Amcam API Manual

1. Version & Platform

- Version: 50.19367.20210815
- Platform
  - Win32:
    - x86: XP SP3 or above; CPU supports SSE2 instruction set or above
    - x64: Win7 or above
    - arm: Win10 or above
    - arm64: Win10 or above
  - WinRT: x86, x64, arm, arm64; Windows10 or above
  - macOS: universal (x64 + x86); macOS 10.10 or above
  - Linux: kernel 2.6.27 or above; GLIBC 2.17 or above
    - x86: CPU supports SSE3 instruction set or above; built by gcc 5.4.0
    - x64: built by gcc 5.4.0
    - armel: built by arm-linux-gnueabi (version 4.9.2)
    - armhf: built by arm-linux-gnueabihf (version 4.9.2)
    - arm64: built by aarch64-linux-gnu (version 4.9.2)
  - Android: arm, arm64, x86, x64; built by android-ndk-r18b; __ANDROID_API__ = 23

2. Introduction

Amcam cameras support various kinds of APIs (Application Program Interface), namely Native C/C++, .NET/C#/VB.NET, Python, Java, DirectShow, Twain, LabView, MabLab and so on. Compared with other APIs, Native C/C++ API, as a low level API, don't depend any other runtime libraries. Besides, this interface is simple, flexible and easy to be integrated into the customized applications. The SDK zip file contains all of the necessary resources and information:

- inc

  amcam.h, C/C++ head file

- win: For Microsoft Windows
  - dotnet

    amcam.cs, for C#. The amcam.cs use P/Invoke to call into amcam.dll. Please copy amcam.cs to your C# project to use it.
    amcam.vb, for VB.NET. The amcam.vb use P/Invoke to call into amcam.dll. Please copy amcam.vb to your VB.NET project to use it.

  - x86

    amcam.lib, lib file for x86
    amcam.dll, dll file for x86
    democpp.exe, x86 C++ demo exe file

  - x64

    amcam.lib, lib file for x64
    amcam.dll, dll file for x64
    democpp.exe, x64 C++ demo exe file

  - arm

    amcam.lib, lib file for arm
    amcam.dll, dll file for arm

  - arm64

    amcam.lib, lib file for arm64
    amcam.dll, dll file for arm64

  - winrt

    DLL files for WinRT / UWP (Universal Windows Platform) / Windows Store App.
    These dll files are compatible with Windows Runtime, and can be consumed from a Universal Windows Platform app.
    If use C# to develop the UWP, amcam.cs wrapper class can be used to P/Invoke into amcam.dll.
    Please pay attation to:

    1. uwp must use winusb, cannot use the proprietary driver. If the proprietary driver has already been installed, please uninstall it in the device manager, after this, Windows will use winusb automatically.
    2. DeviceCapability of uwp, see: How to add USB device capabilities to the app manifest. (It seems that microsoft limits the number of Device items to 100)

    demouwp.zip is a simple demo, before build and run this demo, please change the value of vidpid in file Package.appxmanifest

- drivers (**The cameras produced after Jan. 1, 2017 support WinUSB, it's no longer need to install driver in Windows8 and above versions**)
  1. x86 folder contains the kernel mode drivers for x86, including amcam.cat, amcam.inf and amcam.sys.
  2. x64 folder contains the kennel mode driver for x64, including amcam.cat, amcam.inf and amcam.sys.
     It is recommended to use DPInst.exe to automatically install the driver. If you use NSIS to make the installation file, you can use a statement similar to the following:

     ```
     ExecWait '"$INSTDIR\drivers\x64\DPInst.exe" /SA /SW /PATH "$INSTDIR\drivers\x64"'
     ```

- samples
  1. democpp, C++ sample. It demonstrates to enumerate device, open device, video preview, image capture, set the preview resolution, trigger, multi-format image saving (.bmp, .jpg, .png, etc), wmv format video recording, trigger mode, IO control and so on. This sample use Pull Mode. To keep the code clean, this sample uses the WTL library which can be downloaded from http://sourceforge.net/projects/wtl
  2. AutoTestTool, auto test tool used to automatically test, such as open/close the camera, change the resolution, snap, ROI, bitdepth, etc
  3. demopush, C++ sample, this sample use Push Mode, StartPushModeV3
  4. demomfc, a simple C++ sample. It use MFC as the GUI library. It demonstrates to open device, video preview, image capture, set the preview resolution, multi-format image saving (.bmp, .jpg, .png, etc). This sample use Pull Mode.
  5. demomono, a simple C++ sample. It demonstrates to use mono camera with 8 or 16 bits.
  6. demowinformcs1, C# winform sample. This sample use Pull Mode, StartPullModeWithWndMsg.
  7. demowinformcs2, C# winform sample. This sample use Pull Mode, StartPullModeWithCallback.
  8. demowinformcs3, C# winform sample. This sample use Push Mode, StartPushModeV3.
  9. demowinformvb, VB.NET winform sample. This sample use Pull Mode.

- linux: For Linux
  - udev: 99-amcam.rules, udev rule file. Please see: http://reactivated.net/writing_udev_rules.html
  - c#: amcam.cs, for .Net Core C#. The amcam.cs use P/Invoke to call into libamcam.so. Please copy amcam.cs to your C# project to use it
  - x86: libamcam.so, so file for x86
  - x64: libamcam.so, so file for x64
  - armel: libamcam.so, so file for armel, use toolchain arm-linux-gnueabi
  - armhf: libamcam.so, so file for armhf, use toolchain arm-linux-gnueabihf
  - arm64: libamcam.so, so file for arm64, use toolchain aarch64-linux-gnu

- android: libamcam.so for Android on arm, arm64, x86 and x64

- mac: For macOS

- python: amcam.py and sample code (Console and pyQt)

- java: amcam.java and sample code (Console and Swing)

- doc: User manuals in English and Simplified Chinese

- sample
  - demosimplest: a simplest sample, about 60 lines of code
  - demoraw: raw data and snap still image, about 120 lines of code
  - demoqt, Qt sample.

- extras
  - directshow: DirectShow SDK and demo
  - twain: TWAIN SDK
  - labview: Labview SDK and demo
  - matlab: MatLab demo

---

3. Concepts and terminology

---

a. Modes for accessing image data: "Pull Mode" vs "Push Mode"

Amcam offers two modes to obtain image data: Pull Mode and Push Mode. The former is recommended since it's simpler and the application seldom gets stuck in multithreading conditions, especially when using windows message to notify the events.

- In Pull Mode, amcam plays a passive role and the application 'PULL' image data from amcam. The internal thread of amcam obtains image data from the camera hardware and saves them to the internal buffers, then notify the application (see below). The application then call functions Amcam_PullImage(WithRowPitch)(V2) and Amcam_PullStillImage(WithRowPitch)(V2) to access image data.

  There are to ways to notify applications:

  a) Use Windows message: Start pull mode by using the function Amcam_StartPullModeWithWndMsg. When event occurs, amcam will post message (PostMessage) to the specified window. Parameter WPARAM is the event type, refer to the definition of AMCAM_EVENT_xxxx. This model avoids the multithreading issues, so it's the most simple way. (Obviously, this is only supported in Windows systems, and not supported in Linux and macOS.)

  b) Use Callback function: Start pull mode by using the function Amcam_StartPullModeWithCallback. When event occurs, amcam will callback the function PAMCAM_EVENT_CALLBACK.

  In Pull Mode, the SDK could not only notify the application that the image data or still image are available for 'PULL', but also inform you of the other events, such as:

AMCAM_EVENT_EXPOSURE	exposure time changed
AMCAM_EVENT_TEMPTINT	white balance changed. Temp/Tint Mode, please see here.

AMCAM_EVENT_WBGAIN	white balance changed. RGB Gain Mode, please see here.
AMCAM_EVENT_IMAGE	Video image data arrives. Use Amcam_PullImage(V2) to 'pull' the image data
AMCAM_EVENT_STILLIMAGE	Still image which is triggered by function Amcam_Snap or Amcam_SnapN arrives. Use Amcam_PullStillImage(V2) to 'pull' the image data
AMCAM_EVENT_ERROR	Generic error, data acquisition cannot continue
AMCAM_EVENT_DISCONNECTED	Camera disconnected, maybe has been pulled out
AMCAM_EVENT_NOFRAMETIMEOUT	Grab image no frame timeout error, data acquisition cannot continue
AMCAM_EVENT_NOPACKETIMEOUT	Grab image no packet timeout
AMCAM_EVENT_TRIGGERFAIL	trigger failed (for example, bad frame data or timeout)
AMCAM_EVENT_BLACK	black balance changed
AMCAM_EVENT_FFC	flat field correction status changed
AMCAM_EVENT_DFC	dark field correction status changed
AMCAM_EVENT_ROI	roi changed
AMCAM_EVENT_LEVELRANGE	level range changed
AMCAM_EVENT_EXPO_START	exposure start
AMCAM_EVENT_EXPO_STOP	exposure stop
AMCAM_EVENT_TRIGGER_ALLOW	next trigger allow
AMCAM_EVENT_FACTORY	restore factory settings. Please note that restoring factory settings may cause resolution changes.

- In Push Mode, amcam plays an active role. Once the video data is obtained from camera by internal thread, amcam will 'PUSH' the image data to the application through PAMCAM_DATA_CALLBACK. Call the function Amcam_StartPushMode to start push mode. Push mode is more complex. There are some special precautions, such as multithread issues, being impossible to call Amcam_Close and Amcam_Stop in callback function PAMCAM_DATA_CALLBACK, etc.

b. Still Capture (Still Image)

Most cameras support the so-called still capture capability. This function switches the camera to another resolution temporarily when the camera is in preview mode, after a "still" image in the new resolution is captured and then switch back to the original resolution and resume preview mode.

