VIYEE High-end Camera Application Development Manual

Version 3.1.3

I. SDK File List

iCam.dll
iCyUsb.dll
iGigE.dll
iImg.dll
iWinUsb.dll
msvcp110.dll
msvcr110.dll
Packet.dll
tbb.dll
winusb.dll
wpcap.dll

II. API Reference

A. Camera Control

DLL: iCam.dll

Header: iCam.h

Library: iCam.lib

1. iCamInit

Description: Detects and initializes connected cameras, and returns camera list.

Definition: void iCamInit(HANDLE* hCamList, int* pCamCount)

Parameters:

hCamList: Returns list of camera handles

pCamCount: Returns number of camera

Tips: iCamInit must be called before any other camera control API to ensure proper initialization of the camera.

2. iCamUninit

Description: Releases camera handle and all the resources allocated by iCamInit.

Definition: void iCamUninit()

3. iCamGetCamType

Description: Retrieves camera type.

Definition: int iCamGetCamType(HANDLE hCam)

Parameters: hCam: Camera handle

Return value:

0x00000 Unknown

0x10000 MU204C

0x10001 MU204M

0x10100 MU267C

0x10101 MU267M

0x10110 MU285C

0x10111 MU285M

0x10300 MU412C

0x10500 MU282C

0x20000 CF255M

0x20100 CF285AM

0x20104 CF285BC

0x20105 CF285BM

0x20108 CF285CM

0x20300 CF674M

0x20400 CF4022M

0x20600 CF413C

0x20610 CF694M

0x20800 CF8300AM

0x20804 CF8300BM

0x20900 CF814M

0x30100 SC035C

0x30110 SC120C
0x30200 SC230M
0x30600 SC600C
0x30601 SC600M
0x31200 SC1200C
0x32000 SC2000C
0x40100 AP120C
0x40101 AP120M
0x40110 AP130M
0x40300 AP300C
0x40500 AP500C
0x40501 AP500M
0x50100 MD285C
0x50101 MD285M
0x50300 MD674C
0x50301 MD674M
0x50600 MD694C
0x50601 MD694M
0x50900 MD814C
0x50901 MD814M
0x60100 SL130M
0x60200 SL200M
0x60500 SL500M
0x60800 SL8300M
0x80500 IE500C
0x80501 IE500M
0x80600 IE600C
0x81400 IE1400C
0xA0500 MH655C
0xA0600 MH694AC
0xA0601 MH694AM
0xA0604 MH694BC
0xB1600 SD1600AC
0xB1601 SD1600AM
0xB1604 SD1600BC

4. iCamGetFrameBufferSize
Description: Retrieves frame buffer size needed to allocate for current camera type.
Definition: int iCamGetFrameBufferSize(HANDLE hCam)
Parameters: hCam: Camera handle
Return value: Buffer size in bytes

5. iCamSetScene
Description: Sets scene.
Definition: bool iCamSetScene(HANDLE hCam, unsigned char scene)
Parameters:

hCam: Camera handle

scene: Scene, general by default

10: General

20: Microscopy

21: Enhanced Microscopy

30: Fluorescence

40: Metallography

Return value: Whether setting succeeds

6. iCamSetMode

Description: Sets camera readout mode. See Appendix A for supported mode of variant camera types.

Definition: void iCamGetFrameSize(HANDLE hCam, unsigned char mode)

Parameters:

hCam: Camera handle

mode: Readout mode, full pixel by default

10: Full pixel

20: 2*2 binning

21: 3*3 binning

22: 4*4 binning

7. iCamSetResolution**(Deprecated)**

Description: Sets resolution. See Appendix A for supported resolution of variant camera types.

Definition: bool iCamSetResolution(HANDLE hCam, int x, int y)

Parameters:

hCam: Camera handle

x: Horizontal resolution

y: Vertical resolution

Return value: Whether setting succeeds

8. iCamSetBinning**(Deprecated)**

Description: Sets binning mode. See Appendix A for supported binning mode of variant camera types.

