Driver Circuit for Image Sensor Control Library

DCamUSB : Version 2.1.1.0
DCamTmpCtrl : Version 2.1.0.0

Function Specifications

Document Number : K46-B60011
Document Revision : 2.02 29.Jul.2014

HAMAMATSU

Software License Terms

Please carefully review the following terms and conditions.

Hamamatsu Photonics permits the use of this software only to those companies which accepts the License Terms for using "MCD Head Products" as well as "Amprefer Board Products" ("our products"). By copying these files, or installing this software, user agrees to be bound by the License Terms. In case, the user disagrees with the complete or a part of the License Terms, he will not be able to install or use this software.

1. The purpose of this software

Considering the convenience of the users who wants to use our product easily, the software is licensed free of charge with no warranty. Please use this software at your own discretion and responsibility.

2. License

Only the user who agrees with the License Terms has the right to install and use our software application solely for purposes of implementing the measurement and control of our products.

3. Ownership of copyrights and other rights

Proprietary software and related documentation, and all other intellectual property rights will belong to us. International treaties and laws on intellectual property rights and copyright law protect this software. The user is not allowed to change or modify the attached Software or documentations.

Except for matters that are explicitly licensed under License Terms, we will not transfer or grant any rights to users. All rights on this software and documentation are reserved to us.

4. Reproduction

Users can reproduce this software for backup purposes in compliance with the terms of any provision of this license.

5. Prohibition clause

The user can not do the following things. However, in case user rents or leases or transfers our products to a third party user and the third party agrees to the License Terms of the product of our company, our company continues to permit the use of this software to a third party under this License Terms.

1) Sublicense, distribute or otherwise transfer the Product or any component thereof to any third party.
2) Re-license or transfer the rights to use this software to the third party.
3) Lease, rent or loan the Product to any unlicensed third party.
4) Modify or remove part or all of this software including this License Terms and other attached annexes. The change in the file name is included in the modification.
5) Reverse engineer, decompile, disassemble, or otherwise attempt toderive the source code of this Software.

6. Limitation of Liability

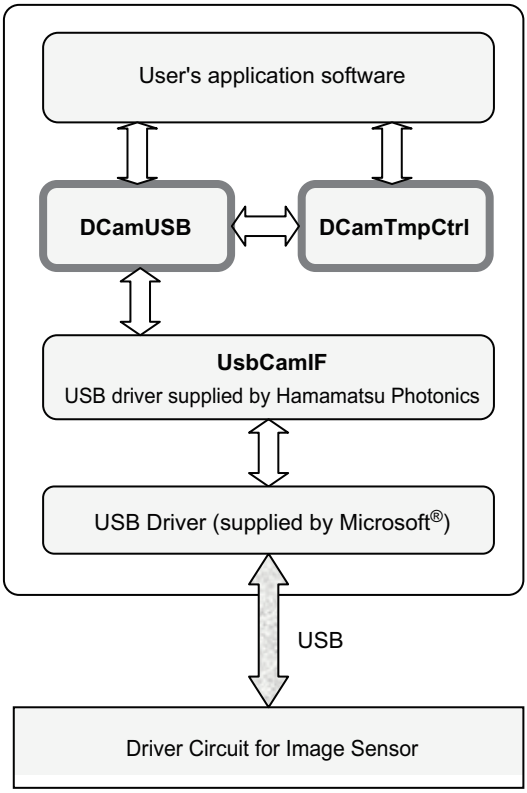
Hamamatsu Photonics will guarantee neither this software nor the attached annex including adaptability to the quality, the performance or the particular purpose at all. In any case, computer failure or damage arising from use of or inability to use this software and documentation, loss of information, all other direct and indirect damages, we will not be responsible. Moreover, not only maintenance and support but also in the event of any defect or failure, etc., we do not bear responsibility including repair and restoration of this software.

Hamamatsu Photonics can do changes without notice for improvement of this software.

Table of contents

1 Overview

The Driver Circuit for Image Sensor Control Library (Hereafter abbreviated as "DCamUSB") and Temperature Control Library (Hereafter abbreviated as "DCamTmpCtrl") is a library for controlling our Driver Circuit for Image Sensor. With this library, software for controlling the operations of the Driver Circuit can be easily developed.



2 Operating environment

Supported OS

Microsoft® Windows 7® (Service Pack 1 or later)

CPU

To conform the environment of OS recommended.

Memory

To conform the environment of OS recommended.

3 Development environment construction

Please use the copy of DCamUSB.dll and DCamTmpCtrl.dll in the prescribed folder of the development environment.
A copy of DCamUSB.h and DCamTmpCtrl.h and DCamUSB.lib and DCamTmpCtrl.lib will be required when the need arises. In that case, please copy DCamStatusCode.h.

Additionally, please refer the "Driver Software for USB Camera Module INSTALLATION MANUAL" for how to install the driver using the installer.

NOTE

In the drive circuit where the temperature control library is not included, DcomTmpCtrl.dll and DcomTmpCtrl.h and DcomTmpCtrl.lib is not necessary.

4 Required files

This library consists of the following files.

Library file	:	DCamUSB.dll, DCamTmpCtrl.dll
Header file	:	DCamUSB.h, DCamTmpCtrl.h, DCamStatusCode.h
Import library	:	DCamUSB.lib, DCamTmpCtrl.lib
Driver	:	UsbCamIF.sys
Driver information file	:	UsbCamIF.inf

5 Function list

Following are the functions available in this library.

DCamUSB.dll

1.	DcamInitialize	Initializes the library.
2.	DcamUninitialize	Uninitializes and unloads the library.
3.	DcamOpen	Opens the device.
4.	DcamClose	Closes the device.
5.	DcamGetDeviceState	Retrieves the device state.
6.	DcamGetImageSize	Retrieves the image size.
7.	DcamGetBitPerPixel	Retrieves the number of bits per pixel.
8.	DcamSetCCDType	Sets the CCD sensor type.
9.	DcamGetCCDType	Retrieves the CCD sensor type.
10.	DcamSetMeasureDataCount	Sets the measurement line count.
11.	DcamGetMeasureDataCount	Retrieves the measurement line count.
12.	DcamGetCaptureBytes	Retrieves the number of bytes of one frame.
13.	DcamGetTotalCaptureBytes	Retrieves the total number of bytes per capture size.
14.	DcamCapture	Captures an image.
15.	DcamCaptureReverseX	Captures an image. The acquired data is reversed.
16.	DcamStop	Stops image acquisition.
17.	DcamStopEx	Stops image acquisition.
18.	DcamWait	Waits till the image is captured.
19.	DcamSetGain	Sets the gain.
20.	DcamGetGain	Retrieves the gain.
21.	DcamSetOffset	Sets the offset.
22.	DcamGetOffset	Retrieves the offset.
23.	DcamSetBinning	Sets the binning mode.
24.	DcamGetBinning	Retrieves the binning mode.
25.	DcamSetTriggerMode	Sets the trigger mode.
26.	DcamGetTriggerMode	Retrieves the trigger mode.
27.	DcamSetTriggerPolarity	Sets the trigger polarity.
28.	DcamGetTriggerPolarity	Retrieves the trigger polarity.
29.	DcamSetExposureTime	Sets the exposure time.
30.	DcamGetExposureTime	Retrieves the exposure time.
31.	DcamSetOperatingMode	Sets the CCD operating mode.
32.	DcamGetOperatingMode	Retrieves the CCD operating mode.
33.	DcamSetLEDOperatingMode	Sets the LED light-operating mode.
34.	DcamGetLEDOperatingMode	Retrieves the LED light-operating mode.
35.	DcamSetStandardTimeUnit	Function to set the standard time unit type
36.	DcamGetStandardTimeUnit	Function to retrieve the standard time unit type
37.	DcamSetOutPulse	Sets the out pulse information.
38.	DcamGetOutPulse	Retrieves the out pulse information.
39.	DcamLoadParameters	Loads the parameters from the device.
40.	DcamStoreParameters	Stores the parameters in the device.
41.	DcamGetVersion	Retrieves the library version information.
42.	DcamGetDriverVersion	Retrieves the driver version information.

43.	DcamGetFirmwareVersion	Retrieves the firmware version information.
44.	DcamGetDeviceInformation	Retrieves the device information.
45.	DcamGetTransferRateType	Retrieves the USB transfer rate type.
46.	DcamGetLastError	Retrieves the last error code.
47.	DcamSetOverClock	Sets the over clock.
48.	DcamGetOverClock	Retrieves the over clock.
49.	DcamSetMPPMode	Sets the MPP mode.
50.	DcamGetMPPMode	Retrieves the MPP mode.
51.	DcamSetLineTime	Sets the Line Time.
52.	DcamGetLineTime	Retrieves the Line Time.
53.	DcamSetIntegralCapacity	Sets the integral capacity.
54.	DcamGetIntegralCapacity	Retrieves the integral capacity.
55.	DcamSetDriveMode	Set CCD Drive Mode.
56.	DcamGetDriveMode	Get CCD Drive Mode.
57.	DcamSetElectronicShutter	Sets the electronic shutter mode.
58.	DcamGetElectronicShutter	Retrieves the electronic shutter mode.
59.	DcamSetSensorSignalPulseWidth	Sets the TG pulse width.
60.	DcamGetSensorSignalPulseWidth	Retrieves the TG pulse width.

