FASTCAM Control
SDK Library "PccLib"
Reference Manual

Rev.2.976

English Edition

PHOTRON LIMITED

Table of contents

1. SDK Introduction

This FASTCAM Control SDK Library is a Windows-based class library that controls, from the PC screen, all operational functions – setting parameters, recording and downloading of image data - of the Photron FASTCAM series high-speed video cameras. Its main functions are as follows:

■Control Functions of High-Speed Video Cameras from PC

Connects to PC via the PCI, IEEE1394, Optical or Ethernet interface.
Sets camera parameters for recording.
Selects a trigger mode to start recording.

■Download of Recorded Image Data

Downloads image data from the camera memory to PC for subsequent processing.
Corrects image data using LUT (look up table).
Stores image data in BMP, TIFF, JPEG or RAW image format.

2. How to Use the SDK Library

This section discusses how to install the FASTCAM Control SDK Library in software development environment for actual use.

To use this Library, you need to have your high-speed video camera set up for recording and your PC and necessary hardware devices connected and installed. See the relevant instruction manuals for connection and installation.
Installation of hardware driver varies by the camera model, and each hardware driver must be installed before using this Library. See the driver installation manual of each hardware device.

2.1. Who Should Read This Reference Manual

This reference manual has been prepared for users of Photron high-speed video cameras who wish to develop PC control application software programs. Such readers should have basic knowledge of and expertise in computer languages such as the C, C++ and Visual C++, and shold be able to understand the programming words used in this book.

2.2. Required Operational Environment for This Library

The following environment is required to run this Library:

PC：	PC/AT Compatible
OS:	Microsoft Windows98、98SE、Me、2000 Professional、XP Professional、XP Home ※1 ※2
CPU：	Intel Pentium or equivalent PentiumIII 1GHz or higher recommended ※2
Memory：	64MB minimum 256MB or more recommended* *Required for multiple cameras or high-resolution cameras
HDD：	5MB or more needed for installing Library 500MB or more recommended for application development works
Development Environment：	Microsoft VisualC++5.0 or later ※2
Other requirements：	Large capacity HDD or removable recording media recommended for image data storage. CD-ROM drive required for installaion.

※1 Note: The combination of camera and PC may require a particular OS. See the instruction manual of the camera

※2 Note: Trademarks or registered trademarks belong to their respective owners.

2.3. Structure of The Library

The classes contained in this library and the structure of library files needed for application development are as follows:

2.3.1. Recording Classes

CCameraControl Class (Camera Control Class)
- Setting up camera status
- Getting camera status
- Starting a recording
- Downloading recorded image data

CImageData Class (Still Image Data Management Class)
- Input and output of still image data file(in BMP, TIFF, JPEG, RAW, PNM, PNG or RAWW format)
- Converting still image data by lookup table

2.3.2 File Structure

The file structures contained in the SDK Library is as follows:

Dll¥

PccLib.dll	PccLib library main DLL
FcamPCI.dll	FASTCAM-PCI device DLL
FcamPCI2.dll	FASTCAM-PCI R2 device DLL
FcamPLMV.dll	FASTCAM -X 1280PCI device DLL
Fcam1394.dll	IEEE1394I/F-compatible FASTCAM series device DLL
FcamEth.dll	Ethernet-compatible(100Base-TX) FASTCAM series device DLL
FcamOpt.dll	Optical I/F-compatible FASTCAM series device DLL
FcamNPCI.dll	FASTCAM-512PCI device DLL
FcamPPCI.dll	FASTCAM-1024PCI device DLL
FcamGEth.dll	1000Base-T compatible FASTCAM series device DLL
GEthLib.dll	1000Base-T compatible FASTCAM series communication DLL
Ijl15.dll	JPEG Library DLL
IrigLib.dll	IRIG library DLL

Lib¥

PccLib.lib	PccLib import library file

Include¥

PccLib.h	PccLib header file
ConstantValues.h	Constant value declaration header file
CameraControl.h	CCameraControl Class header file
ImageData.h	CImageData Class header file
LutControl.h	LUT header file
IrigLib.h	IRIG header file

2.4. Installation in Development Environment

This section shows the procedure for installing the Library in an application development environment (VisualC++ 6.0/VisualC++.NET)

2.4.1. Setup for Visual C++ Project/ Visual C++ .NETProject

1) Setup for Visual C++ Project

Add the Library to Visual C++ in the following manner:

1.Start up Visual C++ and open a project in which this Library is to be installed.

2.Select “Project” from the Visual C++ menu and open the “Project Setup” dialog.

3.Select the link tab “General” and add “PccLib.lib” to “Object/Library Module”.

4.Select the link tab “Input” and add a path name, with PccLib.lib copied, to “Additional Library Path”.

5.Select “C/C++”, “Preprocessor” and then add a path name, with the header file of this Library copied, to “Include File Path”.

2) Setup for VisualC++.NET Project

Add the Library to VisualC++.NET in the following manner:

1. Start up VisualC++.NET and open a project in which this Library is to be installed.

2. Select “Project” from the VisualC++.NET menu and open the “Project Property” dialog.

3. Select the link tab “Inout” and add "PccLib.lib" to " Additional dependence file".

4. Select the link tab “General” and add a path name, with PccLib.lib copied, to “Additional Library Path”.

5. Select the C/C++ tab “General” and add a path name, with the header file of this Library copied, to “Include File Path”.

2.4.2. To run applications

When you run applications with this Library incorporated, you need to place relevant DLL files in the same folders as the execuction files or in the folders in the system where a path has been set.

Note: DLL files may be copied in the Windows system folder. In such a case, however, the specification of each file must usually be updated at software revision. If the DLL files are left without being updated, the revised SDK may erroneously refer to those old DLL files.

3. Fastcam Control SDK common items

This section presents information on the details of the SDK Library.

Items discussed in this section:

3.1. Error Codes

With the CCameraControl Class, the returned values are error codes.

Error codes are shown bellow:

PCC_ERROR_NOERROR	1	Normal
PCC_ERROR_UNINITIALIZE	-1	Not initialized
PCC_ERROR_FUNCTION_FAILED	-2	Command failed
PCC_ERROR_FUNCTION_TIMEOUT	-3	Command time out
PCC_ERROR_FUNCTION_DISABLE	-4	Disabled command or no such function
PCC_ERROR_NO_DEVICE	-5	Device not existent
PCC_ERROR_NO_DATA	-6	Image data not existent
PCC_ERROR_UNKNOWN_FRAME	-7	Frame number not existentor incorrect
PCC_ERROR_CAMERAMODE	-8	Incorrect camera mode
PCC_ERROR_NO_ENDLESS	-9	Trigger mode not endless
PCC_ERROR_FILEREAD_FAILED	-10	File reading failed
PCC_ERROR_FILEWRITE_FAILED	-11	File writing failed
PCC_ERROR_IMAGE_SIZEOVER	-12	Image size error
PCC_ERROR_FRAME_AREAOVER	-13	Frame erea error (exceeding)
PCC_ERROR_PLAYMODE	-14	Playback mode error
PCC_ERROR_NO_CAMERALIB	-15	Unknown camera library (not found)
PCC_ERROR_NO_MCDL	-16	MCDL data not existent in camera
PCC_ERROR_NO_IRIG	-17	IRIG data not existent in camera
PCC_ERROR_NOT_SUPPORTED	-18	Specified command not supported

3.2. Camera Model Codes

[GetDeviceType] member function of the CcameraControl class gets the model code of cameras being used.

The model codes and the corresponding model names are as follows:

PCAM_TYPE_FASTCAM_NET_COLOR	FASTCAM-Super (color model)
PCAM_TYPE_ULTIMA_1024	FASTCAM-Ultima1024 (monochrome model)
PCAM_TYPE_ULTIMA_1024_ZERO	FASTCAM-Ultima1024 (monochrome model)
PCAM_TYPE_ULTIMA_1024_COLOR	FASTCAM-Ultima1024 (color model)
PCAM_TYPE_ULTIMA_1024_COLOR_ZERO	FASTCAM-Ultima1024 (color model)
PCAM_TYPE_ULTIMA_SE	FASTCAM-UltimaSE (monochrome model)
PCAM_TYPE_ULTIMA_SE_COLOR	FASTCAM-UltimaSE (color model)
PCAM_TYPE_FASTCAM_PCI	FASTCAM-PCI (monochrome model)
PCAM_TYPE_FASTCAM_PCI_COLOR	FASTCAM-PCI (color model)
PCAM_TYPE_FASTCAM_1280PCI	FASTCAM-X1280PCI (monochrome model)
PCAM_TYPE_FASTCAM_1280PCI_COLOR	FASTCAM-X1280PCI (color model)
PCAM_TYPE_FASTCAM_MAX	FASTCAM-UltimaAPX (monochrome model)
PCAM_TYPE_FASTCAM_MAX_COLOR	FASTCAM-UltimaAPX (color model)
PCAM_TYPE_FASTCAM_NEO	FASTCAM-Ultima512 (monochrome model)
PCAM_TYPE_FASTCAM_NEO_COLOR	FASTCAM-Ultima512 (color model)
PCAM_TYPE_FASTCAM_PCI2	FASTCAM-PCI R2 (monochrome model)
PCAM_TYPE_FASTCAM_PCI2_COLOR	FASTCAM-PCI R2 (color model)
PCAM_TYPE_FASTCAM_512PCI	FASTCAM-512PCI (monochrome model)
PCAM_TYPE_FASTCAM_512PCI_COLOR	FASTCAM-512PCI (color model)
PCAM_TYPE_FASTCAM_APX_RS	FASTCAM-APX RS (monochrome model)
PCAM_TYPE_FASTCAM_APS_RS_COLOR	FASTCAM-APX RS (color model)
PCAM_TYPE_FASTCAM_1024PCI	FASTCAM-1024PCI (monochrome model)
PCAM_TYPE_FASTCAM_1024PCI_COLOR	FASTCAM-1024PCI (color model)

3.3. Camera Parameters

All the information pertaining to the camera is included in the CAMERA_PARAMS structure. It is possible to change recording rate and other parameters by directly resetting the CMARA_PARAMS structure, but it may take much time to reset all the items with some of the camera models. So, it is advisable to use CMARA_PARAMS for data reference only, and use individual setting functions for camera setup. Model-dependant information or information on extra commands are found in the CAMERA_PARAM_EX structure.

For parameters common with CAMERA_PARAMS and CAMERA_PARAMS_EX, it is advisable to use the CAMERA_PARAM_EX structure as much as possible.

3.3.1[CAMERA_PARAMS] Structure

Note: LMN stands for LIST_MAX_NUMBER

typedef struct {

char	device_name[64];	Camera name
int	device_type;	Camera model code
int	device_version	Camera hardware version
int	color;	Color mode（0:MONO 1:COLOR）
int	session_number;	Recording session number
int	max_frame_num;	Number of frames
int	camera_mode;	Camera mode
BOOL	is_live;	Display mode（0:PLAY 1:LIVE）
BOOL	is_record_ready;	Record ready（0:OFF 1:ON）
int	record_rate;	Recording rate
int	record_rate_num;	Number of recording rates (list)
int	record_rate_list[LMN];	Recording rate list
DWORD	resoluton;	Resolution
int	resolution_num;	Number of resolutions (list)
DWORD	resolution_list[LMN];	Resolution list
int	shutter_speed;	Shutter speed
int	shutter_num;	Number of shutter speeds (list)
int	shutter_list[LMN];	Shutter speed list
int	trigger_mode;	Trigger mode
int	trigger_frame;	Trigger frame
int	manual_trigger_frame;	Manual trigger frame
int	random_trigger_times	Random trigger time
int	random_manual_frame	Random manual frame
int	random_trigger_num;	Random trigger frame number
int	manual_trigger_pos;	Manual trigger position
int	trigger_mode_num;	Number of trigger modes (list)
int	trigger_mode_list[LMN];	Trigger mode list
int	random_number_num;	Number of triggers for random trigger
int	random_number_list[30];	Number of frames for random trigger
BOOL	enable_random_trigger;	Enable random trigger Yes/No
BOOL	enable_manual_trigger;	Enable manual trigger Yes/No
BOOL	enable_random_manual	Enable random man trigger Yes/No
int	gamma_value;	Gamma value
BOOL	enable_gamma;	Enable gamma Yes/No
int	gamma_value_num	Number of gamma lists
int	gamma_value_list[LMN]	Gamma list
int	gain_level;	Gain level
BOOL	enable_gain;	Enable gain Yes/No
int	gain_level_num	Number of gain lists
int	gain_level_lits[LMN]	Gain list
BOOL	external_sync;	External sync

```
    BOOL    external_sync_nega;          External sync neg signal mode
    int     external_sync_in             External sync in
    int     external_sync_out            External sync out
    BOOL    enable_external_sync;        Enable external sync Yes/No
    BOOL    enable_external_sync_in;     Enable external sync in Yes/No
    BOOL    enable_external_sync_out;    Enable external sync out Yes/No
    BOOL    enable_external_in;          General purpose ext input Yes/No
    BOOL    enable_external_out          General purpose ext output Yes/No
    int     external_in_num              Number of gen purpose ext input
    int     external_out_num             Number of gen purpose ext output
    int     external_in[LMN]             Gen purpose ext input list
    int     external_out[LMN]            Gen purpose ext output list
    int     color_temp_mode;             Color temperature mode
    int     color_temp_mode_num;         Number of color temperature modes
    int     color_temp_mode_list[LMN];   Color temperature mode list
    int     color_temp_user_num;         Number of user-set modes
    int     color_temp_user[LMN][3];     User-set color temperature (RGB)
    BOOL    enable_color_temp;           Enable color temperature Yes/No
    int     external_out[2];             External signal output mode(Port No.)
    int     external_out_num;            Number of external signal output ports
    BOOL    enable_external_out;         Enable external signal output Yes/No
    int     monitor_play_rate            External monitor playback rate
    int     monitor_out_mode;            External monitor output mode
    int     monitor_edge_mode;           Ext monitor edge enhancement mode
    BOOL    monitor_zoom_on;             External monitor output zoom mode
    BOOL    monitor_block_on;            Ext monitor output block playback
                                          mode
    int     monitor_block_start;         Block playback start position
    int     monitor_block_end;           Block playback end position
    BOOL    enable_monitor_out;          Enable external monitor output
                                          Yes/No
    BOOL    enable_monitor_out_mode;     Enabel ext monitor output setting
                                          Yes/No
    BOOL    enable_monitor_edge;         Enable ext monitor output edge
                                          enhancement Yes/No
    BOOL    enable_monitor_zoom;         Enable ext monitor output zoom
                                          Yes/No
    BOOL    mcdl_on;                     MCDL function ON/OFF setting
    BOOL    enable_mcdl;                 Enable MCDL function Yes/No
    BOOL    irig_on                      IRIG function ON/OFF setting
    BOOL    enable_irig                  Enable IRIG function Yes/No
    BOOL    restriction_time_on          Rec time restriction ON/OFF setting
    BOOL    enable_restriction_time      Enable rec time restriction Yes/No
    char    camera_type[64];             Camera type
    BOOL    exist_variablefunc;          Availability Variable mode
    Int     variable_channel             Using Variable channel no.
} CAMERA_PARAMS;
```

3.3.2. [CAMERA_PARAMS_EX] Structure

Note: PMN stands for PARTITION_MAX_NUMBER.

```
typedef struct {
    BOOL    exist_mcdl;                 Availability MCDL function Yes/No
    BOOL    exist_monitor_out;          Availability ext monitor output Yes/No
    BOOL    exist_monitor_out_mode;     Availability ext monitor output mode
                                          Yes/No
    BOOL    exist_monitor_edge;         Availability ext monitor output edge
                                          enhanncemt Yes/No
    BOOL    exist_monitor_zoom;         Availability ext monitor output zoom
                                          Yes/No
    BOOL    exist_irig;                 Availability IRIG function Yes/No
    BOOL    exist_restriction_time;     Availability rec time restriction Yes/No
    BOOL    exist_device_id;            Availability camera ID acquisition
                                          Yes/No
    BOOL    exist_reset_trigger;        Availability reset trigger Yes/No
    BOOL    exist_sensor_bitshift;      Availability sensor bitshift correction
                                          Yes/No
    BOOL    exist_partition;            Availability partitioning Yes/No
    BOOL    exist_shading_compensation; Availability shading calibration Yes/No
    BOOL    exist_live_resolution;      Availability live resolution change
                                          Yes/No
    BOOL    exist_lut_mode;             Availability LUT change Yes/No
    BOOL    exist_user_lut;             Availability user-set LUT Yes/No
    BOOL    exist_chroma_mode;          Availability chroma change Yes/No
    BOOL    exist_ds_shutter;           Availability dynamic range change
                                          Yes/No
    BOOL    monitor_block_on;           Ext monitor block playback mode ON
    int     monitor_block_start;        Ext monitor block playback start frame
    int     monitor_block_end;          Ext monitor block playback end frame
    int     monitor_play_rate;          Ext monitro playback rate
    int     monitor_out_mode;           Ext monitor output mode
    int     monitor_edge_mode;          Ext monitor output edge enhancement
                                          mode
    BOOL    monitor_zoom_on;            Ext monitor output zoom
    BOOL    mcdl_on;                    MCDL function ON/OFF setting
    BOOL    irig_on;                    IRIG function ON/OFF setting
    BOOL    restriction_time_on;        Rec time restriction ON/OFF setting
    int     device_id;                  Camera ID
    BOOL    reset_trigger_on;           Reset trigger ON/OFF setting
    int     sensor_bitshift;            Grayscale (bitshift) change
    int     max_partition_num;          Max number of partitions
    int     max_partition_block_num;    Number of blocks in max partition
    int     partition_frame_per_block;  Number of frames per block
    int     current_partition;          Current partition in use
    int     partition_num;              Number of partitions
    int     partition_size_list[PMN];   Partition block list
    int     shading_compensation_mode;  Shading calibration mode
    int     live_resolution_mode;       Live resolution change
    int     lut_mode;                   LUT
    int     chroma_mode;                Chroma
    int     ds_shutter;                 Dynamic range expansion
    int     irig_offset;                IRIG offset value
    BOOL    exist_irig_offset;          Availability IRIG offset Yes/No
    Int     hard_partition;             Hardware partition
```

```
    BOOL    exist_hard_partition;       Availability hardware partition Yes/No
    Int     shutter_mode;               Shutter mode
    BOOL    exist_shutter_mode;         Availability shutter mode Yes/No
    BOOL    II_power_on;                I.I. power
    DWORD   II_gain;                    I.I. gain
    DWORD   II_gate_select;             I.I. gate mode
    DWORD   II_gate_cycle;              I.I. cycle value
    DWORD   II_gate_width;              I.I. width value
    DWORD   II_gate_times;              I.I. times value
    DWORD   II_gate_delay;              I.I. delay value
    BOOL    exist_ii;                   Availability I.I.
    BOOL    exist_event_frame;          Availability event frame
    Int     head_total_num;             Total Number of exchanging head
    Int     current_head                Using head no.
    BOOL    exist_edge_enhancement      Availability Edge Enhancement
    Int     edge_enhancement            Edge Enhancement
    BOOL    exist_shutter_type2;        Availability Shutter speed type2
    Int     shutter_type2_list[LMN];    Shutter speed type2
    Int     shutter_speed_type2;        Number of Shutter speed type2
    BOOL    exist_auto_exposure;        Availability auto exposure Yes/No
    Int     auto_exposure;              Auto Exposure
    Int     auto_exposure_reso;         Auto Exposure resolution
    Int     auto_exposure_value;        Auto Exposure value
    Int     auto_exposure_range;        Auto Exposure range
    Int     auto_exposure_maxshutter;   Auto Exposure max exposure time
    BOOL    exist_ii_protection;        Availability I.I. Protection
    DWORD   II_protection;              I.I. Hardware protection level
} CAMERA_PARAMS_EX;
```

3.4. Frame Parameters

3.4.1. Frame Number Management

In this Class Library, the following two types of frame number systems are used to manage movie image data.

