

*FASTCAM Control  
SDK Library "Pcc4VB"  
Programming Manual*

---

*Rev.1.00*

*English Edition*

*Pcc4VB Ver2.9.3.7 or later*

*PHOTRON LIMITED  
2001-2004*

---

## Table of contents

### Chapter 1. [Outline of This Programming Manual](#)

- 1.1. [Who Should Read This Manual](#)
- 1.2. [Operating Environment for The Library](#)

### Chapter 2. [Operation Flow Charts](#)

- 2.1. [SDK General Operation Flow](#)
- 2.2. [SDK Preparation, Live Operation/Recording and Shutdown](#)
  - 2.2.1. [Prepare SDK File](#)
  - 2.2.2. [Merge SDK File in Development Environment](#)
  - 2.2.3. [Connect Camera Systems](#)
  - 2.2.4. [Initialize Camera](#)
  - 2.2.5. [Obtain Camera Status](#)
  - 2.2.6. [Set up Camera Status](#)
  - 2.2.7. [Set up Camera for Live Operation](#)
  - 2.2.8. [Obtain Live Image](#)
  - 2.2.9. [Prepare for Recording](#)
  - 2.2.10. [Start Recording](#)
  - 2.2.11. [End Recording](#)
  - 2.2.12. [Cut off Camera](#)
- 2.3. [Obtain and Store Recorded Image](#)
  - 2.3.1. [Setup for Obtaining Recorded Image](#)
  - 2.3.2. [Obtain Recorded Image](#)
  - 2.3.3. [Store Recorded Image](#)
- 2.4. [Processing Image Obtained from Camera](#)
  - 2.4.1. [Prepare Image Processing Data](#)
  - 2.4.2. [Initialize Image Processing Data](#)
  - 2.4.3. [Setup for Live Operation and Obtaining Recorded Image](#)
  - 2.4.4. [Obtain Live Operation and Recorded Image](#)
  - 2.4.5. [Apply Obtained Image to Image Processing Data](#)
  - 2.4.6. [Store Image Processing Data](#)
  - 2.4.7. [Terminal Processing of Image Processing Data](#)
- 2.5. [Read and Write Data Files](#)
  - 2.5.1. [Prepare Image Processing Data](#)
  - 2.5.2. [Initialize Image Processing Data](#)
  - 2.5.3. [Obtain Filed Image Data](#)
  - 2.5.4. [Apply Obtained Image to Image Processing Data](#)
  - 2.5.5. [Store Image Processing Data](#)
  - 2.5.6. [Terminal Processing of Image Processing Data](#)

### *Chapter 3. [Other Operations](#)*

- 3.1. [Connection to Multiple Cameras \(Ref. 2.2.3.\)](#)
- 3.2. [Access to Multiple Files \(Ref. 2.5.3.\)](#)
- 3.3. [Trigger Setting and Recording Operation \(Ref. 2.2.8. to 2.2.10.\)](#)
  - 3.3.1. [Start Trigger](#)
  - 3.3.2. [Center Trigger](#)
  - 3.3.3. [End Trigger](#)
- 3.4. [Partitioning \(Ref. 2.2.6.\)](#)
  - 3.4.1. [Setups](#)
  - 3.4.2. [Setups \(Block\)](#)
  - 3.4.3. [Switching](#)

### *Chapter 4. [Procedure or Operation – Camera Connection to Recording](#)*

- 4.1. [Examples of Programming \(VisualBasic6.0/VisualBasic.NET\)](#)

## *Chapter 1. Outline of This Programming Manual*

Congratulations on the acquisition of your Photron High-Speed Video Camera.

This chapter describes how to use the **FASTCAM Control SDK Library** and the required environment for it.

Before using this library, connection of a high-speed camera(s) to a computer (PC) and installation of the hardware driver are necessary. See the user's manuals of hardware equipment for instructions for connection.

The method of hardware driver installation varies by the model of high-speed camera. See the driver installation manual attached to your high-speed camera and install in the correct manner before using the SDK library.

### **Section 1.1. Who Should Read This Manual**

This Manual has been compiled for users who wish to develop application software programs using the FASTCAM Control SDK, the basic software for generating applications using Photron high-speed video cameras. It is written assuming the users have experience in using development tools for VisualBasic6.0 and VisualBasic.NET, and sufficient knowledge of technical terms for programming.

\* This SDK uses 32-bit variables for the VisualBasic6.0 throughout.

When you wish to use this SDK with the VisualBasic.NET, you need to convert the variables to VisualBasic.NET 32-bit variables to interpret them.

## Section 1.2. Operating Environment for The Library

The environment required for operating this library is as follows:

Computer:	PC/AT compatible computer
Operating System:	Microsoft Windows98/98Se/Me, Windows2000 Professional, XP Professional, XP Home *1, *2
CPU:	Intel Pentium or compatible CPU, *3 PentiumIII 1GHz or higher recommended
Memory:	64MB minimum; 256MB recommended for multiple camera or high-resolution camera operation
HDD:	Over 20MB space needed of library installation; 500MB recommended for program development works
Development Environment:	Microsoft VisualBasic6.0 or later
Others:	Large capacity HDD or removable media recommended to store recorded image data; CD-ROM drive recommended for installation of software

Notes:

\*1: OS depends on the camera and PC to be used. Refer to the operation manual of the camera.

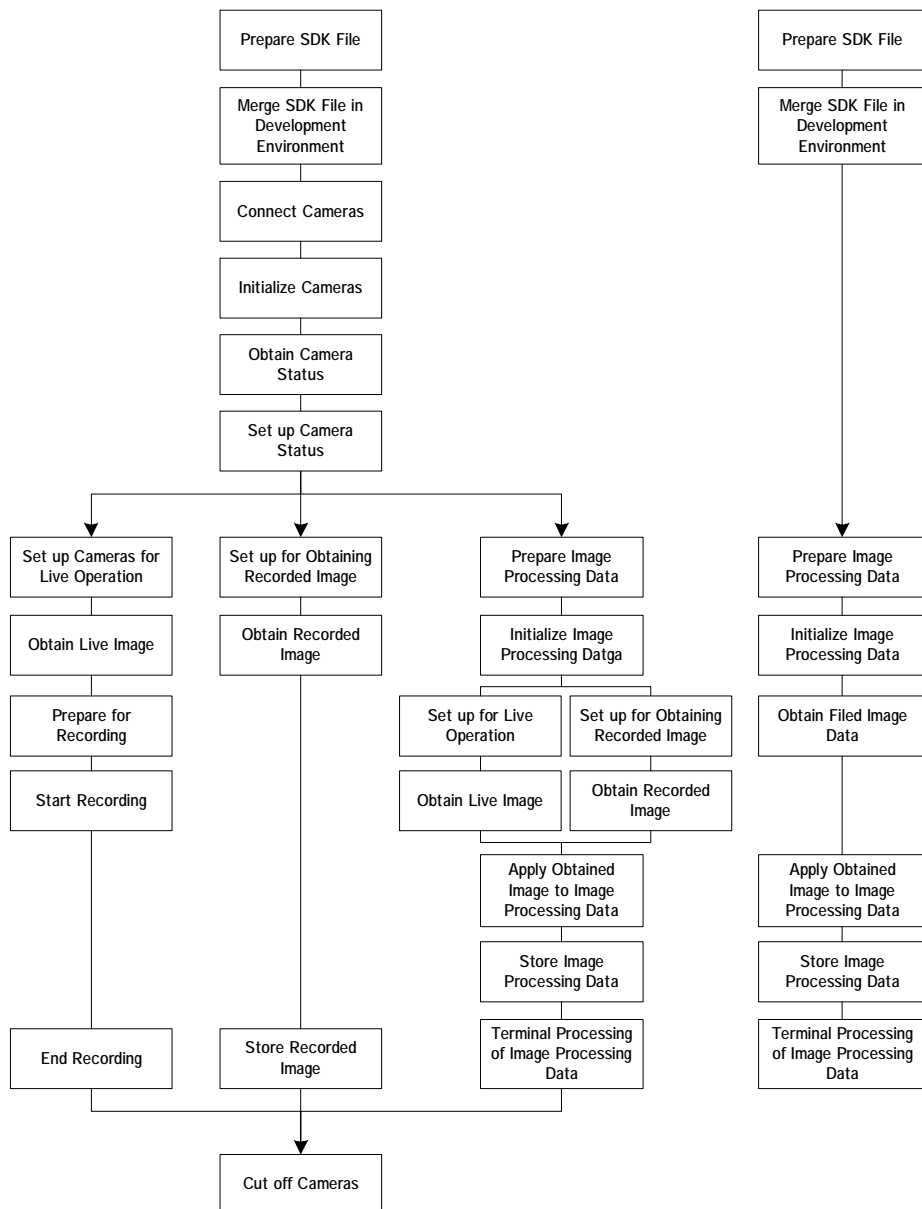
\*2: Microsoft, Windows, VisualBasic are registered trademarks or trademarks of Microsoft Corporation in the US and other countries.