Definition: bool iCamSetBinning(HANDLE hCam, unsigned char h, unsigned char v)

Parameters:

hCam: Camera handle

h: Horizontal binning

v: Vertical binning

Return value: Whether setting succeeds

9. iCamSetDepth**(Deprecated)**

Description: Sets color depth. See Appendix A for supported color depth of variant camera types.

Definition: bool iCamSetDepth(HANDLE hCam, unsigned char depth)

Parameters:

hCam: Camera handle

depth: Color depth

10: 8 bit

20: 16 bit

Return value: Whether setting succeeds

10. iCamSetReadoutSpeed**(Deprecated)**

Description: Sets readout speed.

Definition: bool iCamSetReadoutSpeed(HANDLE hCam, unsigned char readoutSpeed)

Parameters:

hCam: Camera handle

readoutSpeed: Readout speed

10: High speed

20: Medium speed

30: Low speed

Return value: Whether setting succeeds

Tips: Camera works at high readout speed by default. For AP series CMOS camera, reducing readout speed can solve the problem of read stalling caused by low USB performance on certain computers.

11. iCamSetUsbTraffic**(Deprecated)**

Description: Sets USB traffic limit. Supported camera type: AP series CMOS camera.

Definition: bool iCamSetUsbTraffic(HANDLE hCam, unsigned char traffic)

Parameters:

hCam: Camera handle

traffic: Traffic limit, range: 0 - 100, 100 by default which means highest traffic without limit

Return value: Whether setting succeeds

Tips: Setting appropriate USB traffic limit can solve the problem of read stalling caused by low USB performance on certain computers at the expense of less frame rate loss.

12. iCamSetAutoExposure

Description: Sets whether to enable auto exposure. Only works for live capture mode.

Definition: bool iCamSetAutoExposure(HANDLE hCam, bool ae)

Parameters:

hCam: Camera handle

ae: Whether to enable auto exposure

Return value: Whether setting succeeds

13. iCamSetAeCompensation

Description: Sets auto exposure compensation.

Definition: bool iCamSetAeCompensation(HANDLE hCam, int aeCompensation)

Parameters:

hCam: Camera handle

aeCompensation: Exposure compensation, range: -2 - 2, 0 for no compensation

Return value: Whether setting succeeds

14. iCamSetExposure

Description: Sets exposure. Unavailable when auto exposure is on.

Definition: bool iCamSetExposure(HANDLE hCam, double exp)

Parameters:

hCam: Camera handle

exp: Exposure, in milliseconds

Return value: Whether setting succeeds

15. iCamGetExposure

Description: Retrieves current exposure.

Definition: double iCamGetExposure(HANDLE hCam)

Parameters: hCam: Camera handle

Return value: Exposure, in milliseconds

16. iCamSetGain

Description: Sets gain.

Definition: bool iCamSetGain(HANDLE hCam, unsigned char gain)

Parameters:

hCam: Camera handle

gain: Gain, range: 0 - 100

Return value: Whether setting succeeds

Tips: Gain setting decides the ISO speed of camera. Higher gain brings higher ISO speed but introduces more noise. Choose appropriate gain setting according to the actual capture environment.

17. iCamSetLineNoiseRmv

Description: Sets whether to enable line noise removal. Supported camera type: AP130.

Definition: bool iCamSetLineNoiseRmv(HANDLE hCam, bool lnr)

Parameters:

hCam: Camera handle

lnr: Whether to enable line noise removal

Return value: Whether setting succeeds

18. iCamSetHighGainBoost

Description: Sets whether to enable high gain boost. Supported camera type: AP120.

Definition: bool iCamSetHighGainBoost(HANDLE hCam, bool hgb)

Parameters:

hCam: Camera handle

hgb: Whether to enable high gain boost

Return value: Whether setting succeeds

19. iCamBeginCapture

Description: Begins capture. If camera is already in capture progress, the call will fail.