■ DCamTmpCtrl.dll

1.	DcamTmpCtrlInitialize	Initializes the library.
2.	DcamTmpCtrlUninitialize	Uninitializes and unloads the library.
3.	DcamTmpCtrlGetLastError	Retrieves the last error code.
4.	DcamTmpCtrlSetCoolingControl	Set the cooling control status.
5.	DcamTmpCtrlGetCoolingControl	Retrieves the cooling control status.
6.	DcamTmpCtrlLoadCoolingTemperature	Load the cooling temperature status.
7.	DcamTmpCtrlSaveCoolingTemperature	Save the cooling temperature status.
8.	DcamTmpCtrlGetCoolingTemperature	Retrieves the cooling temperature.
9.	DcamTmpCtrlSetCoolingTemperature	Set the cooling temperature.
10.	DcamTmpCtrlGetCoolingTemperatureCurrent	Retrieves the cooling current temperature.
11.	DcamTmpCtrlGetCoolingStatus	Retrieves the cooling temperature status.
12.	DcamTmpCtrlGetThermistorStatus	Retrieves the cooling thermistor status.

5.1 Parameter definition

5.1.1 DCamUSB.h

[Device state]

DCAM_DEVSTATE_NON	Non-connection, No device found
DCAM_DEVSTATE_DEVICE	Non-connection, Device found
DCAM_DEVSTATE_NODEVICE	Connection, No device found
DCAM_DEVSTATE_CONNECT	Connection, Device found
DCAM_DEVSTATE_BOOT	Connection, Device found (during the boot process)

[Number of bits per pixel]

DCAM_BITPIXEL_8	8 bits
DCAM_BITPIXEL_10	10 bits
DCAM_BITPIXEL_12	12 bits
DCAM_BITPIXEL_14	14 bits
DCAM_BITPIXEL_16	16 bits

[Image acquisition]

DCAM_WAITSTATUS_COMPLETED	Image acquisition is complete.
DCAM_WAITSTATUS_UNCOMPLETED	Image acquisition is not complete.
DCAM_WAIT_INFINITE	Wait until image acquisition is complete.

[Binning]

DCAM_BINNING_AREA	Area scanning
DCAM_BINNING_FULL	Full line binning

[Trigger mode]

DCAM_TRIGMODE_INT	Internal Mode
DCAM_TRIGMODE_EXT_EDGE	External Trigger Edge Mode
DCAM_TRIGMODE_EXT_LEVEL	External Trigger Level Mode
DCAM_TRIGMODE_GS_INT	Global Shutter Internal Mode
DCAM_TRIGMODE_GS_EXT_EDGE	Global Shutter External Trigger Edge Mode
DCAM_TRIGMODE_GS_EXT_GATED	Global Shutter External Gated Mode
DCAM_TRIGMODE_GS_EXT_ONE_SHOT	Global Shutter External One Shot Mode
DCAM_TRIGMODE_RS_INT	Rolling Shutter Internal Mode
DCAM_TRIGMODE_RS_EXT_EDGE	Rolling Shutter External Trigger Edge Mode
DCAM_TRIGMODE_RS_EXT_GATED	Rolling Shutter External Gated Mode
DCAM_TRIGMODE_RS_EXT_ONE_SHOT	Rolling Shutter External One Shot Mode

Note : Refer the "DCamUSB.h" for another trigger mode.

[Trigger polarity]

DCAM_TRIGPOL_POSITIVE	Positive
DCAM_TRIGPOL_NEGATIVE	Negative

[Sensor type]

DCAM_CCD_TYPE0	2068x1(S10420-1106), 2068x70(S10420-1106-01, S11071-1106)
DCAM_CCD_TYPE2	2068x22(S10420-1104-01, S11071-1104)
DCAM_CCD_TYPE3	1044x1(S10420-1006), 1044x70(S10420-1006-01, S11071-1006)
DCAM_CCD_TYPE5	1044x22(S10420-1004-01, S11071-1004)
DCAM_CCD_TYPE10	64x64(G11097)

Note : Refer the "DCamUSB.h" for another sensor type.

[CCD drive mode]

DCAM_CCDDRVMODE_SUSPEND	Suspend
DCAM_CCDDRVMODE_STANDBY	Standby

[CCD operating mode]

DCAM_OPMODE_DARKCURRENT	Low Dark Current Mode
DCAM_OPMODE_SATURATION	High Saturation Charge Mode

[LED light operating mode]

DCAM_LEDOPMODE_OFF	LED Off Mode
DCAM_LEDOPMODE_ON	LED On Mode

[Standard time unit type]

DCAM_TIME_UNIT_TYPE1	Trigger setting = [msec], Pulse Out setting = [msec]
DCAM_TIME_UNIT_TYPE2	Trigger setting = [usec], Pulse Out setting = [usec]
DCAM_TIME_UNIT_TYPE3	Trigger setting = [msec], Pulse Out setting = [usec]
DCAM_TIME_UNIT_TYPE4	Trigger setting = [Clock], Pulse Out setting = [Clock]

[Out pulse mode]

DCAM_OUTMODE_NOTOUTPUT	No output
DCAM_OUTMODE_PLS_DT_PW	Output (Delay time + Pulse width)
DCAM_OUTMODE_PLS_ACCUM	Output (Accumulation time width)

[Out pulse polarity]

DCAM_OUTPOL_POSITIVE	Positive
DCAM_OUTPOL_NEGATIVE	Negative

[Device information]

DCAM_DEVINF_TYPE	Device Type
DCAM_DEVINF_SERIALNO	Serial Number of Device
DCAM_DEVINF_VERSION	Device Version

[USB transfer rate type]

DCAM_TRANSRATE_USB11	USB 1.1 standard
DCAM_TRANSRATE_USB20	USB 2.0 standard

[MPP mode]

DCAM_CCDMPPMODE_OFF	MPP Off Mode
DCAM_CCDMPPMODE_ON	MPP On Mode

[Electronic shutter mode]

DCAM_CCDESHUTTER_OFF	Electronic shutter Off Mode.
DCAM_CCDESHUTTER_ON	Electronic shutter On Mode.

5.1.2 DCamTmpCtrl.h

[Cooling Contorl status]

DCAM_COOLING_CONTROL_OFF	Cooling control Off
DCAM_COOLING_CONTROL_ON	Cooling control On

[Cooling Temperature status]

DCAM_COOLING_STATUS_NORMAL	The cooling temperature is normal
DCAM_COOLING_STATUS_LOWER	The cooling temperature is lower
DCAM_COOLING_STATUS_HIGHER	The cooling temperature is higher

[Thermistor status]

DCAM_THERMISTOR_STATUS_NOERROR	Thermistor no error
DCAM_THERMISTOR_STATUS_ERROR	Thermistor error
DCAM_THERMISTOR_STATUS_OVER	Thermistor temperature over

5.2 Error Code Table (Run status)

5.2.1 DCamStatusCode.h

0	dcCode_Success	Normal termination.
1	dcCode_Unknown	An unknown error has occurred.
2	dcCode_NoInit	Library is not initialized.
3	dcCode_AlreadyInit	Already in-use.
4	dcCode_NoDriver	No driver was detected.
5	dcCode_NoMemory	Memory is insufficient.
6	dcCode_NotConnected	The device is not connected.
9	dcCode_InvalidParam	Invalid parameter.
100	dcCode_DeviceDefect	The device is not functioning.
111	dcCode_Timeout	Timeout has occurred.
120	dcCode_AlreadyStarted	Already started.
200	dcCode_CoolingOn	Already cooling control started.
201	dcCode_CoolingOff	Cooling control stopped.

6 Function details

6.1 DCamUSB.dll

6.1.1 DcamInitialize: Function to initialize the library

BOOL DcamInitialize(VOID)

[Summary]

Initializes the library.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

Always run this function first before running other functions.
An error occurs if the library has already been initialized.
Only one process can use this library.
In addition, this function should be used in combination with the "DcamUninitialize".

[Reference]

DcamUninitialize, DcamOpen, DcamClose

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;
if(DcamInitialize() != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.2 DcamUninitialize: Function to uninitialize the library

BOOL DcamUninitialize(VOID)

[Summary]

Unloads the library resources and closes the device driver.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

This function should be used in combination with the "DcamInitialize".
In addition,this function should be called when quitting the program or when this library is not required.

[Reference]

DcamInitialize, DcamOpen, DcamClose

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;
if(DcamUninitialize() != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.3 DcamOpen: Function to open the device

BOOL DcamOpen(VOID)

[Summary]

Opens the device.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamClose, DcamInitialize, DcamUninitialize

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;

// Initialize Library
if(DcamInitialize() != TRUE){
    dwErrCode = DcamGetLastError();
    return;
}

// Open Device
if(DcamOpen() != TRUE){
    dwErrCode = DcamGetLastError();
    return;
}
```

6.1.4 DcamClose: Function to close the device

BOOL DcamClose(VOID)

[Summary]

Closes the device.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamOpen, DcamInitialize, DcamUninitialize

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;

// Close Device
if(DcamClose() != TRUE){
    dwErrCode = DcamGetLastError();
    return;
}

// Uninitialize Library
if(DcamUninitialize() != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.5 DcamGetDeviceState: Function to retrieve device state

BOOL DcamGetDeviceState(INT* pState)

[Summary]

Retrieves the device state type.

[Arguments]

pState — Specifies the address of the variable where the device state type is to be stored. Any one of the following values is obtained.

DCAM_DEVSTATE_NON	:	Non-connection, No device found
DCAM_DEVSTATE_DEVICE	:	Non-connection, Device found
DCAM_DEVSTATE_NODEVICE	:	Connection, No device found
DCAM_DEVSTATE_CONNECT	:	Connection, Device found
DCAM_DEVSTATE_BOOT	:	Connection, Device found (during the boot process)

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamOpen, DcamClose, DcamInitialize, DcamUninitialize

[Example]

The following example shows how this function is called.