For example, UCMOS05100KPA support 3 resolutions and the current one in preview mode is 1280 * 960. Call Amcam_Snap(h, 0) to "still capture" an image in 2592 * 1944 resolution. To realize this function, the camera will temporarily switch to 2592 * 1944 firstly, get an image in 2592 * 1944 resolution and then switch back to 1280 * 960 and resume preview.

a) In pull mode operation, after the still capture, AMCAM_EVENT_STILLIMAGE will be sent out for external acknowledgement. The external application should call Amcam_PullStillImage(V2) to get the still captured image.

b) In push mode operation, after the still capture, the callback function PAMCAM_DATA_CALLBACK_V3 will be called with bSnap parameter setting TRUE. The image information including the resolution information will be obtained via the parameter pHeader.

To check whether the camera have the still capture capability, call Amcam_get_StillResolutionNumber function or check the still field of the struct AmcamModelV2.

c. Data format: RGB vs RAW

Amcam supports two data formats: RGB format (default) and RAW format. RAW format could be enabled by assigning AMCAM_OPTION_RAW parameter to 1 when calling Amcam_put_Option function.

- RGB format: The output of every pixel contains 3 componants which stand for R/G/B value respectively. This output is a processed output from the internal color processing engine.
- RAW format: In this format, the output is the raw data directly output from the sensor. The RAW format is for the users that want to skip the internal color processing and obtain the raw data for user-specific purpose. With the raw format output enabled, the functions that are related to the internal color processing will not work, such as Amcam_put_Hue or Amcam_AwbOnce function and so on.

Users could switch these two format by calling Amcam_put_Option function with different value setting to AMCAM_OPTION_RAW. You must call this function BEFORE the camera start function (Amcam_StartPullModeWithWndMsg or Amcam_StartPullModeWithCallback or Amcam_StartPushMode).

d. White Balance and Auto White Balance: Temp/Tint mode vs RGB Gain mode

1. Amcam sdk supports two independent modes for white balance: a) Temp/Tint Mode; b) RGB Gain Mode

a) Temp/Tint mode is the default white balance mode. In this mode, temp and tint are the parameters that could be used to control the white balance. Amcam_get_TempTint function is used to acquire the temp and tint values and Amcam_put_TempTint is used to set the temp and tint values. Function Amcam_AwbOnce is used to execute the auto white balance. When the white balance parameters change, AMCAM_EVENT_TEMPTINT event will be notified for external use.

b) In RGB Gain mode, the while balace is controled by the gain values of the R,G,B channels. Amcam_get_WhiteBalanceGain is used to acquire the parameters and Amcam_put_WhiteBalanceGain is used to set the white balance parameters. Amcam_AwbInit is used to execute the execute the auto white balance. When the white balance parameters change, AMCAM_EVENT_WBGAIN event will be notified for external use.

The functions for these two modes cannot be misused:

a) In Temp/Tint mode, please use Amcam_get_TempTint and Amcam_put_TempTint and Amcam_AwbOnce. Amcam_get_WhiteBalanceGain and Amcam_put_WhiteBalanceGain and Amcam_AwbInit cannot be used, they always return E_NOTIMPL.
b) In RGB Gain mode, please use Amcam_get_WhiteBalanceGain and Amcam_put_WhiteBalanceGain and Amcam_AwbInit. Amcam_get_TempTint and Amcam_put_TempTint and Amcam_AwbOnce cannot be used, they always return E_NOTIMPL

When calling Amcam_Open function, whether to add a '@' character at the beginning of the id parameter will determine the white balance mode. Add a '@' character at the beginning of the id parameter means the RGB gain mode. If you want to use the RGB Gain mode, for example, if the id parameter is "abcdef", please call the Amcam_Open function with the id parameter "@abcdef".

2. There are two auto white balance mechanisms available in this field: one is continuous auto white balance and the other is a "once" auto white balance. The white balance parameters will be always calculated and updated for the continuous auto white balance mechanism. For "once" auto white balance mechanism, the white balance parameters will be calculated and updated only when triggered. Amcam cameras support "once" auto white balance mechanism since it is more accurate and propriate for the microscope application, especially when the background is in pure color. Continuous white balance mechanism will encounter some problem in some cases.

3. Monochromatic camera does not support white balance. The functions metioned above always return E_NOTIMPL.

e. Trigger Mode

1. What is Trigger Mode

   Amcam camera has two working modes: video mode and trigger mode. When in trigger mode, no images will be available until the trigger conditions are met. Cameras have 2 types for triggering according to the types of trigger source, including software trigger mode, external trigger mode and simulated trigger mode.

2. The Difference between Trigger and Snap

   Trigger mode is designed for accurate control of the camera and images would be acquired only when the conditions are met. Users could get the images by controlling the trigger conditions. Trigger mode must be pre-specified. Once the trigger mode is entered, no images will come out from the camera until the trigger conditions are met. The triggered images will stay the same property as the pre-specified resolution. Snap is designed to acquire the images from the camera in video mode. The resolution of the snapped image could be the same resolution as the video or could be different.

3. Software Trigger Mode

   Camera could be triggered by software. In software trigger mode, images burst out only when users trigger the camera from the software. Numbers of the images of the triggering could also be controlled by software.

4. External Trigger Mode

   Camera could be triggered by external trigger signal. By now Amcam camera only supports positive-edge trigger mode.

5. Mix Trigger Mode

   Both external and software trigger are enabled.

6. Simulated Trigger Mode

   For cameras that do not support software trigger and external trigger, simulated trigger mode could be available. When in simulated trigger mode, the camera hardware actually works in the same mode as the video mode. But the up-level software will not obtain any images from the camera. The software buffer will stay empty until the user set the trigger conditions by software.

7. How to Enter the Trigger Mode

   After the numeration of the connected camera, you can check the flag and find out what trigger mode does the camera support according to the following definition.

```
#define AMCAM_FLAG_TRIGGER_SOFTWARE   0x00080000  /* support software trigger */
#define AMCAM_FLAG_TRIGGER_EXTERNAL   0x00100000  /* support external trigger */
#define AMCAM_FLAG_TRIGGER_SINGLE     0x00200000  /* only support trigger single: one trigger, one image */
```

   Function Amcam_put_Option(HAmcam h, unsigned iOption, int iValue) could be used to set the camera to trigger mode when assign AMCAM_OPTION_TRIGGER to the iOption parameter. iValue is used to specify the types of the trigger modes. Please see the following for reference.

```
#define AMCAM_OPTION_TRIGGER   0x0b    /* 0 = video mode, 1 = software or simulated trigger mode, 2 = external trigger mode, 3 = external + sof
```

   Function Amcam_get_Option(HAmcam h, unsigned iOption, int* piValue) could be used to get what type of trigger mode the camera is in.

8. How to Trigger the camera

   Function Amcam_Trigger(HAmcam h, unsigned short nNumber) could be used to trigger the camera. Assigning different value to nNumber means different:

   nNumber = 0 means stop the trigger.
   nNumber = 0xFFFF means trigger continuously, the same as video mode;
   nNumber = other valid values means nNumber images will come out from the camera.

   If the AMCAM_FLAG_TRIGGER_SINGLE flag is checked, the nNumber parameter must be assigned to 1 and 1 image will come out from the camera when Amcam_Trigger is called.
   Enter the trigger mode first and then call Amcam_Trigger function to trigger the camera.

9. Trigger timeout

   The timeout is recommended for not less than (Exposure Time * 102% + 4 Seconds).

---

4. Functions

- **HRESULT return value**

HRESULT is not uncommon on the Windows platform. It's borrowed to macOS and Linux, see the table:

Please note that the return value >=0 means success (especially S_FALSE is also successful, indicating that the internal value and the value set by the user is equivalent, which means "no operation"). Therefore, the SUCCEEDED and FAILED macros should generally be used to determine whether the return value is successful or failed.
(Unless there are special needs, do not use "==S_OK" or "==0" to judge the return value)

Name	Description	Value
S_OK	Operation successful	0x00000000
S_FALSE	Operation successful, nothing changed	0x00000001
E_FAIL	Unspecified failure	0x80004005
E_ACCESSDENIED	General access denied error	0x80070005
E_INVALIDARG	One or more arguments are not valid	0x80070057
E_NOTIMPL	Not supported or not implemented	0x80004001
E_POINTER	Pointer that is not valid	0x80004003
E_UNEXPECTED	Unexpected failure	0x8000FFFF
E_WRONG_THREAD	Called in wrong thread	0x8001010E
E_GEN_FAILURE	Device not functioning	0x8007001F
E_PENDING	The data necessary to complete this operation is not yet available	0x8000000a

```
#define SUCCEEDED(hr)   (((HRESULT)(hr)) >= 0)
#define FAILED(hr)      (((HRESULT)(hr)) < 0)
```

- ## Calling Convention

Win: __stdcall, please see here

macOS, Linux and Android: __cdecl

- ## Callback: PAMCAM_EVENT_CALLBACK and PAMCAM_DATA_CALLBACK_V3

These callback functions are called back from the internal thread in amcam.dll, so great attention should be paid to multithread issue.
Please ensure that the callback funcion is simple and return quickly.
Otherwise, in callback mode, AMCAM_OPTION_CALLBACK_THREAD can be setted to use a dedicated thread for callback.

There are limitations in the callback context:

(a) Do NOT call Amcam_Stop and Amcam_Close in this callback function, otherwise, deadlocks.