Definition: bool iCamBeginCapture(HANDLE hCam, bool live)

Parameters:

hCam: Camera handle

live: live capture mode

true: live capture

false: capture one frame

Return value: Whether capture begins

Tips: iCamBeginCapture begins capture in asynchronous mode and returns immediately. Use iCamGetFrame function to check if capture is completed and to retrieve frame.

20. iCamGetFrame

Description: Retrieves frame.

Definition: bool iCamGetFrame(HANDLE hCam, void* data, int* pW, int* pH, int* pBpp)

Parameters:

hCam: Camera handle

data: Returns frame data

pW: Returns frame width

pH: Returns frame height

pBpp: Returns bits per pixel

Return value: Whether capture is completed

21. iCamStopCapture

Description: Aborts capture.

Definition: void iCamStopCapture(HANDLE hCam)

Parameters: hCam: Camera handle

22. iCamSetThermalNoiseRmv

Description: Sets whether to enable thermal noise removal. Only works for one frame capture mode.

Definition: bool iCamSetThermalNoiseRmv(HANDLE hCam, bool tnr)

Parameters:

hCam: Camera handle

tnr: Whether to enable thermal noise removal

Return value: Whether setting succeeds

23. iCamAutoWhiteBalance

Description: Performs auto white balance. Takes effect when next frame is captured.

Definition: bool iCamAutoWhiteBalance(HANDLE hCam, int roiLeft, int roiTop, int roiWidth, int roiHeight)

Parameters:

hCam: Camera handle

roiLeft: Left coordinate of region of interest for white balance

roiTop: Top coordinate of region of interest for white balance

roiWidth: Width of region of interest for white balance

roiHeight: Height of region of interest for white balance

Return value: Whether setting succeeds

Tips: When roiLeft, roiTop, roiWidth, roiHeight are all set to 0, whole image will be used for white balance.

24. iCamSetWhiteBalanceParams

Description: Sets white balance parameters.

Definition: bool iCamSetWhiteBalanceParams(HANDLE hCam，double paramR, double paramB)

Parameters:

hCam: Camera handle

paramR: Red channel white balance parameter, range: 0.1 - 10

paramB: Blue channel white balance parameter, range: 0.1 - 10

Return value: Whether setting succeeds

25. iCamGetWhiteBalanceParams

Description: Retrieves white balance parameters.

Definition: void iCamGetWhiteBalanceParams(HANDLE hCam, double* pParamR, double* pParamB)

Parameters:

hCam: Camera handle

pParamR: Returns red channel white balance parameter

pParamB: Returns blue channel white balance parameter

Tips: To retrieve proper auto adjusted white balance parameters after calling iCamAutoWhiteBalance, do not call iCamGetWhiteBalanceParams immediately but wait for the next frame is captured.

26. iCamSetSaturation

Description: Sets saturation.

Definition: bool iCamSetSaturation(HANDLE hCam, double saturation)

Parameters:

hCam: Camera handle

saturation: Saturation, range: 0 - 2, 1 by default

Return value: Whether setting succeeds

27. iCamSetContrast

Description: Sets contrast.

Definition: bool iCamSetContrast(HANDLE hCam, double contrast)

Parameters:

hCam: Camera handle

contrast: Contrast, range: -1 - 1, 0 by default

Return value: Whether setting succeeds

28. iCamSetSharpness

Description: Sets sharpness.

Definition: bool iCamSetSharpness(HANDLE hCam, double sharpness)

Parameters:

hCam: Camera handle

sharpness: Sharpness, range: 0 - 2, 0 by default

Return value: Whether setting succeeds

29. iCamSetFlip

Description: Sets frame flip.

Definition: bool iCamSetFlip(HANDLE hCam, bool horz, bool vert)

Parameters:

hCam: Camera handle

horz: Whether to flip horizontally

vert: Whether to flip vertically

Return value: Whether setting succeeds

30. iCamSetTargetTemperature

Description: Sets cooling target temperature. Only works for deep cooled CCD cameras.

Definition: bool iCamSetTargetTemperature(HANDLE hCam, double temp)

Parameters:

hCam: Camera handle

temp: Cooling target temperature, range: -50 to 30, in centi-degrees

Return value: Whether setting succeeds

31. iCamBeginCooling

Description: Begins cooling. Only works for deep cooled CCD cameras.