\*3: Pentium is a registered trademark of Intel Corporation.

## *Chapter 2. SDK Library Operation*

This chapter describes the four basic operations to use the FASTCAM Control SDK.

## Section 2.1. SDK General Operation Flow Charts.



Section 2.2. discusses from preparation of the SDK to live operation to recording to ending the SDK operation.

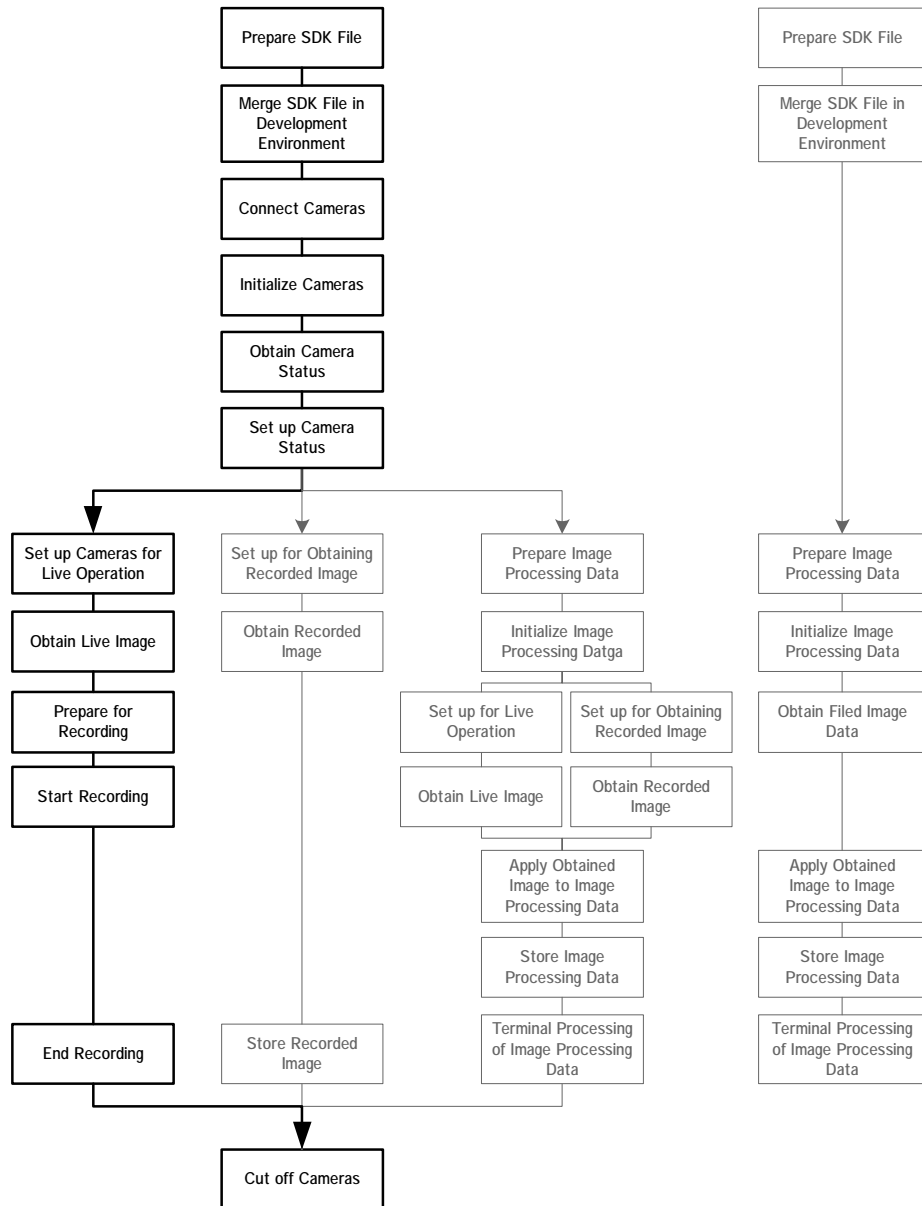
Section 2.3. discusses how to obtain and store image data recorded by camera.

Section 2.4. discusses how to store image data, obtained from a camera, using image processing library prepared under the SDK.

Section 2.5. discusses how to use the image processing library to read and write stored image data.



## Section 2.2. Flow Chart of Preparation to Live Operation to Recording to Ending of SDK Operation



Section 2.2. discusses from preparation of SDK to live operation to recording to ending SDK operation.

### Subsection 2.2.1. Preparation of SDK File

The structure of files contained in the SDK Library is as follows:

#### Dll¥

<b>PccLib.dll</b>	PccLib main DLL
<b>Pcc4VB.dll</b>	Pcc4VB library main DLL (VisualBasic6.0/VisualBasic.NET Only)
<b>FcamPCI.dll</b>	Device DLL for FASTCAM-PCI
<b>FcamPCI2.dll</b>	Device DLL for FASTCAM-PCI R2
<b>FcamPLMV.dll</b>	Device DLL for FASTCAM-X 1280PCI
<b>Fcam1394.dll</b>	Device DLL for IEEE1394I/F compatible FASTCAM series cameras
<b>FcamEth.dll</b>	Device DLL for Ethernet compatible FASTCAM series cameras
<b>FcamOpt.dll</b>	Device DLL for Optical I/F compatible FASTCAM series cameras
<b>FcamNPCI.dll</b>	Device DLL for FASTCAM-512PCI
<b>Ij15.dll</b>	JPEG library DLL
<b>IrigLib.dll</b>	IRIG library DLL

#### Modules¥

<b>FastcamDIIDriver.bas(VisualBaic6.0)</b>
<b>FastcamDIIConstantValues.bas(VisualBaic6.0)</b> Pcc4VB library definition File
<b>FastcamDIIDriver.vb(VisualBaic.NET)</b>
<b>FastcamDIIConstantValues.vb(VisualBaic.NET)</b> Pcc4VB library definition File

### **Subsection 2.2.2. Merge SDK File in Development Environment**

This subsection describes how to merge this library in the development environment (Visual Basic 6.0/Visual Basic.NET) and necessary setups.

#### **1) To set up a project in Visual Basic 6.0 environment**

Add the library to Visual Basic 6.0 in the following manner:

Add the Library to Visual Basic 6.0 in the following manner:

1. Start up Visual Basic 6.0 and open a project in which this Library is to be installed.
2. Select "Project" from the Visual Basic 6.0 menu and open the "Add Files" dialog.
3. add two definition files for libraries to a project  
( **FastcamDIIDriver.bas/FastcamDIIConstantValues.bas**)

#### **2) To set up a project in Visual Basic.NET environment**

Add the Library to VisualBasic.NET in the following manner:

1. Start up VisualBasic.NET and open a project in which this Library is to be installed.
2. Select "Project" from the VisualBasic.NET menu and open the "Add Existing Item" dialog.
3. add two definition files for libraries to a project  
( **FastcamDIIDriver.bas/FastcamDIIConstantValues.bas**)

#### **Note on Operation of Applications:**

To operate applications that are made by incorporating this library, all DLL files must be placed in the same folder as the files to be executed or in a folder assigned with an effective PATH within the system.

It is possible to keep copies of DLL files in the Windows system folder. But you must be careful because the specification of files may have to be changed at a version upgrade and newly revised SDK library may erroneously refer to the DLL files of older version remaining in the system folder.after revision.

### **Subsection 2.2.3. Connect to Cameras**

Right after the application with FASTCAM Control SDK incorporated has started up, the FASTCAM Control SDK has no control over the connected camera. The FASTCAM Control SDK and the cameras must be connected by the following process:

#### **1) Construct Camera Control**

By constructing a camera control , the FASTCAM Control SDK detects a camera connected to the PC and turns it controllable.

The following shows an example of program to construct a camera control:

```
// construct camera control  
Ret = CameraControl(DEVICE_SELECT_AUTO)
```

DEVICE\_SELECT\_AUTO is an argument that makes detectable all the cameras that can be connected. By changing this argument, only cameras that are connected with relevant interface can be detected.