①Device-independent Frame

In this Class Library, the sequence of movie image data is managed in the way where the first frame is numbered “0” (zero) and the last frame “the total number of frames minus 1”.
When a frame number is used to specify a member function argument, this numbering system, Device-independent Frame, is used.

The frame number of a trigger frame varies by the trigger mode as shown below:

Example: When the total number of frames is 100:

Trigger Mode	Trigger Frame
START CENTER END MANUAL	0 50 99 ny number between 0 and 99

②Device-dependent Frame

The frame number of the start and end frames changes by the trigger mode. The particular frame where a trigger is sent is named Frame 1, regardless of the trigger mode. There is no Frame “0”, and the frame right before Frame 1 is Frame –1.
For working out applications, it is necessary to take the factor into consideration, that is, a Device-independent frame number must be converted to a Device-dependent frame number, and vice versa, as necessary.

3.4.2[FRAME_PARAMS] Structure

Parameters for frame number management are included in the FRAME_PARAMS structure.

```
typedef struct {
        int     start;          Start frame
        int     end;            End frame
        int     total;          Total number of frames
        int     area_start;     Start frame of an area
        int     area_end;       End frame of an area
        int     area_total;     Total number of frames in an area
        int     current;        Current frame
        int     trigger;        Trigger frame
        int     key;            Key frame
} FRAME_PARAMS;
```

3.5. IRIG Timecode

3.5.1.[**IRIGLIB_TIMECODE**] Structure

IRIG parameters are included in the IRIGLIB_TIMECODE Structure.

```
typedef struct {
        UINT    year;           Year
        UCHAR   month;          Month
        UCHAR   day;            Day
        UINT    day_of_year;    Day 1 (Jan. 1) to Day 365 or 366 (Dec. 31)
        UCHAR   hour;           Hours
        UCHAR   minute;         Minutes
        UCHAR   second;         Seconds
        LONG    below_second;   Micro seconds
} IRIGLIB_TIMECODE;
```

Note: Normally, year, month and day are not used.

4.CCameraControl Class

This section describes the features and member function specifications the CcameraControl Class in the following subsections:

4.1. CCameraControl Class Overview

4.1.1. About CCameraControl Class

The CCameraControl Class extends overall control on camera parameter setting and opearation including recording. It also controls playback and displays recorded image on the monitor screen while downloading the image data to the PC memory.
Note: There may be functions that cannot be used with a camera having no monitor output.

Each camera requires its own CCameraControl class. An ID number is assigned to the camera from an **Init** member function after a CcameraControl class instance has been genrerated. The assigned camera will be controlled by the instance until **Exit** member function is called out.

■Class Member Functions

- Initialization Function
- Camera Information Get Function
- Camera Setup Function
- Camera Setup Get Function
- Camera Mode Function
- Recording Function
- Playback Function
- Playback Setup Function
- Image Data Get Function
- Other Special Functions

4.1.2. Extra Commands

Extra commands are issued for each of camera models using the CcameraControl Class [ExtraCommand] member functions.

4.2. CCameraControl Class Member Function List

List of Member Functions of CCameraControl Class

Construction	
CCameraControl	Construct CCameraControl Class
Initialization	
GetNumberOfDevice	Get the number of cameras connected
Init	Initialize specified cameras
Exit	End processing
IsInit	Confirm initialization status
Get Camera Information	
GetDeviceName	Get the camera name
GetDeviceType	Get the camera model code
GetDeviceVersion	Get the camera hardware version information
IsColor	Identify the camera is color or monochrome
HaveGammaCorrection	Gamma correction feature Yes/No
HaveGainLevel	Gain level setup feature Yes/No
HaveColorTemperature	Color temperature setup feature Yes/No
HaveExternalSyncMode	External sync drive feature Yes/No
HaveExternalSyncInMode	External sync-in feature Yes/No
HaveExternalSyncOutMode	External sync-out feature Yes/No
HaveExternalIntput	External general purpose signal-in feature Yes/No
HaveExternalOutput	External general purpose signal-out feature Yes/No
HaveEdgeEnhancement	Edge enhancement feature Yes/No (on monitor output)
HaveZoomMode	Zoom display feature Yes/No (on monitor output)
HaveBlockPlay	Block (range) playback feature Yes/No (on monitor output)
HaveMonitorOutMode	Extermal monitor output feature Yes/No
HaveMCDL	MCDL feature Yes/No
HaveIRIG	IRIG feature Yes/No

Camera Setups	
SetCameraParams	Set camera parameters
SetCameraParamsEx	Set camera parameters simultaneously
SetRecordRate	Set frame rate
SetShutterSpeed	Set shutter speed
SetTriggerMode	Set trigger mode
SetRandomTriggerNumber	Set random trigger frame number
SetManualTriggerPosition	Set manual trigger frame position
SetRandomManualPosition	Set random-manual trigger frame position
SetResolution	Set resolution
SetGammaCorrection	Set gamma
SetGainLevel	Set gain level
SetColorTemperature	Set color temperature
SetColorTempUser	Set user color temperature
SetExternalSyncMode	Set external sync drive (Mater/Slave)
SetExternalSyncInMode	Set external sync-in drive
SetExternalSyncModeEx	Set external sync drive
SetExternalSyncOutMode	Set external sync-out drive
SetExternalInMode	Set external general purpose signal-in
SetExternalOutMode	Set external general purpose signal-out
SetEdgeEnhancementMode	Set edge enhancement (on monitor output)
SetZoomMode	Set zoom mode On/Off (on monitor output)
SetMonitorOutMode	Set external monitor output setting.
SetEnableMCDL	Set MCDL feature On/Off
SetEnableIRIG	Set IRIG feature On/Off
SetPartition	Set partition No.
SetPartitionBlockList	Set partition blocks
ExtraCommand	Execute extended command
SetVariableSetting	Set Variable mode channel parameter
SetGeometricConvert	Set output image rotation and mirroring.
SetAutoExposure	Set values for Auto Exposure function

Get Camera Setups	
GetMaxFrame	Get max frame count to record
GetCameraParams	Get camera parameters
GetCameraParamsEx	Get extended camera parameters
UpdateCameraParams	Get current camera parameters
UpdateCameraParamsEx	Get current extended camera parameters

GetRecordRate	Get recording rate
GetShutterSpeed	Get shutter speed
GetTriggerMode	Get trigger mode
GetRandomTriggerNumber	Get random trigger frame number
GetManualTriggerPosition	Get manual trigger position
GetRandomManualPosition	Get random-manual trigger position
GetResolution	Get resolution
GetGammaCorrection	Get gamma correction value
GetGainLevel	Get gain level
GetColorTemperature	Get color temperature
GetColorTempUser	Get user color temperature
GetExternalSyncMode	Get external sync drive setting
GetExternalSyncInMode	Get external sync-in drive setting
GetExternalSyncModeEx	Get external sync drive ex setting
GetExternalSyncOutMode	Get external sync-out drive setting
GetExternalInMode	Get external general purpose signal-in setting
GetExternalOutMode	Get external general purpose signal-out setting
GetEdgeEnhancementMode	Get edgenhancement setting (on monitor output)
GetZoomMode	Get zoom display On/Off (on monitor output)
GetMonitorOutMode	Get external monitor output setting
GetEnabletMCDL	Get MCDO feature enable On/Off
GetEnabletIRIG	Get IRIG feature enable On/Off
GetRecordRateList	Get recording rate list
GetShutterSpeedList	Get shutter speed list
GetTriggerModeList	Get trigger mode list
GetResolutionList	Get resolution list
GetRandomTriggerNumberList	Get random trigger frame number list
GetColorTemperatureList	Get color temperature mode list
GetGammaCorrectionList	Get gamma correction value list
GetGainLevelList	Get gain level list
GetPartitionInfo	Get partition information
GetPartition	Get current partition ID number
GetPartitionBlockList	Get partition block list
GetFrameParams	Get frame management information
GetShutterSpeedType2List	Get shutter speed type2 list
GetShutterSpeedType2	Get shutter speed type2
GetHeadName	Get camera head name

GetVariableSetting	Get Variable mode channel parameter
GetVariableCamMode	Get Variable mode
GetGeometricConvert	Get status of rotation and mirroring of output image
GetAutoExposureArea	Get target area of Auto Exposure function
GetAutoExposureParam	Get image output level of Auto Exposure function
GetShutterSpeedAEList	Get available exposure periods for Auto Exposure
Camera Mode	
OnLive	Switch display Live/Playback
IsLive	Get display status Live/Playback
GetCameraMode	Get camera status
Recording	
OnRecordReady	Set record ready On/Off
OnRecord	Recording Start/Stop
IsRecordReady	Get record reasy status On/Off
IsRecord	Get status recording/pause
TriggerIn	Software trigger input (other than START mode)
IsEndlessRec	Get endless recording status
playback	
Play	Play (on monitor output)
Pause	Pause (on monitor output)
Stop	Stop (on monitor output)
StepFoward	Jog forward (on monitor output)
StepBack	Jog backward (on monitor output)
GoAnyFrame	Go to specified frame (on monitor output)
GoStart	Go to start frame (on monitor output)
GoEnd	Go to end frame (on monitor output)
GoTrigger	Go to trigger frame (on monitor output)
Playback Setting	
SetPlayRate	Set playback rate (on monitor output)
SetBlockArea	Set block (range) (on monitor output)
SetBlockMode	Set block (range) playback On/Off (on monitor output)
GetPlayRate	Get playback rate (on monitor output)
GetBlockArea	Get block (range) playback (on monitor output)
GetBlockMode	Get block (range) playback On/Off (on monitor output)

Getting Image Data	
TransferFrame	Get image data of specified frame
TransferRawBayer	Get Bayer image data of specified frame
Transfer16BitFrame	Get 16-bit image data of specified frame
Transfer16BitRawBayer	Get Bayer image data of specifies frame

Other Special Items	
TransferMCDL	Get MCDL data of specified frame
TransferIRIG	Get IRIG data of specified frame
GetTimeCodeFromFrame	Get IRIG timecode from specified frame number
ShadingCompensation	Correct shading
GetDateOfRecording	Get date of recording
GetTimeOfRecording	Get time of recording
GetIICycleRange	Get I.I. CYCLE value range
GetIIWidthRange	Get I.I. WIDTH value range
GetIITimesRange	Get I.I. TIMES value range
GetIIDelayRange	Get I.I. DELAY value range
GetVariableRateList	Get Variable mode record rate list
GetVariableMaxResolution	Get Variable mode max square resolution
GetVariableMaxWidth	Get Variable mode max width
GetVariableMaxHeight	Get Variable mode max height
GetVariableMaxFrameRate	Get Variable mode max record rate
SetGEtherConfig	Set conncetion conditions for Giga-bit Ether I/F
GetGEtherConfig	Get connection conditions for Giga-bit Ether I/F

Extra Commands	
EXTRA_SET_ENABLE_RESTRICTION_TIME	Set rec time restriction function
EXTRA_GET_ENABLE_RESTRICTION_TIME	Get rec time restriction function
EXTRA_GET_CAMERA_ID	Get camera ID
EXTRA_GET_ENABLE_RESET_TRIGGER	Get reset trigger status
EXTRA_SET_ENABLE_RESET_TRIGGER	Set reset trigger status
EXTRA_GET_SENSOR_BITSHIFT	Get grayscale setting
EXTRA_SET_SENSOR_BITSHIFT	Set grayscale setting
EXTRA_SET_LIVE_RESOLUTION	Set live resolution change
EXTRA_GET_LIVE_RESOLUTION	Get live resolution setting
EXTRA_SET_LUT_MODE	Set LUT status
EXTRA_GET_LUT_MODE	Get LUT status

EXTRA_SET_USER_LUT_PARAMS	Set LUT parameter change
EXTRA_GET_USER_LUT_PARAMS	Get LUT parameters
EXTRA_UPLOAD_LUT_DATA	Upload LUT setting to camera
EXTRA_SET_CHROMA_MODE	Set chroma setting change
EXTRA_GET_CHROMA_MODE	Get chroma setting
EXTRA_SET_DS_SHUTTER_MODE	Set dynamic range expansion mode change
EXTRA_GET_DS_SHUTTER_MODE	Get dynamic range expansion setting
EXTRA_SET_INTERFACE_INFO	Set interface setting change
EXTRA_GET_INTERFACE_INFO	Get interface setting
EXTRA_SET_SHADING_MODE	Set calibration mode
EXTRA_GET_SHADING_MODE	Get calibration mode setting
EXTRA_GET_SHADING_LOCK	Get calibration performance satus
EXTRA_SET_IRIG_OFFSET	Set IRIG offset value
EXTRA_GET_IRIG_OFFSET	Get IRIG offset value
EXTRA_SET_HARD_PARTITION	Set hard partition mode
EXTRA_GET_HARD_PARTITION	Get hard partition mode
EXTRA_SET_SHUTTER_MODE	Set shutter mode
EXTRA_GET_SHUTTER_MODE	Get shutter mode
EXTRA_SET_II_POWER	Set I.I. power mode
EXTRA_GET_II_POWER	Get I.I. power mode
EXTRA_SET_II_GAIN	Set I.I. gain value
EXTRA_GET_II_GAIN	Get I.I. gain value
EXTRA_SET_II_GATE_SELECT	Set I.I. gate mode
EXTRA_GET_II_GATE_SELECT	Get I.I. gate mode
EXTRA_SET_II_GATE_CYCLE	Set I.I. cycle value
EXTRA_GET_II_GATE_CYCLE	Get I.I. cycle value
EXTRA_SET_II_GATE_WIDTH	Set I.I. width value
EXTRA_GET_II_GATE_WIDTH	Get I.I. width value
EXTRA_SET_II_GATE_TIMES	Set I.I. times value
EXTRA_GET_II_GATE_TIMES	Get I.I. times value
EXTRA_SET_II_GATE_DELAY	Set I.I. delay value
EXTRA_GET_II_GATE_DELAY	Get I.I. delay value
EXTRA_GET_II_GAIN_LIMIT	Get I.I. Gain's under limit value
EXTRA_GET_EVENT_FRAME	Get event frame
EXTRA_SET_VARIABLE_LOAD	Load Variable mode channel
EXTRA_SET_VARIABLE_ERASE	Erase Variable mode channel
EXTRA_GET_II_GATE_WIDTH_LIMIT	Get I.I. width minimum value

EXTRA_SET_EDGE_ENHANCEMENT	Set Edge enhancement mode
EXTRA_GET_EDGE_ENHANCEMENT	Get Edge enhancement mode
EXTRA_SET_AUTO_EXPOSURE	Set status of Auto Exposure function
EXTRA_GET_AUTO_EXPOSURE	Get status of Auto Exposure function
EXTRA_SET_AUTO_EXPOSURE_MAXSHUTTER	Set max exposure period for Auto Exposure
EXTRA_GET_AUTO_EXPOSURE_MAXSHUTTER	Get max exposure period for Auto Exposure
EXTRA_SET_II_PROTECTION	Set protection level of I.I. hardware
EXTRA_GET_II_PROTECTION	Get protection level of I.I. hardware

4.3. CCameraControl Class Member Function Specifications

This section shows the feature of member functions of the CCameraControl Class.

CCameraControl( );
CCameraControl(int device_select = DEVICE_SELECT_AUTO);

Functions Generates a CCameraControl class instance.

Arguments **device_select**

Specifies the camera type.

DEVICE_SELECT_AUTO	Automatic recognition (default)
DEVICE_SELECT_1394	FASTCAM UltimaSE & FASTCAM Ultima1024 & FASTCAM Ultima APX & FASTCAM Ultima512 & FASTCAM APX RS （IEEE1394）
DEVICE_SELECT_ETHER	FASTCAM Ultima APX & FASTCAM Ultima512 （100Base-TX）
DEVICE_SELECT_G_ETHER	FASTCAM APX RS （1000Base-T）
DEVICE_SELECT_OPT	FASTCAM Ultima APX & FASTCAM Ultima512 & FASTCAM APX RS （OpticalI/F）
DEVICE_SELECT_PCI	FASTCAM PCI
DEVICE_SELECT_PCI2	FASTCAM PCI R2
DEVICE_SELECT_PLMV	FASTCAM-X 1280PCI
DEVICE_SELECT_512PCI	FASTCAM 512PCI
DEVICE_SELECT_1024PCI	FASTCAM 1024PCI

Returned values None

Remarks One CCameraControl class controls one camera.

