PCIe-CPL64

PCI Express x4
Dual Base / Single Medium Configuration
Camera Link Frame Grabber

User's Manual

Manual Rev. 2.01

Revision Date: January 28, 2009

Part No: 50-11158-1010

Recycled Paper

Advance Technologies; Automate the World.

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

NuDAQ, NuIPC, DAQBench are registered trademarks of ADLINK TECHNOLOGY INC.

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Getting Service from ADLINK

Contact us should you require any service or assistance.

ADLINK Technology Inc.
Address: 9F, No.166 Jian Yi Road, Chungho City,
Taipei County 235, Taiwan
台北縣中和市建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology Inc.
Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室
(100085)
Rm. 801, Power Creative E, No. 1, B/D
Shang Di East Rd., Beijing 100085, China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8625
Email: market@adlinktech.com

ADLINK Technology Shanghai
Address: 上海市漕河泾高科技开发区钦江路 333 号 39 幢 4 层
(200233)
Tel: +86-21-6495-5210
Fax: +86-21-5450-0414
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园
A1 栋 2 楼 C 区 (518057)
2F, C Block, Bld. A1, Cyber-Tech Zone,
Gao Xin Ave. Sec 7, High-Tech Industrial Park S.,
Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

ADLINK Technology Inc. (German Liaison Office)
Address: Nord Carree 3, 40477 Duesseldorf, Germany
Tel: +49-211-495-5552
Fax: +49-211-495-5557
Email: emea@adlinktech.com

ADLINK (French Liaison Office)
Address: 15 rue Emile Baudot, 91300 MASSY Cedex, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 151-0072 東京都渋谷区幡ヶ谷
1-1-2 朝日生命幡ヶ谷ビル 8F
Asahiseimei Hatagaya Bldg. 8F
1-1-2 Hatagaya, Shibuya-ku, Tokyo 151-0072, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5333-6040
Email: japan@adlinktech.com

ADLINK Technology Inc. (Korean Liaison Office)
Address: 서울시 강남구 논현동 60-12 동성빌딩 4 층 402 호
No.402, Dongsung B/D, 60-12, Nonhyeon-Dong
Gangnam-gu, Seoul, 135-010, Korea.
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre,
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte Ltd. (Indian Liaison Office)
Address: No. 1357, "Anupama", Sri Aurobindo Marg, 9th Cross,
JP Nagar Phase I, Bangalore - 560078, India
Tel: +91-80-65605817
Fax: +91-80-22443548
Email: india@adlinktech.com

Table of Contents

List of Tables

List of Figures

1 Introduction

ADLINK's PCIe-CPL64 is a Camera Link frame grabber that is based on the PCI Express x4 interface, and supports two-channel Camera Link "base" or one-channel "medium" configurations, multi-tap area and line scan cameras.

The PCIe-CPL64 frame grabber strikes a perfect balance between performance and cost. It is capable of simultaneous image acquisition from two completely independent Camera Link base configuration cameras, and supports image transfers rates up to 255 MB/s per channel (dual base mode), or 320 MB/s (single medium mode).

Camera Link is an industrial high-speed serial data and cabling standard. Created for easy connectivity between the PC and the camera.

PCI-express (or PCIe) is the latest standard serial interconnect architecture, offering high performance on standard PCs. The PCIe-CPL64 combines Camera Link acquisition capabilities and the high-performance PCI Express bus interface, and provides the high bandwidth and high transfer speed required by advanced machine vision applications such as line scan and 3D inspection.

1.1 Features

- Support two-channel "base" or one-channel "medium" Camera Link configuration
- PCI Express x4 interface
- Acquisition pixel clock rates up to 85 MHz
- Supports PoCL (Power Over Camera Link)
- 128 MB of 200 MHz DDR SDRAM for acquisition
- 4 TTL digital input/output, and 2 trigger input
- Supports Windows XP/Vista
- Supports 64-bits memory addressing
- Supports VC++ 6.0, BCB 6.0, VC++.NET
- Two serial communication ports

2 Hardware Reference

2.1 PCIe-CLP64 Specifications

Video Input

- Camera Link LVDS deferential signals
- Single medium or dual base configurations: Using two MDR26 pins connectors
- Maximum camera link data rate: 85 MHz
- Supports PoCL and standard Camera Link interfaces with auto detection

Camera Control

- LVDS camera control : CC1-CC4 control signal in two MDR26 pins connectors

External Signal Input

- External RS422 level A, B, Z phase deferential signals for encoder input, 1 MHz maximum input frequency
- Two-channel TTL level Line /Area trigger input
- Two-channel TTL level Line trigger start input
- Two-channel TTL level Exposure output
- Line trigger bypass output (encoder mode only)
- Four channels of digital input and four channels of digital output

Power over Camera Link (PoCL)

- Power line output per channel: DC +12 V ,Max. 1 A
- Over-current Protection function, auto detect non-PoCL cable or PoCL camera connected.