(b) Do NOT call Amcam_put_Option with AMCAM_OPTION_TRIGGER, AMCAM_OPTION_BITDEPTH, AMCAM_OPTION_PIXEL_FORMAT, AMCAM_OPTION_BINNING, AMCAM_OPTION_ROTATE, the return value is E_WRONG_THREAD.

- ## Coordinate

Functions with coordinate parameters, such as Amcam_put_Roi, Amcam_put_AEAuxRect, etc., the coordinate is **always relative to the original resolution**, even that the video has been flipped, rotated, digital binning, ROIed, or combination of the previous operations.

If the image is upsize down (see here), the coordinate must be also upsize down.

- ## Amcam_EnumV2

Return value: non-negative integer, enumerated camera number

Parameters:

AmcamDeviceV2 pti[AMCAM_MAX]: AmcamDeviceV2 buffer

Remarks: call this function to enumerate Amcam cameras that are currently connected to computer and when it is returned, AmcamDeviceV2 buffer contains the information of each camera instance enumerated.**If we don't care about that multiple cameras connect to the computer simultaneously, it's optional to call this function to enumerate the camera instances**.

The code snippet shows as below:

```
AmcamDeviceV2 arr[AMCAM_MAX];
unsigned cnt = Amcam_EnumV2(arr);
for (unsigned i = 0; i < cnt; ++i)
    ......
```

```
typedef struct{
#ifdef _WIN32
    const wchar_t*     name;     /* model name */
#else
    const char*        name;
#endif
    unsigned long long flag;     /* AMCAM_FLAG_xxx */
    unsigned           maxspeed; /* maximum speed level, Amcam_get_MaxSpeed, the speed range = [0, maxspeed], closed interval */
    unsigned           preview;  /* number of preview resolution, Amcam_get_ResolutionNumber */
```

```
    unsigned           still;    /* number of still resolution, Amcam_get_StillResolutionNumber */
    unsigned           maxfanspeed; /* maximum fan speed */
    AmcamResolution  res[AMCAM_MAX];
}AmcamModelV2;
```

name	The name of this model	
flag	Bitwise flag	
	AMCAM_FLAG_CMOS	cmos sensor
	AMCAM_FLAG_CCD_PROGRESSIVE	progressive ccd sensor
	AMCAM_FLAG_CCD_INTERLACED	interlaced ccd sensor
	AMCAM_FLAG_ROI_HARDWARE	support hardware ROI. Hardware ROI means only the ROI part of image is output from the sensor and the software cropping operation is not required. Higher frame rate is achieved when using hardware ROI method. Software ROI means the image with the complete field of view of the sensor will be output and software cropping operation is required to obtain the ROI image.
	AMCAM_FLAG_MONO	monochromatic sensor
	AMCAM_FLAG_BINSKIP_SUPPORTED	support bin/skip mode, see Amcam_put_Mode and Amcam_get_Mode
	AMCAM_FLAG_USB30	usb3.0
	AMCAM_FLAG_TEC	Thermoelectric Cooler
	AMCAM_FLAG_USB30_OVER_USB20	usb3.0 camera connected to usb2.0 port
	AMCAM_FLAG_ST4	ST4 port
	AMCAM_FLAG_GETTEMPERATURE	support to get the temperature of the sensor, Amcam_get_Temperature
	AMCAM_FLAG_RAW10	Pixel format, RAW 10 bits
	AMCAM_FLAG_RAW12	Pixel format, RAW 12 bits
	AMCAM_FLAG_RAW14	Pixel format, RAW 14 bits
	AMCAM_FLAG_RAW16	Pixel format, RAW 16 bits
	AMCAM_FLAG_FAN	cooling fan
	AMCAM_FLAG_TEC_ONOFF	Thermoelectric Cooler can be turn on or off, target temperature of TEC, see: AMCAM_OPTION_TEC AMCAM_OPTION_TECTARGET
	AMCAM_FLAG_ISP	ISP (Image Signal Processing) chip
	AMCAM_FLAG_TRIGGER_SOFTWARE	support software trigger
	AMCAM_FLAG_TRIGGER_EXTERNAL	support external trigger
	AMCAM_FLAG_TRIGGER_SINGLE	only support trigger single, one trigger, one image
	AMCAM_FLAG_BLACKLEVEL	support set and get the black level
	AMCAM_FLAG_FOCUSMOTOR	support focus motor
	AMCAM_FLAG_AUTO_FOCUS	support auto focus
	AMCAM_FLAG_BUFFER	frame buffer
	AMCAM_FLAG_DDR	use very large capacity DDR (Double Data Rate SDRAM) for frame buffer
	AMCAM_FLAG_CG	Conversion Gain: LCG, HCG
	AMCAM_FLAG_CGHDR	Conversion Gain: LCG, HCG, HDR
	AMCAM_FLAG_EVENT_HARDWARE	hardware event, such as exposure start & stop
	AMCAM_FLAG_YUV411	pixel format, yuv411
	AMCAM_FLAG_YUV422	pixel format, yuv422
	AMCAM_FLAG_YUV444	pixel format, yuv444
	AMCAM_FLAG_RGB888	pixel format, RGB888
	AMCAM_FLAG_RAW8	pixel format, RAW 8 bits
	AMCAM_FLAG_GMCY8	pixel format, GMCY, 8bits
	AMCAM_FLAG_GMCY12	pixel format, GMCY, 12 btis
	AMCAM_FLAG_GLOBALSHUTTER	global shutter
	AMCAM_FLAG_PRECISE_FRAMERATE	support precise framerate & bandwidth, see AMCAM_OPTION_PRECISE_FRAMERATE & AMCAM_OPTION_BANDWIDTH
	AMCAM_FLAG_HEAT	support heat to prevent fogging up, see AMCAM_OPTION_HEAT & AMCAM_OPTION_HEAT_MAX
	AMCAM_FLAG_LOW_NOISE	support low noise mode, see AMCAM_OPTION_LOW_NOISE
	AMCAM_FLAG_LEVELRANGE_HARDWARE	hardware level range
maxspeed	Maximum speed level, same with Amcam_get_MaxSpeed. The speed range is [0, maxspeed]. see Amcam_put_Speed and Amcam_get_Speed	
preview	Number of preview resolution. Same with Amcam_get_ResolutionNumber	
still	Number of still resolution, zero means still capture is not supported. Same with Amcam_get_StillResolutionNumber	
res	Resolution, width and height	

- ## Amcam_HotPlug

Return value: NA

Parameters:

PAMCAM_HOTPLUG pHotPlugCallback: callback function

typedef void (*PAMCAM_HOTPLUG)(void* pCallbackCtx);

void* pCallbackCtx: callback context

Remarks:

This function is only available on macOS, Linux and Android.

To process the device plug in / pull out in Windows, please refer to the MSDN(Device Management, Detecting Media Insertion or Removal).

To process the device plug in / pull out in Linux/macOS/Android, please call this function to register the callback function. When the device is inserted or pulled out, you will be notified by the callback funcion, and then call Amcam_EnumV2(...) again to enum the cameras.

On macOS, IONotificationPortCreate series APIs can also be used as an alternative.

Recommendation: for better rubustness, when notify of device insertion arrives, don't open handle of this device immediately, but open it **after delaying a short time (e.g., 200 milliseconds)**.

- ## Amcam_Open

Return value: HAmcam handle. Return NULL when fails (Such as the device has been pulled out).

Parameters:

id: Amcam camera instance, enumerated by Amcam_EnumV2. **If id is NULL, Amcam_Open will open the first camera which connects to the computer. So, if we don't care about that multiple cameras connect to the computer simultaneously, Amcam_EnumV2 is optional, we can simply use NULL as the parameter.**

Remarks: open the camera instance.

Camera Configuration: No parameter is required after id parameter if automatic parameter saving or loading is not required.

Add the specific parameters after the id parameter when automatic parameter saving (when closing the camera) or loading (when starting the camera) is required, for example, Amcam_Open(L"0-1010;registry=").

Please refer to the following description for details:

Registry (Windows only)	;registry=xxxx\yyyy	use the registry that the camera parameters are saved to or load from. Empty after "=" means that the default registry location will be used
ini file	;ini=x:\yyyy\zzzz.ini	indicates that x:\yyyy\zzzz.ini is the file where camera parameters are saved to or load from. Complete directory must be specified and empty is not allowed. Please Make sure that the target directory exists and is readable and writeable
json file	;json=x:\yyyy\zzzz.json	indicates that x:\yyyy\zzzz.json is the file where camera parameters are saved to or load from. Complete directory must be specified and empty is not allowed. Please Make sure that the target directory exists and is readable and writeable
EEPROM	;eeprom=xxxx	indicates that EEPROM is the device where the camera parameters are saved to or load from. xxxx is the starting address in EEPROM and empty means the starting address is 0

- ## Amcam_Close

Return value: void

Parameters:

HAmcam h: camera instance handle

Remarks: close the camera instance. After it is closed, never use the HAmcam handle any more.

- ## Amcam_StartPullModeWithWndMsg, Amcam_StartPullModeWithCallback

Return value: HRESULT type means "success / failure"

Parameters:

HAmcam h: instance handle opened by Amcam_Open

HWND hWnd: event occurs, message will be posted in this window

UINT nMsg: Windows custom message type. Its WPARAM parameter means event type AMCAM_EVENT_xxxx, LPARAM is useless (always zero)

PAMCAM_EVENT_CALLBACK pEventCallback, void* pCallbackContext: callback function specified by user's application and callback context parameter.

```
typedef void (*PAMCAM_EVENT_CALLBACK)(unsigned nEvent, void* pCallbackCtx);
```

see here.

Remarks: Obviously, Amcam_StartPullModeWithWndMsg is only supported in Windows OS.

