells if EDR is supported	BOOL	✓	×
gsSupportsAutoExposure	Tells if auto-exposure is supported	BOOL	✓	×
gsHasV2AutoExposure	Tells if auto-exposure is V1 or V2 type	BOOL	✓	×
gsHasV2LockAtTrigger	Tells if auto exposure supports Lock At Trigger	BOOL	✓	×
gsSupportsTurbo	Tells if turbo mode is supported	BOOL	✓	×
gsSupportsBurstMode	Tells if burst mode is supported	BOOL	✓	×
gsSupportsShutterOff	Tells if shutter off mode is supported	BOOL	✓	×
gsSupportsDualSDIOutput	Tells if dual SDI output is present	BOOL	✓	×
gsSupportsRecordingCines	Tells if this device can record cines. (A CineStation can never record cines.)	BOOL	✓	×
gsSupportsV444	Tells if HD-SDI 4:4:4 output is available	BOOL	✓	×
gsSupportsInterlacedSensor	Tells if sensor has interlaced channels.	BOOL	✓	×
gsSupportsRampFRP	Tells if camera supports frame rate profile using the ramp camera field.	BOOL	✓	×
gsSupportsOffGainCorrections	Tells if camera implements offset gain corrections internally.	BOOL	✓	×
gsSupportsQuietMode	Tells if camera supports quiet mode (i.e. turning off the fan)	BOOL	✓	×
gsMaxPartitionCount	Maximum partition count.	UINT	✓	×
gsSupportsProgIO	Tells if camera supports programmable I/O	BOOL	✓	
gsSupportsSensorModes	Tells if camera supports sensor modes	BOOL	✓	
gsSupportsBattery	Tells if the camera supports a internal battery	BOOL	✓	×
gsSupportsExpIndex	Tells if the camera supports exposure index	BOOL	✓	
gsSupportsPIV	Tells if the camera supports PIV	BOOL	✓	
Camera Current Status Information				
gsHasMechanicalShutter	Mechanical shutter present or not	BOOL	✓	×
gsHasBlackLevel4	Tells if the camera uses a non-zero value for black level.	BOOL	✓	×
gsHasCardFlash	Tells if the camera has a card flash inserted and mounted	BOOL	✓	×
gsHas10G	Tells if the camera has a 10G connection up and running	BOOL	✓	×
Camera Current Parameters				
gsMechanicalShutter	Shutter control: on or off	INT	✓	✓
gsImageTrigThreshold	Amount a pixel value must change in order to be counted as an active pixel for auto trigger purposes	INT	✓	✓
gsImageTrigAreaPercentage	Percentage of the area of the auto trigger region that must be active in order for an auto trigger event to be generated	INT	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

<table>
<tr><th>Name</th><th>Meaning</th><th>Type</th><th>Get</th><th>Set</th></tr>
<tr><td>gsImageTrigSpeed</td><td>Number of frames between updates of the auto trigger reference memory</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsImageTrigMode</td><td>Auto trigger mode:
0 – auto trigger disabled
1 – the camera will both drive the autotrigger and trigger itself when an autotrigger condition is detected. If the autotrigger signal is pulled low by an external device, the camera will be triggered.
2 – the image changes are analyzed, and when an auto trigger condition is detected, the autotrigger signal is pulled low, as in mode 1, but the camera will not trigger itself. An external device pulling low the
autotrigger signal will not trigger the camera either.</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsImageTrigRect</td><td>Autotrigger region. Top left corner has (0, 0) coordinates.</td><td>RECT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsAutoProgress</td><td>Number of frames still to do by the camera auto process.</td><td>INT</td><td>✓</td><td>✗</td></tr>
<tr><td>gsAutoBlackRef</td><td>Auto black reference enabled or not (performing a CSR before each new recording)</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsCardFlashError</td><td>Last compact flash error</td><td>INT</td><td>✓</td><td>✗</td></tr>
<tr><td>gsGenlock</td><td>Video genlock</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsGenlockStatus</td><td>Genlock status</td><td>char[MAXSTDSTRSZ]</td><td>✓</td><td>✗</td></tr>
<tr><td>gsTurboMode</td><td>Turbo mode support</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsQuiet</td><td>Camera fan: on or off</td><td>INT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsClockPeriod</td><td>Obtain the camera clock period as a double value (seconds)</td><td>double</td><td>✓</td><td></td></tr>
<tr><td>gsPortCount</td><td>How many ports (fixed & programmable) are available on this camera (0 =indicates none)</td><td>UINT</td><td>✓</td><td></td></tr>
<tr><td>gsTriggerEdgeAndVoltage</td><td>bit 0 rising edge, bit 1 high voltage (6v threshold instead of 1.5v)
<table><tr><th>Bit 0 Value</th><th>Meaning</th></tr><tr><td>0</td><td>Trigger on falling edge</td></tr><tr><td>1</td><td>Trigger on rising edge</td></tr></table>
<table><tr><th>Bit 1 Value</th><th>Meaning</th></tr><tr><td>0</td><td>Use 1.5 Voltage threshold</td></tr><tr><td>1</td><td>Use high Voltage threshold
(6v threshold instead of 1.5v)</td></tr></table></td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsTriggerFilter</td><td>Filter constant (seconds)</td><td>double</td><td>✓</td><td>✓</td></tr>
<tr><td>gsTriggerDelay</td><td>Trigger delay in seconds available at camera. Only valid for cameras with programmable I/O capability</td><td>double</td><td>✓</td><td>✓</td></tr>
</table>