Form Factor

- PCI Express x4 interface

Dimension

- W x L: 167.65 mm x 111.15 mm

Operating Environment

- Temperature: 0 to 50°C
- Humidity: 5 to 90% RHNC

Storage Environment

- Temperature: 0 to 70°C
- Humidity: 0 to 95% RHNC

Power Requirements

- +12 V max 0.5 A, +3.3 V max 1.6 A

2.2 PCIe-CPL64 Outline

2.2.1 PCIe-CPL64 Connectors & Pin Definitions

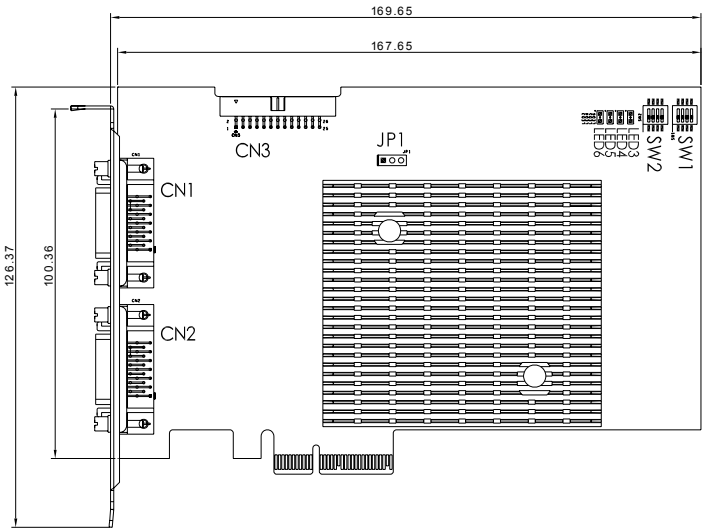


Figure 2-1: PCIe-CPL64 Layout

2.2.2 CN1 and CN2 Video Inputs

Dual Base Mode

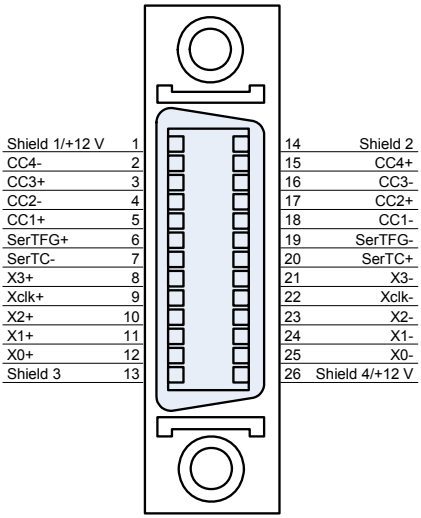


Figure 2-2: CN1 and CN2 Video Inputs (Dual Base Mode)

Medium Mode

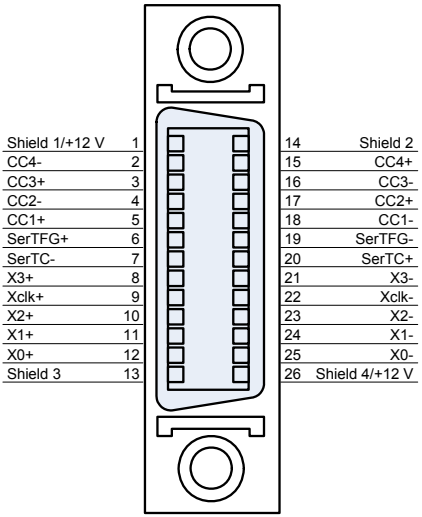


Figure 2-3: CN1 Video Inputs (Medium Mode)