Definition: bool iCamBeginCooling(HANDLE hCam)

Parameters: hCam: Camera handle

Return value: Whether cooling begins

32. iCamStopCooling

Description: Aborts cooling. Only works for deep cooled CCD cameras.

Definition: void iCamStopCooling(HANDLE hCam)

Parameters: hCam: Camera handle

33. iCamGetCoolerStatus

Description: Retrieves current cooler status. Only works for deep cooled CCD cameras.

Definition: bool iCamGetCoolerStatus(HANDLE hCam, double* pTemp, unsigned char* pPower)

Parameters:

hCam: Camera handle

pTemp: Returns current cooler temperature, in centi-degrees

pPower: Returns current cooler power percentage

Return value: Whether status is retrieved successfully

34. iCamGetAuthToken

Description: Gets an authorization token from camera. Authorization tokens are used when calling image processing APIs that need authorization.

Definition: HANDLE iCamGetAuthToken(HANDLE hCam)

Parameters: hCam: Camera handle

Return value: Authorization token

B. Image Processing

DLL: iImg.dll

Header: iImg.h

Library: iImg.lib

1. iImgLoad

Description: Loads an image. Supported format: BMP, JPEG, PNG, TIFF.

Definition: bool iImgLoad(char* fileName, void* data, int* pW, int* pH, int* pBpp)

Parameters:

fileName: Image path

data: Returns image data

pW: Returns image width

pH: Returns image height

pBpp: Returns image bits per pixel

Return value: Whether image is loaded successfully

2. iImgSave

Description: Saves an image. Supported format: BMP, JPEG, PNG, TIFF.

Definition: void iImgSave(char* fileName, void* data, int w, int h, int bpp)

Parameters:

fileName: Image path

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixe

3. iImgAdaptBpp

Description: Adjusts image color depth.

Definition: void iImgAdaptBpp(void* dataIn, int w, int h, int bpp, void* dataOut, int newBpp)

Parameters:

dataIn: Input image data

w: Image width

h: Image height

bpp: Image bits per pixel

dataOut: Output image data

newBpp: New image bits per pixel

4. iImgGamma

Description: Adjusts image gamma.

Definition: void iImgGamma(void* data, int w, int h, int bpp, double gamma)

Parameters:

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixel

gamma: Gamma, range 0.1 - 10

5. iImgSaturation

Description: Adjusts image saturation.

Definition: void iImgSaturation(void* data, int w, int h, int bpp, double saturation)

Parameters:

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixel

saturation: Saturation, range 0 - 2

6. iImgGetLevelStats

Description: Retrieves level statistics of image.

Definition: void iImgGetLevelStats(void* data, int w, int h, int bpp, int* stat, int* statR, int* statG, int* statB)

Parameters:

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixel

stat: Returns overall level statistics, integer array of 256 elements, indicating number of pixels for each luma level

statR: Returns red channel level statistics

statG: Returns green channel level statistics

statB : Returns blue channel level statistics

7. iImgGetAutoLevelParams

Description: Retrieves auto level parameters.

Definition: void iImgGetAutoLevelParams(int* lvlStat, double* pParam1, double* pParam2)

Parameters:

lvlStat: Level statistics

pParam1: Returns lower threshold of auto level adjustment

pParam2: Returns upper threshold of auto level adjustment

8. iImgLevel

Description: Adjusts image level.

Definition: void iImgLevel(void* data, int w, int h, int bpp, double* params)

Parameters:

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixel

params: Level adjustment parameters, double array of 8 elements, range: 0 - 1

params[0]: Lower threshold of red channel adjustment

params[1]: Upper threshold of red channel adjustment

params[2]: Lower threshold of green channel adjustment

params[3]: Upper threshold of green channel adjustment

params[4]: Lower threshold of blue channel adjustment

params[5]: Upper threshold of blue channel adjustment

params[6]: Lower threshold of overall adjustment

params[7]: Upper threshold of overall adjustment

9. iImgCrop

Description: Crops image.