```
INT             nState;
DWORD           dwErrCode;

// Get device state
if(DcamGetDeviceState(&nState) != TRUE){
    dwErrCode = DcamGetLastError();
}

// Remove the device
if(nState == DCAM_DEVSTATE_NODEVICE){
    DcamClose();
}
```

6.1.6 DcamGetImageSize: Function to retrieve the image size

BOOL DcamGetImageSize (INT* pWidth, INT* pHeight)

[Summary]

Retrieves the width and height of the one frame data.

[Arguments]

pWidth	Specifies the address of the variable where the image width is to be stored.
pHeight	Specifies the address of the variable where the image height is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

Call this function for obtaining the size of image after calling DcamSetBinning or DcamSetMeasureDataCount that can be helpful in determining the memory size to be allocated.

[Reference]

DcamGetBitPerPixel, DcamGetCaptureBytes, DcamSetBinning, DcamSetMeasure-DataCount, DcamSetCCDType, DcamGetCCDType

[Example]

The following example shows how this function is called.

```
INT         nWidth = 0;
INT         nHeight = 0;
DWORD       dwErrCode;

// Get size of image
if(DcamGetImageSize(&nWidth, &nHeight) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.7 DcamGetBitPerPixel:Function to retrieve total number of bits per pixel

BOOL DcamGetBitPerPixel(INT* pBit)

[Summary]

Retrieves the number of bits per pixel.

[Arguments]

pBit — Specifies the address of the variable where the number of bits per pixel is to be stored. Any one of the following values is obtained.

DCAM_BITPIXEL_8	:	8 bits
DCAM_BITPIXEL_10	:	10 bits
DCAM_BITPIXEL_12	:	12 bits
DCAM_BITPIXEL_14	:	14 bits
DCAM_BITPIXEL_16	:	16 bits

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetImageSize, DcamGetCaptureBytes

[Example]

The following example shows how this function is called.

```
INT             nBit = 0;
DWORD           dwErrCode;

// Get size of image
if(DcamGetBitPerPixel(&nBit) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.8 DcamSetCCDType: Function to set the CCD sensor type

BOOL DcamSetCCDType(INT nType)

[Summary]

Sets the CCD sensor type.

[Arguments]

nType To specify the type of sensor, that are specified in "DCamUSB.h".

DCAM_CCD_TYPE0	:	2068x1(S10420-1106), 2068x70(S10420-1106-01, S11071-1106)
DCAM_CCD_TYPE2	:	2068x22(S10420-1104-01, S11071-1104)
DCAM_CCD_TYPE3	:	1044x1(S10420-1006), 1044x70(S10420-1006-01, S11071-1006)
DCAM_CCD_TYPE5	:	1044x22(S10420-1004-01, S11071-1004)
DCAM_CCD_TYPE10	:	64x64(G11097)

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetCCDType, DcamGetImageSize, DcamGetCaptureBytes

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetCCDType(DCAM_CCD_TYPE0) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.9 DcamGetCCDType: Function to retrieve the CCD sensor type

BOOL DcamGetCCDType(INT* pType)

[Summary]

Retrieves the CCD sensor type.

[Arguments]

pType — Specifies the address of the variable where the type of CCD sensor that is currently set is to be stored. Acquire type of sensors that are retrieved, that is specified in the "DcamUSB.h".

DCAM_CCD_TYPE0	:	2068x1(S10420-1106), 2068x70(S10420-1106-01, S11071-1106)
DCAM_CCD_TYPE2	:	2068x22(S10420-1104-01, S11071-1104)
DCAM_CCD_TYPE3	:	1044x1(S10420-1006), 1044x70(S10420-1006-01, S11071-1006)
DCAM_CCD_TYPE5	:	1044x22(S10420-1004-01, S11071-1004)
DCAM_CCD_TYPE10	:	64x64(G11097)

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetCCDType

[Example]

The following example shows how this function is called.

```
INT             nType;
DWORD           dwErrCode;

if(DcamGetCCDType(&nType) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.10 DcamSetMeasureDataCount:Function to set the measurement line count

BOOL DcamSetMeasureDataCount(INT nCount)

[Summary]

Sets the measurement frame count.

[Arguments]

nCount	Specifies the measurement frame count. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetBinning, DcamGetImageSize, DcamGetBitPerPixel, DcamDcamGetCaptureBytes,
DcamGetMeasureDataCount

[Example]

The following example shows how this function is called.

```
DWORD        dwErrCode;

if(DcamSetMeasureDataCount (10) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.11 DcamGetMeasureDataCount:Function to retrieve the measurement line count

BOOL DcamGetMeasureDataCount(INT* pCount)

[Summary]

Retrieves the measurement frame count.

[Arguments]

pCount	Specifies the address of the variable where the measurement frame count is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetBinning, DcamGetImageSize, DcamGetBitPerPixel, DcamGetCapture-Bytes,
DcamSetMeasureDataCount

[Example]

The following example shows how this function is called.

```
INT         nCount = 0;
DWORD       dwErrCode;

// Get the measurement line count
if(DcamGetMeasureDataCount (&nCount) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.12 DcamGetCaptureBytes : Function to retrieve number of bytes of one frame

BOOL DcamGetCaptureBytes(INT* pBytes)

[Summary]

Retrieves the number of bytes of one frame.

[Arguments]

pBytes	Specifies the address of the variable where the number of bytes of one frame is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetImageSize, DcamGetBitPerPixel, DcamGetMeasureDataCount, DcamGetTotalCaptureBytes

[Example]

The following example shows how this function is called.

```
INT             nBytes = 0;
DWORD           dwErrCode;

// Get total bytes of capture size.
if(DcamGetFrameBytes(&nBytes) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.13 DcamGetTotalCaptureBytes:Function to retrieve total number of bytes per capture size

BOOL DcamGetTotalCaptureBytes(INT* pBytes)

[Summary]

Retrieves the total number of bytes per capture size.

[Arguments]

pBytes	Specifies the address of the variable where the total number of bytes per capture size is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetImageSize, DcamGetBitPerPixel, DcamGetMeasureDataCount, DcamGetCaptureBytes

[Example]

The following example shows how this function is called.

```
INT         nBytes = 0;
DWORD       dwErrCode;

// Get total bytes of capture size.
if(DcamGetFrameBytes(&nBytes) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.14 DcamCapture: Function to capture the image

BOOL DcamCapture(LPVOID pImageBuff, INT nBuffSize)

[Summary]

Starts to capture one image from the device.

[Arguments]

pImageBuff	Specifies the starting address of the buffer where the image data is to be stored.
nBuffSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

(1) This function issues an instruction to start capturing the image. Since the image capturing is not complete even when this function ends, use the DcamWait function to check whether image capturing is complete.
(2) The necessary buffer size can be obtained with the DcamGetCaptureBytes function.

[Reference]

DcamWait, DcamStop, DcamStopEx, DcamGetCaptureBytes

[Example]

Please refer to "7.1 About the flow of data acquisition".

6.1.15 DcamCaptureReverseX: Function to capture the image

BOOL DcamCaptureReverseX(LPVOID pImageBuff, INT nBuffSize)

[Summary]

Starts to capture one image from the device. The X-axis of the acquired image data is reversed.

[Arguments]

pImageBuff	Specifies the starting address of the buffer where the image data is to be stored.
nBuffSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

(1) This function issues an instruction to start capturing the image. Since the image capturing is not complete even when this function ends, use the DcamWait function to check whether image capturing is complete.
(2) The necessary buffer size can be obtained with the DcamGetCaptureBytes function.

[Reference]

DcamWait, DcamStop, DcamStopEx, DcamGetCaptureBytes

[Example]

Please refer to "7.1 About the flow of data acquisition".

6.1.16 DcamStop: Function to stop image capturing

BOOL DcamStop(VOID)

[Summary]

Stops capturing the image.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamCapture, DcamCaptureReverseX, DcamWait, DcamStopEx

[Example]

Please refer to "7.1 About the flow of data acquisition".

6.1.17 DcamStopEx: Function to stop image capturing

BOOL DcamStopEx(VOID)

[Summary]

Stops capturing the image.
The function of "DcamStopEx" is same as "DcamStop".
But if use the following models, please use "DcamStopEx".

Model	Version
C11287	"V1.01C_V1.01F" Earlier than this. (including this version.)
C11288	"V1.01C_V1.01F" Earlier than this. (including this version.)
C11860	"V0.02C_V0.03F" Earlier than this. (including this version.)
C11861	"V0.02C_V0.03F" Earlier than this. (including this version.)

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamCapture, DcamCaptureReverseX, DcamWait, DcamStop

[Example]

Please refer to "7.1 About the flow of data acquisition".

6.1.18 DcamWait: Function to wait till image is captured

BOOL DcamWait(DWORD* pStatus, INT nTimeout)

[Summary]

Waits till the image is captured.

[Arguments]

pStatus	Specifies the address of the variable where the image capturing completion status is to be stored. Whether image capturing is complete or not can be checked by the value in this variable. The value is either of the following: DCAM_WAITSTATUS_COMPLETED : Image capturing is complete. DCAM_WAITSTATUS_UNCOMPLETED : Image capturing is not complete.
nTimeout	Specifies the length of timeout in milliseconds. When "DCAM_WAIT_INFINITE" is specified here, the process waits until image capturing is finished. When "0" is specified, control is returned immediately after checking the status.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamCapture, DcamCaptureReverseX, DcamStop, DcamStopEx

[Example]

Please refer to "7.1 About the flow of data acquisition".