- ## Amcam_PullImageV2, Amcam_PullStillImageV2, Amcam_PullImageWithRowPitchV2, Amcam_PullStillImageWithRowPitchV2, Amcam_PullImage, Amcam_PullStillImage, Amcam_PullImageWithRowPitch, Amcam_PullStillImageWithRowPitch

Return value: HRESULT type means "success/ failure". Return E_PENDING when there isn't image ready for pull.

Parameters:

HAmcam h: instance handle opened by Amcam_Open

void* pImageData: Data buffer. Users have to make sure that the data buffer capacity is enough to save the image data, data buffer capacity must >= rowPitch * nHeight.

int bits: 24, 32, 48, 8, 16, means RGB24, RGB32, RGB48, 8 bits gray or 16 bits gray images. This parameter is ignored in RAW mode.

int rowPitch: the distance from one row to the next row. rowPitch = 0 means using the default row pitch. rowPitch = -1 means zero padding

unsigned* pnWidth, unsigned* pnHeight: out parameter. width and height of image.

AmcamFrameInfoV2* pInfo: out parameter, frame info. Some cameras support frame sequence and frame timestamp, for other unsigned cameras, sequence and timestamp are always 0

Remarks: when pImageData is NULL, while pnWidth and pnHeight are not NULL, you can "peek" the width and height of images.

Please ensure that the pImageData buffer is large enough to hold the entire frame data, see the table below:

Format		=0 means Default Row Pitch	=-1 means zero paddding
RGB	RGB24	TDIBWIDTHBYTES(24 * Width)	Width * 3
	RGB32	Width * 4	Width * 4
	RGB48	TDIBWIDTHBYTES(48 * Width)	Width * 6
	GREY8 grey image	TDIBWIDTHBYTES(8 * Width)	Width
	GREY16 grey image	TDIBWIDTHBYTES(16 * Width)	Width * 2
RAW	8bits Mode	Width	Width
	10bits, 12bits, 14bits, 16bits Mode	Width * 2	Width * 2

```
#ifndef TDIBWIDTHBYTES
#define TDIBWIDTHBYTES(bits)    ((unsigned)(((bits) + 31) & (~31)) / 8)
#endif
```

- ## Amcam_StartPushModeV3

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: instance handle opened by Amcam.

PAMCAM_DATA_CALLBACK_V3 pDataCallback, void* pDataCallbackCtx: the callback function and callback context parameters that are specified by the user's program. Amcam.dll gets image data from the camera, then calls back this function.

typedef void (*PAMCAM_DATA_CALLBACK_V3)(const void* pData, const AmcamFrameInfoV2* pInfo, int bSnap, void* pCallbackCtx);

see here.

when calls back, if Parameter pData == NULL, then internal error occurs (eg: the camera is pulled out suddenly).
The row pitch of pData is always the default value.
For parameter int bSnap, TRUE means still image snap by Amcam_Snap or Amcam_SnapN function, FALSE means ordinary previewed pictures / videos.

Remarks: start camera instance.

- ## Amcam_Stop

Return value: HRESULT type means success or failure

Parameters:

HAmcam handle

Remarks: stop the camera instance. After stopped, it can be restart again. For example, switching the video resolution:

Step 1: call Amcam_Stop to stop

Step 2: call Amcam_put_Size or Amcam_put_eSize to set the new resolution

Step 3: call Amcam_StartPullModeWithWndMsg or Amcam_StartPullModeWithCallback or Amcam_StartPushModeV3 to restart

- ## Amcam_Pause

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

Remarks: pause/continue camera instance.

- ## Amcam_Snap, Amcam_SnapN

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned nResolutionIndex: resolution index.

unsigned nNum: the number to be snapped.

Remarks: snap 'still' image, please see here. When snap successfully:

a) If we use Pull Mode, it will be notified by AMCAM_EVENT_STILLIMAGE.

b) If we use Push Mode, the image will be returned by callback function PAMCAM_DATA_CALLBACK_V3 with the parameter int bSnap is TRUE.

Most cameras can snap still image with different resolutions under continuous preview. For example, UCMOS03100KPA's previewed resolution is 1024*768, if we call Amcam_Snap(h, 0), we get so called "still image" with 2048*1536 resolution.
Some cameras hasn't this ability, so nResolutionIndex must be equal the preview resolution which is set by Amcam_put_Size, or Amcam_put_eSize. Whether it supports "still snap" or not, see "still" domain in AmcamModelV2.

Amcam_Snap(h, index) == Amcam_SnapN(h, index, 1)

- ## Amcam_Trigger

Return value: HRESULT type means success or failure.

Parameters:

HAmcam h: camera instance handle

unsigned short nNumber: 0xffff(trigger continuously), 0(stop / cancel trigger), others(number of images to be triggered)

Remarks: in trigger mode, call this function to trigger an image:

a) If we use Pull Mode, it will be notified by AMCAM_EVENT_IMAGE.

b) If we use Push Mode, the image will be returned by callback function PAMCAM_DATA_CALLBACK_V3 with the parameter int bSnap is FALSE.

- ## Amcam_put_Size, Amcam_get_Size, Amcam_put_eSize, Amcam_get_eSize

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned nResolutionIndex: current/present resolution index

int nWidth, int nHeight: width and height of current resolution index

Remarks: set/get current resolution

Set resolution before running Amcam_StartPullModeWithWndMsg or Amcam_StartPullModeWithCallback or Amcam_StartPushModeV3
There are two ways to set current resolution: one is by resolution index, the other by width/height. Both ways are equivalent. For example, UCMOS03100KPA supports the following three kinds of resolution:
  Index 0: 2048, 1536
  Index 1: 1024, 768
  Index 2: 680, 510
So Amcam_put_Size(h, 1024, 768) is as effective as Amcam_put_eSize(h, 1)

- ## Amcam_put_Roi, Amcam_get_Roi

Return value: HRESULT type means success or failure.

Parameters:

HAmcam h: camera instance handle

unsigned xOffset: x offset, must be even number

unsigned yOffset: y offset, must be even number

unsigned xWidth: width, must be even number and must not be less than 16

unsigned yHeight: height, must be even number and must not be less than 16

Remarks: set/get the ROI. Amcam_put_Roi(h, 0, 0, 0, 0) means to clear the ROI and restore the original size.

Important: It is forbidden to call Amcam_put_Roi in the callback context of PAMCAM_EVENT_CALLBACK and PAMCAM_DATA_CALLBACK_V3, the return value is E_WRONG_THREAD.

Pay attention to that the coordinate is always relative to the original resolution, see here.

The second resolution of UHCCD03100KPB, UHCCD05000KPA, UHCCD05100KPA don't support ROI, so the return value is E_NOTIMPL.

- ## Amcam_get_ResolutionNumber, Amcam_get_Resolution

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned nResolutionIndex: resolution index

int* pWidth, int* pHeight: width/height

Remarks: Amcam_get_ResolutionNumber means the number of resolution supported. Take UCMOS03100KPA as an example, if we call the function Amcam_get_ResolutionNumber and get "3", which means it can support three kinds of resolution. Amcam_get_Resolution gets the width/height of each kind of resolution.

These parameters have also been contained in AmcamModelV2.

- ## Amcam_get_RawFormat

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned* nFourCC: raw format, see the table below

unsigned* bitdepth: bit depth, such as 8, 10, 12, 14, 16

```
#ifndef MAKEFOURCC
#define MAKEFOURCC(a, b, c, d) ((unsigned)(unsigned char)(a) | ((unsigned)(unsigned char)(b) << 8) | ((unsigned)(unsigned char)(c) << 16) | ((unsigned)(unsigned char)(d) << 24))
#endif
```

MAKEFOURCC('G', 'B', 'R', 'G')	GBGBGB... RGRGRG... GBGBGB... RGRGRG... ... see here
MAKEFOURCC('R', 'G', 'G', 'B')	RGRGRG... GBGBGB... RGRGRG... GBGBGB... ...
MAKEFOURCC('B', 'G', 'G', 'R')	BGBGBG... GRGRGR... BGBGBG... GRGRGR... ...
MAKEFOURCC('G', 'R', 'B', 'G')	GRGRGR... BGBGBG... GRGRGR... BGBGBG... ...
MAKEFOURCC('V', 'U', 'Y', 'Y')	YUV4:2:2, please see: http://www.fourcc.org
MAKEFOURCC('U', 'Y', 'V', 'Y')	YUV4:2:2
MAKEFOURCC('Y', 'Y', 'Y', 'Y')	Black / White camera