© 2023 Vision Research - AMETEK Materials Analysis Division

<table>
<tr><th>Name</th><th>Meaning</th><th>Type</th><th>Get</th><th>Set</th></tr>
<tr><td>gsSensorModesList</td><td>Get list of camera Sensor modes</td><td>char[MAXIPSTRSZ]</td><td>✓</td><td></td></tr>
<tr><td>gsSensorMode</td><td>Get/Set Sensor Mode</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryEnable</td><td>Enable/Disable battery operation</td><td>BOOL</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryCaptureEnable</td><td>Enable/Disable battery operation during capture</td><td>BOOL</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryPreviewEnable</td><td>Enable/Disable battery operation during preview</td><td>BOOL</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryWtrRuntime</td><td>Time in seconds the camera will run on battery in WTR if a cine is not triggered.</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryVoltage</td><td>Battery voltage level</td><td>double</td><td>✓</td><td>×</td></tr>
<tr><td>gsBatteryState</td><td>Battery control status
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>battery not present</td></tr><tr><td>1</td><td>battery charging</td></tr><tr><td>2</td><td>battery charging, voltage above 1/2 capacity threshold;</td></tr><tr><td>3</td><td>battery charged;</td></tr><tr><td>4</td><td>battery discharging;</td></tr><tr><td>5</td><td>battery low;</td></tr><tr><td>16</td><td>battery charging fault.</td></tr></table>When the battery is armed, bit 3 is set (8 is added to the values above).</td><td>UINT</td><td>✓</td><td>×</td></tr>
<tr><td>gsBatteryArmDelay</td><td>Delay, in seconds, from the moment the camera is placed into WTR and the time the battery is armed.</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryPrevRuntime</td><td>Time in seconds the camera will run on battery in preview if a cine is not triggered.</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryPwrOffDelay</td><td>Time in seconds the battery still supplies power to the camera after it has been disarmed.</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsBatteryReadyGate</td><td>Set/Get Battery readygate control</td><td>BOOL</td><td>✓</td><td>✓</td></tr>
<tr><td>gsExpIndex</td><td>Set/Get exposure index ISO value</td><td>UINT</td><td>✓</td><td>✓</td></tr>
<tr><td>gsExpIndexPresets</td><td>Get list of camera ISO values</td><td>UINT[10]</td><td>✓</td><td></td></tr>
<tr><td colspan="5">Network Interface Alias</td></tr>
<tr><td>gsEthernetDefaultAddress</td><td>Primary/Default IP address</td><td>char[MAXIPSTRSZ]</td><td>✓</td><td>✓</td></tr>
<tr><td>gsEthernetAddress</td><td>Secondary IP address
DHCP Control:
Enable: Set this to the word “auto”
Disable: Remove the word “auto”

Note: DHCP is only functional on selective cameras</td><td>char[MAXIPSTRSZ]</td><td>✓</td><td>✓</td></tr>
</table>