6.1.19 DcamSetGain: Function to set the gain

BOOL DcamSetGain(INT nGain)

[Summary]

Sets the gain.

[Arguments]

nGain	Specifies the gain value. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetGain

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetGain(5) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.20 DcamGetGain: Function to retrieve the gain

BOOL DcamGetGain(INT* pGain)

[Summary]

Retrieves the gain.

[Arguments]

pGain Specifies the address of the variable where the gain is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetGain

[Example]

The following example shows how this function is called.

```
INT             nGain = -1;
DWORD           dwErrCode;

if(DcamGetGain(&nGain) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.21 DcamSetOffset: Function to set the offset

BOOL DcamSetOffset(INT nOffset)

[Summary]

Sets the offset.

[Arguments]

nOffset	Specifies the offset value. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetOffset

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetOffset(10) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.22 DcamGetOffset: Function to retrieve the offset

BOOL DcamGetOffset(INT* pOffset)

[Summary]

Retrieves the offset.

[Arguments]

pOffset	Specifies the address of the variable where the offset is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetOffset

[Example]

The following example shows how this function is called.

```
INT         nOffset = -1;
DWORD       dwErrCode;

if(DcamGetOffset(&nOffset) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.23 DcamSetBinning: Function to set the binning

BOOL DcamSetBinning(INT nBinning)

[Summary]

Sets the binning.

[Arguments]

nBinning Specifies the binning. Either one of the following can be specified.

DCAM_BINNING_AREA : Area scanning
DCAM_BINNING_FULL : Full line binning

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

When this function is run, the number of bytes per frame size may change. Check the capture size with the DcamGetCaptureBytes function.

[Reference]

DcamGetBinning, DcamGetCaptureBytes

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetBinning(DCAM_BINNING_FULL) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.24 DcamGetBinning: Function to retrieve the binning

BOOL DcamGetBinning(INT* pBinning)

[Summary]

Retrieves the binning.

[Arguments]

pBinning	Specifies the address of the variable where the currently set binning is to be stored. Either one of the following values is obtained. DCAM_BINNING_AREA : Area scanning DCAM_BINNING_FULL : Full line binning

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetBinning

[Example]

The following example shows how this function is called.

```
INT             nBinning;
DWORD           dwErrCode;

if(DcamGetBinning(&nBinning) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.25 DcamSetTriggerMode: Function to set the trigger mode

BOOL DcamSetTriggerMode(INT nMode)

[Summary]

Sets the trigger mode.

[Arguments]

nMode — Specifies the trigger mode. Any one of the following can be specified.

DCAM_TRIGMODE_INT	:	Internal Mode
DCAM_TRIGMODE_EXT_EDGE	:	External Trigger Edge Mode
DCAM_TRIGMODE_EXT_LEVEL	:	External Trigger Level Mode
DCAM_TRIGMODE_GS_INT	:	Global Shutter Internal Mode
DCAM_TRIGMODE_GS_EXT_EDGE	:	Global Shutter External Trigger Edge Mode
DCAM_TRIGMODE_GS_EXT_GATED	:	Global Shutter External Gated Mode
DCAM_TRIGMODE_GS_EXT_ONE_SHOT	:	Global Shutter External One Shot Mode
DCAM_TRIGMODE_RS_INT	:	Rolling Shutter Internal Mode
DCAM_TRIGMODE_RS_EXT_EDGE	:	Rolling Shutter External Trigger Edge Mode
DCAM_TRIGMODE_RS_EXT_GATED	:	Rolling Shutter External Gated Mode
DCAM_TRIGMODE_RS_EXT_ONE_SHOT	:	Rolling Shutter External One Shot Mode

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetTriggerMode, DcamSetTriggerPolarity, DcamGetTriggerPolarity, DcamSetExposureTime, DcamGetExposureTime

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetTriggerMode(DCAM_TRIGMODE_INT) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.26 DcamGetTriggerMode: Function to retrieve the trigger mode

BOOL DcamGetTriggerMode(INT* pMode)

[Summary]

Retrieves the trigger mode.

[Arguments]

pMode — Specifies the address of the variable where the currently set trigger mode is to be stored. Any one of the following values is obtained.

DCAM_TRIGMODE_INT	:	Internal Mode
DCAM_TRIGMODE_EXT_EDGE	:	External Trigger Edge Mode
DCAM_TRIGMODE_EXT_LEVEL	:	External Trigger Level Mode
DCAM_TRIGMODE_GS_INT	:	Global Shutter Internal Mode
DCAM_TRIGMODE_GS_EXT_EDGE	:	Global Shutter External Trigger Edge Mode
DCAM_TRIGMODE_GS_EXT_GATED	:	Global Shutter External Gated Mode
DCAM_TRIGMODE_GS_EXT_ONE_SHOT	:	Global Shutter External One Shot Mode
DCAM_TRIGMODE_RS_INT	:	Rolling Shutter Internal Mode
DCAM_TRIGMODE_RS_EXT_EDGE	:	Rolling Shutter External Trigger Edge Mode
DCAM_TRIGMODE_RS_EXT_GATED	:	Rolling Shutter External Gated Mode
DCAM_TRIGMODE_RS_EXT_ONE_SHOT	:	Rolling Shutter External One Shot Mode

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetTriggerMode, DcamSetTriggerPolarity, DcamGetTriggerPolarity, DcamSetExposureTime, DcamGetExposureTime

[Example]

The following example shows how this function is called.

```
INT         nMode;
DWORD       dwErrCode;

if(DcamGetTriggerMode(&nMode) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.27 DcamSetTriggerPolarity: Function to set the trigger polarity

BOOL DcamSetTriggerPolarity(INT nPolarity)

[Summary]

Sets the trigger polarity.

[Arguments]

nPolarity — Specifies the trigger polarity. Either one of the following can be specified.

DCAM_TRIGPOL_POSITIVE : Positive
DCAM_TRIGPOL_NEGATIVE : Negative

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetTriggerPolarity, DcamSetTriggerMode, DcamGetTriggerMode, DcamSetExposureTime, DcamGetExposureTime

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetTriggerPolarity(DCAM_TRIGPOL_POSITIVE) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.28 DcamGetTriggerPolarity: Function to retrieve the trigger polarity

BOOL DcamGetTriggerPolarity(INT* pPolarity)

[Summary]

Retrieves the trigger polarity.

[Arguments]

pPolarity	Specifies the address of the variable where the currently set trigger polarity is to be stored. Either one of the following values is obtained. DCAM_TRIGPOL_POSITIVE : Positive DCAM_TRIGPOL_NEGATIVE : Negative

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetTriggerPolarity, DcamSetTriggerMode, DcamGetTriggerMode, DcamSetExposureTime, DcamGetExposureTime

[Example]

The following example shows how this function is called.

```
INT             nPolarity;
DWORD           dwErrCode;

if(DcamGetTriggerPolarity(&nPolarity) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.29 DcamSetExposureTime: Function to set the exposure time

BOOL DcamSetExposureTime(INT nTime)

[Summary]

Sets the exposure time.

[Arguments]

nTime	Specifies the exposure time. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetExposureTime, DcamSetTriggerMode, DcamGetTriggerMode, DcamSetTriggerPolarity, DcamGetTriggerPolarity

[Example]

The following example shows how this function is called.

```
DWORD        dwErrCode;

if(DcamSetExposureTime(120 /* 120 msec */) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.30 DcamGetExposureTime: Function to retrieve the exposure time

BOOL DcamGetExposureTime(INT* pTime)

[Summary]

Retrieves the exposure time.

[Arguments]

pTime	Retrieves the exposure time that is currently set in standard time units

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetExposureTime, DcamSetTriggerMode, DcamGetTriggerMode, DcamSetTriggerPolarity, DcamGetTriggerPolarity

[Example]

The following example shows how this function is called.

```
INT         nTime;          // msec
DWORD       dwErrCode;

if(DcamGetExposureTime(&nTime) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.31 DcamSetOperatingMode: Function to set the CCD operating mode

BOOL DcamSetOperatingMode(INT nMode)

[Summary]

Sets the CCD operating mode.

[Arguments]

nMode	Specifies the CCD operating mode. Any one of the following can be specified. DCAM_OPMODE_DARKCURRENT : Low Dark Current Mode DCAM_OPMODE_SATURATION : High Saturation Charge Mode

[Return Value]
If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]
None.

[Reference]
DcamGetOperatingMode

[Example]
The following example shows how this function is called.

```
DWORDdwErrCode;

if(DcamSetOperatingMode(DCAM_OPMODE_DARKCURRENT) != TRUE){
dwErrCode = DcamGetLastError();
}
```

6.1.32 DcamGetOperatingMode: Function to retrieve the CCD operating mode

BOOL DcamGetOperatingMode(INT* pMode)

[Summary]

Retrieves the CCD operating mode.

[Arguments]

pMode — Specifies the address of the variable where the CCD operating mode is to be stored. Any one of the following values is obtained.

DCAM_OPMODE_DARKCURRENT : Low Dark Current Mode
DCAM_OPMODE_SATURATION : High Saturation Charge Mode

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetOperatingMode

[Example]

The following example shows how this function is called.

```
INT             nMode;
DWORD           dwErrCode;

if(DcamGetOperatingMode(&nMode) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.33 DcamSetLEDOperatingMode: Function to set the LED light operating mode

BOOL DcamSetLEDOperatingMode(INT nMode)

[Summary]

Sets the LED light operating mode.