- ## Amcam_put_Option, Amcam_get_Option

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned iOption: see the table

int iValue: see the table

Option	Description	Default	The val Amcan or Amcan

			or Amc **The va** **fly?**
AMCAM_OPTION_RAW	0 means RGB mode. 1 means RAW mode, read the CMOS or CCD raw data.	0	**No** (The re E_UNE option a
AMCAM_OPTION_BITDEPTH	Some cameras support the bit depth which is more than 8 such as 10, 12, 14, 16. 0 = use 8 bits depth. 1 = use the maximum bits depth of this camera.	NA	Yes. But not frequen is best t
AMCAM_OPTION_FAN	Some cameras support the cooling fan. 0 = turn off the cooling fan [1, max] = fan speed	NA	Yes
AMCAM_OPTION_TEC	Some cameras support to turn on or off the thermoelectric cooler. 0 = turn off the thermoelectric cooler 1 = turn on the thermoelectric cooler	1	Yes
AMCAM_OPTION_TRIGGER	0 = video mode 1 = software or simulated trigger mode 2 = external trigger mode 3 = external + software trigger	0	Yes
AMCAM_OPTION_RGB	0 = RGB24 1 = enable RGB48 format when bitdepth > 8 2 = RGB32 3 = 8 bits gray (only for mono camera) 4 = 16 bits gray (only for mono camera and bitdepth > 8)	0	**No** (The re E_UNE option a
AMCAM_OPTION_BYTEORDER	Byte order: 1: BGR 0: RGB	Win: 1 Linux/MacOS/Android: 0	Yes. But not frequen is best t
AMCAM_OPTION_UPSIDE_DOWN	Upside down: 1: yes 0: no Please distinguish it from Amcam_put_VFlip, which requires the CPU to perform data moving work on each frame of data	Win: 1 Linux/MacOS/Android: 0	Yes
AMCAM_OPTION_TECTARGET	get or set the target temperature of the thermoelectric cooler, in 0.1°C. For example, 125 means 12.5°C, -35 means -3.5°C	NA	Yes
AMCAM_OPTION_AUTOEXP_POLICY	Auto Exposure Policy: 0: Exposure Only 1: Exposure Preferred 2: Gain Only 3: Gain Preferred	1	Yes
AMCAM_OPTION_AUTOEXP_THRESHOLD	threshold of auto exposure, range: [2~15]	5	Yes
AMCAM_OPTION_FRAMERATE	limit the frame rate, range=[0, 63]. frame rate control is disabled automatically in trigger mode.	0 (means no limit)	**No** (The re E_UNE option a
AMCAM_OPTION_BLACKLEVEL	Black Level Always return E_NOTIMPL for camera that don't support black level.	0	Yes
AMCAM_OPTION_MULTITHREAD	multithread image processing	1	**No** (The re E_UNE option a
AMCAM_OPTION_BINNING	digital binning: 0x01 (no binning) 0x02 (add, 2*2) 0x03 (add, 3*3) 0x04 (add, 4*4) 0x05 (add, 5*5) 0x06 (add, 6*6) 0x07 (add, 7*7) 0x08 (add, 8*8) 0x82 (average, 2*2) 0x83 (average, 3*3) 0x84 (average, 4*4) 0x85 (average, 5*5) 0x86 (average, 6*6) 0x87 (average, 7*7) 0x88 (average, 8*8) The final image size is rounded down to an even number, such as 640/3 to get 212	1	Yes
AMCAM_OPTION_ROTATE	rotate clockwise: 0, 90, 180, 270	0	Yes
AMCAM_OPTION_CG	Conversion Gain: 0: LCG 1: HCG 2: HDR	NA	Yes
AMCAM_OPTION_PIXEL_FORMAT	pixel format	NA	Yes. But not

			frequen is best t
AMCAM_OPTION_DDR_DEPTH	the number of the frames that DDR can cache: 1: DDR cache only one frame 0: Auto: ->>one for video mode when auto exposure is enabled ->>full capacity for others -1: DDR can cache frames to full capacity	0	Yes
AMCAM_OPTION_FFC	Flat Field Correction: set: 0: disable 1: enable -1: reset (0xff000000 | n): set the average number to n, [1~255] get: (val & 0xff): 0 ->> disable, 1 ->> enable, 2 ->> inited ((val & 0xff00) >> 8): sequence ((val & 0xff0000) >> 8): average number	0	Yes
AMCAM_OPTION_DFC	Dark Field Correction: set: 0: disable 1: enable -1: reset (0xff000000 | n): set the average number to n, [1~255] get: (val & 0xff): 0 ->> disable, 1 ->> enable, 2 ->> inited ((val & 0xff00) >> 8): sequence ((val & 0xff0000) >> 8): average number	0	Yes
AMCAM_OPTION_SHARPENING	Sharpening, (threshold << 24) | (radius << 16) | strength) strength: [0, 500], default: 0 (disable) radius: [1, 10] threshold: [0, 255]	0	Yes
AMCAM_OPTION_FACTORY	restore the factory settings Only put	Always 0	Yes
AMCAM_OPTION_TEC_VOLTAGE	get the current TEC voltage in 0.1V, 59 mean 5.9V; Please do not get this value too frequently, the recommended interval is 2 seconds or more Only get	NA	NA
AMCAM_OPTION_TEC_VOLTAGE_MAX	get the TEC maximum voltage in 0.1V Only get	NA	NA
AMCAM_OPTION_POWER	get power consumption, unit: milliwatt Only get	NA	NA
AMCAM_OPTION_GLOBAL_RESET_MODE	global reset mode	0	Yes
AMCAM_OPTION_DEVICE_RESET	reset usb device, simulate a replug Only put	NA	NA
AMCAM_OPTION_AFPOSITION	auto focus sensor board positon	NA	Yes
AMCAM_OPTION_AFMODE	auto focus mode: 0: manul focus 1: auto focus 2: once focus 3: conjugate calibration	NA	Yes
AMCAM_OPTION_AFZONE	auto focus zone	NA	Yes
AMCAM_OPTION_AFFEEDBACK	auto focus information feedback: 0: unknown 1: focused 2: focusing 3: defocus	NA	Yes

	4: up 5: down Only get		
AMCAM_OPTION_TESTPATTERN	test pattern: 0: TestPattern Off 3: monochrome diagonal stripes 5: monochrome vertical stripes 7: monochrome horizontal stripes 9: chromatic diagonal stripes Only put	0	Yes
AMCAM_OPTION_NOFRAME_TIMEOUT	Report error if cannot grab frame in maximum exposure time. 1 = enable this feature; 0 = disable this feature.	0	Yes
AMCAM_OPTION_NOPACKET_TIMEOUT	Report event if cannot grab packet in the set time. 0 = disable this feature positive value = timeout milliseconds	0	Yes
AMCAM_OPTION_MAX_PRECISE_FRAMERATE	get the precise frame maximum value in 0.1fps, such as 115 means 11.5fps the maximum value depends the resolution/bitdepth/ROI, etc E_NOTIMPL means not supported	NA	NA
AMCAM_OPTION_MIN_PRECISE_FRAMERATE	get the precise frame minimum value in 0.1fps, such as 15 means 1.5fps the minimum value depends the resolution/bitdepth/ROI, etc E_NOTIMPL means not supported	NA	NA
AMCAM_OPTION_PRECISE_FRAMERATE	in 0.1fps, such as 115 means 11.5fps. range:[1~maximum]	90% of the maximum	Yes
AMCAM_OPTION_BANDWIDTH	bandwidth, [1-100]%	90%	Yes
AMCAM_OPTION_RELOAD	Reload the last frame in the trigger mode get return value S_OK means supporting this feature, E_NOTIMPL means not supported	NA	Yes
AMCAM_OPTION_CALLBACK_THREAD	dedicated thread for callback, only available in pull mode	0	No
AMCAM_OPTION_FRONTEND_DEQUE_LENGTH or AMCAM_OPTION_FRAME_DEQUE_LENGTH	frontend frame deque length, range: [2, 1024]	3	No
AMCAM_OPTION_BACKEND_DEQUE_LENGTH	backend frame deque length (Only available in pull mode), range: [2, 1024]	3	No
AMCAM_OPTION_SEQUENCER_ONOFF	sequencer trigger: on/off	0	Yes
AMCAM_OPTION_SEQUENCER_NUMBER	sequencer trigger: number, range = [1, 255]	NA	Yes
AMCAM_OPTION_SEQUENCER_EXPOTIME	sequencer trigger: exposure time iOption = AMCAM_OPTION_SEQUENCER_EXPOTIME | index iValue = exposure time For example, to set the exposure time of the third group to 50ms, call Amcam_put_Option(AMCAM_OPTION_SEQUENCER_EXPOTIME | 3, 50000)	NA	Yes
AMCAM_OPTION_SEQUENCER_EXPOGAIN	sequencer trigger: exposure gain iOption = AMCAM_OPTION_SEQUENCER_EXPOGAIN | index iValue = gain	NA	Yes
AMCAM_OPTION_DENOISE	Denoise strength range: [0, 100], 0 means disable	0	Yes
AMCAM_OPTION_HEAT_MAX	get maximum level: heat to prevent fogging up [0, max], 0 means off	NA	NA
AMCAM_OPTION_HEAT	heat to prevent fogging up	max level	Yes
AMCAM_OPTION_LOW_NOISE	low noise mode (Higher signal noise ratio, lower frame rate)	0	Yes
AMCAM_OPTION_DEFECT_PIXEL	Defect Pixel Correction 0 => disable, 1 => enable	enable (1)	Yes
AMCAM_OPTION_THREAD_PRIORITY	set the priority of the internal thread which grab data from the usb device. Win: 0 = THREAD_PRIORITY_NORMAL; 1 = THREAD_PRIORITY_ABOVE_NORMAL; 2 = THREAD_PRIORITY_HIGHEST; 3 = THREAD_PRIORITY_TIME_CRITICAL; Please refer to SetThreadPriority Linux & macOS: The high 16 bits for the scheduling policy, and the low 16 bits for the priority. Please refer to pthread_setschedparam	Win: 1 Linux / macOS: NA	Yes
AMCAM_OPTION_LINEAR	0 = turn off the builtin linear tone mapping 1 = turn on the builtin linear tone mapping	1	Yes
AMCAM_OPTION_CURVE	0 = turn off the builtin curve tone mapping 1 = turn on the builtin polynomial curve tone mapping 2 = turn on the builtin logarithmic curve tone mapping	2	Yes
AMCAM_OPTION_COLORMATIX	0 = turn off the builtin color matrix 1 = turn on the builtin color matrix	1	Yes