When multiple cameras with different types of interface are present in a system and when DEVICE_SELECT_AUTO is selected, DEVICE_SELECT_1394, DEVICE_SELECT_OPT ,DEVICE_SELECT_PCI, , DEVICE_SELECT_PCI2, DEVICE_SELECT_PLMV ,DEVICE_SELECT_512PCI and DEVICE_SELECT_1024PCI are recognized in this order.

Note: DEVICE_SELECT_ETHER and DEVICE_SELECT_G_ETHER are not recognized when DEVICE_SELECT_AUTO has been selected.

int GetNumberOfDevice(int &number);

Functions Gets the maximum number of connected cameras

Arguements **number**
Specifies an integer that represents the maximum number of cameras.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks One CCameraControl class controls one camera.
See “On Error Codes” for details of error codes On Error Codes

int Init(int camera_num);

Functions Initializes CCameraControl class and specifies the cameras to be controlled.

Arguments **camera_num**
Specifies the camera number to be controlled.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks One CCameraControl class controls one camera.
The camera number begins with 1 and goes up to the total number of cameras connected to the system.
See “On Error Codes” for details of error codes On Error Codes

int Exit( );

Functions	Gets a CCameraControl class initialization status.
Arguements	None
Returned values	Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
Remarks	This function is normally called out from the default destractor: no need to call it directly. See “On Error Codes” for details of error codes On Error Codes.

int IsInit( );

Functions Confirms the initialization status of CCameraControl class.

Arguments None

Returned Values

TRUE:	Initialized
FALSE:	Not initialized

Remarks

int GetDeviceName(LPTSTR name);

Functions Gets the name of a camera to control.

Arguements **name**

Specifies the address of a LPTSTR type buffer that gets the letter string of the camera name.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes.

int GetDeviceType(int &type);

Functions Gets the camera type code of the camera to control.

Arguements **type**
Specifies the integer argument that gets the camera type code.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int GetDeviceVersion(int &version);

Functions Gets the hardware version number of the camera to control.

Arguments **version**
Specifies the integer parameter that gets the hardware version number of the camera.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int IsColor(BOOL &color);

Functions Gets information whether the camera is a color or monochrome model.

Arguments **color**

Specifies the Bool type parameter that receives the result of judging.
TRUE: Color
FALSE: Monochrome

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveGammaCorrection(BOOL &have);

Functions Tells whether or not the camera has a gamma setting function.

Arguments **have**

Specifies the Bool type parameter that receives the result of judgement.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveGainLevel(BOOL &have);

Functions Tells whether or not the camera has a gain level control function.

Arguments **have**

Specifies the Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveColorTemperature(BOOL &have);

Functions Tells whether or not the camera has a color temperature setting function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveExternalSyncMode(BOOL &have);

Functions Tells whether or not the camera has an external clock sync function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveExternalSyncInMode(BOOL &have);

Functions Tells whether or not the camera has an external clock sync input functin.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveExternalSyncOutMode(BOOL &have);

Functions Tells whether or not the camera has an external clock sync output functin.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveExternalInput(BOOL &have);

Functions Tells whether or not the camera has an external general purpose signal input functin.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveExternalOutput(BOOL &have);

Functions Tells whether or not the camera has an external general purpose signal output functin.

Arguments **have**

Specifies a Bool type parameter that receives the result of judgment.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int HaveEdgeEnhancement(BOOL &have);

Functions Tells whether or not the camera has an edge enhancement function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Has effect on monitor output signal only.
See “On Error Codes” for details of error codes On Error Codes

Note: This function is not used with present camera models. This is only included here to match the older version of SDK manual.

int HaveZoomMode(BOOL &have);

Functions Tells whether or not the camera has a zooming display function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Has effect on monitor output signal only.

See “On Error Codes” for details of error codes On Error Codes

Note: This function is not used with present camera models. This is only included here to match the older version of SDK manual.

int HaveBlockPlay(BOOL &have);

Functions Tells whether or not the camera has a block playback function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Has effect on monitor output signal only.
See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member fuctions. This is only included here to match the older version of SDK manual.
Note: This member function is not supported by FASTCAM Ultima 1024 or later models.

int HaveMonitorOutMode(BOOL &have);

Functions Tells whether the camera has an external monitor selecting function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member fuctions. This is only included here to match the older version of SDK manual.

int HaveMCDL(BOOL &have);

Functions Tells whether or not the camera has an MCDL function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member fuctions.This is only included here to match the older version of SDK manual.

int HaveIRIG(BOOL &have);

Functions Tells whether or not the camera has an IRIG function.

Arguments **have**

Specifies a Bool type parameter that receives the result of judging.

TRUE	Yes
FALSE	No

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member fuction. This is only included here to match the older version of SDK manual.

int SetCameraParams(CAMERA_PARAMS params);

Functions Sets camera parameters.

Arguments **params**

Specifies the CAMERA_PARAMS structure that has the camera parameters.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Values of unchangeable items are neglected.
All the changeable parameters are reset. It may take some time to reset depending on the camera type. It is advisable to use a different function on each camera.

See “CAMERA_PARAMS Structure” for details
CAMARA_PARAMS Structure

See “On Error Codes” for details of error codes On Error Codes

int SetCameraParamsEx(CAMERA_PARAMS_EX params);

Functions Sets extra camera parameters.

Arguments **params**

Specifies a CAMERA_PARAMS structure that has extra camera parameters.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Values of unchangeable items are neglected.
All the changeable parameters are reset. It may take some time to reset depending on the camera type. It is advisable to use a different function on each camera.

See “CAMERA_PARAMS Structure” for details
CAMARA_PARAMS_EX Structure

See “On Error Codes” for details of error codes On Error Codes

int SetRecordRate(int rate);

Functions Sets a recording rate.

Arguments **rate**

Specifies a recording rate (FPS) to be set.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Resettable rates are received from the GetRecordRateList member function.

See “On Error Codes” for details of error codes On Error Codes

Note: In FASTCAM-1024PCI, the shutter speed, which can be set up with resolution width, is changed. For the reason, after this function execution performs the GetShutterSpeedList member function, and needs to reacquire a shutter speed list. Moreover, the thing for which the present shutter speed is also re-acquired in the GetShutterSpeed member function.

int SetShutterSpeed(int speed);

Functions Sets a shutter speed.

Arguments **speed**
Specifies a shutter speed (1/speed) to set

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR

Remarks Settable speeds are received from GetShutterSpeedList member function.

See “On Error Codes” for details of error codes On Error Codes

Note: In FASTCAM-1024PCI, since the shutter speed, which can be set up with resolution width, is changed, it is careful.

int SetTriggerMode(int mode);

Functions Sets the trigger mode.

Arguments **mode**

Specifies the trigger mode to be set.

TRIGGER_START	Start Trigger
TRIGGER_CENTER	Center Trigger
TRIGGER_END	End Trigger
TRIGGER_RANDOM	Random Trigger
TRIGGER_MANUAL	Manual Trigger
TRIGGER_RANDOMRESET	Random Reset Trigger
TRIGGER_RANDOMCENTER	Random Center Trigger
TRIGGER_RANDOMMANUAL	Random Manual Trigger
TRIGGER_TWOSTAGE	Dual-speed Trigger

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR

Remarks Settable values are received from the GetTriggerModeList member function.

See “On Error Codes” for details of error codes On Error Codes

int SetRandomTriggerNumber(int number);

Functions Sets the number of frames to record per one trigger pulse in random trigger mode.

Arguments **number**
Specifies the number of frames to be set.

Returned values Returnes and error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Settable numbers are received from the GetRandomTriggerNumberList member function.

See “On Error Codes” for details of error codes On Error Codes

int SetManualTriggerPosition(int position);

Functions Sets the position of the trigger in manual trigger mode.

Arguments **position**
Specifies the trigger position (frame number) to be set.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “Frame Number” for details of frame number Frame Parameters

See the hardware manual of each camera for details of trigger modes.

See “On Error Codes” for details of error codes On Error Codes

int SetRandomManualPosition(int position);

Functions Sets the position of the trigger in random manual trigger mode.

Arguments **position**
Specifies the trigger position (frame number) to be set.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “Frame Number” for details of frame number Frame Parameters

See the hardware manual of each camera for details of trigger modes.

See “On Error Codes” for details of error codes On Error Codes

int SetResolution(DWORD resolution);

Functions Sets the resolution

Arguments **resolution**

Specifies the value of resolution to be set:
DWORD Upper 16 bits for Width
DWORD Lower 16 bits for Height

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Settable values are dependand on the recording rate.

Settable numbers are received from GetResolutionList member function.

See “On Error Codes” for details of error codes On Error Codes

Note:In FASTCAM-1024PCI, the shutter speed, which can be set up with resolution width, is changed. For the reason, after this function execution performs the GetShutterSpeedList member function, and needs to reacquire a shutter speed list. Moreover, the thing for which the present shutter speed is also re-acquired in the GetShutterSpeed member function.

int SetGammaCorrection(int mode);

Functions Sets gamma correction value.

Arguments **mode**

Specifies a gamma correction mode.

GAMMA_1_0	γ = 1.0
GAMMA_0_9	γ = 0.9
GAMMA_0_8	γ = 0.8
GAMMA_0_7	γ = 0.7
GAMMA_0_6	γ = 0.6
GAMMA_0_5	γ = 0.5
GAMMA_0_4	γ = 0.4

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
With a camera without gamma correction function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Settable numbers are received from GetGammaCorrectionList member function.

See “On Error Codes” for details of error codes On Error Codes

int SetGainLevel(int mode);

Functions Sets the gain level mode.

Arguments **mode**

Specifies the gain level mode.

GAIN_0DB	0dB
GAIN_3DB	+3dB
GAIN_6DB	+6dB
GAIN_12DB	+12dB
GAIN_18DB	+18dB
GAIN_24DB	+24dB
GAIN_30DB	+30dB
GAIN_36DB	+36dB

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On a camera without gain level setting function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Settable numbers are received from GetGainLevelList member function.

See “On Error Codes” for details of error codes On Error Codes

int SetColorTemperature(int mode);

Functions Sets color temperature mode (white balance).

Arguments **mode**

Specifies a color temperature (white balance) mode.

COLORTEMPMODE_DEF1	Default setting 1
COLORTEMPMODE_DEF2	Default setting 2
COLORTEMPMODE_USER1	User setting 1
COLORTEMPMODE_USER2	User setting 2
COLORTEMPMODE_USER3	User setting 3

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras (color or monochrome) without color temperature mode setting function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Settable values are received from 「GetColorTemperatureList」 member function.
See “On Error Codes” for details of error codes On Error Codes

Note: **COLORTEMPMODE_USER2 and COLORTEMPMODE_USER3 are not currently used. COLORTEMPMODE_DEF1** and **COLORTEMPMODE_DEF2 are set to 5100K and 3100K, respectively.**

int SetColorTempeUser(int mode, int r, int g, int b);

Functions

Sets white balance in user-set color temperature mode.

Arguments

mode

Specifies the user-set color temperature mode.

COLORTEMPMODE_USER1	user setting 1
COLORTEMPMODE_USER2	user setting 2
COLORTEMPMODE_USER3	user setting 3

r

0 to 63 Red level (default 16)

g

0 to 63 Green level (default 16)

b

0 to 63 Blue level (default 16)

Returned values

Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On cameras (color or monochrome) without color temperature setting function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks

Settable values are received from GetColorTemperatureList.

See “On Error Codes” for details of error codes On Error Codes

Note: **COLORTEMPMODE_USER2 and COLORTEMPMODE_USER3 are not currently used.**

int SetExternalSyncMode(BOOL on);

Functions Sets ON/OFF of the external sync mode.

Arguments **on**

Specifies ON/OFF of external sync

FALSE:	OFF (Master mode)
TRUE :	ON (Slave mode)

Returned values Returns an erro code. When normal, returns a PCC_ERROR_NOERROR..

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with the SetExternalSyncModeEx
member function. This is included here to match the old version of SDK manual.

int SetExternalSyncInMode(int &mode);

Functions Sets external sync mode.

Arguements **mode**

Specifies an external sync mode from:

EXTSYNC_NONE	OFF (Master mode)
EXTSYNC_POSI	ON (Slave mode with positive signal input)
EXTSYNC_NEGA	ON (Slave mode with negative signal input)
EXTSYNC_OTHERS_POSI	ON (Positive signal)
EXTSYNC_OTHERS_NEGA	ON (Negative signal)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks With cameras without Pos/Neg setting of input signal, switching between ON and OFF is only effective.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with the SetExternalSyncModeEx member function. This is only included here to match the old version of the SDK manual.

int SetExternalSyncModeEx(int mode);

Functions Sets external sync mode.

Arguements **mode**

Specifies an external sync mode from:

EXTSYNC_NONE	OFF (Master mode)
EXTSYNC_POSI	ON (Slave mode with positive signal input)
EXTSYNC_NEGA	ON (Slave mode with negative signal input)
EXTSYNC_OTHERS_POSI	ON (Positive signal)
EXTSYNC_OTHERS_NEGA	ON (Negative signal)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks With cameras without Pos/Neg setting of input signal, switching between ON and OFF is only effective.

See “On Error Codes” for details of error codes On Error Codes

int SetExternalSyncOutMode(int mode);

Functions Sets external sync output mode.

Arguements **mode**

Specifies an external sync output mode from:

EXTOUT_VSYNC_POSI	VSYNC signal (Positive going)
EXTOUT_VSYNC_NEGA	VSYNC signal (Negative going)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int SetExternalInMode(int port, int mode);

Functions Sets external general purpose signal input mode.

Arguements

port

Specifies an output port number beginning with a " 0".

mode

Specifies an output general purpose signal input mode from:

EXTIN_EVENT_POSI	Event signal (Positive going)
EXTIN_EVENT_NEGA	Event signal (Negative going)
EXTIN_TRIGGER_POSI	Trigger signal (Positive going)
EXTIN_TRIGGER_NEGA	Trigger signal (Negative going)
EXTIN_READY_POSI	Ready signal (Positive going)
EXTIN_READY_NEGA	Ready signal (Negative going)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See "On Error Codes" for details of error codes On Error Codes

int SetExternalOutMode(int port, int mode);

Functions Sets the external signal output mode.

Arguments

port

Specifies the output port number beginning with a “0”.

mode

Specifies the external general purpose signal input mode from:

EXTOUT_RECORD_POSI	Recording period signal (Positive going)
EXTOUT_RECORD_NEGA	Recording period signal (Negative going)
EXTOUT_TRIGGER_POSI	Trigger signal (Positive going)
EXTOUT_TRIGGER_NEGA	Trigger signal (Negative going)
EXTOUT_EXPOSE_POSI	Exposure period signal (Positive going)
EXTOUT_EXPOSE_NEGA	Exposure period signal (Negative going)
EXTOUT_READY_POSI	Ready signal (Positive going)
EXTOUT_READY_NEGA	Ready signal (Negative going)
EXTOUT_IRIG_RESET_POSI	IRIG reset signal (Positive going)
EXTOUT_IRIG_RESET_NEGA	IRIG reset signal (Negative going)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int SetEdgeEnhancementMode(int mode);

Functions Sets edge enhancement mode of monitor output signal

Arguments **mode**

Specifies the edge enhancement mode from:

EDGEMODE_NONE	No edge enhancement
EDGEMODE_1	Edge enhancement 1
EDGEMODE_2	Edge enhancement 2
EDGEMODE_3	Edge enhancement 3

Returned values Returns an error code. When normar, returns
a PCC_ERROR_NOERROR.
With cameras without monitor output or edge enhancement function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Effective only on the monitor output video signal.
See “On Error Codes” for details of error codes On Error Codes

Note: This member function is not used with currently supported cameras.
This is only included here to match the old version of SDK manual.

int SetZoomMode(BOOL on);

Functions Sets zooming display mode of monitor output signal.

Arguments **on**

Specifies ON/OFF of zooming display.

TRUE	ON
FALSE	OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

On cameras without zooming display function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Effective only on the monitor output video signal.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function is not used with currently supported cameras. This is only included here to match the old version of SDK manual.

int SetMonitorOutMode(int mode);

Functions Sets external monitor output mode.

Arguments **mode**

Specifies an external monitor output mode from:

MONITOROUT_VGA	VGA monitor output
MONITOROUT_NTSC	NTSC monitor output

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without external monitor output setting function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member function. This is only included here to match the old version of SDK manual.

int SetEnableMCDL(BOOL on);

Functions Sets ON/OFF of MCDL function.

Arguments **on**

Specifies ON/OFF of MCDL function.

TRUE	MCDL function ON
FALSE	MCDL function OFF

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On cameras without MCDL function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member function. This is only included here to match the old version of SDK manual.

int SetEnableIRIG(BOOL on);

Functions Sets ON/OFF of IRIG function.

Arguments **on**

Specifies ON/OFF of IRIG function.

TRUE	IRIG function ON
FALSE	IRIG function OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without IRIG function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with ExtraCommand member function. This is only included here to match the old version of SDK manual.

int SetPartition(int partition);

Functions Sets partition numbers to use.

Arguements **partition**
Partition number beginning with “1”.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int SetPartitionBlockList(int * list, int listsize);

Functions Sets a list of partition blocks.

Arguements

list

Specifies a pointer for integer-type memory array containing the number of blocks in each partition.

listsize

Specifies the size of list memory array (number of partitions).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks When the frame_per_block value indicated by GetPartitionInfo is 1, you cannot set the number of blocks within each partition as you desire. In such a case, first decide on the number of partitions to set and then set [the total number of blocks] divided by [the number of partitions] as the number of blocks per partition.

See “On Error Codes” for details of error codes On Error Codes

---

int **ExtraCommand(DWORD command,**
DWORD param1,DWORD param2 = 0,
DWORD param3 = 0,DWORD param4 = 0);

Functions Executes extra commands on each of cameras

Arguements

command
Specifies an extra command

param1
First argument of extra command (cannot be omitted)

param2
Second argument of extra command (default 0 when omitted)

param3
Third argument of extra command (default 0 when omitted)

param4
Fouth argument of extra command (default 0 when omitted)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without the specified command, a PCC_ERROR_NOT_SUPPORTED is returned as an error code.

Remarks See “On Extra Commands” for details of extra commands
On Extra Commands

See “On Error Codes” for details of error codes On Error Codes

int SetVariableSetting(int nChannel, int nFrameRate, int nWidth, int nHeight, int nXPos, int nYPos);

Functions Sets the setting value to a specification channel by variable setup.