[Arguments]

nMode	Specifies the LED light operating mode. Any one of the following can be specified. DCAM_LEDOPMODE_OFF : LED Off Mode DCAM_LEDOPMODE_ON : LED On Mode

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetLEDOperatingMode

[Example]

The following example shows how this function is called.

```
DWORD       dwErrCode;

if(DcamSetLEDOperatingMode(DCAM_LEDOPMODE_OFF) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.34 DcamGetLEDOperatingMode: Function to retrieve the LED light operating mode

BOOL DcamGetLEDOperatingMode(INT* pMode)

[Summary]

Retrieves the LED light operating mode.

[Arguments]

pMode — Specifies the address of the variable where the LED light operating mode is to be stored. Any one of the following values is obtained.

DCAM_LEDOPMODE_OFF : LED Off Mode
DCAM_LEDOPMODE_ON : LED On Mode

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetLEDOperatingMode

[Example]

The following example shows how this function is called.

```
INT             nMode;
DWORD           dwErrCode;

if(DcamGetLEDOperatingMode(&nMode) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.35 DcamSetStandardTimeUnit: Function to set the standard time unit type

BOOL DcamSetStandardTimeUnit(INT nType)

[Summary]

Sets the standard time unit type.

[Arguments]

nType — Specifies the standard time unit from among the following types.

DCAM_TIME_UNIT_TYPE1 :
Trigger setting [msec], Pulse Out setting [msec]

DCAM_TIME_UNIT_TYPE2 :
Trigger setting [usec], Pulse Out setting [usec]

DCAM_TIME_UNIT_TYPE3 :
Trigger setting [msec], Pulse Out setting = [usec]

DCAM_TIME_UNIT_TYPE4 :
Trigger setting = [Clock], Pulse Out setting = [Clock]

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamGetStandardTimeUnit, DcamSetExposureTime, DcamGetExposureTime, DcamSetOutPulse, DcamGetOutPulse

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;

if(DcamSetStandardTimeUnit(DCAM_TIME_UNIT_TYPE1) !=TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.36 DcamGetStandardTimeUnit: Function to retrieve the standard time unit type

BOOL DcamGetStandardTimeUnit(INT *pType)

[Summary]

Retrieves the standard time unit type.

[Arguments]

pType — Specifies the address of the variable where the type of standard time unit that is currently set is to be stored. Any one of the following values is obtained.

DCAM_TIME_UNIT_TYPE1:
Trigger setting [mSec], Pulse Out setting [mSec]

DCAM_TIME_UNIT_TYPE2 :
Trigger setting [uSec], Pulse Out setting [uSec]

DCAM_TIME_UNIT_TYPE3 :
Trigger setting [mSec], Pulse Out setting = [uSec]

DCAM_TIME_UNIT_TYPE4 :
Trigger setting = [Clock], Pulse Out setting = [Clock]

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamSetStandardTimeUnit, DcamSetExposureTime, DcamGetExposureTime, DcamSetOutPulse, DcamGetOutPulse

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT   nType;

if(DcamGetStandardTimeUnit(&nType) !=TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.37 DcamSetOutPulse: Function to set the out pulse information

BOOL DcamSetOutPulse(INT nMode, INT nPolarity, INT nDelayTime,
INT nPulseWidth)

[Summary]

Sets the out pulse information.

[Arguments]

nMode	Specifies the output mode. Any one of the following can be specified. DCAM_OUTMODE_NOTOUTPUT : Low Dark Current Mode DCAM_OUTMODE_PLS_DT_PW : Output (Delay Time + Pulse width) DCAM_OUTMODE_PLS_ACCUM : Output (Accumulation time width)
nPolarity	Specifies the polarity of out pulse. Any one of the following can be specified. DCAM_OUTPOL_POSITIVE : Positive DCAM_OUTPOL_NEGATIVE : Negative
nDelayTime	Specifies the delay time. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.
nPulseWidth	Specifies the pulse width. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetOperatingMode

[Example]

The following example shows how this function is called.

```
DWORD          dwErrCode;

if(DcamSetOutPulse(DCAM_OUTMODE_PLS_DT_PW,
 DCAM_OUTPOL_NEGATIVE,
 10,
 100) != TRUE){
     dwErrCode = DcamGetLastError();
}
```

6.1.38 DcamGetOutPulse: Function to retrieve the out pulse information

BOOL DcamGetOutPulse (INT* pMode, INT* pPolarity, INT* pDelayTime, INT* pPulseWidth)

[Summary]

Retrieves the out pulse information.

[Arguments]

pMode	Specifies the address of the variable where the output mode is to be stored. Any one of the following values is obtained.
	DCAM_OUTMODE_NOTOUTPUT : Low Dark Current Mode DCAM_OUTMODE_PLS_DT_PW : Output (Delay Time + Pulse width) DCAM_OUTMODE_PLS_ACCUM : Output (Accumulation time width)
pPolarity	Specifies the address of the variable where the polarity of out pulse is to be stored. Any one of the following values is obtained.
	DCAM_OUTPOL_POSITIVE : Positive DCAM_OUTPOL_NEGATIVE : Negative
pDelayTime	Retrieves the delay time that is currently set.
pPulseWidth	Retrieves the pulse width that is currently set.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetOutPulse

[Example]

The following example shows how this function is called.

```
INT         nMode;
INT         nPolarity;
INT         nDelayTime = 0;
INT         nPulseWidth = 0;
DWORD       dwErrCode;

if(DcamGetOutPulse(&nMode, &nPolarity, &nDelayTime, &nPulseWidth) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.39 DcamLoadParameters: Function to load the parameters

BOOL DcamLoadParamters(INT nTimeout)

[Summary]

Reads the device parameter settings from the internal EEPROM.

[Arguments]

nTimeout Specifies the length of timeout in milliseconds.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamStoreParameters

[Example]

The following example shows how this function is called.

```
DWORD        dwErrCode;

if(DcamLoadParamters() != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.40 DcamStoreParameters: Function to store the parameters

BOOL DcamStoreParamters(VOID)

[Summary]

Writes the current parameter settings of the device into the internal EEPROM.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamLoadParameters

[Example]

The following example shows how this function is called.

```
DWORD          dwErrCode;

if(DcamStoreParamters() != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.41 DcamGetVersion: Function to retrieve the version

BOOL DcamGetVersion(char* szVersion, INT nBufSize)

[Summary]

Retrieves the library version number, in string format.

[Arguments]

szVersion	Specifies the starting address of the character string buffer where the version of the library is to be stored.
nBufSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Reference]

DcamFirmwareVersion, DcamGetDriverVersion, DcamGetDeviceInformation

[Note]

None.

[Example]

The following example shows how this function is called.

```
char        szVersion[256];
DWORD       dwErrCode;

if(DcamGetVersion(szVersion, sizeof(szVersin)) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.42 DcamGetDriverVersion: Function to retrieve driver information

BOOL DcamGetDriverVersion(char* szVersion, INT nBufSize)

[Summary]

Retrieves the driver version number, in string format.

[Arguments]

szVersion	Specifies the starting address of the character string buffer where the version of the driver is to be stored.
nBufSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamVersion, DcamGetFirmwareVersion, DcamGetDeviceInformation

[Example]

The following example shows how this function is called.

```
char        szVersion[256];
DWORD       dwErrCode;

if(DcamGetDriverVersion(szVersion, sizeof(szVersin)) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.43 DcamGetFirmwareVersion: Function to retrieve the firmware information

BOOL DcamGetFirmwareVersion(char* szFirmVersion, INT nBufSize)

[Summary]

Retrieves the firmware version number, in a character string format.

[Arguments]

szFirmVersion	Specifies the starting address of the character string buffer where the version of the firmware is to be stored.
nBufSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamVersion, DcamGetDriverVersion, DcamGetDeviceInformation

[Example]

The following example shows how this function is called.

```
char        szVersion[256];
DWORD       dwErrCode;

if(DcamGetFirmwareVersion(szVersion, sizeof(szVersin)) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.44 DcamGetDeviceInformation:Function to retrieve the device information

BOOL DcamGetDeviceInformation(INT nType, char* pszBuff, INT nBufSize)

[Summary]

Retrieves the device information.

[Arguments]

nType	Specifies any one of the following type of information. DCAM_DEVINF_TYPE : Device type DCAM_DEVINF_SERIALNO : Serial number of device DCAM_DEVINF_VERSION : Device version
pszBuff	Specifies the starting address of the character string buffer where the device information is to be stored.
nBufSize	Specifies the buffer size (number of bytes).

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

DcamVersion, DcamGetDriverVersion, DcamGetFirmwareVersion,

[Example]

The following example shows how this function is called.

```
char        szInfo[256];
DWORD       dwErrCode;

if(DcamGetDeviceInformation(DCAM_DEVINF_TYPE, szInfo, sizeof(szInfo)) !=
TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.45 DcamGetTransferRateType:Function to retrieve the USB transfer rate type

BOOL DcamGetTransferRateType (INT* pType)

[Summary]

Retrieves the USB transfer rate type.

[Arguments]

pType	Specifies the address of the variable where the type of USB transfer rate that is currently set is to be stored. Any one of the following values is obtained. DCAM_TRANSRATE_USB11 : USB 1.1 standard DCAM_TRANSRATE_USB20 : USB 2.0 standard

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None

[Reference]

None

[Example]

The following example shows how this function is called.

```
INT             nType;
DWORD           dwErrCode;

if(DcamGetTransferRateType(&nType) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.46 DcamGetLastError : Function to retrieve the last error code

DWORDDcamGetLastError(VOID)

[Summary]

Retreives the last-error code.