AMCAM_OPTION_WBGAIN	0 = turn off the builtin white balance gain 1 = turn on the builtin white balance gain	1	Yes
AMCAM_OPTION_DEMOSAIC	Demosaic mothod for both video and still image: (Please refrence https://en.wikipedia.org/wiki/Demosaicing) 0 = BILINEAR 1 = VNG(Variable Number of Gradients) 2 = PPG(Patterned Pixel Grouping) 3 = AHD(Adaptive Homogeneity Directed) 4 = EA(Edge Aware) Always return E_NOTIMPL for monochromatic camera.	0	Yes
AMCAM_OPTION_DEMOSAIC_VIDEO	Demosaic mothod for video	0	Yes
AMCAM_OPTION_DEMOSAIC_STILL	Demosaic mothod for still image	0	Yes
AMCAM_OPTION_OPEN_USB_ERRORCODE	open usb error code. get only	NA	NA
AMCAM_OPTION_FLUSH	1: hard flush, discard frames cached by camera DDR (if any) 2: soft flush, discard frames cached by amcam.dll (if any) 3: both Amcam_Flush means 'both'	NA	NA
AMCAM_OPTION_NUMBER_DROP_FRAME	get the number of frames that have been grabbed from the USB but dropped by the software	NA	NA
AMCAM_OPTION_DUMP_CFG	explicitly dump configuration to ini, json, or EEPROM. when camera is closed, it will dump configuration automatically	NA	NA
AMCAM_OPTION_LINUX_USB_ZEROCOPY	global option for linux platform: enable or disable usb zerocopy (helps to reduce memory copy and improve efficiency. Requires kernel version >= 4.6 and hardware platform support). if the image is wrong, this indicates that the hardware platform does not support this feature, please disable it when the program starts: Amcam_put_Option((this is a global option, the camera handle parameter is not required, use nullptr), AMCAM_OPTION_LINUX_USB_ZEROCOPY, 0)	disable(0): android or arm32 enable(1): others	No (Must b (Amcar

Important:

a. Some options cannot be changed after the camera is started. This is to say the option cannot be changed on-the-fly.

b. It is forbidden to call Amcam_put_Option with AMCAM_OPTION_TRIGGER, AMCAM_OPTION_BITDEPTH, AMCAM_OPTION_PIXEL_FORMAT, AMCAM_OPTION_BINNING, AMCAM_OPTION_ROTATE in the callback context of PAMCAM_EVENT_CALLBACK and PAMCAM_DATA_CALLBACK_V3, the return value is E_WRONG_THREAD.

c. The second resolution of UHCCD03100KPB, UHCCD05000KPA, UHCCD05100KPA don't support RAW mode.

- ## Amcam_put_RealTime, Amcam_get_RealTime

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

int val:
0: stop grab frame when frame buffer deque is full, until the frames in the queue are pulled away and the queue is not full
1: realtime
use minimum frame buffer. When new frame arrive, drop all the pending frame regardless of whether the frame buffer is full
If DDR present, also limit the DDR frame buffer to only one frame. 2: soft realtime
Drop the oldest frame when the queue is full and then enqueue the new frame

Remarks: If you set RealTime mode as 1, then you get shorter frame delay but lower frame rate which damages fluency. The default value is 0.

- ## Amcam_get_AutoExpoEnable, Amcam_put_AutoExpoEnable, Amcam_get_AutoExpoTarget, Amcam_put_AutoExpoTarget, Amcam_put_MaxAutoExpoTimeAGain, Amcam_get_MaxAutoExpoTimeAGain, Amcam_put_MinAutoExpoTimeAGain, Amcam_get_MinAutoExpoTimeAGain

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

int bAutoExposure: TRUE or FALSE

unsigned short Target: auto-exposure target

unsigned maxTime, unsigned short maxAGain: the maximum time and maximum gain of auto exposure. The default values are 350ms and 500.

unsigned minTime, unsigned short minAGain: the minimal time and minimal gain of auto exposure. The default values are 0 and 100.

Remarks: If auto exposure is enabled, the exposure time and gain are controlled by software to make the average brightness of the target rectangle as close as possible to the auto exposure target. The auto exposure target value is the target brightness of the target rectangle (see Amcam_put_AEAuxRect,

Amcam_get_AEAuxRect).

- ## Amcam_get_ExpoTime, Amcam_put_ExpoTime, Amcam_get_ExpTimeRange, Amcam_get_RealExpoTime

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  unsigned Time: exposure time, unit: microsecond

  unsigned* nMin, unsigned* nMax, unsigned* nDef: the minimum, maximum and default value of exposure time.

  Remarks: exposure time related. Amcam_get_RealExpoTime get the real exposure time based on 50HZ/60HZ.

- ## Amcam_get_ExpoAGain, Amcam_put_ExpoAGain, Amcam_get_ExpoAGainRange

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  unsigned short AGain: gain, shown in percentage, eg, 200 means the gain is 200%

  unsigned short* nMin, unsigned short* nMax, unsigned short* nDef: the minimum, maximum and default value of gain.

  Remarks: gain related.

- ## Amcam_put_Hue, Amcam_get_Hue, Amcam_put_Saturation, Amcam_get_Saturation, Amcam_put_Brightness, Amcam_get_Brightness, Amcam_get_Contrast, Amcam_put_Contrast, Amcam_get_Gamma, Amcam_put_Gamma

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  Remarks: set or get hue, saturation, brightness, contrast and gamma.

- ## Amcam_get_Chrome, Amcam_put_Chrome

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  int bChrome: TRUE or FALSE

  Remarks: color or gray mode

- ## Amcam_get_VFlip, Amcam_put_VFlip, Amcam_get_HFlip, Amcam_put_HFlip

  Return value: HRESULT type means success or failure

  Parameters: HAmcam h: camera instance handle

  Remarks: vertical/horizontal flip.

- ## Amcam_put_Speed, Amcam_get_Speed, Amcam_get_MaxSpeed

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  unsigned short nSpeed: frame rate level

  Remarks: the minimum frame rate level is "0", the maximum one can be achieved through Function "Amcam_get_MaxSpeed" which equals to maxspeed in AmcamModelV2.

- ## Amcam_put_HZ, Amcam_get_HZ

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

int nHZ: 0: 60Hz alternating current, 1: 50Hz alternating current, 2: direct current

Remarks: set the light source power frequency

- ## Amcam_get_Temperature, Amcam_put_Temperature

Return value: HRESULT type means success or failure, E_NOTIMPL means not supporting get or set the temperature

Parameters:

HAmcam h: camera instance handle

short nTemperature: in 0.1℃ (32 means 3.2℃, -35 means -3.5℃).

Remarks: get the temperature of the sensor. see AMCAM_FLAG_GETTEMPERATURE.

set the target temperature of the sensor, equivalent to Amcam_put_Option(, AMCAM_OPTION_TECTARGET,).

- ## Amcam_put_Mode, Amcam_get_Mode

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

int bSkip: Bin mode or Skip mode.

Remarks: set Bin mode or Skip mode. Cameras with higher resolution can support two sampling modes, one is Bin mode (Neighborhood averaging), the other is Skip (sampling extraction). In comparison, the former is with better image effect but decreasing frame rate while the latter is just the reverse.

- ## Amcam_put_TempTint, Amcam_get_TempTint

Return value: HRESULT type means success or failure. Works int Temp/Tint mode. Does not work int RGB Gain mode, E_NOTIMPL will be return.

Parameters:

HAmcam h: camera instance handle

int nTemp, int nTint: color temperature and Tint

Remarks: set/get the color temperature and Tint parameters of white balance (Temp/Tint Mode, please see here).

- ## Amcam_AwbOnce

Return value: HRESULT type means success or failure. Works int Temp/Tint mode. Does not work int RGB Gain mode, E_NOTIMPL will be return.

Parameters:

HAmcam h: camera instance handle

PIAMCAM_TEMPTINT_CALLBACK fnTTProc, void* pTTCtx: callback function and callback context when the automatic white balance completes.

Remarks: Call this function to perform one "auto white balance" in Temp/Tint Mode. When the "auto white balance" completes, AMCAM_EVENT_TEMPTINT event notify the application (In Pull Mode) and callback happens. In pull mode, this callback is useless, set it to NULL.

- ## Amcam_put_WhiteBalanceGain, Amcam_get_WhiteBalanceGain

Return value: HRESULT type means success or failure. Works int RGB Gain mode. Does not work int Temp/Tint Gain mode, E_NOTIMPL will be return.

Parameters:

HAmcam h: camera instance handle

int aGain[3]: RGB Gain

Remarks: set/get the RGB gain parameters of white balance (RGB Gain Mode, please see here).

- ## Amcam_AwbInit

Return value: HRESULT type means success or failure. Works int RGB Gain mode. Does not work int Temp/Tint mode, E_NOTIMPL will be return.

Parameters:

HAmcam h: camera instance handle

PIAMCAM_WHITEBALANCE_CALLBACK fnWBProc, void* pWBCtx: callback function and callback context when the automatic white balance completes.

Remarks: Call this function to perform one "auto white balance" in RGB Gain Mode. When the "auto white balance" completes, AMCAM_EVENT_WBGAIN event notify the application (In Pull Mode) and callback happens. In pull mode, this callback is useless, set it to NULL.

- ## Amcam_AbbOnce

Return value: HRESULT type means success or failure.

Parameters:

HAmcam h: camera instance handle

PIAMCAM_BLACKBALANCE_CALLBACK fnBBProc, void* pBBCtx: callback function and callback context when the automatic black balance completes.

Remarks: Call this function to perform one "auto black balance". When the "auto black balance" completes, AMCAM_EVENT_BLACK event notify the application (In Pull Mode) and callback happens. In pull mode, this callback is useless, set it to NULL.