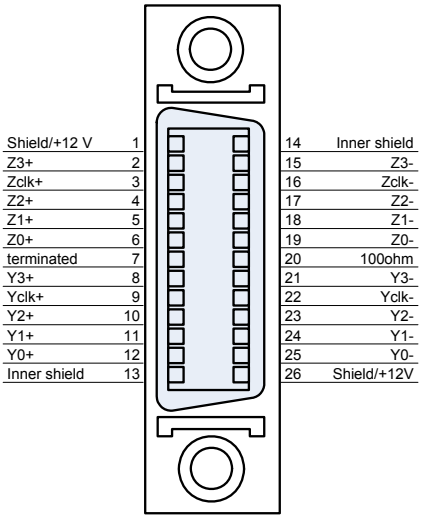
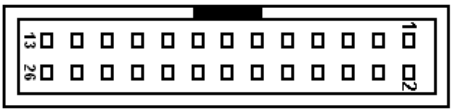


Figure 2-4: CN2 Video Inputs (Medium Mode)

2.2.3 CN1/CN2 Camera Link Connector Pin Assignment

	Dual Base Configuration Mode	Signal Medium Configuration Mode
CN1	Channel 0	Control Data 1
CN2	Channel 1	Data 2

2.2.4 CN3 Encoder and GPIO



	PIN NAME	TYPE	PIN	PIN NAME	TYPE
1	Encoder A phase A+	IN RS422	2	Encoder A phase A-	IN RS422
3	Encoder B phase B+	IN RS422	4	Encoder B phase B-	IN RS422
5	Encoder Z phase Z+	IN RS422	6	Encoder Z phase Z-	IN RS422
7	GND	--	8	GND	
9	Channel 0 Line/Area Trigger in	IN TTL	10	Channel 1 Line/Area Trigger in	IN TTL
11	Channel 0 Line Trigger start Logic High enable	IN TTL	12	Channel 1 Line Trigger start Logic High enable	IN TTL
13	Line trigger bypass output	OUT TTL	14	GND	--
15	Channel 0 Exposure Out	OUT TTL	16	Channel 1 Exposure Out	OUT TTL
17	DI channel 0	IN TTL	18	DO channel 0	OUT TTL
19	DI channel 1	IN TTL	20	DO channel 1	OUT TTL
21	DI channel 2	IN TTL	22	DO channel 2	OUT TTL
23	DI channel 3	IN TTL	24	DO channel 3	OUT TTL
25	GND	--	26	+5V Output	CN3 Encoder and GPIO

Table 2-1: CN3 Encoder and GPIO

2.2.5 JP1 Z-Phase Input Level Setting

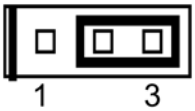
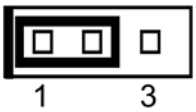
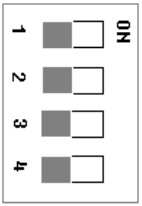
JP1	Setting	Function
	Pin 2-3 Short/Closed	Z phase input TTL level
	Pin 1-2 Short/Closed	Z phase input RS422 level (Default)

Table 2-2: JP1 Z-Phase Input Level Setting

2.2.6 SW1 Card ID Select



Pin no	Signal Name	Default
1	S1	ON
2	S2	ON
3	S3	ON
4	S4	ON

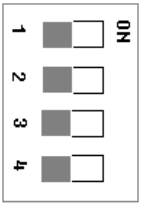
Table 2-3: SW1 Card ID Select

Card ID Select Table

Card ID	S1	S2	S3	S4
0	ON	ON	ON	ON
1	ON	ON	ON	OFF
2	ON	ON	OFF	ON
3	ON	ON	OFF	OFF

Table 2-4: Card ID Select Table

2.2.7 SW2 DO Pull +5 V Select



Pin no	Signal Name	Default
1	DO_3 pull +5V	ON(+5V)
2	DO_2 pull +5V	ON(+5V)
3	DO_1 pull +5V	ON(+5V)
4	DO_0 pull +5V	ON(+5V)

Table 2-5: SW2 DO Pull +5 V Select

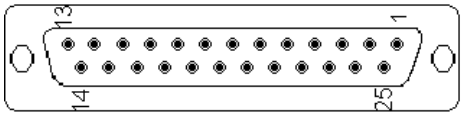
2.2.8 Status LED

LED No.	Function	Status
LED 3	Channel A PoCL +12 ON	**ON**: PoCL +12 turn on **OFF**: PoCL +12 turn off
LED 4	Channel A grab image	**Blinking**: Grabbing image **OFF** :Stop grabber
LED 5	Channel B PoCL +12 ON	**ON**: PoCL +12 turn on **OFF**: PoCL +12 turn off
LED 6	Channel B grab image	**Blinking**: Grabbing image **OFF** :Stop grabber