Example) To detect cameras that can only be connected with IEEE1394 interface:

```
Ret = CameraControl(DEVICE_SELECT_1394)
```

One camera control controls any one single camera that can be connected (when DEVICE\_SELECT\_AUTO is selected). Therefore, in order to control all cameras connected to the PC, the same number of camera control, as the connected cameras, must be prepared.

```
// Construct camera control (to control camera 1)  
Ret = CameraControl(DEVICE_SELECT_AUTO)  
// Construct camera control (to control camera 2)  
Ret = CameraControl(DEVICE_SELECT_AUTO)  
// Construct camera control (to control camera 3)  
Ret = CameraControl(DEVICE_SELECT_AUTO)
```

## **2) Obtain The Number of Cameras to Connect**

When the construction of camera control library is successful, the number of the cameras connected within the camera control class can be set up.

The number of cameras connected is obtained using the `GetNumberOfDevice` function. If the number of connected cameras counts 1 or more, it indicates the construction of camera control class was successful.

Example of program for obtaining the number of connected cameras is shown below:

```
// Obtain number of connected cameras
Ret = GetNumberOfDevice(nMaxCameranumber)
If(Ret <> PCC_ERROR_NO_ERROR) then
    Ret = ExitCameraControl()Endif
```

See the reference manual for the reset value of the function.

### **Subsection 2.2.4 Initialize Camera**

Using the control for a successfully connected camera, initialize each camera control library prepared for the camera model and interface type. The status of the connected camera is confirmed by initialization.

The following is an example of a program to initialize a connected camera:

```
// Initialize FASTCAM Control SDK and each camera control library
Ret = SetCameraNumber(1)

Ret = Init(1)
```

The argument for initialization command is the control number of the connected camera. The highest value is the maximum number obtained by the `GetNumberOfDevice` function.

See the reference manual for the reset value of the function.

## **Subsection 2.2.5 Obtain Status to Camera**

Once the normal operation of the connected camera has been confirmed by initialization of camera control library prepared for each camera, obtain the current setting status of the connected camera to confirm it.

By confirming the setting status, it is possible to confirm the setting status for framing conditions and recording method with the connected camera. The items with which you should confirm the setting status of a camera varies by the camera model. Refer to the hardware manual of each camera to obtain and confirm the setting status of each camera as necessary.

The setting status of a camera is managed by dividing items into two groups – basic functions and extended functions. To obtain the setting status of a camera, you can use either of the two methods – group acquisition of all items or one-by-one acquisition of each of items (functions).

However, because some of the setting status can only be obtained by the group acquisition method, you must be careful when selecting a method depending on the nature of the software to develop.

### **1) Group acquisition – to obtain the camera status at one time**

This method obtains, at one time, all setting status that was obtained at initialization of the camera within the camera control library or the camera status after any change was made.

The following is an example of program to obtain camera status at one time:

```
Dim params As CAMERA_PARAMS // Base camera parameter structure
Dim params_ex As CAMERA_PARAMS_EX // Extended camera parameter structure

// Group acquisition of setting status of camera's basic functions
Ret = GetCameraParams(params)

// Group acquisition of setting status of camera's extended functions
Ret = GetCameraParamsEx(params_ex)
```

Information on camera is contained in CAMERA\_PARAMS and CAMERA\_PARAMS\_EX structures.

Decision or setting of functions that can be set on camera can be done by obtaining or resetting the both structures.

See the reference manual for setting status of basic and extended functions that are obtainable.

## **2) One-by-one acquisition of camera status**

This method lets you obtain, one by one, setting status all functions that was obtained at initialization of the camera within the camera control library or the camera status after any change was made.

The following shows an example of program to obtain camera setting status one by one:

```
Dim list(LIST_MAX_NUMBER - 1) As Integer // Record rate list
Dim nRecordRate As Integer // Record rate
```

The LIST\_MAX\_NUMBER being used in the size which was prepared to obtain each of setting values is the maximum list size defined in the header file of the camera control library.

```
// Obtain monochrome/color mode of the connected camera
Ret = IsColor(color)
```

This decides whether the connected camera is color or monochrome model. [1] for the argument of the IsColor function means color model and [0] monochrome.

```
// Obtain record rate list
Ret = GetRecordRateList(list(0), LIST_MAX_NUMBER)
```

Any of the record rates can be obtained that are set on the connected camera. Note that some camera models have record rates other than that are normally set by this function.

When you wish to obtain a list of set values, be sure to have a size for variables for acquisition using LIST\_MAX\_NUMBER.

```
// Obtain current record rate
Ret = GetRecordRate(m_nRecordRate)
```

Use this to obtain the framing rate currently set on the connected camera.

See the reference manual for the reset value of function.



### **Subsection 2.2.6 Set up Camera Status**

Recording conditions (resolution, recording method, etc.) may be set by changing the camera setting status.

By setting the camera status, you can set the status, such as record rate and mode of recording, that the connected camera can work in.

After setting the status, confirm the newly set status on the camera and make sure if the camera is correctly operating as it has been set.

The items to set vary by the camera model. Refer to the hardware manual of each camera before making necessary setting.

The camera setting status is managed by dividing the items into two groups – basic functions and extended functions. To set a camera, you can use either of the two methods – group setting of all items or one-by-one setting of each of items (functions).

However, because some of the setting status can only be attained by the group setting method, you must know the nature of the software that you wish to develop before selecting a method..

#### **1) Group setting – to set the camera status at one time**

This method sets all setting status on a camera at one time.

The following is an example of program to set camera status at one time:

```
Dim params As CAMERA_PARAMS // Camera parameter structure
Dim params_ex As CAMERA_PARAMS_EX // Extended camera parameter structure

// Group setting of camera's basic functions
Ret = SetCameraParams(params)

// Group setting of camera's extended functions
Ret = SetCameraParamsEx(params_ex)
```

When you change settings on a camera, all of items that can be set on CAMERA\_PARAMS or CAMERA\_PARAMS\_EX structure must be reset. On some models of camera, it may be a time-consuming procedure..

See the reference manual for basic and extended functions that are resettable.

## **2) One-by-one setting of camera functions**

This method lets you set functions of the camera one by one.

The following shows an example of program to set functions of camera one by one:

```
Dim nRecordRate As Integer // Record rate  
  
// Set current record rate  
Ret = SetRecordRate(m_nRecordRate)
```

The SetRecordRate function changes the record rate.

The record rate to be set as the argument is a value obtained by the GetRecordRateList function.

See the reference manual for the reset value of functions.

### **Subsection 2.2.7 Set up Camera for Live Operation**

This subsection discusses how to set up the connected camera in the live operation mode to work under the recording conditions (record rate, resolution, etc.) currently set on it. By setting the connected camera in the live mode, you can obtain the live image from the camera as image data.

#### **1) Confirm Live Operation**

Confirm if the connected camera is the live operating status.

```
// Decide if camera is live  
Ret = IsLive(check)
```

When the argument for the IsLive function is [1], the camera is live. When it is [0], the camera is playing image recorded in memory.

#### **2) Set up for Live Operation**

Set up the connected camera in live status.

```
// Set up in live status  
Ret = OnLive(1)
```

The argument for the OnLive function is used to set up the camera status.

When the argument is [1], the camera is set up in live status. When it is [0], the camera is set to play back recorded image.

```
// Decide if camera is live  
Ret = IsLive(check)
```

After changing the camera status using the OnLive function, use the Is Live function to confirm if the camera has been correctly set in the new status.

See the reference manual for the reset values of the functions.

### **Subsection 2.2.8 Obtain Live Image**

After confirming the live status of the connected camera, obtain image to make sure the live image is transferred to the PC correctly.

The size of image data is determined by the image size obtained by camera status and the type of the camera – monochrome or color.

The live image data obtained from a monochrome camera is binary data with origin at the upper left corner of the image, while image data from a color camera is of interleaved format (RGBRGB.....).

The following shows an example of program to obtain live image from the camera:

```
// Obtain live image  
Ret = TransferFrame(-1, ImageDataBuf(0))
```

To obtain live image data, the first argument for the TransferFrame is [-1] and the second argument denotes the memory area to store the image data obtained from the camera.

Before transferring the image data, determine the second argument taking into consideration the camera resolution and necessary memory area.

See the reference manual for the reset value of functions.

### **Subsection 2.2.9 Prepare for Recording**

The cameras that are controlled by the FASTCAM Control SDK must be prepared for recording. By the preparation procedure described here, the cameras are made ready for a command to start recording.

The following is an example of program to make the connected camera ready for recording:

```
// Set up for recording ready  
Ret = OnRecordReady(1)
```

The argument for the OnRecordReady function is used to set the camera into record-ready status. [1] for the argument makes the camera ready for recording and [0] cancels the setting of ready status.

```
// Confirm record ready status  
Ret = IsRecordReady(check)
```

After setting the camera by the OnRecordReady function, confirm the record-ready status using the IsRecordReady function. If the argument of the IsRecordReady function is [1], the camera is ready to record, but if it is [0], the camera is not ready.