- ## Amcam_put_BlackBalance, Amcam_get_BlackBalance

Return value: HRESULT type means success or failure.

Parameters:

HAmcam h: camera instance handle

unsigned short aSub[3]: RGB Offset

Remarks: set/get the RGB offset parameters of black balance.

- ## Amcam_put_AWBAuxRect, Amcam_get_AWBAuxRect, Amcam_put_AEAuxRect, Amcam_get_AEAuxRect, Amcam_put_ABBAuxRect, Amcam_get_ABBAuxRect

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

Remarks: set/get the rectangle of automatic white balance and auto-exposure and automatic black balance. The default Rectangle is in the center of the image, its width is 20% image with, its height is 20% image height.

Pay attention to that the coordinate is always relative to the original resolution, see here.

- ## Amcam_get_MonoMode

Return value: S_OK means mono mode, S_FALSE means color mode

Parameters:

Amcam h: camera instance handle

Remarks: gray camera or not, find the flag in AmcamModelV2: AMCAM_FLAG_MONO

- ## Amcam_get_MaxBitDepth

Return value: the maximum bit depth this camera supports.

Parameters:

HAmcam h: camera instance handle

Remarks: Some cameras support the bit depth which is more than 8 such as 10, 12, 14, 16.

- ## Amcam_get_StillResolutionNumber, Amcam_get_StillResolution

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned nResolutionIndex: resolution index

int* pWidth, int* pHeight: width/height

Remarks: Amcam_get_StillResolutionNumber means the number of supported still resolution. Take UCMOS03100KPA as an example, if we call the function Amcam_get_StillResolutionNumber and get "3", which means it can support three kinds of resolution. If it doesn't support "still snap", then we get "0". Amcam_get_Resolution gs the width/height of each kind of resolution.

- ## Amcam_get_SerialNumber, Amcam_get_FwVersion, Amcam_get_HwVersion, Amcam_get_ProductionDate, Amcam_get_Revision

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

char sn[32]: buffer to the serial number, such as: TP110826145730ABCD1234FEDC56787

char fwver[16]: buffer to the firmware version, such as: 3.2.1.20140922

char hwver[16]: buffer to the hardware version, such as: 3.12

char pdate[10]: buffer to the production date, such as: 20150327

unsigned short pRevision: revision

Remarks: each camera has a unique serial number with 31 letters.

- ## Amcam_get_PixelSize

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned nResolutionIndex: resolution index

float* x, float* y: physical pixel size(μm)

- ## Amcam_put_LEDState

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned short iLed: the index of LED light

unsigned short iState: LED status, 1 -> Ever bright; 2 -> Flashing; other -> Off

unsigned short iPeriod: Flashing Period (>= 500ms)

Remarks: One or more LED lights installed on some camera. This function controls the status of these lights.

- ## Amcam_read_EEPROM, Amcam_write_EEPROM

Return value: HRESULT type means failure or byte(s) transferred

Parameters:

HAmcam h: camera instance handle

unsigned addr: EEPROM address

const unsigned char* pBuffer: data buffer to be written

unsigned char* pBuffer: read EEPROM to buffer

unsigned nBufferLen: buffer length

Remarks: In some cameras, EEPROM is available for read and write. If failed to read or write, a negative HRESULT error code will be return, when success, the bytes number has been read or written will be return.

- ## Amcam_IoControl

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned index: IO port index

unsigned nType: type of control

int outVal: output control value

int* inVal: input control value

AMCAM_IOCONTROLTYPE_GET_SUPPORTEDMODE	get the supported mode: 0x01->>Input 0x02->>Output (0x01 | 0x02)->>support both Input and Output
AMCAM_IOCONTROLTYPE_GET_GPIODIR	0x00->>Input, 0x01->>Output
AMCAM_IOCONTROLTYPE_SET_GPIODIR	
AMCAM_IOCONTROLTYPE_GET_FORMAT	format: 0x00->>not connected 0x01->>Tri-state 0x02->>TTL 0x03->>LVDS 0x04->>RS422 0x05->>Opto-coupled
AMCAM_IOCONTROLTYPE_SET_FORMAT	
AMCAM_IOCONTROLTYPE_GET_OUTPUTINVERTER	boolean, only support output signal
AMCAM_IOCONTROLTYPE_SET_OUTPUTINVERTER	
AMCAM_IOCONTROLTYPE_GET_INPUTACTIVATION	0x00->>Positive, 0x01->>Negative
AMCAM_IOCONTROLTYPE_SET_INPUTACTIVATION	
AMCAM_IOCONTROLTYPE_GET_DEBOUNCERTIME	debouncer time in microseconds, [0, 20000]
AMCAM_IOCONTROLTYPE_SET_DEBOUNCERTIME	
AMCAM_IOCONTROLTYPE_GET_TRIGGERSOURCE	0x00->> Opto-isolated input 0x01->> GPIO0 0x02->> GPIO1 0x03->> Counter 0x04->> PWM 0x05->> Software
AMCAM_IOCONTROLTYPE_SET_TRIGGERSOURCE	
AMCAM_IOCONTROLTYPE_GET_TRIGGERDELAY	Trigger delay time in microseconds, [0, 5000000]
AMCAM_IOCONTROLTYPE_SET_TRIGGERDELAY	
AMCAM_IOCONTROLTYPE_GET_BURSTCOUNTER	Burst Counter: 1, 2, 3 ... 1023
AMCAM_IOCONTROLTYPE_SET_BURSTCOUNTER	
AMCAM_IOCONTROLTYPE_GET_COUNTERSOURCE	0x00->> Opto-isolated input 0x01->> GPIO0 0x02->> GPIO1
AMCAM_IOCONTROLTYPE_SET_COUNTERSOURCE	
AMCAM_IOCONTROLTYPE_GET_COUNTERVALUE	Counter Value: 1, 2, 3 ... 1023
AMCAM_IOCONTROLTYPE_SET_COUNTERVALUE	
AMCAM_IOCONTROLTYPE_SET_RESETCOUNTER	
AMCAM_IOCONTROLTYPE_GET_PWMSOURCE	0x00->> Opto-isolated input 0x01->> GPIO0 0x02->> GPIO1
AMCAM_IOCONTROLTYPE_SET_PWMSOURCE	
AMCAM_IOCONTROLTYPE_GET_OUTPUTMODE	0x00->> Frame Trigger Wait 0x01->> Exposure Active 0x02->> Strobe 0x03->> User output
AMCAM_IOCONTROLTYPE_SET_OUTPUTMODE	
AMCAM_IOCONTROLTYPE_GET_STROBEDELAYMODE	boolean, 0->> pre-delay, 1->> delay; compared to exposure active signal
AMCAM_IOCONTROLTYPE_SET_STROBEDELAYMODE	
AMCAM_IOCONTROLTYPE_GET_STROBEDELAYTIME	Strobe delay or pre-delay time in microseconds, [0, 5000000]
AMCAM_IOCONTROLTYPE_SET_STROBEDELAYTIME	
AMCAM_IOCONTROLTYPE_GET_STROBEDURATION	Strobe duration time in microseconds, [0, 5000000]
AMCAM_IOCONTROLTYPE_SET_STROBEDURATION	
AMCAM_IOCONTROLTYPE_GET_USERVALUE	bit0->> Opto-isolated output bit1->> GPIO0 output bit2->> GPIO1 output
AMCAM_IOCONTROLTYPE_SET_USERVALUE	
AMCAM_IOCONTROLTYPE_GET_UART_ENABLE	UART enable: 1-> on; 0-> off
AMCAM_IOCONTROLTYPE_SET_UART_ENABLE	
AMCAM_IOCONTROLTYPE_GET_UART_BAUDRATE	baud rate: 0-> 9600 1-> 19200 2-> 38400 3-> 57600 4-> 115200
AMCAM_IOCONTROLTYPE_SET_UART_BAUDRATE	
AMCAM_IOCONTROLTYPE_GET_UART_LINEMODE	Line Mode: 0->TX(GPIO_0)/RX(GPIO_1) 1->TX(GPIO_1)/RX(GPIO_0)

AMCAM_IOCONTROLTYPE_SET_UART_LINEMODE	

- ## Amcam_read_UART, Amcam_write_UART

Return value: HRESULT type means failure or byte(s) transferred

Parameters:

HAmcam h: camera instance handle

const unsigned char* pBuffer: data buffer to be written

unsigned char* pBuffer: read buffer

unsigned nBufferLen: buffer length

Remarks: If failed to read or write, a negative HRESULT error code will be return, when success, the bytes number has been read or written will be return.

- ## Amcam_FfcOnce

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

- ## Amcam_DfcOnce

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

- ## Amcam_LevelRangeAuto

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

Remarks: auto Level Range.

- ## Amcam_put_LevelRange, Amcam_get_LevelRange

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned short aLow[4], unsigned short aHigh[4]: Level Range of R, G, B, and Gray. RGB is only available for color image, and gray is only available for gray image.

Remarks: level range related.

- ## Amcam_put_LevelRangeV2, Amcam_get_LevelRangeV2

Return value: HRESULT type means success or failure

Parameters:

HAmcam h: camera instance handle

unsigned short mode:

AMCAM_LEVELRANGE_MANUAL	Manual mode
AMCAM_LEVELRANGE_ONCE	Once
AMCAM_LEVELRANGE_CONTINUE	Continue mode
AMCAM_LEVELRANGE_ROI	Update the ROI rectangle

RECT* pRoiRect: ROI rectangle

unsigned short aLow[4], unsigned short aHigh[4]: Level Range of R, G, B, and Gray. RGB is only available for color image, and gray is only available for gray image.

Remarks: level range related.