Arguments

nChannel
Specifies setting channel (1-20)

nFrameRate
Specifies frame rate

nWidth
Specifies width of resolution

nHeight
Specifies height of resolution

nXPos
Specifies upper left X coordinates of rectangle

nYPos
Specifies upper left Y coordinates of rectangle

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Each specification value sets up equivalent to the execution result of a GetVariableMaxXXX function, or a low speed and low resolution. Moreover, fulfill the following conditions.

nWidth	128 multiples
nHeight	16 multiples
nXPos	64 multiples (0 is included)
nYPos	8 multiples (0 is included)

In addition, nWidth+nXPos or nHeight+nYPos should not exceed the highest resolution (if it is FASTCAM-APX RS 1024x1024) of a camera.

int SetGeometricConvert(int mode);

Functions Sets rotation and mirroring of output image.

Arguments **mode**

Sets rotation and mirroring modes.

IMAGE_CONV_DEFAULT	No rotation
IMAGE_CONV _ROTATE90	90-degree rotation to right
IMAGE_CONV _ROTATE180	180-degree rotation to right
IMAGE_CONV _ROTATE270	270-degree rotation to right
IMAGE_CONV _MIRROR_H	Horizontal mirroring
IMAGE_CONV _MIRROR_V	Vertical mirroring

Combination of rotation (any of 0, 90, 180 or 270 degrees) and mirroring possible

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int SetAutoExposure(int nWidth, int nHeight, int nXPos, int nYPos, int nValue, int nRange);

Functions Sets values for the Auto Exposure function.

Arguments

nWidth
Sets the width for a target area.

nHeight
Sets the height for a target area.

nXPos
Sets the x-coordinate of the upper left corner of a target area.

nYPos
Sets the Y-coordinate of the upper left corner of a target area.

nValue
Sets the image output level (0 to 255).

nRange
Sets the range for image output level (0 to 255).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Each of setting values in the above should meet the following condition:

nWidth	128 multiples
nHeight	16 multiples
nXPos	64 multiples (0 is included)
nYPos	8 multiples (0 is included)

Note 1: nWidth + nXPos or nHeight + nYPos should not exceed the resolution of the camera.

Note 2: nXPos and nYPos represent the coordinate of the maximum resolution of the camera (with FASTCAM-APX-RS, for example, 1024 x 1024). As an example, for a resolution of 512 x 512 around an optical axis, the coordinate of the origin in the upper left corner is X = 256 and Y = 256.

int GetMaxFrame(int &frame);

Functions

Gets the maximum number of frames in the current mode (recording rate and resolution).

Arguments

frame

Specifies an integer parameter that receives the maximum number of frames.

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

The maximum number is different by the mode the camera is in: LIVE or PLAY. The maximum number of frames available to record is given in the LIVE mode and the maximum number of recorded frames in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetCameraParams(CAMERA_PARAMS &params);

Functions Gets camera parameters.

Arguments **params**

Specifies a CAMERA_PARAMS structure that receives camera parameters.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The acquired structure is a copy. So, you must use setting functions such as SetCameraParams or SetRecordRate to change the parameters like the recording rate. Likewise, when you change settings directly on the camera, you must call out the UpdateCameraParams member function before getting parameters by this member function.

See “CAMERA_PARAMS Structure” for details of structure CAMARA_PARAMS Structure

See “On Error Codes” for details of error codes On Error Codes

int GetCameraParamsEx(CAMERA_PARAMS_EX &params);

Functions Gets extra camera parameters.

Arguments

params

Specifies a CAMERA_PARAMS_EX structure that receives extra camera parameters.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The acquired structure is a copy. So, you must use setting functions such as SetCameraParamsEx to change the parameters like the recording rate. Likewise, when you change settings directly on the camera, you must call out the UpdateCameraParamsEx member function before getting parameters by this member function.

See “CAMERA_PARAMS_EX Structure” for details of structure CAMARA_PARAMS EX Structure

See “On Error Codes” for details of error codes On Error Codes

int UpdateCameraParams();

Functions	Gets parameters from cameras.
Arguments	None
Returned values	Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
Remarks	The function updates itself with information on current settings of the cameras.

int UpdateCameraParamsEx();

Functions	Gets extra parameters from cameras.
Arguments	None
Returned values	Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
Remarks	The function updates itself with information on current settings of the cameras.

int GetRecordRate(int &rate);

Functions Gets the current recording rate.

Arguments **rate**
Specifies the integer parameter that receives a recording rate (FPS).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current recording rate is given in the LIVE mode and the recording rate of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetShutterSpeed(int &speed);

Functions Gets the current shutter speed.

Arguments **speed**

Specifies the integer parameter that receives a shutter speed (1/speed).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current recording rate is given in the LIVE mode and the recording rate of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetTriggerMode(int &mode);

Functions Gets the current trigger mode.

Arguments **mode**

Specifies the integer parameter that receives a trigger mode:

TRIGGER_START	START trigger
TRIGGER_CENTER	CENTER trigger
TRIGGER_END	END trigger
TRIGGER_RANDOM	RANDOM trigger
TRIGGER_MANUAL	MANUAL trigger
TRIGGER_RANDOMRESET	RANDOM RESET trigger
TRIGGER_RANDOMCENTER	RANDOM CENTER trigger
TRIGGER_RANDOMMANUAL	RANDOM MANUAL trigger
TRIGGER_TWOSTAGE	DUAL SPEED trigger

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current recording rate is given in the LIVE mode and the recording rate of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetRandomTriggerNumber(int &number);

Functions Gets the number of frames to record per one trigger in RANDOM trigger mode.

Arguments **number**

Specifies the integer parameter that receives the number of frames to record.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The number of frames to be recorded per currently set trigger is given in the LIVE mode and the number of frames recorded per trigger of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetManualTriggerPosition (int &position);

Functions Gets the position of trigger frame in MANUAL trigger mode.

Arguments **position**

Specifies integer parameters that receive a trigger frame position.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current trigger setting position to start recording is given in the LIVE mode and the trigger setting position of playback data is given in the PLAY mode.

See the hardware manual of each camera for details of trigger modes.
See On Frame Parameters for details of frame numbers.
See “On Error Codes” for details of error codes On Error Codes

int GetRandomManualPosition(int &position);

Functions Gets the trigger position for RANDOM MANUAL trigger mode.

Arguments **position**

Specifies integer parameters that receive a trigger position (frame number).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current trigger setting position to start recording is given in the LIVE mode and the trigger setting position of playback data is given in the PLAY mode.

See the hardware manual of each camera for details of trigger modes.
See On Frame Parameters for details of frame numbers.
See “On Error Codes” for details of error codes On Error Codes

int GetResolution(DWORD &resolution);

Functions Gets the current resolution.

Arguments **resolution**

Specifies a DWORD-type parameter that receives the resolution.
Upper 16 bits: Width of resolution
Lower 16 bits: Height of resolution

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current resolution setting is given in the LIVE mode and the resolution setting of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetGammaCorrection(int &mode);

Functions Gets the current gamma correction mode.

Arguments **mode**

Specifies integer parameters that receive a gamma correction value mode.

GAMMA_1_0	γ = 1.0
GAMMA_0_9	γ = 0.9
GAMMA_0_8	γ = 0.8
GAMMA_0_7	γ = 0.7
GAMMA_0_6	γ = 0.6
GAMMA_0_5	γ = 0.5
GAMMA_0_4	γ = 0.4

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without gamma correction function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current gamma correction setting is given in the LIVE mode and the gamma correction setting of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetGainLevel(int &mode);

Functions Gets the current gain level mode.

Arguments **mode**

Specifies integer parameters that receive a gain level value mode.

GAIN_0DB	0dB
GAIN_3DB	3dB
GAIN_6DB	6dB
GAIN_12DB	12dB
GAIN_18DB	18dB
GAIN_24DB	24dB
GAIN_30DB	30dB
GAIN_36DB	36dB

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras (color or monochrome) without gain level setting function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current gain level setting is given in the LIVE mode and the gain level setting of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetColorTemperature(int &mode);

Functions Gets the current color temperature mode.

Arguments **mode**

Specifies integer parameters that receive a color temperature mode.

COLORTEMPMODE_DEF1	Default setting 1
COLORTEMPMODE_DEF2	Default setting 2
COLORTEMPMODE_USER1	User setting 1
COLORTEMPMODE_USER2	User setting 2
COLORTEMPMODE_USER3	User setting 3

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks On cameras (color or monochrome) without color temperature setting function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current color temperature mode setting is given in the LIVE mode and the color temperature mode setting of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

Note: COLORTEMPMODE_USER2 and COLORTEMPMODE_USER3 are not currently used. COLORTEMPMODE_DEF1 and COLORTEMPMODE_DEF2 are set 5100K and 3100K, respectively.

int GetColorTempeUser(int mode, int &r, int &g, int &b);

Functions

Gets the value of a specified user-setting color temperature.

Arguments

mode

Specifies a user-setting color temperature mode to get white balance value.

COLORTEMPMODE_USER1	user setting 1
COLORTEMPMODE_USER2	uset setting 2
COLORTEMPMODE_USER3	user setting 3

r

Specifies the integer parameter that receives Red level.

g

Specifies the integer parameter that receives Green level.

b

Specifies the integer parameter that receives Blue lebel.。

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras (color or monochrome) without color temperature setting function, a PCC_ERROR_FUNCTION_DISABLE is returened as an error code.

Remarks

The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current white balance value setting is given in the LIVE mode and the white balance value of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

Note: COLORTEMPMODE_USER2, and COLORTEMPMODE_USER3 are not currently used.

int GetExternalSyncMode(BOOL &on);

Functions Gets ON or OFF information of the current external sync mode.

Arguments **on**

Specifies Bool type parameter that receives ON/OFF status of external sync setting.

FALSE: OFF (Master mode)

TRUE: ON (Slave mode)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with GetExternalSyncEx member function and is only included here to match the old version SDK manual.

int GetExternalSyncInMode(int &mode);

Functions Gets the current external sync mode status.

Arguments **mode**

Specifies integer parameter that receives an external sync mode.

EXTSYNC_NONE	OFF (Master mode)
EXTSYNC_POSI	ON (Slave mode with positive signal input)
EXTSYNC_NEGA	ON (Slave mode with negative signal input)
EXTSYNC_OTHERS_POSI	ON (Positive signal)
EXTSYNC_OTHERS_NEGA	ON (Negative signal)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks On cameras without Pos/Neg input signal setting, the function gets an ON or OFF only.

See “On Error Codes” for details of error codes On Error Codes

int GetExternalSyncModeEx(int &mode);

Functions Gets the current external sync mode setting.

Arguments **mode**

Specifies integer parameter that receives external sync mode status.

EXTSYNC_NONE	OFF (Master mode)
EXTSYNC_POSI	ON (Slave mode with positive signal input)
EXTSYNC_NEGA	ON (Slave mode with negative signal input)
EXTSYNC_OTHERS_POSI	ON (Positive signal)
EXTSYNC_OTHERS_NEGA	ON (Negative signal)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks On cameras without Pos/Neg input signal setting, the function gets an ON or OFF only.

See “On Error Codes” for details of error codes On Error Codes

int GetExternalSyncOutMode(int &mode);

Functions Gets the current external sync mode setting.

Arguments **mode**

Specifies integer parameter that receives external sync output mode status.

EXTOUT_VSYNC_POSI	VSYNC signal (Positive going)
EXTOUT_VSYNC_NEGA	VSYNC signal (Negative going)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int GetExternalInMode(int port, int &mode);

Functions Gets the current external general purpose signal input mode setting.

Arguments

port

Specifies an output port number beginning with a “0”.

mode

Specifies integer parameter that receives external general purpose signal input status.

EXTIN_EVENT_POSI	Event signal (Positive going)
EXTIN_EVENT_NEGA	Event signal (Negative going)
EXTIN_TRIGGER_POSI	Trigger signal (Positive going)
EXTIN_TRIGGER_NEGA	Trigger signal (Negative going)
EXTIN_READY_POSI	Ready signal (Positive going)
EXTIN_READY_NEGA	Ready signal (Negative going)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int GetExternalOutMode(int port, int &mode);

Functions

Gets the current external general purpose signal output mode setting.

Arguments

port

Specifies an output port number beginning with a “0”.

mode

Specifies integer parameter that receives the external signal output mode.

EXTOUT_RECORD_POSI	Recording period signal (Positive going)
EXTOUT_RECORD_NEGA	Recording period signal (Negative going)
EXTOUT_TRIGGER_POSI	Trigger signal (Positive going)
EXTOUT_TRIGGER_NEGA	Trigger signal (Negative going)
EXTOUT_EXPOSE_POSI	Exposure period signal (Positive going)
EXTOUT_EXPOSE_NEGA	Exposure period signal (Negative going)
EXTOUT_READY_POSI	Ready signal (Positive going)
EXTOUT_READY_NEGA	Ready signal (Negative going)
EXTOUT_IRIG_RESET_POSI	IRIG reset signal (Positive going)
EXTOUT_IRIG_RESET_NEGA	IRIG reset signal (Negative going)

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras (color or monochrome) without external signal output function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks

The number designation of settable ports varies by the camera model.
The kind of settable output modes varies by the camera model and port number.

See “On Error Codes” for details of error codes On Error Codes

int GetEdgeEnhancementMode(int &mode);

Functions Gets the current edge enhancement mode on the monitor output signal.

Arguments **mode**

Specifies integer parameter that receives the current edge enhancement mode status.

EDGEMODE_NONE	No edge enhancement in effect
EDGEMODE_1	Edge enhancement level 1
EDGEMODE_2	Edge enhancement level 2
EDGEMODE_3	Edge enhancement level 3

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without edge enhancement function, a PCC_ERROR_FUNCTION_DISABLE is returened as an error code.

Remarks Effective on monitor output only.
See “On Error Codes” for details of error codes On Error Codes

Note: This function is not used on currently supported cameras, but is only included here to match the old version SDK manual.

int GetZoomMode(BOOL &on);

Functions Gets the current zooming display mode of monitor output signal.

Arguments **on**

Specifies the integer parameter that receives the status of ON/OFF of zooming display.

TRUE ON

FALSE OFF

Returned values Returns an error code. Whenk normal, returns a PCC_ERROR_NOERROR.
On cameras without zooming display function or monitor output, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Effective on the monitor output signal only.

See “On Error Codes” for details of error codes On Error Codes

Note: This function is not used on currently supported cameras, but is only included here to match the old version SDK manual.

int GetMonitorOutMode(int &mode);

Functions Gets the external monitor output mode.

Arguments **mode**

Specifies integer parameter that receives the extenal monitor output mode.
MONITOROUT_VGA VGA monitor output
MONITOROUT_NTSC NTSC monitor output

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On cameras without error code function,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks See “On Error Codes” for details of error codes On Error Codes

Note: This function has been combined with ExtraCommand member function. It is only included here to match the old version of SDK manual.

int GetEnableMCDL(BOOL &on);

Functions Gets ON/OFF status of the MCDL function.

Arguments **on**

Specifies BOOL parameter that receives ON/OFF status of the MCDL status.

TRUE	MCDL function ON
FALSE	MCDL function OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR. With a camera without MCDL function, a PCC_ERROR_FUNCTION_DISABLE is returned.

Remarks See the hardware manual of each camera for details of MCDL function.

See “On Error Codes” for details of error codes On Error Codes.

Note: This member function has been combined with ExtraCommand member function, and is only included here to match the old version of SDK manual.

int GetEnableIRIG(BOOL &on);

Functions Gets ON/OFF status of the IRIG function.

Arguments **on**

Specifies BOOL parameter that receives the ON/OFF status of the IRIG function.

TRUE	MCDL function ON
FALSE	MCDL function OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR. With a camera without IRIG function, a PCC_ERROR_FUNCTION_DISABLE is returned.

Remarks See the hardware manual of each camera for details of IRIG function.

See “On Error Codes” for details of error codes On Error Codes.

Note: This member function has been combined with ExtraCommand member function, and is only included here to match the old version of SDK manual.

---

int **GetRecordRateList(int *list,**
int listsize = LIST_MAX_NUMBER);

Functions Gets a list of recording rates that can be set on the camera.

Arguments

list

Specifies the integer address of memory array that receives the recording rate list

listsize

Specifies the size of the list memory array.
The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The size of list memory array is normally determined by a LIST_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes

int GetShutterSpeedList(int *list, int listsize = LIST_MAX_NUMBER);

Functions

Gets a list of shutter speeds that are currently available to set.

Arguments

list

Specifies integer address of memory array that receives the shutter speed list.

listsize

Specifies the size of the list memory array.
The default size is LIST_MAX_NUMBER.

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

For list memory array, secure the size of LIST_MAX_NUMBER unless there is any problem.

See “On Error Codes” for details of error codes On Error Codes

If a “1” exists in a shutter speed list, the camera is compatible with special shutter speeds (eg. FASTCAM-ultima APX camera).
A camera compatible with special shutter speeds can get information on the following:

list[n] = 1	A camera compatible with special shutter speeds
list[n+1]	Shutter speed variable unit
list[n+2]	Minimum shutter speed
list[n+3]	Maximum shutter speed

With a camera compatible with special shutter speeds, any shutter speed between the minimum and maximum shutter speeds can be set with an iterval of the above variable unit.

Also, a shutter speed that is the same as the one currently used can be set.

---

int **GetTriggerModeList(int *list,**
int listsize = LIST_MAX_NUMBER);

Functions Gets a list of trigger mode that is available with the camera.

Arguments **list**

Specifies integer address of memory array that receives the trigger mode list.

listsize

Specifies the size of memory array .
The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The size of list memory array is nusually determined, and secured, by a LIST_MAX_NUMBER.

See the hardware manual of each camera for details of trigger modes.

See “On Error Codes” for details of error codes On Error Codes

---

int **GetResolutionList(DWORD *list,**
int listsize = LIST_MAX_NUMBER);

Functions Gets a list of resolutions that are currently available for setting.

Arguments **list**

Specifies integer address of memory array that receives the resolution list.
DWORD Upper 16 bits Width
DWORD Lower 16 bits Height

listsize

Specifies the size of list memory array.
The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The size of list memory array is usually determined, and secured, by a LIST_MAX_NUMBER.
Because the resolutions available for setting varies by the current recording rate, it may be necessary to get another list when changing the recording rate.