See the reference manual for the reset values of functions

### **Subsection 2.2.10 Start Recording**

The procedures in the previous subsections have set the camera into the record-ready status. The camera is now ready to start recording at a trigger.

The trigger mode (i.e. START mode), which requires issuance of a record trigger at the beginning of recording, a command for the TriggerIn function is also issued, at the start of recording, which sets the base point of recording.

For other trigger modes (i.e. CENTER, RANDOM, etc.) a command for TriggerIn function must be issued to determine the base point frame number for a recording, in addition to one for the OnRecord function.

Note: Descriptions on recording operations in this manual are for operation by software only. For recording by hardware triggers, see the user's manual for each camera.

The following shows an example of program to record on the connected camera:

```
// Set to start recording  
Ret = OnRecord(1)
```

Set [1] as the argument for the OnRecord function, and the camera starts recording.

```
// Confirm recording status  
Ret = IsRecord(check)
```

Use the IsRecord function to confirm if the camera has started recording. If the argument for the IsRecord function is [1], the camera is now recording, and if the argument is [0], the camera has finished recording.

See the reference manual for the reset values of functions.

### **Subsection 2.2.11 End Recording**

The camera control library does not monitor the camera to end recording. You must confirm and monitor the status while the camera is in the process of recording.

Confirm the end of recording on the camera in the process below:

#### **1) How to Confirm End of Recording**

```
// Decide end of recording  
Ret = IsRecord(check)
```

Use the IsRecord function to confirm the end of recording as well as start. While the camera is recording, the argument of the IsRecord function is [1]. When a recording ends, the argument turns to [0].

#### **2) Force to End Recording**

You can force the camera to end recording at any time you wish to cancel or abort the recording. If a recording is cancelled halfway, the recorded image up to the moment of cancellation remains in memory.

The following shows an example of program to force the camera to end recording:

```
// Force to end recording  
Ret = OnRecord(0)
```

Set [0] as the argument for the OnRecord function and the camera is forced to end recording.

See the reference manual for the reset values of functions.

### **Subsection 2.2.12 Cut off Camera**

As soon as control finishes on the connected camera, the camera control library deletes the information on the camera and returns to the status prior to initialization. The connection between the Control SDK and the camera is retained within the Control SDK until the parameters of constructed camera control library are released.

To restart to operate the camera after cutoff, you must go back to [Subsection 2.2.4.](#) and redo the whole procedure from initialization of the camera.

The following shows an example of program to end operating a camera.

```
// End camera control  
Ret = ExitCameraControl()
```

See the reference manual for the reset values of functions.



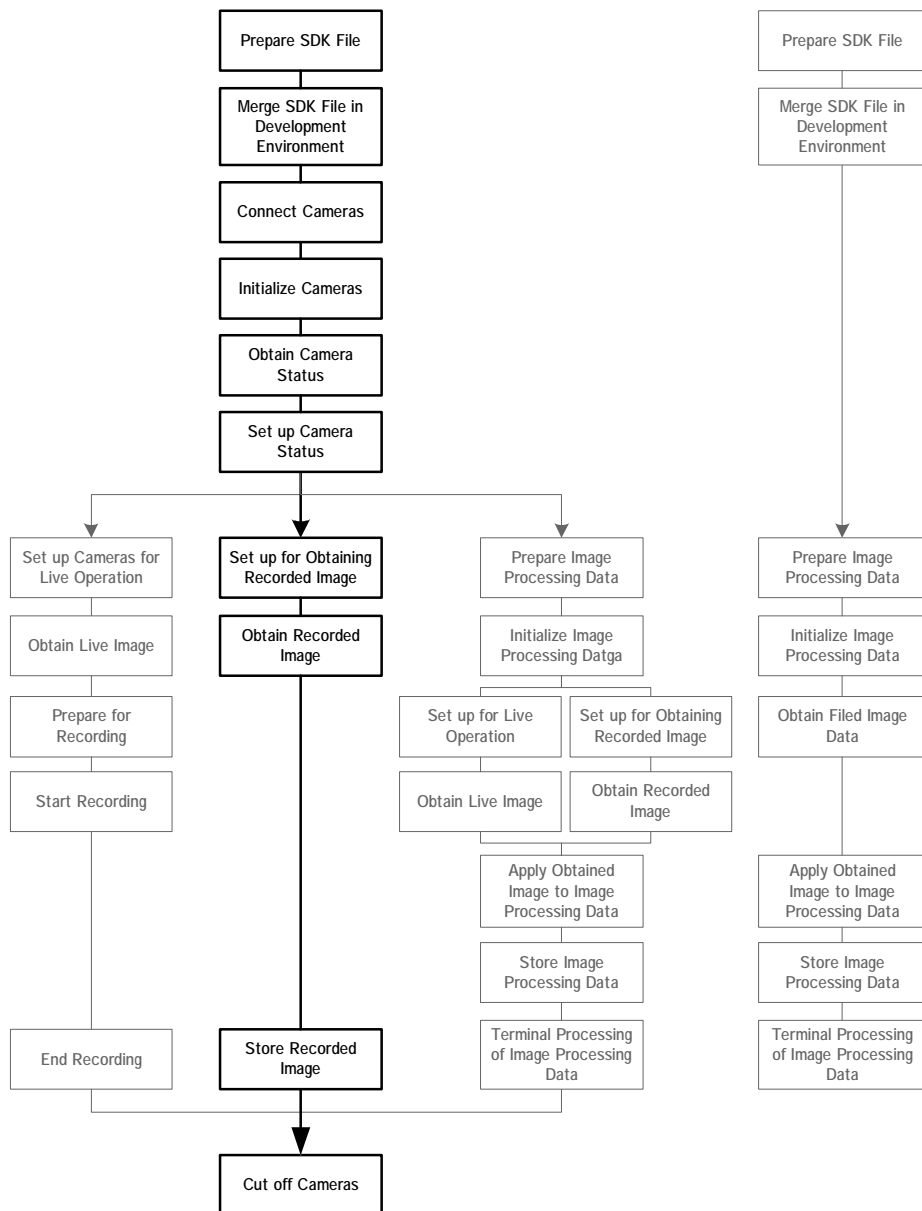
### **Section 2.3. Obtain and Store Recorded Image**

Recorded image is retained in the hardware memory of the camera. The FASTCAM Control SDK obtains any image frame from the camera memory.

As soon as the camera, or the PC for a PCI type camera, is switched off, the image recorded in the hardware memory is lost. So, if you wish to retain any recorded image for future use, the image must be obtained and stored in an appropriate manner.

This section describes how to obtain and store image that is recorded in the hardware memory of the camera.

See the previous sections for procedure for camera connection to start and end of recording.



Section 2.3. discusses how to obtain and store image data recorded by camera.

### **Subsection 2.3.1 Setup for Obtaining Recorded Image**

In order for the FASTCAM Control SDK to obtain recorded image from the camera memory, the camera must be reset from the live recording status to display (playback) status. Unless this change of status is made, any necessary image cannot be obtained from the camera memory. So be sure to change the camera status.

The following shows an example of program to reset the camera to the playback status:

```
// Set camera to playback status  
Ret = OnLive(0)
```

Use the OnLive function to reset the camera status. Setting [1] as the argument for the OnLive function makes the camera set in the live status, and [0] in the playback status.

```
// Decide if camera is in live status  
Ret = IsLive(check)
```

After changing the camera status by OnLive function, use the IsLive function to confirm if the camera has been set in the playback status.

See the reference manual for the reset values of functions.

### **Subsection 2.3.2 Obtain Recorded Image**

This subsection describes how to obtain recorded image of the specified frame number. Image is obtained by the frame. The camera manages the recorded image giving a frame number to each of the recorded frames. The FASTCAM Control SDK obtains recorded image from the camera memory by assigning frame numbers of the frames of interest.

As is the case with the obtained data from live image, the monochrome image data is binary with its origin in the upper left corner, and color image data is of interleaved format (RGBRGB.....) with its origin in the upper left corner.

The size of image data transferred from the memory depends on the image size and the type of the image, monochrome or color, when it was originally recorded in the memory from the camera. Image data should be stored in such a storage format that will allow other applications to view.

To obtain the image frame at the trigger point, set the first argument for the TransferFrame [1]. See the reference manual for the details of frame number management within the camera memory.

## **1) Obtain Camera Status at Recording**

This confirms the image status when the recorded image was transferred to memory.

Camera status at recording

```
// Obtain current resolution  
Ret = GetResolution(m_dwResolution)
```

Use the GetResolution function to confirm the resolution at recording to find out the image size when it was transferred to memory.