- ## Amcam_Update

  Return value: HRESULT type means success or failure

  Parameters:

  camId: camera ID

  filePath: ufw file full path

  pFun, pCtx: progress percent callback

  Remarks: firmware update. Please do not unplug the camera or lost power during the upgrade process, this is very very important. Once an unplugging or power outage occurs during the upgrade process, the camera will no longer be available and can only be returned to the factory for repair.

- ## Amcam_GetHistogram

  Return value: HRESULT type means success or failure

  Parameters:

  HAmcam h: camera instance handle

  PIAMCAM_HISTOGRAM_CALLBACK fnHistogramProc, void* pHistogramCtx: callback function and callback context of histogram data.

  Remarks: get histogram data.

- ## Ranges and default value of some parameters

Parameters			Range	Default value	Get	Set	Auto
Auto Exposure Target			16~220	120	Amcam_get_AutoExpoTarget	Amcam_put_AutoExpoTarget	
White Balance	Temp/Tint Mode	Color Temperature	2000~15000	6503	Amcam_get_TempTint	Amcam_put_TempTint	Amcam_AwbOnce
		Tint	200~2500	1000			
	RGB Gain Mode	Red Gain	-127~127	0	Amcam_get_WhiteBalanceGain	Amcam_put_WhiteBalanceGain	Amcam_AwbInit
		Green Gain					
		Blue Gain					
LevelRange		Software	0~255	Low = 0 High = 255	Amcam_get_LevelRange	Amcam_put_LevelRange	Amcam_LevelRangeAu
		Hardware			Amcam_get_LevelRangeV2	Amcam_put_LevelRangeV2	
Contrast			-100~100	0	Amcam_get_Contrast	Amcam_put_Contrast	
Hue			-180~180	0	Amcam_get_Hue	Amcam_put_Hue	
Saturation			0~255	128	Amcam_get_Saturation	Amcam_put_Saturation	
Brightness			-64~64	0	Amcam_get_Brightness	Amcam_put_Brightness	
Gamma			20~180	100	Amcam_get_Gamma	Amcam_put_Gamma	
Black Level			0~31 (bit depth=8) 0~31 * 4 (bit depth=10) 0~31 * 16 (bit depth=12) 0~31 * 64 (bit depth=14) 0~31 * 256 (bit depth=16)		Amcam_get_Option	Amcam_put_Option	
Auto Exposure	Max	Exposure Time		350ms	Amcam_get_MaxAutoExpoTimeAGain	Amcam_put_MaxAutoExpoTimeAGain	
		Gain		500			
	Min	Exposure Time		0	Amcam_get_MinAutoExpoTimeAGain	Amcam_put_MinAutoExpoTimeAGain	
		Gain		100			
TEC Target			-300 ~ 300 -30℃ ~ 30℃	0	AMCAM_OPTION_TECTARGET	AMCAM_OPTION_TECTARGET	

---

5. .NET and C# and VB.NET

Amcam does support .NET development environment (C# and VB.NET).

amcam.cs use P/Invoke to call into amcam.dll. Copy amcam.cs to your C# project, please reference demowinformcs1, demowinformcs2 and demowinformcs3 in the samples directory.

Please pay attation to that **the object of the C# class Amcam. Amcam must be obtained by static mothod Open or OpenByIndex, it cannot be obtained by obj = new Amcam** (The constructor is private on purpose).

Most properties and methods of the Amcam class P/Invoke into the corresponding Amcam_xxxx functions of amcam.dll/so. So, the descriptions of the Amcam_xxxx function are also applicable for the corresponding C# properties or methods. For example, the C# Snap method call the function Amcam_Snap, the descriptions of the Amcam_Snap function is applicable for C# Snap method:

```
[DllImport("amcam.dll", ExactSpelling = true, CallingConvention = CallingConvention.StdCall)]
private static extern int Amcam_Snap(SafeHAmcamHandle h, uint nResolutionIndex);

public bool Snap(uint nResolutionIndex)
{
    if (_handle == null || _handle.IsInvalid || _handle.IsClosed)
        return false;
    return (Amcam_Snap(_handle, nResolutionIndex) >= 0);
}
```

VB.NET is similar with C#, not otherwise specified.

---

6. Python

Amcam does support Python (version 3.0 or above), please import amcam to use amcam.py and reference the sample code simplest.py and qt.py.

Please pay attation to that **the object of the python class amcam.Amcam must be obtained by classmethod Open or OpenByIndex, it cannot be obtained by obj = amcam.Amcam()**

Most methods of the Amcam class use ctypes to call into the corresponding Amcam_xxxx functions of amcam.dll/so/dylib. So, the descriptions of the Amcam_xxxx function are also applicable for the corresponding python methods.

Please reference __errcheck in amcam.py, the original HRESULT return code is mapped to HRESULTException exception (in win32 it's inherited from OSError).

Please make sure the amcam dll/so/dylib library file is in the same directory with amcam.py.

7. Java

Amcam does support Java, amcam.java use JNA (https://github.com/java-native-access/jna) to call into amcam.dll/so/dylib. Copy amcam.java to your java project, please reference the sample code simplest.java (Console), javafx.java, swing.java.

Please pay attation to that **the object of the java class amcam must be obtained by static method Open or OpenByIndex, it cannot be obtained by obj = new amcam()**(The constructor is private on purpose).

Most methods of the amcam class use JNA to call into the corresponding Amcam_xxxx functions of amcam.dll/so/dylib. So, the descriptions of the Amcam_xxxx function are also applicable for the corresponding java methods.

Please reference errcheck in amcam.java, the original HRESULT return code is mapped to HRESULTException exception.

Remark: (1) Download jna-*.jar from github; (2) Make sure amcam.dll/so/dylib is placed in the correct directory.

---

8. Changelog

---

v50: x86 and x64 SIMD optimized Demosaic in Windows: EA, VNG, AHD. Please see here
    frontend and backend deque length: AMCAM_OPTION_FRONTEND_DEQUE_LENGTH, AMCAM_OPTION_BACKEND_DEQUE_LENGTH

v49: Add support to save & load configuration. Please see here

v48: hardware event. Please see here

v47: hardware level range. Please see here and here

v46: Add support denose. Please see here

v45: Add sequencer trigger, UART, mix trigger (external and software trigger both are enabled)

v44: Extend the realtime mode, Please see here
    Add AMCAM_OPTION_CALLBACK_THREAD and AMCAM_OPTION_FRAME_DEQUE_LENGTH

v43: Reload the last frame in the trigger mode. Please see AMCAM_OPTION_RELOAD

v42: Precise frame rate and bandwidth. Please see here and AMCAM_FLAG_PRECISE_FRAMERATE

v41: no packet timeout. Please see here

v40: Auto test tool, see samples\AutoTestTool

v39: Update C#/VB.NET/Java/Python

v38: Add support to byte order, change BGR/RGB. Please see here

v37: Add Android support
Add Amcam_StartPushModeV3 (Amcam_StartPushModeV2 and Amcam_StartPushMode are obsoleted)

v36: Add Java support. Please see here

v35: Add Python support. Please see here

v34: Auto Focus and Focus Motor

v33: extend AMCAM_OPTION_AGAIN to AMCAM_OPTION_AUTOEXP_POLICY, support more options. Please see here

v32: Addd support to Windows 10 on ARM and ARM64, both desktop and WinRT

v31: Add Amcam_deBayerV2, support RGB48 and RGB64

v30: Add AMCAM_FLAG_CGHDR

v29: Add AmcamFrameInfoV2, a group of functions (PullImageV2 and StartPushModeV2), some cameras support frame sequence number and timestamp. Please see here

v28: Add Amcam_read_Pipe, Amcam_write_Pipe, Amcam_feed_Pipe

v27: Add Amcam_SnapN, support to snap multiple images, please see here and democpp

v26: Add support to restore factory settings, AMCAM_OPTION_FACTORY

v25: Add sharpening, AMCAM_OPTION_SHARPENING

v24: Add support to Auto Exposure with the 50/60 HZ constraint

v23: Add support to Linux armhf, armel and arm64
Add FFC and DFC, please see here and here

v22: Add AMCAM_OPTION_DDR_DEPTH, please see here

v21: Add Amcam_IoControl

v20: Add Amcam_EnumV2, AmcamModelV2, AmcamDeviceV2; (Amcam_Enum, AmcamModel and AmcamDevice are obsoleted)
Add Pixel Format, see AMCAM_OPTION_PIXEL_FORMAT; (AMCAM_OPTION_PIXEL_FORMAT is the super set of AMCAM_OPTION_BITDEPTH)
Add Flat Field Correction

v19: Add Black Balance: please see here

v18: Add Amcam_get_Revision

v17: Add AMCAM_OPTION_ROTATE

v16: Add AMCAM_FLAG_DDR, use very large capacity DDR (Double Data Rate SDRAM) for frame buffer

v15: Add AMCAM_OPTION_BINNING

v14: Add support to WinRT / UWP (Universal Windows Platform) / Windows Store App

v13: support row pitch, please see Amcam_PullImageWithRowPitch and Amcam_PullStillImageWithRowPitch

v12: support RGB32, 8 bits Gray, 16 bits Gray, please see here

v11: black level: please see here

v10: demosaic method: please see here

v9: change the histogram data type from double to float

v8: support simulated trigger, please see here

v7: support RGB48 when bit depth > 8, please see here

v6: support trigger mode, please see here

v5: White Balance: Temp/Tint Mode vs RGB Gain Mode, please see here

v4: ROI (Region Of Interest) supported: please see here

v3: more bit depth supported: 10bits, 12bits, 14bits, 16bits

v2: support RAW format, please see here and here; support Linux and macOS

v1: initial release