See “On Error Codes” for details of error codes On Error Codes

---

int GetRandomTriggerNumberList(int *list,
int listsize = LIST_MAX_NUMBER);

Functions

Gets a list of the numbers of frames that can be set on the camera for recording in RANDOM trigger mode.

Arguments

list

Specifies integer address of memory array that receives the list of recording frames.

listsize

Specifies the size of list memory array.
The default size is LIST_MAX_NUMBER.

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

The size of list memory array is usually determined, and secured, by a LIST_MAX_NUMBER.

See the hardware manual of each camera for details of trigger modes.

See “On Error Codes” for details of error codes On Error Codes.

int GetColorTemperatureList(int *list, int listsize = LIST_MAX_NUMBER);

Functions

Gets a list of color temperature (white balance) mode available for setting on cameras.

Arguments

list

Specifies integer memory array address that receives the list of color temperature (white balance).

COLORTEMPMODE_DEF1	Default setting 1
COLORTEMPMODE_DEF2	Default setting 2
COLORTEMPMODE_USER1	User setting 1
COLORTEMPMODE_USER2	User setting 2
COLORTEMPMODE_USER3	User setting 3

listsize

Specifies the size of list memory array. The default size is a LIST_MAX_NUMBER.

Returned values

Returns an arror code. When normal, returns a PCC_ERROR_NOERROR. With a camera without color temperature setting function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks

The size of list memory array is usually determined, and secured, by LIST_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes.

Note: COLORTEMPMODE_USER2 and COLORTEMPMODE_USER3 are not currently used. COLORTEMPMODE_DEF1 and COLORTEMPMODE_DEF2 are set 5100K and 3100K, respectively.

int GetGammaCorrectionList(int *list, int listsize = LIST_MAX_NUMBER);

Funtions Gets a list of gamma correction values available to be set on cameras.

Arguments

list

Specifies integer memory array address that receives the gamma correction mode list.

listsize

Specifies the size of memory array. The default size is LIST_MAX_NUMBER.

Returned values Returns an arror code. When normal, returns a PCC_ERROR_NOERROR. With a camera without gamma correction function, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks The size of list memory array is usually determined, and secured, by a LIST_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes.

int GetGainLevelList(int *list, int listsize = LIST_MAX_NUMBER);

Functions Gets a list of gain level modes that can be set on cameras.

Arguments

list

Specifies an address for integer memory array that receives the mode list of gain levels.

listsize

pecifies the size of the list memory array. The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
With a camera without gainsetting function,
a PCC_ERROR_FUNCTION_DISABLE is returned.

Remarks The size of list memory array is usually determined, and secured, by a LIST_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes.

int GetPartitionInfo(int &max_partition, int &max_partition_block, int &frame_per_block);

Functions Gets partition information.

Arguments

max_partition
The maximum number of available partitions.

max_partition_block
The maximum number of available blocks.

frame_per_block
The number of frames per block.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Arguments [max_partition_block] and [frame_per_block] vary by the current status of respective cameras.
When the value for [frame_per_block] is [-1] on a camera, you cannot decide the number of blocks per partition for it.

See “On Error Codes” for details of error codes On Error Codes.

int GetPartition(int &partition);

Functions Gets a number for the partition being used.

Arguments **partition**

Specifies integer parameter that gets a partition number beginning with “1”.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The size of list memory array is usually determined, and secured, by a PARTITIONT_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes.

int GetPartitionBlockList(int * list, int listsize);

Functions Gets a list of partitions.

Arguements

list

Specifies a pointer for integer-type memory array containing the number of blocks in each partition.

listsize

Specifies the size of list memory array. The default size is determined by a PARTITION_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes.

int GetFrameParams(FRAME_PARAMS &params);

Functions Gets frame management information.

Arguments **params**
Specifies a pointer of FRAME_PARAMS structure to store.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes.

---

int **GetShutterSpeedType2List(int *list,**
int listsize = LIST_MAX_NUMBER);

Functions Gets a list of shutter speeds that are currently available to set.

Arguments **list**

Specifies integer address of memory array that receives the shutter speed list(usec).

listsize

Specifies the size of the list memory array.
The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks For list memory array, secure the size of LIST_MAX_NUMBER unless there is any problem.

See “On Error Codes” for details of error codes On Error Codes

int GetShutterSpeedType2(int &speed);

Functions Gets the current shutter speed.

Arguments **speed**

Specifies the integer parameter that receives a shutter speed (usec).

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The number the function gets is different by the mode the camera is in: LIVE or PLAY. The current recording rate is given in the LIVE mode and the recording rate of playback data is given in the PLAY mode.

See “On Error Codes” for details of error codes On Error Codes

int GetHeadName(LPTSTR name, int no);

Functions Gets the camera head name

Arguments

name
Specifies the address of LPTSTR type character sequence buffer which receives a camera head name

no
Specifies camera head number to acquire (1-3)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Use only with the dissociated type camera corresponding to the head exchange function is possible.

int GetVariableSetting(int nChannel, int *nFrameRate, int * nWidth, int * nHeight int * nXPos, int * nYPos);

Functions Gets the setting value of a specification channel by variable setting.

Arguments

nChannel
Specifies acquisition channel (1-20)

nFrameRate
Integer parameter that frame rate is received.

nWidth
Integer parameter that receives the width of resolution

nHeight
Integer parameter that receives the height of resolution

nXPos
Integer parameter that receives upper left X coordinates of rectangle

nYPos
Integer parameter that receives upper left Y coordinates of rectangle

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int GetVariableCamMode(int *nMode, int * nChannel);

Functions Gets the current variable mode.

Arguments

nMode

Integer parameter that variable mode is received.

CAMMODE_DEFAULT	Default mode
CAMMODE_VARIABLE	Variable mode
CAMMODE_EXTERNAL	External (Signal input) mode

nChannel

Integer parameter that receives current variable channel.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks This function is used on with a “Variable function” cameras

int GetGeometricConvert(int &mode);

Functions Gets a status of rotation and mirroring of the output image.

Arguments **mode**

Gets rotation and mirroring mode.。

IMAGE_CONV_DEFAULT	No rotation
IMAGE_CONV _ROTATE90	90-degree rotation to right
IMAGE_CONV _ROTATE180	180-degree rotation to right
IMAGE_CONV _ROTATE270	270-degree rotation to right
IMAGE_CONV _MIRROR_H	Horizontal mirroring
IMAGE_CONV _MIRROR_V	Vertical mirroring

Gets a combined value.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int GetAutoExposureArea(int *nWidth, int *nHeight, int *nXPos, int *nYPos);

Functions Gets a target area for Auto Exposure function.

Arguments

nWidth
Gets the width for a target area.

nHeight
Sets the height for a target area.

nXPos
Sets the X-coordinate of the upper left corner of a target area.

nYPos
Sets the Y-coordinate of the upper left corner of a target area.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks XPos and nYPos represent the coordinate of the maximum resolution of the camera (with FASTCAM-APX-RS, for example, 1024 x 1024). As an example, for a resolution of 512 x 512 around an optical axis, the coordinate of the origin in the upper left corner is X = 256 and Y = 256.

int GetAutoExposureParam(int *nValue, int *Range);

Functions Gets the image output level for the Auto Exposure function.

Arguments

nValue
Image output level (0 to 255)

nRange
Range of image output level (0 to 255)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

---

int GetShutterSpeedAEList(int *list,
int listsize = LIST_MAX_NUMBER_EX,
int *listnum);

Functions

Gets an exposure period (shutter speed) that can be set for the Auto Exposure function.

Arguments

list

Specifies the address of an int-type memory array that receives an exposure period (1/shutter speed) that can be set at the current frame rate and resolution.

listsize

Specifies the size of a **list** memory array. The default size is **LIST_MAX_NUMBER_EX**

listnum

The number of lists to get.

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

For the size of a **list** memory array, secure a size that is the same as that for LIST_MAX_NUMBER_EX (not **LIST_MAX_NUMBER)** unless there is a specific problem.

int OnLive(BOOL on);

Functions Switches between LIVE and PLAY modes.

Arguments **on**

Specifies a camera mode.
TRUE: LIVE mode
FALSE: PLAY mode

Retruned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int IsLive(BOOL &on);

Functions Gets the current camera mode.

Arguments **on**

Specifies a Bool type parameter that receives the camera mode.

TRUE	LIVE mode
FALSE	PLAY mode

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int GetCameraMode(int &mode);

Functions Gets the current camera mode.

Arguments **mode**

Gets a camera mode as a flag.

CAMERA_MODE_PLAY	PLAY
CAMERA_MODE_LIVE	LIVE
CAMERA_MODE_RECREADY	REC READY
CAMERA_MODE_REC	REC
CAMERA_MODE_ENDLESS	ENDLESS
CAMERA_MODE_RECORDED	RECORDED

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes.

int OnRecordReady(BOOL on);

Functions Sets ON or OFF of the ready-to-record (REC READY) mode.

Arguments **on**

Specifies ON or OFF of the ready-to-record status.

TRUE	REC READY status ON
FALSE	REC READY status OFF (LIVE mode)

Retruned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
When the camera is in the PLAY mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function can only be set when the camera is in the LIVE mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

int OnRecord(BOOL on);

Functions Starts and stops recording.

Arguments **on**

Specifies start or stop recording.

TRUE	Starts recording (ON)
FALSE	Stops recording (OFF)

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
When the camera is in the PLAY mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks A start of recording is only possible when the camera is in the REC READY mode. See the [OnRecordReady] member function.

See “On Error Codes” for details of error codes On Error Codes

int IsRecordReady(BOOL &on);

Functions Gets information whether or not the camera is in REC READY mode.

Arguments **on**

Specifies a Bool type parameter that receives REC READY mode.

TRUE	REC READY Mode ON
FALSE	REC READY Mode OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int IsRecord(BOOL &on);

Functions Gets information whether or not the camera is currently recording.

Arguments **on**

Specifies a Bool type parameter that receives the recording status.

TRUE	Camera is currently recording.
FALSE	Camera is not recording.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int TriggerIn();

Functions Inputs software triggers

Auguments None

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
When the camera is in the PLAY mode,
a PCC_ERROR_CAMERAMODE is returned.

Remarks In the START trigger mode, this function can be used only when the camera is in the REC READY mode. In other trigger modes, it can be used any time the camera is recording. See [OnRecordReady] and [OnRecord] member functions.

See “On Error Codes” for details of error codes On Error Codes

Note: In a multple-camera operation with FASTCAM-PCI, PCI-R2 or 1280PCI cameras, the master camera only receives this command. Because slave cameras automatically receive a trigger, this member function does not have to be called out for each of the slave cameras.

int IsEndlessRec(BOOL &on);

Functions

Gets information whether or not the camera is currently recording in the endless mode.

Arguments

on

Specifies a Bool type parameter that receives the recording status.

TRUE	Recording in the endless mode
FALSE	Not recording in the endless mode

Returned values

Retruns an error code. When normal, returns a PCC_ERROR_NOERROR .

Remarks

See “On Error Codes” for details of error codes On Error Codes

int Play(int mode);

Functions Displays playback image data on the monitor.

Arguments **mode**

Specifies a playback mode.

PLAYMODE_PLAY	Playback
PLAYMODE_PLAYBACK	Reverse playback
PLAYMODE_FASTPLAY	Fast forward
PLAYMODE_FASTPLAYBACK	Fast reverse

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
With a camera that has no monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned.
When the camera is in the LIVE mode,
a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on monitor outpur only. It can only be set when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int Pause(BOOL on);

Functions Sets ON/OFF of pause on playback monitor output.

Arguments **on**

Specifies ON or OFF of pause.
TRUE: Pause ON
FALSE: Pause OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without monitor output, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int Stop();

Functions Stops the monitor output.

Arguments None

Returned values Returns and error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without monitor output, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int StepFoward();

Functions Steps the displayed image one frame forward.

Arguments None.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int StepBack();

Functions Steps the displayed image one frame backward.

Arguments None.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GoAnyFrame(int frame);

Functions

Moves the displayed image to any specified frame.

Arguments

frame

Specifies the number of frames to move.

Returned values

Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks

This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function. The system automatically comes into a temporary stop status after the specified frame has been displayed.

See “Details of Frame Parameters” for details of frame numbers.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GoStart();

Functions Jumps to the first frame of a recording.

Arguments None.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GoEnd();

Functions

Jumps to the last frame of a recording.
Jumps to the last frame of a block in the Block playback mode.

Arguments

None.

Returned values

Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks

This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GoTrigger();

Functions Jumps to a trigger frame

Arguments None.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int SetPlayRate(int rate);

Functions Sets playback rate of the monitor output.

Arguments **rate**

Specifies a playback rate（FPS）.
Available playback rates: 30, 15, 10, 5, 2, 1

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int SetBlockArea(int start, int end);

Functions

Sets a range (block) of frames for playback on monitor output.

Arguments

start

Specifies the first frame of a range (block).

end

Specifies the last frame of a range (block).

Returned values

Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks

This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.
See “Details of Frame Parameters” for details of frame numbers.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int SetBlockMode(BOOL on);

Functions Sets ON/OFF of block playback on monitor output.

Arguments **on**

Specifies ON/OFF of blocck playback.

TRUE	Block playback	ON
FALSE	Block playback	OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR
On cameras without monitor output, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only. It can only be used when the camera is in the PLAY mode. See [OnLive] member function.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GetPlayRate(int &rate);

Functions Gets a playback rate on monitor output.

Arguments **rate**

Specifies integer parameter that receives the display rate of playback (FPS)

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GetBlockArea(int &start, int &end);

Functions

Gets a range (block) of block playback on monitor output.

Arguments

start

Specifies integer parameter that receives the first frame of the range (block) .

end

Specifies integer parameter that receives the last frame of the range (block).

Returned values

Returns an error code. When normal, returns
a PCC_ERROR_NOERROR
On cameras without monitor output,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks

This function is effective on the monitor outpur only.
See “Details of Frame Parameters” for details of frame numbers.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int GetBlockMode(BOOL &on);

Functions Gets ON/OFF status of monitor output in Block playback.

Arguments **on**

Specifies Bool type parameter that receives the status of Block playback.

TRUE	Block playback	ON
FALSE	Block playback	OFF

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR
On cameras without monitor output, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.
When the camera is in the LIVE mode, a PCC_ERROR_CAMERAMODE is returned.

Remarks This function is effective on the monitor outpur only.
See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

Note 2: This function is not supported by cameras that are newer than the FASTCAM ultima 1024.

int TransferFrame(int frame, BYTE *buff);

Functions Gets live and memory image data from the camera.

Arguments

frame

Specifies the frame number to get recorded image.
Specifies –1 to get LIVE image.

buff

Specifies address of the BYTE type memory array that stores the image data.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks Stored image data has its origin at the upper left corner. Color image data is interleaved RGBRGBRGB, etc.

See [Frame Parameters] for details of frame numbers.

See “On Error Codes” for details of error codes On Error Codes

int TransferRawBayer(int frame, BYTE *buff);

unctions Gets Bayer image data from the camera.

Arguments

frame

Specifies a frame number.

buff

Specifies the address of a BYTE-type memory array that stores Bayer image data.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int Transfer16BitFrame(int frame, WORD *buff);

Functions Gets 16-bit recorded image data from the camera.

Arguments

frame

Specifies a frame number.

buff

Specifies the address of a WORD-type memory array that stores image data.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int Transfer16BitRawBayer(int frame, WORD *buff);

Functions Gets 16-bit Bayer image data from the camera.

Arguments

frame

Specifies a frame number.

buff

Specifies the address of a WORD-type memory array that stores Bayer image data.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks See “On Error Codes” for details of error codes On Error Codes

int TransferMCDL(int frame, BYTE *buff);

Functions

Gets MCDL data from the camera.

Arguments

frame

Specifies a frame number that receives the MCDL data.

buff

Specifies the address of a BYTE type memory array that stores the MCDL data. The size of memory array is determined by a SIZE_PF_MCDL_DATA.

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On a camera without MCDL function or a camera with its MCDL function set off, a PCC_ERROR_NO_MCDL is returned as an error code.

Remarks

See [On Frame Parameters] for details of frame numbers.

See the hardware manual of each camera for details of MCDL.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

int TransferIRIG(int frame, BYTE *buff);

unctions Gets IRIG data from the camera.

Arguments

frame

Specifies a frame number that receives the IRIG data.

buff

Specifies the address of a BYTE type memory array that stores the IRIG data. The size of memory array is determined by a SIZE_PF_MCDL_IRIG.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On camera without IRIG function or a camera with its IRIG function set off, a PCC_ERROR_NO_IRIG is returned as an error code.

Remarks

See [On Frame Parameters] for details of frame numbers.

See the hardware manual of each camera for details of IRIG.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

int GetTimeCodeFromFrame(IRIGLIB_TIMECODE &time_code, int frame);

Functions Gets IRIG timecode from a frame number.

Arguments

time_code

Specifies IRIGLIB_TIMECODE structure that receives IRIG timecode.

frame

Specifies a frame number that receives IRIG data.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On camera without IRIG function or a camera with its IRIG function set off, a PCC_ERROR_NO_IRIG is returned as an error code.

Remarks

See [On Frame Parameters] for details of frame numbers.

See the hardware manual of each camera for details of IRIG.

See [IRIGLIB_TIMECODE Structure] for the details of IRIGLIB_TIMECODE structure.

See “On Error Codes” for details of error codes On Error Codes

int ShadingCompensation();

Functions Corrects shading of image sensor.

Arguments None

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On cameras without shading correction,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks Shading correction must be done with the lens completely covered to shut out incomin light.

See the hardware manual of each camera for details of operation.

See “On Error Codes” for details of error codes On Error Codes

Note: This member function has been combined with [ExtraCommand] member functions. It is only included here to match the old version of SDK manual.

int GetDateOfRecording(int& year, int& month, int& day);

Functions Gets the date of recording.

Arguments

year
Specifies integer parameter that shows the year (last two digits).

month
Specifies integer parameter that shows the month.

day
Specifies integer parameter that shows the date.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.
On cameras without shading correction, a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks

int GetTimeOfRecording(int& hour, int& minute, int& second);

Functions Gets the time of recording.