Confirmation of image size at recording must be done only after setting the camera into the playback status. The image size displayed in the live mode does not necessarily match that of recording, and the difference between the image size and the memory size prepared by the software for transfer of image data may result in a transfer error.

## **2) Obtain Recorded Image**

If the target memory area for the image to be transferred into is smaller than the image size, a memory access error may result. Be sure to check the image resolution information (see the previous description) to set the target memory size larger than the image to be obtained.

The following shows an example of program to transfer recorded image assuming the image size is known:

```
// Obtain recorded image  
Ret = TransferFrame(1, ImageDataBuf(0))
```

The first argument for the TransferFrame function denotes the frame number of the recorded image to be transferred. Usually, setting is so made by the CameraControl library that the first frame at the start of recording is numbered frame [1]. The frame numbers that can be used to assign frames for transfer vary by the trigger mode.

To obtain image frames of interest from the camera memory, set the corresponding frame numbers in the above program.

See the reference manual for the reset value for functions.

### Subsection 2.3.3 Store Recorded Image

As explained in [Section 2.3](#), Obtain and Store Recorded Image, the image recorded in the hardware memory is erased when the camera power, or the computer power for PCI type cameras, is switched off. So any image that you wish to retain for future use must be obtained and stored in the computer.

The following shows an example of program to store image (1280 x 1024 resolution) obtained from a camera in the DIB format:

```
// Obtain mono/color mode of connected camera
Ret = IsColor(color)

// Obtain recorded image
Ret = TransferFrame(1, ImageDataBuf(0))

// Decide result of IsColor function
If m_color = 1 Then{
    // To store image from color camera
    // Set up BMP file information
    'BITMAPFILEHEADER
    BFH.bfType = &H4D42S
    BFH.bfSize = 14 + 40 + (1280 * 1024) * 3
    BFH.bfOffBits = 14 + 40
    'BITMAPINFOHEADER
    BIH.biSize = 40
    BIH.biWidth = 1280
    BIH.biHeight = 1024
    BIH.biPlanes = 1
    BIH.biBitCount = 24
    BIH.biCompression = 0
    BIH.biSizeimage = 1280 * 1024 * 3

    Try
        fileNo = FreeFile()
        FileOpen(fileNo, "c:\temp.bmp", OpenMode.Binary)
        FilePut(fileNo, BFH)
        FilePut(fileNo, BIH)
        FilePut(fileNo, ImageDataBuf)
    Catch
        FileClose(fileNo)
    Finally
        FileClose(fileNo)
End TryElse // To store image from monochrome camera
    // Set up BMP file information
    'BITMAPFILEHEADER
    BFH.bfType = &H4D42S
    BFH.bfSize = 14 + 40 + 1024 + (1280 * 1024)
    BFH.bfOffBits = 14 + 40 + 1024
    'BITMAPINFOHEADER
    BIH.biSize = 40
    BIH.biWidth = 1280
    BIH.biHeight = 1024
    BIH.biPlanes = 1
    BIH.biBitCount = 8
    BIH.biCompression = 0
    BIH.biSizeimage = 1280 * 1024
```

```
Try
  fileNo = FreeFile()
  FileOpen(fileNo, "c:\fastcam\temp.bmp", OpenMode.Binary)
  FilePut(fileNo, BFH)
  FilePut(fileNo, BIH)
  'RGBQUAD( )
  For k = 0 To 255
    BitColor.rgbBlue = k
    BitColor.rgbGreen = k
    BitColor.rgbRed = k
    BitColor.rgbReserved = 0
    FilePut(fileNo, BitColor)
  Next k
  FilePut(fileNo, ImageDataBuf)
Catch
  FileClose(fileNo)
Finally
  FileClose(fileNo)
End Try
```

For details of the BITMAP file structure, see Platform SDK: Windows GDI in Microsoft Corporation MSDN Library.

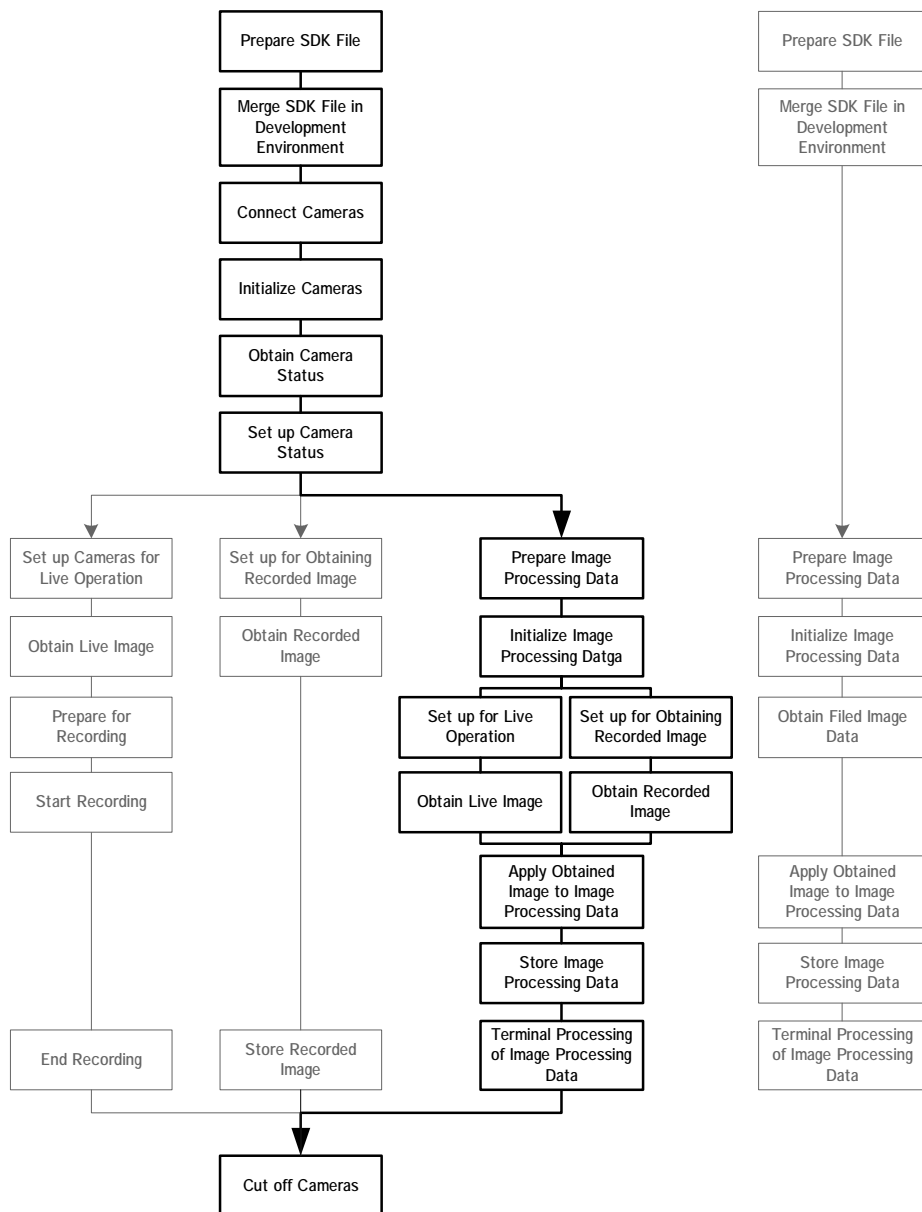
See the reference manual for the reset value of functions.

## **Section 2.4 Processing Image Obtained from Camer**

When it is necessary to manipulate image data, obtained from a camera, in such a way as correction (e.g. lookup table correction) or conversion to a Windows image, the image data library prepared in the FASTCAM Control SDK makes available quick processing and storage means of image data. This section describes how to manipulate and store image data obtained from a camera using the image processing class prepared by the FASTCAM Control SDK.

Process the image data in the following manner after obtaining image data from the camera following the previous relevant descriptions:





Section 2.4. discusses how to store image data obtained from camera, using image processing library prepared under SDK.

## **Subsection 2.4.1 Prepare Image Processing Data**

### **1) Construct Image Data**

By constructing an image data, you can process image data with the FASTCAM Control SDK.

The following shows an example of program to construct an image data for processing and storing image data:

```
// Construct Image Data  
Ret = ImageData()
```

One image data can process one image data set. To process multiple sets of image data, you need to construct sufficient number of image data to cover the sets of image data.

## Subsection 2.4.2 Initialize Image Processing Data

### 1) Secure Image Data Buffer

To introduce image data into the constructed image data, secure an image data buffer, of a size matching the image, within the image data library.