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
gsEthernetMask	Network mask	char[MAXIPSTRSZ]	✓	✓
gsEthernetBroadcast	Broadcast mask	char[MAXIPSTRSZ]	✓	✓
gsEthernetGateway	IP address of default gateway	char[MAXIPSTRSZ]	✓	✓
gsEthernet10GAddress	10G IP address	char[MAXIPSTRSZ]	✓	✓
gsEthernet10GMask	10G Network mask	char[MAXIPSTRSZ]	✓	✓
gsEthernet10GBroadcast	10G Broadcast mask	char[MAXIPSTRSZ]	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

9.4. Appendix 4 – PhCineGet/PhCineSet Selectors

Name	Meaning	Type	Get	Set
Cine Status Parameters				
cgsVideoMarkIn	Start of the videoplay range	INT	✓	✓
cgsVideoMarkOut	End of the videoplay range	INT	✓	✓
cgsIsRecorded	Cine status: recorded or not	BOOL	✓	✗
cgsHqMode	HQ mode status	BOOL	✓	✓
Burst Mode Parameters				
cgsBurstCount	Number of frames in a burst	UINT	✓	✓
cgsBurstPeriod	Interval between two frames in a burst, in nanoseconds	UINT	✓	✓
Lens Information				
cgsLensDescription	Lens description	char[MAXSTDSTRSZ]	✓	✗
cgsLensAperture	Lens aperture f number	float	✓	✗
cgsLensFocalLength	Lens focal length	float	✓	✗
Acquisition Parameters				
cgsShutterOff	Enable/disable maximum exposure for piv mode	BOOL	✓	✓
Time Information				
cgsTriggerTime	Time of trigger	TIME64	✓	✗
cgsTrigTC	Time code of the trigger frame	TC	✓	✗
cgsPbRate	Playback rate (fps) of the video mode active when the cine was captured	float	✓	✗
cgsTcRate	Time code rate (fps) corresponding to the video mode active when the cine was captured	float	✓	✗
Image Processing Parameters				
cgsWb	White balance gains	WBGAIN	✓	✓
cgsBright	Brightness	float	✓	✓
cgsContrast	Contrast	float	✓	✓
cgsGamma	Per component gamma		✓	✓
cgsGammaR	Difference between red channel gamma and overall gamma	float	✓	✓
cgsGammaB	Difference between blue channel gamma and overall gamma	float	✓	✓
cgsSaturation	Saturation	float	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

Name	Meaning	Type	Get	Set
cgsHue	Hue	float	✓	✓
cgsPedestalR	Red channel pedestal	float	✓	✓
cgsPedestalG	Green channel pedestal	float	✓	✓
cgsPedestalB	Blue channel pedestal	float	✓	✓
cgsFlare	Flare adjustment	float	✓	✓
cgsChroma	Chroma gain	float	✓	✓
cgsTone	A label plus a set of control points describing the tone	TONEDESC	✓	✓
cgsUserMatrix	A label plus a 3 by 3 matrix describing user matrix.	CMDESC	✓	✓
cgsEnableMatrices	Disable/enable user matrix	BOOL	✓	✓
Other Parameters				
cgsName	Cine name	char[MAXSTDSTRSZ]	✓	✓
Deprecated				
cgsVideoTone	Integer selector among predefined video tones	INT	✓	✓

© 2023 Vision Research - AMETEK Materials Analysis Division

9.5. Appendix 5 – PhGet1/PhSet1 Selectors

Name	Meaning	Type	Get	Set
gsSigCount	Signal count available at the port	UINT	✓	✗
gsSigSelect	Get/Select the signal to connect to a certain port	UINT	✓	✓✗
gsPulseProc	Parameters for pulse processor (in a structure)	PULSEPARAM	✓	✓✗
gsHasPulseProc	Tells if this port supports pulse processor control	BOOL	✓	✗

© 2023 Vision Research - AMETEK Materials Analysis Division

9.6. Appendix 6 – PhGet2 Selectors

Name	Meaning	Type	Get	Set
gsSigName	Signal name, given the port number and signal number	Char[MAXSTDSTRSZ]	✓	✗

© 2023 Vision Research - AMETEK Materials Analysis Division

9.7. Appendix 7 – PhUpgradeFirmware Selectors

Name	Meaning	Type	Get	Set
FW_PHFW	Upgrade Ph16 camera firmware (.phfw file)	Char[MAXSTDSTRSZ]		✓✖

© 2023 Vision Research - AMETEK Materials Analysis Division