Definition: void iImgCrop(void* dataIn, int w, int h, int bpp, void* dataOut, int cropLeft, int cropTop, int cropW, int cropH)

Parameters:

dataIn: Input image data

w: Image width

h: Image height

bpp: Image bits per pixel

dataOut: Output image data

cropLeft: Crop left margin

cropTop: Crop top margin

cropW: Crop width

cropH: Crop height

10. iImgResize

Description: Changes image size.

Definition: void iImgResize(void* dataIn, int w, int h, int bpp, void* dataOut, int newW, int newH)

Parameters:

dataIn: Input image data

w: Image width

h: Image height

bpp: Image bits per pixel

dataOut: Output Image data

newW: New image width

newH: New image height

11. iImgFlip

Description: Flips image.

Definition: void iImgFlip(void* data, int w, int h, int bpp, bool horz, bool vert)

Parameters:

data: Image data

w: Image width

h: Image height

bpp: Image bits per pixel

horz: Whether to flip horizontally

vert: Whether to flip vertically

Appendix A Camera Parameter Specification

Camera Type	Binning	Resolution	Depth	Cooling
MU204	-	1024*768	8 bit/16 bit	-
		640*480		
MU267	-	1360*1024	8 bit/16 bit	-
		640*480		
MU285	-	1360*1024	8 bit/16 bit	-
		640*480		
MU412	-	2048*1536	8 bit/16 bit	-
		1024*768		
		640*480		
MU282	-	2560*1920	8 bit/16 bit	-
		1280*960		
		640*480		
CF255	-	500*582	16 bit	Cooled
	2*2	250*291		
CF285A/CF285B	-	1360*1024	16 bit	Deep Cooled
	2*2	680*512		
	4*4	340*256		
CF674	-	1932*1452	16 bit	Deep Cooled
	2*2	966*726		
	4*4	483*363		
CF4022	-	2048*2048	16 bit	Deep Cooled
	2*2	1024*1024		
	4*4	512*512		
CF413	-	3000*2000	16 bit	Deep Cooled
	2*2	1500*1000		
	4*4	750*500		
CF694	-	2750*2200	16 bit	Deep Cooled
	2*2	1375*1100		
	4*4	687*550		
CF8300A	-	3326*2504	16 bit	Deep Cooled
	2*2	1663*1252		
	3*3	1108*834		
	4*4	831*626		
CF8300B	-	3326*2504	16 bit	Deep Cooled
	2*2	1663*1252		
	4*4	831*626		
CF814	-	3380*2704	16 bit	Deep Cooled
	2*2	1690*1352		
	4*4	845*676		
SC035	-	1280*1024	8 bit	-

		640*480		
SC600	-	3072*2048	8 bit/16 bit	-
	2*2	1536*1024		
SC1200	-	4000*3000	8 bit/16 bit	-
	2*2	2000*1500		
	3*3	1328*1000		
SC2000	-	5472*3648	8 bit/16 bit	-
	2*2	2736*1824		
	3*3	1824*1216		
AP120	-	1280*960	8 bit/16 bit	-
		1024*768		
		800*600		
		640*480		
		320*240		
AP130	-	320*240 - 1280*1024	8 bit	-
AP300	-	320*240 - 2048*1536	8 bit	-
	2*2	320*240 - 1024*768		
	3*3	320*240 - 640*480		
AP500	-	320*240 - 2560*1920	8 bit	-
	2*2	320*240 - 1280*960		
	4*4	320*240 - 640*480		
IE500	-	320*240 - 2560*1920	8 bit	-
	2*2	320*240 - 1280*960		
	4*4	320*240 - 640*480		
MH655	-	2448*2050	8 bit/16 bit	Deep Cooled
MH694	-	2752*2200	8 bit/16 bit	Deep Cooled
SD1600A	-	4608*3456	8 bit/16 bit	Deep Cooled
	2*2	2304*1728		
	3*3	1536*1152		