The following shows an example of program to initialize the image data library of that matches the size of the image data:

```
// Secure image data buffer matching the size of image data
Ret = SetImageNumber(1)

Ret = InitImageData(ImageDataSize)
```

The image data library has secured, at initialization, a certain memory area of the default size that is needed for subsequent processing of image.

You need to change the memory size for image processing to the memory size necessary for image processing. The image size is measured in bytes and is set up by the argument for the InitImageData function.

### 2) Set up Image Data Information

To set up image data to be processed by the image data library, the image data information must be set up in advance. You set up information regarding the image data to send over to the image data class

The following shows an example of program to set up image data information to send to the image data library:

```
Dim m_Bitmapinfo As BITMAPINFO // BITMAPINFO structure

Ret = SetBmpInfo(m_Bitmapinfo)
```

To the SetBitmapInfo function, assign a BITMAPINFO structure of the image data information with setup needed for image data library

Using the information on the BITMAPINFO structure assigned by the argument for the SetBitmapInfo function, the image data library carries out necessary processing of the image data. If you set up BITMAPINFO structure information irrelevant to the image data library, correct result will not be obtained.

See [Platform SDK: Windows GDI] of Microsoft Corporation MSDN Library for details of BITMAPINFO structure.

See the reference manual for the reset value for functions.

### **Subsection 2.4.3 Setup for Live Operation and Obtaining Recorded Image**

Set up the connected camera for live operation referring to [Subsection 2.2.7](#) Setup Camera for Live Operation.

Set up the connected camera for obtaining recorded image referring to [Subsection 2.3.1](#). Setup for Obtaining Recorded Image.

### **Subsection 2.4.4 Obtain Live Operation and Recorded Image**

Obtain live image from the connected camera referring to [Subsection 2.2.8](#). Obtain Live Image.

Obtain recorded image from the connected camera referring to [Subsection 2.3.2](#). Obtain Recorded Image.

### **Subsection 2.4.5 Apply Obtained Image to Image Processing Data**

This subsection describes how to send the obtained image data library, including relevant image information set up under previous descriptions, to the buffer of the image data library and to exercise various processes of the class on the image data.

If you send, to the buffer of the image data library, image data with different settings from the setup made in the previous subsection, correct result will not be obtained. You must always submit image data with correct image data information

The following shows an example of program to transfer live image data to the image data buffer:

```
// Obtain live image (set the first argument [-1] to request a transfer to the camera)
Ret = TransferFrame(-1, ImageDataBuf(0))

// Copy the live image, obtained from the camera, to the image data buffer
Ret = SetImageData(ImageDataBuf (0))
```

This transfers the obtained image over to the image data library. The transferred image data is retained until the image data library is released.

See the reference manual for the reset value of the functions.

### **Subsection 2.4.6 Store Image Processing Data**

Using the image data library to set up both image data and relevant image data information makes it easy to store the image data in a file format that is supported by the image data library.

The following shows an example of program for the image data library to store the image data in BMP file format using its file storing function.

```
// Store BMP file  
Ret = SaveBmpFile("C:\\temp.bmp ")
```

With the file name assigned by the argument of the SaveBmpFile function, the image data library stores the processed image data within the image data library in the specified file format. When no processing is given, the image data class stores the image data set up in the image buffer of the image data library in the specified file format.

See the reference manual for the reset value of the functions.

### **Subsection 2.4.7 Terminal Processing of Image Processing Data**

This subsection describes how to terminate the image data library in order to release the image data buffer, scaled to the image data, that has been prepared to receive the image shot by image data library.

As soon as the operation of the image data library is terminated, the image data releases the image buffer and returns to the pre-initialization status, which is retained within the FASTCAM Control SDK until the variable of the constructed image data is released.

```
// Release image data class  
Ret = ExitImageData()
```

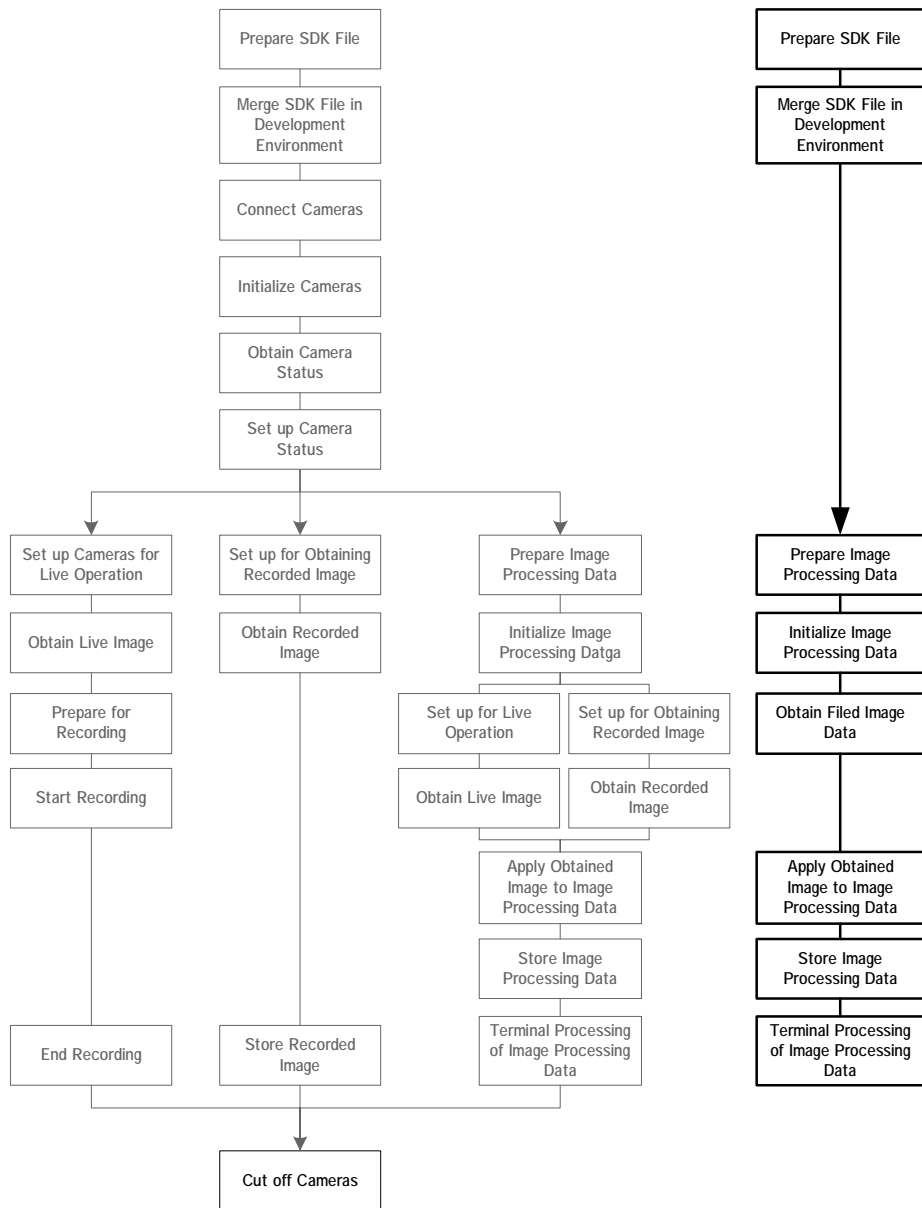
See the reference manual for the reset value of the functions.

## **Section 2.5 Read and Write Data Files**

The image data library has a function to store image data in Photron's special format. By reading a file, stored in the special format, with image data library, you can get the image data from the file. The image data library can also re-store the image data, read from a file of the special format, in another file format.

The following describes how to read, using the image data library, image data shot and stored by Image Data library:





This section discussed how to read and write stored image data using the image data library only.

### Subsection 2.5.1 Prepare Image Processing Data

Prepare image processing data following the instructions given in [Subsection 2.4.1](#).

### Subsection 2.5.2 Initialize Image Processing Data

Refer to the initialization procedure given in [Subsection 2.4.2](#).

### Subsection 2.5.3 Obtain Filed Image Data

Use the LoadFile function to obtain a file of image data.

The LoadFile function must be added with an absolute reference path to read files.

```
Ret = LoadFile("C:\\temp.bmp ")
```

The image data library tells the file format by the file extension of the argument of the files it reads. Be sure to add a file extension to the file name you wish to read.

```
// Decide result of LoadFile function
Select Case Ret// Decide file format type

    case FORMAT_NONE:
        // When file format is not supported by image data library
    case FORMAT_BMP:
        // When file format is BMP file
End Select
```

When a file of different size from the predetermined image data size, the image data library automatically scales the buffer size of the image data within the image data library.