[Arguments]

None.

[Return Value]

The last error code is returned. For details on error code, refer to the error code table.

[Note]

None.

6.1.47 DcamSetOverClock : Function to Set the over clock

BOOL DcamSetOverClock (INT nClock)

[Summary]

Sets the over clock.

[Arguments]

nClock	Specify the over clock. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

When the trigger mode is a DCAM_TRIGMODE_RS_INT and DCAM_TRIGMODE_RS_EXT_EDGE and DCAM_TRIGMODE_RS_EXT_GATED and DCAM_TRIGMODE_RS_EXT_ONE_SHORT, this function can be executed.

[Reference]

DcamGetOverClock , DcamSetTriggerMode, DcamGetTriggerMode

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamSetOverClock (10) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.1.48 DcamGetOverClock: Function to retrieve the over clock

BOOL DcamGetOverClock (INT *pClock)

[Summary]

Retrieves the over clock.

[Arguments]

pClock	Specify the address of the variable where the over clock is to be stored

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None.

[Reference]

DcamSetOverClock

[Example]

The following example shows how this function is called.

```
INT nClock;
DWORD dwErrCode;
if(DcamGetOverClock (&nClock) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.1.49 DcamSetMPPMode: Function to Set MPP mode

BOOL DcamSetMPPMode(INT nMode)

[Summary]

Sets the MPP mode.

[Arguments]

nMode	Specify the MPP mode from among the following modes. DCAM_CCDMPPMODE_OFF : MPP mode is off. DCAM_CCDMPPMODE_ON : MPP mode is on

[Return Value]

If the funciotn is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

When the MPP mode is changed, "Line Time" and "Exposure Time" are updated by hardware automatically. After calling this function, please acquire the "Line Time" and "Exposure Time" from hardware.

[Reference]

DcamGetMPPMode

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
If( DcamSetMPPMode(DCAM_CCDMPPMODE_ON)!= TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.50 DcamGetMPPMode: Function to retrieve the MPP mode

BOOL DcamGetMPPMode(INT* pMode)

[Summary]

Retrieves the MPP mode.

[Arguments]

pMode	Specify the address of the variable where the MPP mode is to be stored. One of the following values is obtained. DCAM_CCDMPPMODE_OFF : MPP mode is off. DCAM_CCDMPPMODE_ON : MPP mode is on

[Return Value]

If the function is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetMPPMode

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nMode
If( DcamGetMPPMode(&nMode)!= TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.51 DcamSetLineTime: Function to Set the Line Time

BOOL DcamSetLineTime(INT nTime)

[Summary]

Sets the Line Time.

[Arguments]

nTime	Specify the line time. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the function is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetLineTime

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
If( DcamSetLineTime(100) TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.52 DcamGetLineTime: Function to retrieve the Line Time

BOOL DcamGetLineTime(INT* pTime)

[Summary]

Retrieves the Line Time.

[Arguments]

pTime	Specify the address of the variable where the line time is to be stored.

[Return Value]

If the funciotn is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetLineTime

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nTime ;
If( DcamGetLineTime(&nTime) TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.53 DcamSetIntegralCapacity: Function to Set the integral capacity

BOOL DcamSetIntegralCapacity (INT nType)

[Summary]

Settings indicate the type of integration capacity.

[Arguments]

nType	Settings indicate the type of integration capacity. Please refer to the "DCamUSB_FunctionParameterList_J.pdf" for input range.

[Return Value]

If the funciotn is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetIntegralCapacity

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
If(DcamSetIntegralCapacity(0) TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.54 DcamGetIntegralCapacity: Function to gets the integral capacity

BOOL DcamGetIntegralCapacity (INT* pType)

[Summary]

Get the type of integration capacity.

[Arguments]

pType Specifies the address of a variable that integration capacity to store.

[Return Value]

If the funciotn is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetIntegralCapacity

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nType
If(DcamGetIntegralCapacity(&nType) TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.55 DcamSetDriveMode: Function to set CCD drive mode

BOOL DcamSetDriveMode(INT nMode, INT nTimeout)

[Summary]

Sets the CCD drive mode.

[Arguments]

nMode	Specify the CCD drive mode. Either one of the following can be specified. DCAM_CCDDRVMODE_SUSPEND : Suspend DCAM_CCDDRVMODE_STANDBY : Standby
nTimeout	Specify the timeout(ms) for sets the CCD drive mode.

[Return Value]

If the function is successful, the return value is TRUE(1). Otherwise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetDriveMode

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamSetDriveMode(DCAM_CCDDRVMODE_SUSPEND, 0) != TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.56 DcamGetDriveMode: Function to retrieve the CCD drive mode

BOOL DcamGetDriveMode(INT* pMode)

[Summary]

Retrieves the TG pulse width.

[Arguments]

pMode	Specify the address of the variable where the CCD drive mode is to be stored. Either one of the following values is obtained. DCAM_CCDDRVMODE_SUSPEND : Suspend DCAM_CCDDRVMODE_STANDBY : Standby

[Return Value]

If the function is successful, the return value is TRUE(1). Otherwise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetDriveMode

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nMode = 0;
if( DcamGetDriveMode(&nMode)!= TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.57 DcamSetElectronicShutter: Function to set electronic shutter mode

BOOL DcamSetElectronicShutter (INT nMode)

[Summary]

Sets the electronic shutter mode.

[Arguments]

nMode	Specify the electronic shutter mode from among the following modes. DCAM_CCDESHUTTER_OFF : Electronic shutter mode is off. DCAM_CCDESHUTTER_ON : Electronic shutter mode is on.

[Return Value]

If the funciotn is successful. The return value is TRUE(1). Otherwise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetElectronicShutter

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
If( DcamSetElectronicShutter (DCAM_CCDESHUTTER_ON)!= TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.58 DcamGetElectronicShutter: Function to retrieve the electronic shutter mode

BOOL DcamGetElectronicShutter (INT* pMode)

[Summary]

Retrieves the electronic shutter mode.

[Arguments]

pMode	Specify the address of the variable where the electronic shutter mode is to be stored. DCAM_CCDESHUTTER_OFF : Electronic shutter mode is off. DCAM_CCDESHUTTER_ON : Electronic shutter mode is on.

[Return Value]

If the function is successful. The return value is TRUE(1). Otherewise, the return value is FALSE(0).

For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetElectronicShutter

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nMode;
If( DcamGetElectronicShutter (&nMode)!= TRUE){
    dwErrCode = DcamGetLastError();
}
```

6.1.59 DcamSetSensorSignalPulseWidth : Function to Set TG pulse width

BOOL DcamSetSensorSignalPulseWidth(INT nSignalSensor, INT nWidth) ;

[Summary]

Sets the TG pulse width.
This value becomes the bottom value of the exposure time. When this value became bigger than exposure time, please change exposure time by all means.

[Arguments]

nSignalSecsor	Specify 0.
nWidth	Specifies the TG pulse width in the range from 2 to 500[usec].

[Return Value]

If the funciotn is successful. The return value is TRUE(1).
Otherewise, the return value is FALSE(0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamGetSensorSignalPulseWidth

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
If( DcamSetSensorSignalPulseWidth( 0, 40 ){
        dwErrCode = DcamGetLastError();
}
```

6.1.60 DcamGetSensorSignalPulseWidth : Function to retrieve the TG pulse width

BOOL DcamGetSensorSignalPulseWidth(INT nSignalSensor, INT* pWidth) ;

[Summary]

Retrieves the TG pulse width.

[Arguments]

nSignalSecsor	Specify 0.
pWidth	Specify the address of the variable where the TG pulse width is to be stored.

[Return Value]

If the funciotn is successful. The return value is TRUE(1).
Otherewise, the return value is FALSE(0).
For details on error information, refer to the DcamGetLastError function.

[Note]

None.

[Reference]

DcamSetSensorSignalPulseWidth

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT      nWidth
If( DcamGetSensorSignalPulseWidth( 0, &nWidth ){
        dwErrCode = DcamGetLastError();
}
```

6.2 DcamTmpCtrl

6.2.1 DcamTmpCtrlInitialize : Function to initialize the library

BOOL DcamTmpCtrlInitialize(VOID)

[Summary]

Initialize the library.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

Always run this function first before running other functions.
An error occurs if the library has already been initialized.
Only one process can use this library.

[Reference]

DcamTmpCtrlUninitialize

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;
if(DcamTmpCtrlInitialize () != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.2 DcamTmpCtrlUninitialize : Function to uninitialize the library

BOOL DcamTmpCtrlUninitialize (VOID)

[Summary]

Unloads the library resources and closes the device driver.

[Arguments]

None.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

Call this function when quitting the program or when the DcamTmpCtrl library is not required.

[Reference]

DcamTmpCtrlInitialize

[Example]

The following example shows how this function is called.

```
DWORDdwErrCode;
if(DcamTmpCtrlUninitialize () != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.3 DcamTmpCtrlGetLastError : Function to retrieve the last error code

DWORD DcamTmpCtrlGetLastError (VOID)

[Summary]

Retreives the last-error code.

[Arguments]

None.

[Return Value]

The last error code is returned. For details on error code, refer to the error code Table.

[Note]

None.

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
dwErrCode = DcamTmpCtrlGetLastError ();
```

6.2.4 DcamTmpCtrlSetCoolingControl : Function to Set the cooling control status

BOOL DcamTmpCtrlSetCoolingControl (BOOL bOnOff)

[Summary]

Sets the cooling control status.

[Arguments]

bOnOff — Specify the Cooling control status from among the following modes.