Arguments

hour
Specifies integer parameter that receives the hours.

minute
Specifies integer parameter that receives the minutes.

second
Specifies integer parameter that receives the seconds.

Returned values Returns an error code. When normal, returns
a PCC_ERROR_NOERROR.
On cameras without shading correction,
a PCC_ERROR_FUNCTION_DISABLE is returned as an error code.

Remarks

void GetIICycleRange(DWORD& dwMin, DWORD& dwMax);

Functions Gets the CYCLE value of Image Intensifier(I.I.).

Arguments

dwMin

Specifies DWORD parameter that receives the Minimum CYCLE value.

dwMax

Specifies DWORD parameter that receives the Maximum CYCLE value.

Returned values None

Remarks When you change frame rate or CYCLE, WIDTH, TIMES, and a DELAY value at the time of I.I. use, be sure to perform.

void GetIIWidthRange(DWORD& dwMin, DWORD& dwMax);

Functions Gets the WIDTH value of Image Intensifier(I.I.).

Arguments

dwMin

Specifies DWORD parameter that receives the Minimum WIDTH value.

dwMax

Specifies DWORD parameter that receives the Maximum WIDTH value.

Returned values None

Remarks When you change frame rate or CYCLE, WIDTH, TIMES, and a DELAY value at the time of I.I. use, be sure to perform.

void GetIITimesRange(DWORD& dwMin, DWORD& dwMax);

Functions Gets the TIMES value of Image Intensifier(I.I.).

Arguments

dwMin

Specifies DWORD parameter that receives the Minimum TIMES value.

dwMax

Specifies DWORD parameter that receives the Maximum TIMES value.

Returned values None

Remarks When you change frame rate or CYCLE, WIDTH, TIMES, and a DELAY value at the time of I.I. use, be sure to perform.

void GetIIDelayRange(DWORD& dwMin, DWORD& dwMax);

Functions Gets the DELAY value of Image Intensifier(I.I.).

Arguments

dwMin

Specifies DWORD parameter that receives the Minimum DELAY value.

dwMax

Specifies DWORD parameter that receives the Maximum DELAY value.

Returned values None

Remarks When you change frame rate or CYCLE, WIDTH, TIMES, and a DELAY value at the time of I.I. use, be sure to perform.

int GetVariableRateList(int * list, int listsize = LIST_MAX_NUMBER);

Functions Gets the frame rate list, which can be used by variable setting

Arguments

list

Specifies the address of the integer memory arrangement that receives a frame rate list

listsize

Specifies the size of the list memory array.
The default size is LIST_MAX_NUMBER.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks The size of list memory array is normally determined by a LIST_MAX_NUMBER.

See “On Error Codes” for details of error codes On Error Codes

int GetVariableMaxResolution(int nFrame, int *nWidth, int * nHeight);

Functions

Gets the Maximum Square Resolution at specification frame rate by variable setting

Arguments

nFrame

Specifies frame rate

nWidth

Integer parameter that receives the width of the Maximum Square Resolution

nHeight

Integer parameter that receives the heigth of the Maximum Square Resolution

Returned values

Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

Maximum square resolution is noticed about the point that is not necessarily the resolution at the time of the maximum area.

---

int GetVariableMaxWidth(int nFrame, int nHeight, int * nWidth);

Functions Gets the maximum width at specification frame rate and height by variable setting

Arguments

nFrame
Specifies frame rate

nHeight
Specifies height of image

nWidth
Integer parameter that image maximum width at the time of the above-mentioned specification value is received.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

---

int GetVariableMaxHeight(int nFrame, int nWidth, int * nHeight);

Functions Gets the maximum height at specification frame rate and width by variable setting

Arguments

nFrame

Specifies frame rate

nWidth

Specifies width of image

nHeight

Integer parameter that image maximum height at the time of the above-mentioned specification value is received.

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int GetVariableMaxFrameRate(int nWidth, int nHeight, int * nRate);

Functions Gets the maximum frame rate at specification resolution by variable setting

Arguments

nWidth
Specifies width of image

nHeight
Specifies height of image

nRate
Integer parameter that maximum frame rate at the time of the above-mentioned specification value is received

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks

int SetGEtherConfig(DWORD PacketSize, DWORD SendPort, DWORD ReceivePort, DWORD ConnectMode);

Functions Sets conditions for connection of 1000Base-T Ethernet I/F.

Arguments

PacketSize
Packet size (default 0)
SendPort
Send port (default 0)
ReceivePort
Receive port (default 0)
ConnectMode
Connection mode
GETHER_CONNECT_NORMAL
Normal connection (default)
GETHER_CONNECT_LOWSPEED
Low-speed connection

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks A setting is applied when CCameraControl class is nullified

Note 1: For packet size, use a number 58 + (multiple of 8).

Note 2: When “0” (zero) is specified, the packet size is automatically readjusted.
If and when communication cannot be established at the specified size, the system operates at the default value (1458).

For normal use, select GETHER_CONNECT_NORMAL in ConnectMode. Take note that, when GETHER_CONNECT_LOWSPEED is used, PacketSize is forced into 722.

int GetGEtherConfig(DWORD *PacketSize, DWORD *SendPort, DWORD *ReceivePort, DWORD *ConnectMode);

Functions Gets setting conditions for connection of 1000Base-T Ethernet I/F

Arguments

PacketSize
Packet size (default 0)

SendPort
Send port (default 0)

ReceivePort
Receive port (default 0)

ConnectMode
Connection mode
GETHER_CONNECT_NORMAL
Normal connection (default)
GETHER_CONNECT_LOWSPEED
Low-speed connection

Returned values Returns an error code. When normal, returns a PCC_ERROR_NOERROR.

Remarks When the packet size is “0” (zero), the system operates as it is automatically readjusted.

EXTRA_SET_ENABLE_RESTRICTION_TIME

unctions Sets recording time restriction function status.

Argument 2 Valid/Invalid

TRUE	Valid (default)
FALSE	Invalid

Argument 3 None

Argument 4 None

Remarks This function switches the recording time restriction between Valid and Invalid on FASTCAM-PCI/R2 cameras. Normally, use the function in the Valid position.

EXTRA_GET_ENABLE_RESTRICTION_TIME

Functions Gets recording time restriction function status.

Argument 2 Valid/Invalid

TRUE	Valid
FALSE	Invalid

Argument 3 None

Argument 4 None

Remarks This function switches the recording time restriction between Valid and Invalid on FASTCAM-PCI/R2 cameras. Normally, use the function in the Valid position.

EXTRA_GET_CAMERA_ID

Functions	Gets a camera ID.
Arbument 2	DWORD-type pointer that contains a camera ID
Argument 3	None
Argument 4	None
Remarks	

EXTRA_GET_ENABLE_RESET_TRIGGER

Functions Gets a reset trigger status.

Argument 2 Pointer that contains Valid/Invalid.

TRUE	Valid
FALSE	Invalid

Argument 3 None

Argument 4 None

Remarks This function is used on FASTCAM-X1280PCI, FASTCAM-PCI R2, FASTCAM-512PCI and FASTCAM-1024PCI cameras.

EXTRA_SET_ENABLE_RESET_TRIGGER

Functions Sets a reset trigger status.

Argument 2 Valid or Invalid.

TRUE	Valid
FALSE	Invalid

Argument 3 None

Argument 4 None

Remarks This function is used on FASTCAM-X1280PCI, FASTCAM-PCI R2, FASTCAM-512PCI and FASTCAM-1024PCI cameras.

EXTRA_GET_SENSOR_BITSHIFT

Functions Gets grayscale setting status obtained by FASTCAM-X1280PCI camera.

Argument 2 DWORD-type pointer that contains grayscale.

0	Upper 8 bits of 10bits (default)
1	Middle 8 bits of 10bits
2	Lower 8 bits of 10bits

Argument 3 None

Argument 4 None

Remarks This function gets the current grayscale setting status obtained by a FASTCAM-X1280PCI camera.

EXTRA_SET_SENSOR_BITSHIFT

Functions Sets grayscale setting change obtained by FASTCAM-X 1280PCI camera.

Argument 2 Grayscale.
Selects one of the following three:

0	Upper 8 bits of 10bits (default)
1	Middle 8 bits of 10bits
2	Lower 8 bits of 10bits

Argument 3 None

Argument 4 None

Remarks This function selects one of three possible settings of grayscale values and sets in a FASTCAM-X1280PCI camera.

EXTRA_SET_LIVE_RESOLUTION

Functions Changes resolution setting in the Live mode.

Argument 2 Resolution settings for live mode

LIVE_RESO_FULL	Full resolution (default)
LIVE_RESO_HALF	1/2
LIVE_RESO_QQRTER	1/4
LIVE_RESO_DQARTER	1/8

Argument 3 None

Argument 4 None

Remarks Setting live resolution to any one other than the full resolution may esult in a faster transfer rate of image data, but the image quality is sacrificed.

EXTRA_GET_LIVE_RESOLUTION

Functions Gets resolution setting in the Live mode.

Argument 2 DWORD-type pointer that contains resolution settings for the live mode

LIVE_RESO_FULL	Full resolution (default)
LIVE_RESO_HALF	1/2
LIVE_RESO_QQRTER	1/4
LIVE_RESO_DQARTER	1/8

Argument 3 None

Argument 4 None

Remarks Setting live resolution to any one other than the full resolution may result in a faster transfer rate of image data, but the image quality is sacrificed.

EXTRA_SET_LUT_MODE

Funtions Sets the LUT status.

Argument 2 Mode setting values

LUT_DEFAULT1	Default 1
LUT_DEFAULT2	Default 2
LUT_DEFAULT3	Default 3
LUT_DEFAULT4	Default 4
LUT_DEFAULT5	Default 5
LUT_USER1	User-set number value

Argument 3 None

Argument 4 None

Remarks The LUT mentioned in the above is a parameter within the camera and has nothing to do with the LUT shown in the CImageCtrl class.

EXTRA_GET_LUT_MODE

Funtions Gets the LUT status.

Argument 2 DWORD-type pointers that contain mode setting values

LUT_DEFAULT1	Default 1
LUT_DEFAULT2	Default 2
LUT_DEFAULT3	Default 3
LUT_DEFAULT4	Default 4
LUT_DEFAULT5	Default 5
LUT_USER1	User-set number value

Argument 3 None

Argument 4 None

Remarks The LUT mentioned in the above is a parameter within the camera and has nothing to do with the LUT shown in the CImageCtrl class.

EXTRA_SET_USER_LUT_PARAMS

Argument 2 Mode setting values

LUT_USER1 Value set by user

Argument 3 Pointers for PCAM_LUT_PARAMS structure.

Argument 4 None

Remarks To make the changes effective on cameras, it is necessary to perform EXTRA_UPLOAD_LUT_DATA separately.

Note 1. This command is not currently supported.
Note 2. The following PCAM_LUT_PARAMS structure must be prepared and executed with appropriate values.

```
typedef struct {
        LONG    brightness_r;
        LONG    brightness_g;
        LONG    brightness_b;
        LONG    contrast_r;
        LONG    contrast_g;
        LONG    contrast_b;
        float   gain_r;
        float   gain_g;
        float   gain_b;
        float   gamma_r;
        float   gamma_g;
        float   gamma_b;
        BOOL    negative_r;
        BOOL    negative_g;
        BOOL    negative_b;
} PCAM_LUT_PARAMS, *PPCAM_LUT_PARAMS;
```

brightness	-512～512
gain	1.0～8.0
gamma	0.0～2.0
contrast	-512～512

EXTRA_GET_USER_LUT_PARAMS

Functions Gets LUT parameters.

Argument 2 Mode setting values

LUT_USER1	Value set by user

Argument 3 Pointers for PCAM_LUT_PARAMS structure structure that contains obtained values.

Argument 4 None

Remarks Note 1. This command is not currently supported.
Note 2. The following PCAM_LUT_PARAMS structure must be prepared and executed with appropriate values.

```
typedef struct {
        LONG    brightness_r;
        LONG    brightness_g;
        LONG    brightness_b;
        LONG    contrast_r;
        LONG    contrast_g;
        LONG    contrast_b;
        float   gain_r;
        float   gain_g;
        float   gain_b;
        float   gamma_r;
        float   gamma_g;
        float   gamma_b;
        BOOL    negative_r;
        BOOL    negative_g;
        BOOL    negative_b;

} PCAM_LUT_PARAMS, *PPCAM_LUT_PARAMS;
```

brightness	-512～512
gain	1.0～8.0
gamma	0.0～2.0
contrast	-512～512

EXTRA_UPLOAD_LUT_DATA

Functions	Uploads LUT to a camera.
Argument 2	None
Argument 3	None
Argument 4	None
Remarks	his function uploads the currently-set LUT to a camera. The camera must be set in LUT_USER1 mode.

EXTRA_SET_CHROMA_MODE

Functions Changes chroma settings.

Argument 2 Chroma setting values:

- CHROMA_NORMAL (Default)
- CHROMA_UP01
- CHROMA_UP02
- CHROMA_UP03
- CHROMA_UP04
- CHROMA_UP05
- CHROMA_UP06
- CHROMA_UP07
- CHROMA_UP08

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_CHROMA_MODE

Functions Gets chroma settings.

Argument 2 DWORD-type pointer that receives chroma settings:

CHROMA_NORMAL (Default)
CHROMA_UP01
CHROMA_UP02
CHROMA_UP03
CHROMA_UP04
CHROMA_UP05
CHROMA_UP06
CHROMA_UP07
CHROMA_UP08

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_DS_SHUTTER_MODE

Functions Sets dynamic range expansion mode.

Argument 2 Dyanamic range expansion mode setting values:

DS_SHUTTER_OFF	OFF (Default)
DS_SHUTTER_MODE1	Exposure control Small
DS_SHUTTER_MODE2	Exposure control Medium
DS_SHUTTER_MODE3	Exposure control Large

Argument 3 None

Argument 4 None

Remarks This function sets a dynamic range expansion mode.

EXTRA_GET_DS_SHUTTER_MODE

Functions Gets dynamic range expansion mode setting.

Argument 2 DWORD-type pointer that contains dyanamic range expansion setting values:

DS_SHUTTER_OFF	OFF (Default)
DS_SHUTTER_MODE1	Exposure control Small
DS_SHUTTER_MODE2	Exposure control Medium
DS_SHUTTER_MODE3	Exposure control Large

Argument 3 None

Argument 4 None

Remarks This function sets a dynamic range expansion mode.

EXTRA_SET_INTERFACE_INFO

Functions Changes interface settings.

Argument 2 Interface parameters:

IEEE1394

INTERFACE_IEEE1394 Fixed value

100Base-TX

INTERFACE_ETHER Fixed value

1000Base-T

INTERFACE_G_ETHER Fixed value

Argument 3 Interface parameters

IEEE1394

400 400Mbps operaion

200 200Mbps operation

100 100Mbps operation

100Base-TX

(Time for time-out):

IP address search time (milliseconds)

1000Base-T

Base address for automatic IP search

IP address is set with a DWORD-type number.
Example: For192.168.12.0, the IP address is 0xC0A80A00

Argument 4 Interface paremeters

IEEE1394

None

100Base-TX / 1000Base-T

(IP address anutomatic search number and IP address list)

Top of the list,

Upper 16 bits automatic search number

Set a “-1” for Using IP list

Lower 16 bits IP address number

Set a “-1” for automatic search.

Set an IP address by DWORD

Example: Set “0xC0A80101” for IP address 192.168.1.1.

(IP address automatic search number)

Maximum value: 64

If a negative number is set, automatic search is turned off.

Remarks Set a “0” for arguments that are not used.

Do not set an interface that is different from the type of the camera.

When you do not use automatic search, be sure to set interface before initialization.

Example

1) 100Base-TX

```
DWORD      dwIPList[CAMERA_DEVICE_MAX + 1];  //  IP List
DWORD      dwAutoTimeout;

dwAutoTimeout = 500;  //  500msec

If (bAutomatic) {
      //  Automatic search (10 Cameras)
      dwIPList[0] = (((DWORD)10 << 16) & 0xFFFF0000) + (DWORD)(-1 & 0x0000FFFF);
      pCamera->ExtraCommand(EXTRA_SET_INTERFACE_INFO,
               INTERFACE_ETHER,
               (DWORD)&dwAutoTimeout,
               (DWORD)dwIPList);
} else {
      //  Using IP List (2 Cameras)
      dwIPList[0] = (((DWORD)-1 << 16) & 0xFFFF0000) + (DWORD)(2 & 0x0000FFFF);
      dwIPList[1] = 0xC0A80101;  //  192.168.1.1
      dwIPList[2] = 0xC0A80102;  //  192.168.1.2
      pCamera->ExtraCommand(EXTRA_SET_INTERFACE_INFO,
               INTERFACE_ETHER,
               (DWORD)&dwAutoTimeout,
               (DWORD)dwIPList);
}
```

2) 1000Base-T

```
DWORD      dwIPList[CAMERA_DEVICE_MAX + 1];  //  IP List
DWORD      dwBaseIP; //Base IP address

dwBaseIP = 0xC0A80100;  //  192.168.1.0

If (bAutomatic) {
      //  Automatic search
      dwIPList[0] = (((DWORD)10 << 16) & 0xFFFF0000) + (DWORD)(-1 & 0x0000FFFF);
      pCamera->ExtraCommand(EXTRA_SET_INTERFACE_INFO,
               INTERFACE_G_ETHER,
               (DWORD)&dwBaseIP,
               (DWORD)dwIPList);
} else {
      //  Using IP List (2 Cameras)
      dwIPList[0] = (((DWORD)-1 << 16) & 0xFFFF0000) + (DWORD)(2 & 0x0000FFFF);
      dwIPList[1] = 0xC0A80101;  //  192.168.1.1
      dwIPList[2] = 0xC0A80102;  //  192.168.1.2
      pCamera->ExtraCommand(EXTRA_SET_INTERFACE_INFO,
               INTERFACE_G_ETHER,
               (DWORD)&dwBaseIP,
               (DWORD)dwIPList);
}
```

EXTRA_GET_INTERFACE_INFO

Functions Gets interface settings.