Confirm the image data information that is just read.

```
// Confirm file information
Dim m_Bitmapinfo As BITMAPINFO // BITMAPINFO structure
Ret = GetBmpInfo(Bitmapinfo)
```

The image data information read by the image data library is obtained by the GetBmpInfo function.

For the detail of the BITMAPINFO structure, see [Platform SDK : Windows GDI] of Microsoft Corporation MSDN Library.

See the reference manual for the reset value of the functions.

#### **Subsection 2.5.4 Apply Obtained Image to Image Processing Data**

Follow the instructions given in [Subsection 2.4.5](#). Apply Obtained Image to Image Processing Data.

#### **Subsection 2.5.5 Store Image Processing Data**

Follow the instructions given in [Subsection 2.4.6](#). Store Image Processing Data.

#### **Subsection 2.5.6 Terminal Processing of Image Processing Data**

Follow the instructions given in [Subsection 2.4.7](#). Terminal Processing of Image Processing Data.

## Chapter 3 Other Operations

This chapter describes other camera control functions that are not explained in previous sections.

### Section 3.1 Connection to Multiple Cameras (application of Subsections 2.2.3. and others)

The FASTCAM Control SDK can control all of multiple cameras connected to the PC. See [Subsection 2.2.3.](#) Connect Camera Systems for instructions how to connect cameras to the PC. After multiple cameras are connected to the PC, the FASTCAM Control SDK recognized all those cameras, provided that the cameras are of one single model.

One camera control operates one camera. So, to operate multiple cameras, you need to construct the same number of camera control as the number of cameras to be controlled.

As discussed in Subsection 2.2.3., camera control classes for the cameras are constructed in the following manner:

```
// Construct camera control class (for camera 1)
Ret = CameraControl(DEVICE_SELECT_AUTO)

// Construct camera control class (for camera 2)
Ret = CameraControl(DEVICE_SELECT_AUTO)

// Construct camera control class (for camera 3)
Ret = CameraControl(DEVICE_SELECT_AUTO)
```

As described in [Subsection 2.2.4.](#), initialize the control libraries that have been prepared for each of the camera models and type of interface. After all the connected cameras are initialized, start operating the cameras.

The following shows an example of program to initialize cameras:

```
// Initialize FASTCAM Control SDK and control library for camera 1
Ret = SetCameraNumber(1)
Ret = Init(1)

// Initialize FASTCAM Control SDK and control library for camera 2
Ret = SetCameraNumber(2)
Ret = Init(2)

// Initialize FASTCAM Control SDK and control library for camera 3
Ret = SetCameraNumber(3)
Ret = Init(3)
```

Each of the above control libraries controls its relevant camera.

## Section 3.2 Access to Multiple Files (application of Section 3.2. and others)

The FASTCAM Control SDK can process multiple image data files. See also [Subsection 2.4.1](#). Prepare Image Processing Data for information regarding preparation of image data.

As is the case with the camera control, one image data controls one corresponding image data file. So, to process multiple image data file at a time, you need to prepare the same number of image data as the image data files.

The following shows an example of program to construct image data class for 3 image data files:

### 1) Construct Image Data

```
// Construct image data class (for image data file 1)
Ret = ImageData()

// Construct image data class (for image data file 2)
Ret = ImageData()

// Construct image data class (for image data file 3)
Ret = ImageData()
```

### 2) Secure Image Data Buffer

As described in [Subsection 2.4.2](#)., secure an image data buffer with a matching size with the image data within the image data library.

The following shows an example of program to initialize the image data library for monochrome image data of 1280 (W) x 1024 (H) resolution.

```
Dim ImageDataSize As Integer
ImageDataSize = 1280 * 1024:// Image size
Dim ret As Integer// Reset value of image data function

// Secure 1st image data buffer for image processing
Ret = SetImageNumber(1)
Ret = InitImageData(ImageDataSize)

// Secure 2nd image data buffer for image processing
Ret = SetImageNumber(2)
Ret = InitImageData(ImageDataSize)

// Secure 3rd image data buffer for image processing
Ret = SetImageNumber(3)
Ret = InitImageData(ImageDataSize)
```

### **3) Set up Image Data Information**

To process multiple image data files, you set up image data information on each image data library following the instructions as described in [Subsection 2.4.2](#). Initialize Image Processing Data, 2) Set up Image Data Information.

```
Dim m_Bitmapinfo As BITMAPINFO // BITMAPINFO structure

// Set up 1st image data information for image processing
Ret = SetImageNumber(1)
Ret = SetBmpInfo(m_Bitmapinfo)

// Set up 2nd image data information for image processing
Ret = SetImageNumber(2)
Ret = SetBmpInfo(m_Bitmapinfo)

// Set up 3rd image data information for image processing
Ret = SetImageNumber(3)
Ret = SetBmpInfo(m_Bitmapinfo)
```

As to details of the BITMAPINFO structure, see [Platform SDK: Windows GDI] in Microsoft Corporation MSDN Library.

See the reference manual for the reset value of the functions .o.

## Section 3.3 Trigger Setting and Recording Operation (application of Subsections 2.2.8 to 2.2.10)

The trigger is a type of signal to make a camera start recording. By selecting a mode of trigger, you can set the trigger status of a connected camera.

For details of recording operation by each mode of triggering, see the hardware reference manual of the camera you are using.

### Subsection 3.3.1 Start Trigger

This trigger mode carries out a recording of image with the start frame being the same as the base point frame at which a trigger is given to start a recording, and automatically stops recording as soon as the memory space available for recording has been filled.

The following shows an example of program to record image in the Start Trigger mode:

```
// Set trigger mode to [Start Trigger]
Ret = SetTriggerMode(TRIGGER_START)
```

TRIGGER\_START is the argument to set the recording condition of the camera to [Start Trigger]. By changing this argument, you can set the camera's triggering mode to a particular one of your choice.

```
// Obtain trigger mode
Ret = GetTriggerMode(m_nTrigMode)
```

After setting the trigger mode, you can verify if the trigger mode has been correctly set.

```
// Set up record ready status
Ret = OnRecordReady(1)

// Verify record ready status
Ret = IsRecordReady(check)

// Set up start of recording
Ret = OnRecord(1)

// Check recording status
Ret = IsRecord(check)
```

The above shows the process of setting up and verifying the record ready status, start recording and check the recording status in a recording operation in the Start Trigger mode. When the recording finishes, you can verify the ending by the IsRecord function.

See the reference manual for the reset value of functions.

### Subsection 3.3.2 Center Trigger

Use this trigger mode to record the same number of frames before and after the base point frame at which a trigger is given.

If a trigger is given after having recorded over half of the maximum number of available frames, the camera overwrites the previously recorded image and stops recording at the frame whose count corresponds to a half of the maximum number of available frames from the base point frame (where the trigger was given). In this case, you have the same number of frames recorded before and after the base point frame.

On the other hand, if a trigger is given before recording half of the maximum number of frames from the start of recording, the camera stops recording when it has recorded half of the maximum number of frames from the base point frame. In this case, however, the number of frames recorded before the base point frame is less than half of the maximum number of frames and the total number of recorded frames does not match the maximum number of frames available for recording.

So, when obtaining the recorded image data after shooting, be careful about the difference between the number of frames actually recorded and the maximum number of frames available for recording.

```
// Set trigger mode to [Center Trigger]
Ret = SetTriggerMode(TRIGGER_CENTER)
```

TRIGGER\_CENTER is the argument that sets the camera's shooting condition to the Center Trigger mode. By changing this argument, you can select a trigger mode of interest.

```
// Obtain trigger mode
Ret = GetTriggerMode(m_nTrigMode)
```

After changing the trigger mode, you can verify if the trigger mode of interest has been correctly set.

```
// Set up record ready status
Ret = OnRecordReady(1)

// Verify record ready status
Ret = IsRecordReady(check)

// Set up start recording
Ret = OnRecord(1)

// Verify loop recording status
Ret = IsEndlessRec(check)

//Enter start-recording trigger
Ret = TriggerIn()

// Enter set-base-point-frame (end-recorging) trigger
Ret = TriggerIn()

// Verify end of recording
Ret = IsRecord(check)
```

To record in the Center Trigger mode, first you set up record ready status and then start recording using the OnRecord function. The camera starts recording when it is given a start-recording trigger



and it continues to record in a loop-recording manner, repeatedly, (the status of "endless" recording is confirmed using the IsEndlessRec function).

The base point frame is set up by the TriggerIn function. Recording stops when half of the maximum number of frames available, after the base point frame, have been recorded. Completion of recording is confirmed by IsRecord function.

See the reference manual for the reset value of functions.

### Subsection 3.3.3 End Trigger

The End Trigger mode makes the base point frame, at which a trigger is given, the last frame of a recording.