DCAM_COOLING_CONTROL_OFF : Cooling control off
DCAM_COOLING_CONTROL_ON : Cooling control on

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlGetCoolingControl

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamTmpCtrlSetCoolingControl (DCAM_COOLING_CONTROL_ON) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.5 DcamTmpCtrlGetCoolingControl : Function to retrieve the cooling control status

BOOL DcamTmpCtrlGetCoolingControl (BOOL *pbOnOff)

[Summary]

Retrieves the cooling control status.

[Arguments]

pbOnOff — Specify the address of the variable where the currently set the cooling control status is to be stored.

DCAM_COOLING_CONTROL_OFF : Cooling control off
DCAM_COOLING_CONTROL_ON : Cooling control on

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlSetCoolingControl

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
BOOL bOnOff;
if(DcamTmpCtrlGetCoolingControl (&bOnOff) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.6 DcamTmpCtrlLoadCoolingTemperature : Function to Load the cooling temperature

BOOL DcamTmpCtrlLoadCoolingTemperature (Void)

[Summary]

Load the cooling temperature.

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlSaveCoolingTemperature

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamTmpCtrlLoadCoolingTemperature () != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.7 DcamTmpCtrlSaveCoolingTemperature : Function to save the cooling temperature

BOOL DcamTmpCtrlSaveCoolingTemperature (Void)

[Summary]

Save the cooling temperature.

[Arguments]

None

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlLoadCoolingTemperature

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamTmpCtrlSaveCoolingTemperature () != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.8 DcamTmpCtrlGetCoolingTemperature : Function to retrieve the coolingTemperature

BOOL DcamTmpCtrlGetCoolingTemperature(INT *pValue)

[Summary]

Retrieves the cooling temperature.

[Arguments]

pValue	Specify the address of the variable where the cooling temperature is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlSetCoolingTemperature

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nValue;
if(DcamTmpCtrlGetCoolingTemperature (&nValue) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.9 DcamTmpCtrlSetCoolingTemperature : Function to Set the cooling temperature

BOOL DcamTmpCtrlSetCoolingTemperature (INT nValue)

[Summary]

Sets the cooling temperature.

[Arguments]

nValue Specify the cooling temperature.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

DcamTmpCtrlGetCoolingTemperature

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
if(DcamTmpCtrlSetCoolingTemperature (10) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.10 DcamTmpCtrlGetCoolingTemperatureCurrent: Function to retrieve the cooling current temperature

BOOL DcamTmpCtrlGetCoolingTemperatureCurrent (INT *pValue)

[Summary]

Retrieves the cooling current temperature.

[Arguments]

pValue	Specify the address of the variable where the cooling current temperature is to be stored.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

None

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nValue;
if(DcamTmpCtrlGetCoolingTemperatureCurrent (&nValue) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.11 DcamTmpCtrlGetCoolingStatus : Function to retrieve the cooling temperature status

BOOL DcamTmpCtrlGetCoolingStatus (INT *pValue)

[Summary]

Retrieves the cooling temperature status.

[Arguments]

pValue — Specify the address of the variable where the cooling status is to be stored.
One of the following values is obtained.

DCAM_COOLING_STATUS_NORMAL : The cooling temperature is normal.
DCAM_COOLING_STATUS_LOWER : The cooling temperature is lower.
DCAM_COOLING_STATUS_HIGHER : The cooling temperature is higher.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

None

[Example]

The following example shows how this function is called.

```
DWORD dwErrCode;
INT nValue;
if(DcamTmpCtrlGetCoolingStatus (&nValue) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

6.2.12 DcamTmpCtrlGetThermistorStatus : Function to retrieve the Thermistor status

BOOL DcamTmpCtrlGetThermistorStatus(INT *pValue)

[Summary]

Retrieves the Thermistor status.

[Arguments]

pValue — Specify the address of the variable where the thermistor status is to be stored.
One of the following values is obtained.

DCAM_THERMISTOR_STATUS_NOERROR	:	Thermistor no error.
DCAM_THERMISTOR_STATUS_ERROR	:	Thermistor error.
DCAM_THERMISTOR_STATUS_OVER	:	Thermistor temperature over.

[Return Value]

If the function is successful, the return value is TRUE (1). Otherwise, the return value is FALSE (0).
For details on error information, refer to the DcamTmpCtrlGetLastError function.

[Note]

None

[Reference]

None

[Example]

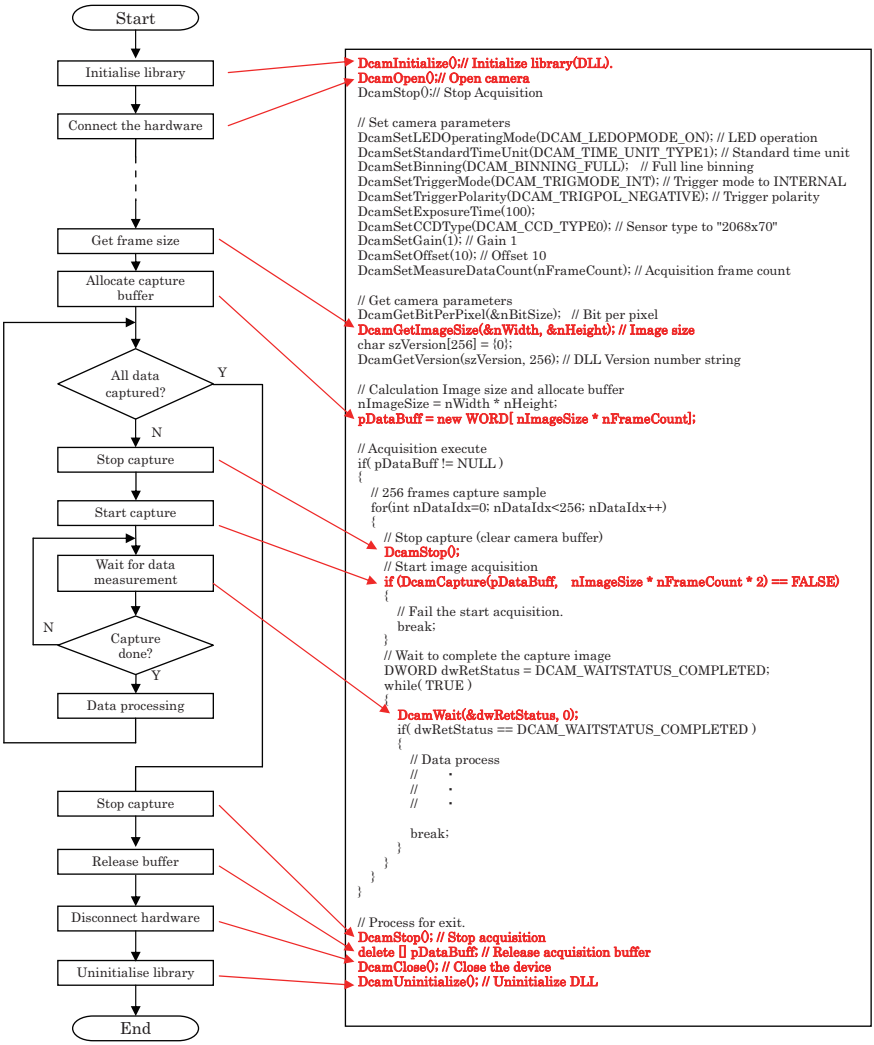
The following example shows how this function is called.