Argument 2 DWORD-type parameters that receives types of interface:

IEEE1394
- INTERFACE_IEEE1394 Fixed value

100Base-TX
- INTERFACE_ETHER Fixed value

1000Base-T
- INTERFACE_G_ETHER Fixed value

Optical I/F
- INTERFACE_OPTICAL Fixed value

Argument 3 DWORD-type parameters that receive interface parameters.

IEEE1394
- 400 400Mbps operaion
- 200 200Mbps operation
- 100 100Mbps operation

100Base-TX

Time-out for automatic IP search

IP address search time (msec)

1000Base-T

Base address for automatic IP search

IP address is set with a DWORD-type number.
Example: For192.168.12.0, the IP address is 0xC0A80A00

Optical I/F

None

Argument 4 DWORD-type parameters that stores interface paremeters

IEEE1394

None

100Base-TX / 1000Base-T

(IP address anutomatic search number and
IP address list)

Top of the list,

Upper 16 bits automatic search number

Set a “-1” for Using IP list

Lower 16 bits IP address number

Set a “-1” for automatic search.

Set an IP address by DWORD

Example: Set “0xC0A80101” for IP address 192.168.1.1.

(IP address anutomatic search number)

Maximum value: 64

If a negative number is set, automatic search is turned off.

Optical I/F

None

Remarks No parameters are needed.for inteface types that are not used,

EXTRA_SET_SHADING_MODE

Functions Sets calibration mode.

Argument 2 Calibration mode setting values:

SHADING_OFF	Will not use Calibration results.
SHADING_ON1	Will use Calibration results.
SHADING_SAVE	Will store Calibration results.
SHADING_LOAD	Will read Calibration results.

Argument 3 None

Argument 4 None

Remarks To check the status in SAVE/LOAD, the function [EXTRA_GET_SHADING_LOCK] is used.

EXTRA_GET_SHADING_MODE

Functions Gets calibration mode.

Argument 2 DWORD-type parameters that receive calibration mode:

SHADING_OFF	Will not use Calibration results.
SHADING_ON1	Will use Calibration results.
SHADING_SAVE	Will store Calibration results.
SHADING_LOAD	Will read Calibration results.

Argument 3 None

Argument 4 None

Remarks To check the status in SAVE/LOAD, the function [EXTRA_GET_SHADING_LOCK] is used.

EXTRA_GET_SHADING_LOCK

Functions Gets an execution status of calibration function.

Argument 2 DWORD-type parameters that receive calibration execution status.

TRUE	Locked
FALSE	Not Locked

Argument 3 None

Argument 4 None

Remarks After storing or reading a calibration, wait until Argument 2 turns to “Not Locked” before starting any other operation.

EXTRA_SET_IRIG_OFFSET

Functions	Sets IRIG offset value.
Argument 2	Offset value -999999 – 999999 (microsecond)
Argument 3	None
Argument 4	None
Remarks	When you do not use offset, make it 0.

EXTRA_GET_IRIG_OFFSET

Functions	Gets IRIG offset value.
Argument 2	DWORD-type parameters that receive calibration execution offset value.
Argument 3	None
Argument 4	None
Remarks	It is set to 0 from a non-corresponding model.

EXTRA_SET_HARD_PARTITION

Functions Sets hard partition mode.

Argument 2 Hard partition mode setting status.

TRUE	Hard partition mode
FALSE	Not hard partition mode (default)

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_HARD_PARTITION

Functions Gets an execution status of hard partition mode.

Argument 2 DWORD-type parameters that receive hard partition mode status.

TRUE	Hard partition mode
FALSE	Not hard partition mode(default)

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_SHUTTER_MODE

Functions Sets shutter mode.

Argument 2 Shutter mode setting status.

TRUE	Locked shutter speed
FALSE	Not Locked(default)

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_SHUTTER_MODE

Functions Gets an execution status of shutter mode.

Argument 2 DWORD-type parameters that receive shutter mode status.

TRUE	Locked shutter speed
FALSE	Not Locked(default)

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_II_POWER

Functions Sets power status of Image Intensifier.

Argument 2 ”Image Intensifier” ‘s setting power status.

II_POWER_OFF	Power off
II_POWER_ON	Power on
II_POWER_ON_LOAD	Power on (Loading configuration on Camera)

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_POWER

Functions Gets an execution status of “Image Intensifier” ‘s power status.

Argument 2 DWORD-type parameters that receive “Image Intensifier” ‘s power status.

II_POWER_OFF	Power off
II_POWER_ON	Power on

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_II_GAIN

Functions Sets gain value of Image Intensifier.

Argument 2 Gain value of Image Intensifier.
1 - 50 (0.1V – 5.0V)

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GAIN

Functions Gets gain value of Image Intensifier.

Argument 2 DWORD-type parameters that receive gain value of Image Intensifier.
1 - 50 (0.1V – 5.0V)

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_II_GATE_SELECT

Functions Sets an execution status of “Image Intensifier” ‘s gate mode.

Argument 2 Gate mode of Image Intensifier.

II_GATE_SELECT_OFF
II_GATE_SELECT_CONTINUOUS
II_GATE_SELECT_EXTERNAL
II_GATE_SELECT_GATING
II_GATE_SELECT_DELAY

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GATE_SELECT

Functions Gets gate mode of Image Intensifier.

Argument 2 DWORD-type parameters that receive gate mode of Image Intensifier.

II_GATE_SELECT_OFF
II_GATE_SELECT_CONTINUOUS
II_GATE_SELECT_EXTERNAL
II_GATE_SELECT_GATING
II_GATE_SELECT_DELAY

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_II_GATE_CYCLE

Functions Sets CYCLE value of Image Intensifier.

Argument 2 CYCLE value of Image Intensifier.

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GATE_CYCLE

Functions	Gets CYCLE value of Image Intensifier.
Argument 2	DWORD-type parameters that receive CYCLE value of Image Intensifier.
Argument 3	None
Argument 4	None
Remarks	

EXTRA_SET_II_GATE_WIDTH

Functions	Sets WIDTH value of Image Intensifier.
Argument 2	WIDTH value of Image Intensifier.
Argument 3	None
Argument 4	None
Remarks	

EXTRA_GET_II_GATE_WIDTH

Functions	Gets WIDTH value of Image Intensifier.
Argument 2	DWORD-type parameters that receive WIDTH value of Image Intensifier.
Argument 3	None
Argument 4	None
Remarks	

EXTRA_SET_II_GATE_TIMES

Functions Sets TIMES value of Image Intensifier.

Argument 2 TIMES value of Image Intensifier.

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GATE_TIMES

Functions	Gets TIMES value of Image Intensifier.
Argument 2	DWORD-type parameters that receive TIMES value of Image Intensifier.
Argument 3	None
Argument 4	None
Remarks	

EXTRA_SET_II_GATE_DELAY

Functions Sets DELAY value of Image Intensifier.

Argument 2 DELAY value of Image Intensifier (nsec).

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GATE_DELAY

Functions Gets DELAY value of Image Intensifier.

Argument 2 DWORD-type parameters that receive DELAY value of Image Intensifier.

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_GAIN_LIMIT

Functions	Gets Gain's limit value of Image Intensifier.
Argument 2	DWORD-type parameters that receive Gain value of Image Intensifier.
Argument 3	None
Argument 4	None
Remarks	

EXTRA_GET_EVENT_FRAME

Functions	Get the event frame number
Argument 2	event number (0-9)
Argument 3	DWORD-type parameter that receives a frame number
Argument 4	long-type parameter that receives the photography number at the time of a random system trigger
Remarks	

EXTRA_SET_VARIABLE_LOAD

Functions	Set the state of a camera that is made into the setting value of a specification channel by variable setting.
Argument 2	The channel to be used (0-19)
Argument 3	None
Argument 4	None
Remarks	

EXTRA_SET_VARIABLE_ERASE

Functions	The setting value of a specification channel is deleted by variable setting.
Argument 2	The channel to delete (0-19)
Argument 3	None
Argument 4	None
Remarks	

EXTRA_GET_II_GATE_WIDTH_LIMIT

Functions Gets mimimum WIDTH value of Image Intensifier.

Argument 2 Minimum WIDTH value of Image Intensifier (nsec).

Argument 3 None

Argument 4 None

Remarks

EXTRA_SET_EDGE_ENHANCEMENT

Functions A state setup of EDGE-ENHANCEMENT.

Argument 2 **mode**

EDGE_ENHANCEMENT_OFF	OFF
EDGE_ENHANCEMENT_MODE1	MODE1
EDGE_ENHANCEMENT_MODE2	MODE2
EDGE_ENHANCEMENT_MODE3	MODE3

Argument 3 None

Argument 4 None

Remarks It is not compatible with EXTRA_SET_MONITOROUT_EDGEENHANCE.

EXTRA_GET_EDGE_ENHANCEMENT

Functions Get the state of EDGE-ENHANCEMENT

Argument 2 DWORD-type pointer that stores the mode value of EDGE-ENHANCEMENT

EDGE_ENHANCEMENT_OFF	OFF
EDGE_ENHANCEMENT_MODE1	MODE1
EDGE_ENHANCEMENT_MODE2	MODE2
EDGE_ENHANCEMENT_MODE3	MODE3

Argument 3 None

Argument 4 None

Remarks It is not compatible with EXTRA_GET_MONITOROUT_EDGEENHANCE.

EXTRA_SET_AUTO_EXPOSURE

Functions Sets status of Auto Exposure function.

Argument 2 **Auto Exposure setting values**

TRUE	Auto Exposure function On
FALSE	Auto Exposure function OFF

Argument 3 None

Argument 4 None

Remarks When the Auto Exposure function is on, the shutter speed that can be gotten is indefinite.

EXTRA_GET_AUTO_EXPOSURE

Functions Gets status of the Auto Exposure function.

Argument 2 A DWORD-type pointer that stores status of the Auto Exposure function.

TRUE	Auto Exposure function On
FALSE	Auto Exposure function Off

Argument 3 None

Argument 4 None

Remarks When the Auto Exposure function is on, the shutter speed that can be gotten is indefinite.

EXTRA_SET_AUTO_EXPOSURE_MAXSHUTTER

Functions Sets the maximum exposure period for the Auto Exposure function.

Argument 2 **Maximum exposure period**
Specfies the maximum exposure period (1/shutter speed) that can be set.

Argument 3 None

Argument 4 None

Remarks Values that can be set can be gotten from the member function [GetShutterSpeedAEList].

EXTRA_GET_AUTO_EXPOSURE_MAXSHUTTER

Functions	Gets the maximum exposure period for the Auto Exposure function.
Argument 2	A DWORD-type pointer that stores status of the Auto Exposure function. Gets the maximum exposure period (1/shutter speed) that is set
Argument 3	None
Argument 4	None
Remarks	

EXTRA_SET_II_PROTECTION

Functions Sets the level of hardware protection function for the image intensifier (I.I.).

Argument 2 **Protection level value**

1 to 5	Protection function level

Argument 3 None

Argument 4 None

Remarks

EXTRA_GET_II_PROTECTION

Functions	Gets the level of hardware protection function for the image intensifier (I.I.).
Argument 2	A DWORD-type pointer that stores the protection level.
Argument 3	None
Argument 4	None
Remarks	

5. CImageData Class

This section discusses the overview of CimageData Class and specifications its meber functions..

Class in the following subsections:

5.1. CImageData Class Overview

5.1.1. About CImageData Class

CImageData Class has such functions as writing still image data, storing, LUT (Look Up Table) Correction and Image Transform into Windows Format.

One still frame requires one class. After generating an instance of CImageData, you need to initialize to secure an image memory buffer using Init member function. When an Exit member function is called out, all buffers are opened.
The size of buffers is automatically readjusted by the size of image file being read.

Note: The total size of buffers secured in a class is normally larger than that of the original image file.

■Class Member Functions

- Initialization Function
- Image operation functions
- File input/Output functions
- LUT functions

5.2. CImageData Class Member Function List

List of Member Functions of CCameraControl Class

Structure	
CImageData	CimageData Class Structure
Initialization functions	
Init	Initialization processes
Exit	Exiting processes
Image functions	
GetImageBuff	Gets image data
GetImageSize	Gets bitmap information
GetWindowsImage	Gets Windows display image
GetWindows16BitImage	Gets 16-bit Windows display image
GetBmpInfo	Gets BITMAP information
SetBmpInfo	Sets BITMAP information
File functions	
LoadFile	Reads file(Returned value:image format)
LoadRawFile	Reads RAW file
LoadRawwFile	Reads RAWW file
SaveFile	Stores File
SaveBmpFile	Stores BMP file
SaveTiffFile	Stores TIFF file
SaveTiffFileEx	Stores TIFF file
SaveJpegFile	Stores JPEG file
SavePnmFile	Stores PNM file
SaveRawFile	Stores RAW file
SaveRawwFile	Stores RAWW file
SavePngFile	Stores PNG file

LUT	
InitLut	Initializes LUT
SetUseLut	Use LUT ON/OFF
GetUseLut	Gets status of Use LUT ON/OFF
SetLut	Sets LUT
SetLutR	Sets LUT Red plane
SetLutG	Sets LUT Green plane
SetLutB	Sets LUT Blue plane
GetLut	Gets LUT
GetLutR	Gets LUT Red plane
GetLutG	Gets LUT Green plane
GetLutB	Gets LUT Blue plane
SetLutGamma	Sets LUT gamma correction
SetLutContrast	Sets LUT contrast correction
SetLutBrightness	Sets LUT brightness correction
GetLutGamma	Gets LUT gamma correction
GetLutContrast	Gets LUT contrast correction
GetLutBrightness	Gets LUT brightness correction

5.3. CImageData Class Member Function Specifications

This section describes specifications of member functions of the CImageData Class.

CImageData( );

Functions	Generates an instance of CImageData class.
Arguments	None
Returned values	None
Remarks	One CImageData class operates one still image.

BOOL Init(int buffer_size = 0xB4000);

Functions Initializes a CImageData class.

Arguments **buffer_size**

Specifies image size to secure an image buffer for internal processing.
Image size = Height x Width x Number of color channels
The size when omitted is 0 x B4000（w512 x h480 x p3）

Returned values A TRUE is returned for successful initialization and a FALSE for failure of initialization.

Remarks One CImageData class operates one still image.

BOOL Exit();

Functions	Exits CImageData class process (releases work memory area).
Arguments	None
Returned values	A TRUE is returned if successful, and a FALSE if failed.
Remarks	This funciton is normally called from the default destructor.

BYTE* GetImageBuff();

Functions	Gets a buffer of image data.
Arguments	None
Returned values	Returns from CImageData class a BYTE type pointer of the buffer that stores the image data. Returns a NULL when it is not initialized by a member function Init. It gets the uncorrected, original data even if LUT correction function is enabled.
Remarks	The image data acquired by this member function is a top-down image with its origin in the upper left corner. To get a Windows-standard bottom-up image, use GetWindowsImage member function.

void GetImageSize(int &width, int &height);

Functions Gets a size for image data.

Arguments

width

Specifies integer parameter that receives the image width.

height

Specifies integer parameter that receives the image height.

Returned values None

Remarks When initialization has not been done with an Init member function, or when there is no image data existing, both arguments - width and height - will have a “0”.

BYTE* GetWindowsImage(BYTE* buff = NULL, BITMAPINFO* bmpinfo = NULL);

Functions

Gets Windows-standard bottom-up image data obtained by correcting original image by the LUT.

Arguments

buff

Specifies the address of a BYTE type parameter for top-down type image data that is to be transferred to Windows-type standard image data.
When this function is omitted, the image data is nullified (specifies “NULL”) and the image buffer stored in the CImageData class is specified instead.

bmpinfo

Specifies the address of a BITMAPINFO structure that contains bitmap information of top-down type image data specified by “buff”.”
When this function is omitted, the image data is nullified (specifies “NULL”) and the bitmap information in the image buffer stored in the CImageData class is specified.

Returned values

Returns the BYTE type pointer of the Windows-standard bottom-up type image data processed by the LUT.
Returns a NULL if the system has not been initialized by an Init member function.
When LUT corrction function is active, the data processed by the LUT correction function is obtained.

Remarks

The pointer obtained in this process is a temporary one and nothing can be written into it.
The image data acquired by this member function is a Windows-standard image with its origin in the lower left corner. To get a top-down type image, use a GetImageBuff member function.

See [Platform SDK: Windows GDI] in the Microsoft Corporation MSDN Library for the details of BITMAPINFO structure.

WORD* GetWindows16BitImage(WORD* buff = NULL, BITMAPINFO* bmpinfo = NULL);

Functions

Gets Windows-standard bottom-up type image data processed in 16-bit data LUT correction.

Arguments

buff

Specifies the address of a WORD type parameter for top-down type image data that is to be transferred to Windows-type standard image data.
When this function is omitted, the image data is nullified (specifies “NULL”) and the image buffer stored in the CImageData class is specified instead.

bmpinfo

Specifies the address of a BITMAPINFO structure that contains bitmap information of top-down type image data specified by “buff”.”
When this function is omitted, the image data is nullified (specifies “NULL”) and the bitmap information in the image buffer stored in the CImageData class is specified.

Returned values

Returns the WORD type pointer of the Windows-standard bottom-up type image data processed by the LUT.
Returns a NULL if the system has not been initialized by an Init member function.
When LUT corrction function is active, the data processed by the LUT correction function is obtained.

Remarks

The pointer obtained in this process is a temporary one and nothing can be written into it.
The image data acquired by this member function is a Windows-standard image with its origin in the lower left corner. To get a top-down type image, use a GetImageBuff member function.

See [Platform SDK: Windows GDI] in the Microsoft Corporation MSDN Library for the details of BITMAPINFO structure.

BITMAPINFO* GetBmpInfo();

Functions Gets bitmap information of image data.

Arguments None

Returned values Returns the pointer of BITMAPINFO structure that contains the bitmap information of the image data.
Returns a NULL when this function has not been initialized by an Init member function.

Remarks When image data is written with a file input member function, the bitmap information of the data is stored here.

See [Platform SDK: Windows GDI] in the Microsoft Corporation MSDN Library for the details of BITMAPINFO structure.

BOOL SetBmpInfo(BITMAPINFO* bmi);

Functions Sets bitmap information of image data.

Arguments **bmi**

The pointer of BITMAPINFO structure with which the bitmap information of the data is stored.

Returned values A TRUE is returned if successful, and a FALSE if failed.