The camera records, when [record ready] status is set up and recording is started by the OnRecord function, in a loop-recording manner, repeatedly, overwriting the previously recorded frames, until a trigger is given.

The frame, at which a trigger is given, is the base point frame of a recording and is named frame number [1]. The frame right before frame [1] is [-1] and all other frames are given a number in the backward direction (toward the first frame of recording) with a [- (minus)] symbol. Because the frame numbers are all negative, except for the base point frame, you must be careful when you obtain image frames before frame [1].

```
// Set trigger mode to [End Trigger]
Ret = SetTriggerMode(TRIGGER_END)
```

*TRIGGER\_END* is the argument to set the camera in the End Trigger mode. By changing this argument, you can set the camera to any recording mode.

```
// Obtain trigger mode
Ret = GetTriggerMode(m_nTrigMode)
```

After changing the trigger mode, verify if the trigger mode has been set correctly.

```
// Set up record ready status
Ret = OnRecordReady(1)

// Verify record ready status
Ret = IsRecordReady(check)

// Set up start recording
Ret = OnRecord(1)

// Verify endless recording status
Ret = IsEndlessRec(check)

//Enter start-recording trigger
Ret = TriggerIn()

// Enter set-base-point-frame (end-recording) trigger
Ret = TriggerIn()

// Verify end of recording
Ret = IsRecord(check)
```

To record in the End Trigger mode, first you set up the camera in the record ready status and then set start-recording using the OnRecord function. The camera starts recording, when it is given a start-recording trigger, and continues to record in a loop-recording manner, repeatedly, until a trigger is given. The status of "endless" recording is confirmed using the IsEndlessRec function. Recording stops at the moment a set-base-point-frame (end-recording) trigger is given. Completion of recording is confirmed by the IsRecord function.

### **Section 3.4 Partitioning (application of Subsection 2.2.6.)**

Some of FASTCAM series cameras have a function that divides the whole memory space into several partitions. With the memory divided, you can record several different recordings into partitioned spaces separately.

The FASTCAM Control SDK can divide the memory space of a camera that has partition capability. The divided memory spaces are managed by [partition number]. By assigning a partition number to each of recordings, the image data is recorded in separate partitions. With the partitioning function, you can make multiple recordings continually without having to store recorded image every time a recording ends.

### Subsection 3.4.1 Setting

To divide the memory space of the camera to be controlled, you must set up divisions of memory.

The following shows an example of program to divide the memory space of a camera:

```
// Obtain status to camera
Dim params_ex As CAMERA_PARAMS_EX // Extended camera parameter structure
Ret = GetCameraParamsEx(params_ex)
```

The information on the partition availability is contained in the extended camera parameter structure. By checking the exist\_partition parameter of the extended camera parameter structure, decide if partitioning is possible.

When exist\_partition is [1], the camera memory can be divided.

```
// Obtain camera partitioning information
Ret = GetPartitionInfo(m_nMaxPartition, m_nMaxPartitionBlock, m_nFramePerBlock)
```

If the connected camera supports memory partitioning, obtain a setup of number of available partitions by the GetPartitionInfo function.

The first argument obtained by the GetPartitionInfo function is the maximum number of available partitions and the second the maximum block size.

```
// Obtain information on camera partition block list
Dim list(PARTITION_MAX_NUMBER - 1) As Integer // Partition parameter list
```

```
Ret = GetPartitionBlockList(list(0), PARTITION_MAX_NUMBER)
```

Obtain and confirm the number of blocks within each partition of the connected camera by the GetPartitionBlockList function.

```
// Calculate (divide memory size by 2) the number of blocks of each partition
m_nPartitionParamList(0) = m_nMaxPartitionBlock / 2;
m_nPartitionParamList(1) = m_nMaxPartitionBlock / 2;
```

If the third argument obtained by the GetPartitionInfo function is [-1], it means the partition size cannot be specified by the number of blocks. The whole blocks must be evenly allocated to each partition. In this case, there should be no surplus of blocks that are allocated to partitions.

```
// Set up camera partition change
Ret = SetPartitionBlockList(list(0), 2)
```

Set up the changed partition information to the camera by the SetPartitionBlock Function. After partitions are set up, the camera should always show partition number [1].

When the third argument obtained by the GetPartitionInfo function is [-1], the number of partitions, which is the second argument for the SetPartitionBlock function, must have the priority of being processed first.

See the reference manual for the reset value of functions.

### Subsection 3.4.2 Setups (Blocks)

With cameras of the PCI type, the memory space can be divided by the unit of Block, and if the memory can be divided by the block, the memory size of each partition can freely be changed.

```
// Obtain status to camera
Dim params_ex As CAMERA_PARAMS_EX // Extended camera parameter structure
Ret = GetCameraParamsEx(params_ex)
```

The information on the partition availability is contained in the extended camera parameter structure. By checking the exist\_partition parameter of the extended camera parameter structure, decide if partitioning is possible.

When exist\_partition is [1], the camera memory can be divided.

```
// Obtain camera partitioning information
Ret = GetPartitionInfo(m_nMaxPartition, m_nMaxPartitionBlock, m_nFramePerBlock)
```

If the connected camera supports memory partitioning, obtain a setup of number of available partitions by GetPartitionInfo function.

The first argument obtained by the GetPartitionInfo function is the maximum number of available partitions and the second the maximum block size.

As long as the description in this subsection is effective, the third argument obtained by the GetPartitionInfo function should always a number other than [-1].

```
// Obtain information on list of camera partition blocks
Dim list(PARTITION_MAX_NUMBER - 1) As Integer // Partition parameter list

Ret = GetPartitionBlockList(list(0), PARTITION_MAX_NUMBER)
```

Obtain and confirm the number of blocks in each partition of the camera by the GetPartitionBlockList function.

```
// Calculate the number of blocks of each partition (any size is specified to any partition)
m_nPartitionParamList(0) = 10;
m_nPartitionParamList(1) = 2;
```

As to the total number of blocks to be set on all partitions, the second argument obtained by the GetPartitionInfo function indicates the maximum number.

```
// Set up change of camera partitions
Ret = SetPartitionBlockList(list(0), 2)
```

A change of partition information is set up in the camera by the SetPartitionBlock function. Right after partitions have been set up in a camera, the camera should show partition number [1] first.

See the reference manual for the reset value of functions.

### **Subsection 3.4.3 Switching**

When you record image or obtain recorded image in any of the partitions, you should set up the camera to allow for independent operation on each partition.

The following shows an example of program to change partitioning of divided memory:

```
// Obtain current partition number  
Ret = GetPartition(m_nPartition)
```

You confirm if change of partitions is possible and obtain the current partition number.

```
// Change partition position to be recorded in camera  
Ret = SetPartition(2)// Set up partition number 2
```

After partition numbers are changed by the SetPartition function, the camera will record in a partition specified by the new number.

As to the argument for the SetPartition function, the maximum number, which was used to set the number of partitions in [Subsection 3.4.1.](#), shows the number that can be set as partition number.

See the reference manual for the reset value of functions.

## Chapter 4 Procedure for Camera Connection to Recording

### Section 4.1 Examples of VisualBaisc6.0/VisualBaisc.NET Programs

An example of procedure of recording and obtaining recorded image using Camera Control Class is shown below:

(1) Declaration of Class  
Ret = CameraControl(DEVICE\_SELECT\_AUTO)

(2) Initialization of Class  
To initialize camera device.

Ret = SetCameraNumber(1)  
Ret = Init(1)

(3) Change Camera to LIVE Status  
Ret = OnLive(1)

(4) Set up Shooting Conditions  
To obtain camera parameters  
Ret = GetCameraParams(params)

To set up recording rate  
Ret = SetRecordRate(m\_nRecordRate)

To re-obtain parameters due to change of list of resolution and shutter speed  
Ret = GetCameraParams(params)

To set up shutter speed  
Ret = SetShutterSpeed (nShutterSpeed)

To set up resolution  
Ret = SetResolution(m\_dwResolution)

To set up trigger mode  
Ret = SetTriggerMode(TRIGGER\_START)

(5) Start Recording  
To set record ready status on  
Ret = OnRecordReady(1)

To start recording  
Ret = OnRecord(1)

To confirm end of recording  
Ret = IsRecord(check)

(6) Change Camera to PLAY status  
Ret = OnLive(0)

(7) Download first frame of image  
Ret = TransferFrame(1, ImageDataBuf(0))

(8) End of Class  
Ret = ExitCameraControl()

*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](mailto: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](mailto: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](mailto:image@photron.com)

**FASTCAM Control SDK Library "PccLib" Programming Manual**

English Edition

January, 2003

Rev. 1.00