```
DWORD dwErrCode;
INT nValue;
if(DcamTmpCtrlGetThermistorStatus (&nValue) != TRUE){
    dwErrCode = DcamTmpCtrlGetLastError ();
}
```

7 Supplementary explanation

7.1 About the flow of data acquisition

The flow of following steps are the flow of data acquisition.



7.2 About the using of the "DcamStop()"

The functions of "DcamStop()" are following.

a) Stop capturing of PC.
b) Stop data acquisition process and reset data memory in hardware.

And therefore, the sample program of this library use the "DcamStop()" at three times of following.

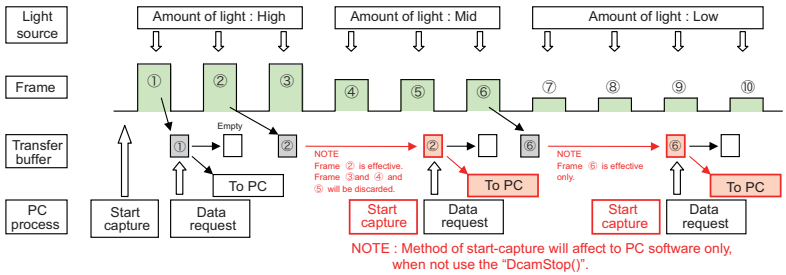
a) Start of capture process. (After calling the "DcamConnect()".)
b) Process of the capturing image. (Before calling the "DcamCapture()".)
c) End of capture process. (Before to release capture buffer.)

"DcamStop ()" is not necessary method.
But, there is a difference of capture data following two case.

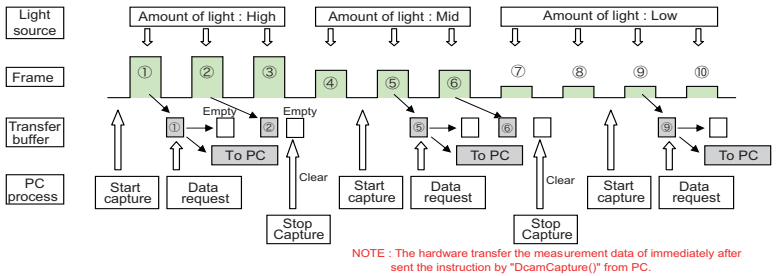
a) DcamCapture() -> DCamWait() -> DcamCapture() ...
b) DcamCapture() -> DCamWait() -> Dcamstop() -> DcamCapture() ...

Please refer to the following and please be careful.

■ Case of not use the "DcamStop()"



■ Case of use the "DcamStop()"



7.3 About operation of the device connection and remove.

The DCamUSB library has the function (DcamGetDeviceState) which checks the connection state of a device.
When the device (or the USB cable) is removed, the application uses this function to confirm the connection state of a device.

The application recognizes the moment when a device is removed.
Because when a device is removed, the OS sends the device change message (WM_DEVICECHANGE) to the top Window of the application.

When an application receives this message, it confirms connection state of a device by a DcamGetDeviceState function. By the status type acquired from this function, the application understands connected device existence.

When the device is disconnected, the state of the device is confirmed and if no device found (DCAM_DEVSTATE_NODEVICE) please perform the disconnect process (DcamClose, DcamUninitialize) in the library.
When a device is connected, the OS recognizes a change of a device, and OS sends a message to the application. The application checks the connection state of a device, and, if a device is found connection process starts.

In the above-mentioned operation, there are some attentions.

1) When a device is connected and removed frequently, it may become the cause of failure of the device. Please use it after understanding the specifications of the device.

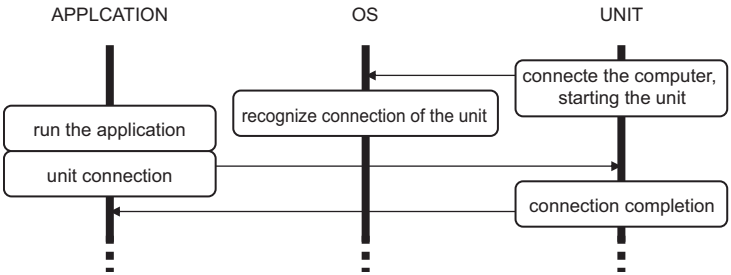
2) In the operation, which removes or connects a device, the message which notifies, state change of the device transmitted from OS is not necessarily surely transmitted only once.
It may transmit two or more times by the device. Moreover, a message is transmitted also for operation of other devices. (For example, when CD is inserted in CD drives etc.) . Please consider this in the case of programming.

3) When the device is removed and reconnected, the settings of the device changes to initial values in contrast to the settings of the device in the application. When the device is reconnection, and the settings before disconnection is required, application holds the settings of the device, and so please set it after reconnection.

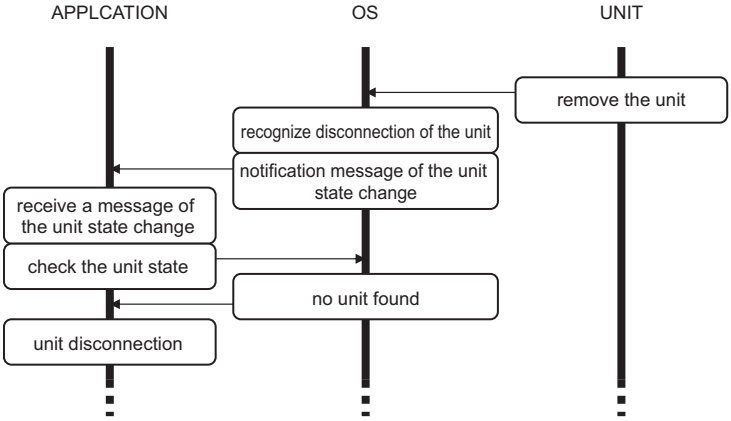
   However depending upon the specification of device there are also devices, which requires time for starting when connected to PC. Even after reconnecting, there may be situations when setting canÅft be done, because device is in the starting process. Please use it after understanding the specifications of a device.

Chart below is a basic procedure when the device connects and removes.

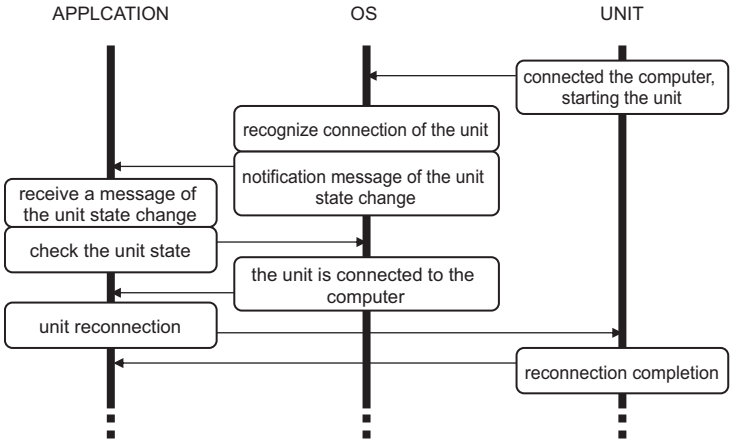
- [The device connection]



- [The device is remove]



- [The device is reconnection]



[Example]

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,
                                        WPARAM wParam, LPARAM lParam)
{
    INT      nState;    // Device State
     :
     :
    switch( message ) {
        :
        :
    case WM_DEVICECHANGE:
         // Get Device State
         if( DcamGetDeviceState(&nState) ) {
              if(nState == DCAM_DEVSTATE_NODEVICE) {    // No device found
                DisconnectionDevice();
         } else if(nState == DCAM_DEVSTATE_DEVICE) { // Device found
                ConnectionDevice();
         }
    }
    break;
     :
```

Document History

Date	Document Revision	Contents
01.Jun.2009	1.00	First Edition
23.Jul.2009	1.10	Added the description of the following functions. • DcamSetStandardTimeUnit • DcamGetStandardTimeUnit
08.Oct.2009	1.20	Corrected misspelling. Added the description of the DCamTmpCtrl.dll.
30.May.2011	1.30	Added the description of the following functions. • DcamSetMPPMode • DcamGetMPPMode • DcamSetLineTime • DcamGetLineTime • DcamSetIntegralCapacity • DcamGetIntegralCapacity • DcamSetSensorSignalPulseWidth • DcamGetSensorSignalPulseWidth
15.Jul.2011	1.40	Delete the description of the following functions. • DcamSetSensorSignalPulseWidth • DcamGetSensorSignalPulseWidth Added the description of the following functions and constant. • DcamSetDriveMode • DcamGetDriveMode • DCAM_CCDMPPMODE_OFF • DCAM_CCDMPPMODE_ON
28.Sep.2011	1.41	Update the version of "DCamUSB" Update the version of "DCamTmpCtrl" Modify the "Required Files".
11.Nov.2011	1.42	Added the description of the follwing functions. • DcamSetElectronicShutter • DcamGetElectronicShutter
01.May.2012	1.43	Updated the description of the following functions. • DcamGetImageSize • DcamGetCCDType • DcamSetCCDType • DcamSetMeasDataCount • DcamGetMeasDataCount • DcamGetCaptureByte • DcamCapture • DcamCaptureReverseX • DcamStop • DcamWait Added the description of the following function. • DcamStopEx • DcamGetTotalCaptureBytes Added the description of the following explanation. • 7.1 About the flow of data acquisition • 7.2 About the using of the "DcamStop()"
28.Jun.2012	2.00	Delete the description of Wndows2000ÂAand added the description of Windows7.
25.Feb.2014	2.01	Added the description of the follwing functions. • DcamSetSensorSignalPulseWidth • DcamGetSensorSignalPulseWidth
29.Jul.2014	2.02	Edited by the end of support of WindowsXP. Add list of offices on the back cover.

Driver Circuit for Image Sensor Control Library
DCamUSB and DCamTmpCtrl Function Specifications

Manufacturer

Japan: Hamamatsu Photonics K.K., Solid State Division
1126-1 Ichino-cho, Higashi-ku, Hamamatsu City, 435-8558 Japan, Telephone:(81)53-434-3311, Fax:(81)53-434-5184
E-mail: export@sys.hpk.co.jp

U.S.A. and Canada: Hamamatsu Corporation, Systems Division
360 Foothill Road,Bridgewater,N.J. 08807-0910,U.S.A. Telephone: (1)908-231-1116, Fax: (1)908-231-0852
E-mail: usa@hamamatsu.com

Germany: Hamamatsu Photonics Deutschland GmbH
Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany Telephone: (49)8152-375-0, Fax: (49)8152-265-8
E-mail: info@hamamatsu.de

France: Hamamatsu Photonics France S.A.R.L.
19,Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France Telephone: (33)1 69 53 71 00, Fax: (33)1 69 53 71 10
E-mal: infos@hamamatsu.fr

United Kingdom: Hamamatsu Photonics UK Limited
2 Howard Court, 10 Tewin Road, Welwyn Garden City Hertfordshire AL7 1BW, United Kingdom Telephone: (44)1707-294888, Fax: (44)1707-325777 E-mail: info@hamamatsu.co.uk

North Europe: Hamamatsu Photonics Norden AB
Smidesvagan 12, SE-171 41 Solna, Sweden Telephone: (46)8-509-031-00, Fax: (46)8-509-031-01
E-mail: info@hamamatsu.se

Italy: Hamamatsu Photonics Italia S.R.L.
Strada della Moia, 1/E 20020 Arese (Milano), Italy Telephone: (39)02-935 81 733, Fax: (39)02-935 81 741

WEB SITE : http://www.hamamatsu.com/

Document Number : K46-B60011
Document Revision 2.02 29.Jul.2014