Remarks When image data is written with a file input member function, the bitmap information of the data is stored here.

See [Platform SDK: Windows GDI] in the Microsoft Corporation MSDN Library for the details of BITMAPINFO structure.

int LoadFile(LPCTSTR filename);

Functions Reads image data from a file.

Arguments **filename**
Specifies the file from which the data is read.

Returned values Returns the format name of the file that has been read.
When in error, a FORMAT_NONE is returned.

Remarks See [On File Format] for information about files that can be read. About File Formats

---

BOOL LoadRawFile(LPCTSTR filename, int size_header, int width, int height, int plane, BOOL interleave = TRUE);

Functions Reads image data from a RAW file.

Arguments

filename

Specifies the file name to read.

size_header

Specifies the size of header information.

width

Specifies the width of image data.

height

Specifies the height of image data.

plane

Specifies the number of color planes.

interleave

Specifies whether the image data is in an interleave format.

TRUE Interlleave

FALSE Non interleave

The default is TRUE.

Returned values Returns a TRUE when successful, and a FALSE if failed.

Remarks

BOOL LoadRawwFile(LPCTSTR filename, int size_header, int width, int height, int plane, BOOL interleave = TRUE);

Functions Reads image data from a RAWW file.

Arguments

filename
Specifies the file name to read.

size_header
Specifies the size of header information.

width
Specifies the width of image data.

height
Specifies the height of image data.

plane
Specifies the number of color planes.

interleave
Specifies whether the image data is in an interleave format.
TRUE Interlleave
FALSE Non interleave
The default is TRUE.

Returned value Returns a TRUE when successful, and a FALSE if failed.

Remarks

BOOL SaveFile(LPCTSTR filename, int compress = 0, int comp_quality = 75, BOOL ascii = FALSE);

Functions Stores image data in a file.

Arguments

filename

Specifies the file name to store

compress

TIFF format:

Specifies whether PACKBIT compression should be applied.

This can be omitted (default FALSE)

RAWW format:

Specifies grayscale of image data (default 16)

PNG format:

PNG_COMPRESS_NORMAL (default)

PNG_COMPRESS_SPEEDL

PNG_COMPRESS_SIZE

comp_quality

TIFF format:

Specifies the number of bits (8 or 16) per color plain.

JPEG format:

Specifies compression ratio with a value between 0 and 100.

The higher the value, the lower the compression rate with higher image quality.

This can be omitted (default 75).

RAWW format:

Specifies 10- or 16-bit transfer

EFFECTIVE_BITS_LOWSIDE lower

EFFECTIVE_BITS_HIGHSIDE higher (default)

PNG format:

Specifies grayscale of image data (default 8)

ascii

PNM format:

Specifies the data recording format.

TRUE Image data is stored in ASCII code.

FALSE Image data is stored in binary code.

This can be omitted (default FALSE).

RAWW format:

Specifies the front buffer.

.

Returned values A TRUE is returned when the data is stored correctly, and a FALSE when in error.

Remarks The file format for storage is determined by the file name extention.
For details of file formats that can be stored, see [File Format].

BOOL SaveBmpFile(LPCTSTR filename);

Functions Stores image data in a file in BMP format.

Arguments **filename**
Specifies the file name.

Returned values A TRUE is returned when the data is properly stored and a FALSE when in error.

Remarks

BOOL **SaveTiffFile(LPCTSTR filename,**
BOOL compress = FALSE);

Functions Stores image data in a file in TIFF format.

Arguments **filename**
Specifies the file name.
compress
Specifies whether PACKBIT compression should be applied.
This can be omitted (default FALSE)

Returned values A TRUE is returned when the data is properly stored and a FALSE when in error.

Remarks

BOOL **SaveTiffFileEx(LPCTSTR filename,**
BOOL compress = FALSE, int depth = 8);

Functions Stores image data in a file in TIFF format.

Arguments

filename
Specifies the file name.

compress
Specifies whether PACKBIT compression should be applied.
This can be omitted (default FALSE)

depth
Specifies the number of bits (8 or 16) per color plain.

Returned values A TRUE is returned when the data is properly stored and a FALSE when in error.

Remarks

BOOL SaveJpegFile(LPCTSTR filename, int quality = 75);

Functions Stores image date in a file in JPEG format.

Arguments

filename

Specifies the file name.

quality

Specifies the required compression rate for JPEG storage by a value between 0 and 100.
The higher the value, the lower the compression rate resulting in higher picture quality.
This can be omitted (default 75)

Returned values A TRUE is returned when image data is properly stored, and a FALSE when in error.

Remarks Supports unreversible compression only.

BOOL **SavePnmFile(LPCTSTR filename,**
BOOL ascii = FALSE);

Functions Stores image data in a file in PNM format.

Arguments **filename**

Specifies the file name.

ascii

Specifies the data storage format.

TRUE Stores image data portion in ASCI code.

FALSE Stores image data portion in binary code.

This can be omitted (default FALSE)

Returned values A TRUE is returned when data is properly stored, and a FALSE when in error.

Remarks See [File Format] for details of PNM format.About File Formats.

BOOL SaveRawFile(LPCTSTR filename);

Functions Stores image data in a file in RAW format.

Arguments **filename**
Specifies the file name.

Returned values A TRUE is returned when data is properly stored and a FALSE when in error.

Remarks See [File Format] for details of RAW format About File Formats.

BOOL **SaveRawwFile(LPCTSTR filename, int bit_depth = 16, int bit_side = RAWW_EFFECTIVE_BITS_HIGHSIDE, WORD* buff = NULL);**

Functions Stores image data in a file in RAWW format.

Arguments

filename

Specifies the file name.

bit_depth

Grayscale of the image data (default 16)

bit_side

Specifies 10- or 16-bit transfer

EFFECTIVE_BITS_LOWSIDE lower

EFFECTIVE_BITS_HIGHSIDE higher (default)

buff

Specifies the top buffer.

Returned values A TRUE is returned when the data is stored properly, and a FALSE when in error.

Remarks See [File Format] for the details of the RAWW format
About File Formats..

BOOL **SavePngFile(LPCTSTR filename,**
int comp_mode = PNG_COMPRESS_NORMAL,
int depth = 8);

Functions Stores image data in a file in PNG format.

Arguments

filename

Specifies the file name.

comp_mode

Selects a compression mode:

PNG_COMPRESS_NORMAL	(default)
PNG_COMPRESS_SPEEDL	priority on speed
PNG_COMPRESS_SIZE	priority on size

depth

Specifies the grayscale of image data.

Returned values A TRUE is returned when the data is stored properly, and a FALSE when in error.

Remarks See [File Format] for the details of the PNG format About File Formats.

void InitLut();

Functions	Initializes the LUT (Look Up Table) Returns to the defaule value (IN = OUT)
Arguments	None
Returned values	None
Remarks	See [LUT (Look Up Table)] for the details of LUT.

void SetUseLut(BOOL on);

Functions Sets ON/OFF of the correction process in LUT.

Arguemnts **on**

Specifies ON or OFF of the LUT correction process

TRUE	Process ON
FALSE	Process OFF

Returned values None

Remarks See [LUT (Look Up Tabel)] for details of LUT.LUT.

BOOL GetUseLut();

Functions Gets ON/OFF status of the LUT correction process.

Arguments None

Returned values Returns ON or OFF status of the LUT process.

TRUE	Process ON
FALSE	Process OFF

Remarks See [LUT (Look Up Table)] for details of LUT.LUT.

void SetLut(int *lut, int plane = LUT_PLANE_ALL);

Functions Sets up the LUT.

Arguments

lut

Specifies the address of an integer memory array that contains table data.
The number of elements is 256.

plane

Planes to be set:

LUT_PLANE_ALL	All planes
LUT_PLANE_R	R (Red) plane
LUT_PLANE_G	G (Green) plane
LUT_PLANE_B	B (Blue) plane

Returned values None

Remarks When the image data is a grayscale (monochrome) or of the 256 palatte color, only the R table is used.

See [On the LUT (Look Up Table)] for the details of the LUT.

void SetLutR(int *lut);

Functions Sets the Red Table of the LUT.

Arguments **lut**

Specifies the address of integer memory array containing the table data.
The number of elements is 256.

Returned values None

Remarks Only this Red Table is used when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT.LUT.

void SetLutG(int *lut);

Functions Sets the Green Table of the LUT.

Arguments **lut**

Specifies the address of integer memory array containing the table data.
The number of elements is 256.

Returned values None

Remarks This table is void when the image data is in grayscale or in 256-color palette color.
See [On the LUT (Look Up Table)] for details of LUT.LUT.

void SetLutB(int *lut);

Functions Sets the Blue Table of the LUT.

Arguments **lut**

Specifies the address of integer memory array containing the table data.
The number of elements is 256.

Returned values None

Remarks This Blue Table is void when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT.

void GetLut(int *lut, int plane = LUT_PLANE_ALL);

Functions Gets settings of the LUT.

Arguments

lut

Specifies the address of an integer memory array that receives the table data.
The number of elements is 256 (1024 for LUT_PLANE_ALL).

plane

Planes to be set:

LUT_PLANE_ALL	All planes
LUT_PLANE_R	R (Red) plane
LUT_PLANE_G	G (Green) plane
LUT_PLANE_B	B (Blue) plane

Returned values None

Remarks When the image data is a grayscale (monochrome) or of the 256-color palatte color, only the R table is used.

See [On the LUT (Look Up Table)] for the details of the LUT.LUT.

void GetLutR(int *lut);

Funcitions Gets the setting of Red Table of the LUT.

Arguments **lut**

Specifies the address of an integer memory array that receives the table data.
The number of elements is 256.

Returned values None

Remarks Only this Red Table is used when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT.LUT.

void GetLutG(int *lut);

Funcitions Gets the setting of Green Table of the LUT.

Arguments **lut**

Specifies the address of an integer memory array that receives the table data.
The number of elements is 256.

Returned values None

Remarks This Green Table is void when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT.LUT.

void GetLutB(int *lut);

Funcitions Gets the setting of Blue Table of the LUT.

Arguments **lut**

Specifies the address of an integer memory array that receives the table data.
The number of elements is 256.

Returned values None

Remarks This Blue Table is void when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT.LUT.

---

void **SetLutGamma(double gamma,**
int plane = LUT_PLANE_ALL);

Functions Sets the LUT with gamma correction values

Arguments

gamma

Sets the gamma correction values.
The range of gamma is 0.1 to 2.0.

plane

Specifies the color plane (table) that performs gamma correction.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values None

Remarks The plane values are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT

---

void **SetLutContrast(int contrast,**
int plane = LUT_PLANE_ALL);

Functions Sets the LUT with contrast correction values

Arguments

gamma

Sets the contrast correction values.
The range of contrast is between –256 and + 256.

plane

Specifies the color plane (table) that performs contrast correction.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values None

Remarks The plane values are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT.

---

void **SetLutBrightness(int brightness,**
int plane = LUT_PLANE_ALL);

Functions Sets the LUT with brightness correction values

Arguments

gamma

Sets the brightness correction values.
The range of gamma is between –256 and +256.

plane

Specifies the color plane (table) that performs brightness correction.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values None

Remarks The plane values are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT.

double GetLutGamma(int plane = LUT_PLANE_ALL);

Functions Gets the LUT gamma correction values.

Arguments **plane**

Specifies the color planes (table) that get a gamma correction value.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values Returns the current gamma correction values.
If LUT_PLANE_ALL is specified after each plane is given a correction value independently, the returned value will be indefinite.

Remarks The values in the plane are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT.

int GetLutContrast(int plane = LUT_PLANE_ALL);

Functions Gets the LUT contrast correction values.

Arguments **plane**

Specifies the color planes (table) that get a contrast correction value.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values Returns the current contrast correction values.
If LUT_PLANE_ALL is specified after each plane is given a correction value independently, the returned value will be indefinite.

Remarks The values in the plane are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT. LUT.

int GetLutBrightness(int plane = LUT_PLANE_ALL);

Functions Gets the LUT brightness correction values.

Arguments **plane**

Specifies the color planes (table) that get a brightness correction value.

LUT_PLANE_R	Red (R) Plane
LUT_PLANE_G	Green (G) Plane
LUT_PLANE_B	Blue (B) Plane
LUT_PLANE_ALL	All Planes

Returned values Returns the current brightness correction values.
If LUT_PLANE_ALL is specified after each plane is given a correction value independently, the returned value will be indefinite.

Remarks The values in the plane are disregarded when the image data is in grayscale or in 256-color palette color.

See [On the LUT (Look Up Table)] for details of LUT.LUT.

6. Compatible with Old Version SDK (Unused) /Compatible with Old Version SDK

Extra commands are issued for each of camera models using the CcameraControl Class [ExtraCommand] member functions.
Extra commands are shown below:

Compatible with Old Version SDK (Unused)	
EXTRA_SET_MONITOROUT_EDGEENHANCE	SetEdgeEnhancementMode
EXTRA_GET_MONITOROUT_EDGEENHANCE	HaveEdgeEnhancement
EXTRA_SET_MONITOROUT_ZOOM	SetZoomMode (HaveZoomMode)
EXTRA_GET_MONITOROUT_ZOOM	GetZoomMode (HaveZoomMode)
EXTRA_SET_MONITOROUT_DISP_STATUS	SetStatusDisplayMode
EXTRA_GET_MONITOROUT_DISP_STATUS	GetStasusDisplayMode
EXTRA_MONITOROUT_PLAY	Play
EXTRA_MONITOROUT_PAUSE	Pause
EXTRA_MONITOROUT_STOP	Stop
EXTRA_MONITOROUT_STEP_FORWARD	StepForward
EXTRA_MONITOROUT_STEP_BACK	StepBack
EXTRA_MONITOROUT_GOTO_ANYFRAME	GoAnyFrame
EXTRA_MONITOROUT_GOTO_START	GoStart
EXTRA_MONITOROUT_GOTO_END	GoEnd
EXTRA_MONITOROUT_GOTO_TRIGGER	GoTrigger
Compatible with Old Version SDK	
EXTRA_SET_MONITOROUT_MODE	SetMonitorOutMode (HaveMonitorOutMode)
EXTRA_GET_MONITOROUT_MODE	GetMonitorOutMode (HaveMonitorOutMode)
EXTRA_SET_ENABLE_MCDL	SetEnableMCDL (HaveMCDL)
EXTRA_GET_ENABLE_MCDL	GetEnableMCDL (HaveMCDL)
EXTRA_SET_ENABLE_IRIG	SetEnebleIRIG (HaveIRIG)
EXTRA_GET_ENABLE_IRIG	GetEnableIRIG (HaveIRIG)
EXTRA_TRANSFER_MCDL	TransferMCDL
EXTRA_TRANSFER_IRIG	TransferIRIG
EXTRA_SHADING_COMPENSATION	ShadingCompensation

7. File Formats

File formats that can be used in this class library are shown below:

7.1. BMP File (*.bmp)

The bitmap format that is used as a standard format in Windows environment.
Compatible with full color, 8-bit palette color and grayscale.

7.2. TIFF File (*.tif)

Another bitmap file format（Tagged Image File Format）
Supports noncompression and PackBit compression files only. Default is noncompression.

7.3. JPEG File (*.jpg)

A file format for compressed still image.
This library supports Lossy (non-reversible) compression only.
The compression parameter can be set between 0 and 100. The highe the number, the lower the compression rate resulting in higher picture quality. Default is 75.

7.4. PNM File (*.ppm / *.pgm)

Another bitmap file format（Tagged Image File Format）.
A file extension changes with contents of a image.
PNM is the general term of the following format.

Color :PPM (Portable Pixel Map)
Glayscale :PGB (Portable Gray Map)
2Value :PBM (Portable Bit Map) (not supported)

Image data portion can be saved in binary data or ASCII format.

7.5. RAW File (*.raw / *.raww)

A binary data file format without header information.
Data of color image is stored in the order of interleave (RGBRGBRGB..etc.).
RAWW file consists of 16-bit data.

7.6. PNG File (*.png)

A file format for compressed still image data which uses reversible compression technique.
Three compression parameters are available to choose from: priority on image quality, normal, and priority on speed.

8. Look Up Table (LUT)

The CImageData Class and its derivative classes have LUT funcitions that offer the capability to freely correct displayed images without affecting the original image data. Image data corrected on LUT is acquired by GetWindowsImage member function of the CImageData Class. Valid/Invalid of LUT can be switched by the [SetUseLut] member function.

The LUT prepares and sets an integer array for each of the color planes. Each array contains the value converted from the 256-step density scale of the original image. For grayscale of 256 steps, only the [SetLutR] member function is used.

***Note: For 256-color palette color, the whole palette is converted.**

8.1. Example of application: Negative inversion of image

```
//LUT data of Red, Green, Blue planes
        int         lut_r[256], lut_g[256], lut_b[256];

//Store density-reversed values

        for (int I = 0; i <= 255; i++) {
                lut_r[i] = 255 – i;
                lut_g[i] = 255 – i;
                lut_b[i] = 255 – i;
        }

// Acquire an original image
        CImageData          image;
        image.LoadFile(”sample.bmp”);

// Set LUT
        image.SetLutR(lut_r);
        image.SetLutG(lut_g);
        image.SetLutB(lut_b);

// Make LUT valid
        image.SetUseLut(TRUE);

// Acquire LUTcorrected image
        BYTE* pBuff = image.GetWindowsImage();
```

www.photron.com

In Americas and Antepodes
PHOTRON USA, INC.
9520 Padgett Street, Suite 110
San Diego, CA 92126-4446, USA
Phone: 858-684-3555
Fax: 858-684-3558
E-mail: image@photron.com

In Europe:
PHOTRON EUROPE LIMITED
Willowbank House
84 Station Road
Marlow, Bucks SL7, UK
Phone:+44(0) 1628 89 4353
Fax: +44(0) 1628 89 4354
E-mail: image@photron.com

In other areas:
PHOTRON LIMITED
Chiyodafujimi BLDG.,
Fujimi 1-1-8, Chiyoda-Ku
Tokyo 102-0071, Japan
Phone:+81 3 3238 2106
Fax: +81 3 3238 2109
E-mail: image@photron.com

FASTCAM Control SDK Library "PccLib" Reference Manual
English Edition
April, 2006
Rev. 2.976