Table 2-6: Status LED

2.2.9 Encoder & GPIO Extension Cable

Extension cable connect CN3



Extension cable connector is 25 pin D-SUB female connector

	PIN NAME	TYPE	PIN	PIN NAME	TYPE
1	Encoder A phase A+	IN RS422	2	Encoder A phase A-	IN RS422
3	Encoder B phase B+	IN RS422	4	Encoder B phase B-	IN RS422
5	Encoder Z phase Z+	IN RS422	6	Encoder Z phase Z-	IN RS422
7	GND	--	8	GND	--
9	Channel 0 Line/Area Trigger in	IN TTL	10	Channel 1 Line/Area Trigger in	IN TTL
11	Channel 0 Line Trigger start Logic High enable	IN TTL	12	Channel 1 Line Trigger start Logic High enable	IN TTL
13	Line trigger bypass output	OUT TTL	14	GND	
15	Channel 0 Exposure Out	OUT TTL	16	Channel 1 Exposure Out	OUT TTL
17	DI channel 0	IN TTL	18	DO channel 0	OUT TTL
19	DI channel 1	IN TTL	20	DO channel 1	OUT TTL
21	DI channel 2	IN TTL	22	DO channel 2	OUT TTL
23	DI channel 3	IN TTL	24	DO channel 3	OUT TTL
25	GND	--			

Table 2-7: CN3 Encoder and GPIO

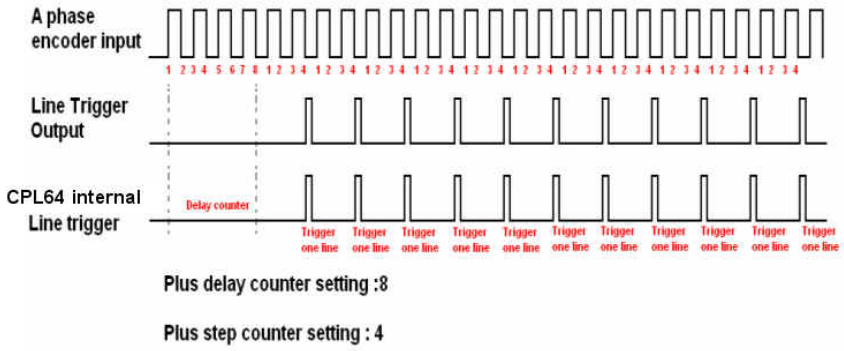
2.3 Trigger Modes

PCIe-CPL64 supported trigger mode list

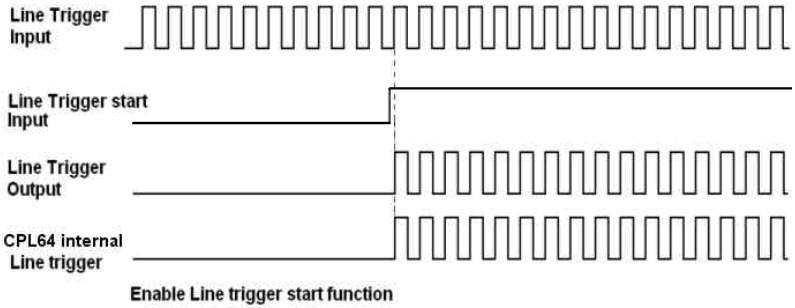
Trigger Mode	Input Signal RS-422 Level	Input Signal TTL Level	Line Trigger Start	Line Trigger Out	Encoder Counter	Line Area Counter	Plus Delay Counter	Plus Step Counter
A phase	Yes	No	No	Yes	Yes	Yes	No	Yes
A,B phase	Yes	No	No	Yes	Yes	Yes	No	Yes
A,B,Z phase	Yes	No	No	Yes	Yes	Yes	Yes	Yes
TTL Line/ Area Trigger	Yes	Yes	Yes	Yes	No	Yes	No	No

- Line trigger start (CN3 pin11/12): If enabling this function, you can use line trigger start signal to control line trigger
- Line trigger out (CN3 pin13) : Bypass CPL64 internal Line trigger output, when use TTL Line trigger mode, it bypass channel 0 Line trigger.
- Encoder counter: Count encoder input plus number
- Line counter: Count line trigger input plus number
- Plus Delay counter: Set encoder delay number
- Plus Step counter: Set encoder step number

Example Timing 1: A phase trigger mode



Example Timing 2: Line trigger mode



3 Installation Guide

3.1 Hardware Installation

PCIe-CPL64 series

Use the following steps to install the PCIe-CPL64 series board on the PCI express bus:

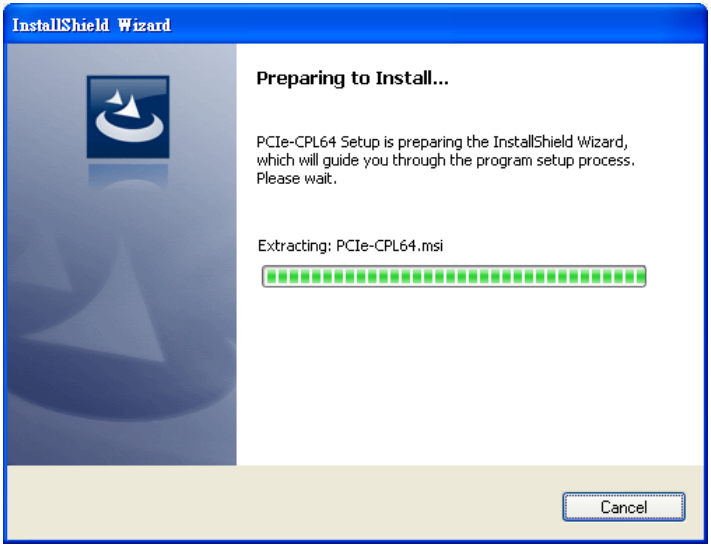
1. Remove the computer cover using the instructions from the computer manual.
2. Check that there is an empty PCI express slot to accommodate the board. If there is not an empty slot, remove a PCI express board from the computer to make room for the PCIe-CPL64 board and take note of the chosen slot number.
3. Remove the blank metal plate located at the back of the selected slot (if any). Keep the removed screw to fasten the PCIe-CPL64 board after installation.
4. Carefully position the PCIe-CPL64 in the selected PCI express slot as illustrated below. If using a tower computer, orient the board to suit the board slots.
5. Once perfectly aligned with an empty slot, press the board firmly but carefully into the connector.
6. Anchor the board by replacing the screw.

Note: The PCIe-CPL64 can install at PCI Express x4, x8, x16 slot. Some PCI Express x16 slots on certain motherboards may only support a VGA card and if the PCIe-CPL64 is installed in such slot, the speed will decrease to PCI Express x1 speeds.

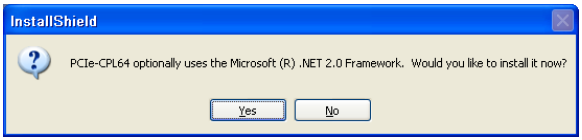
3.2 Driver Installation

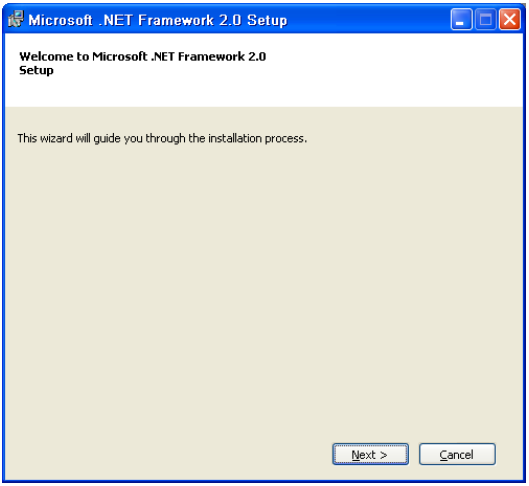
3.2.1 WDM Driver Installation

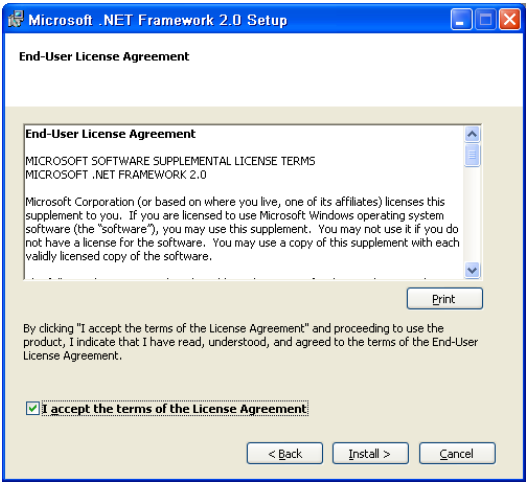
1. Click setup.
2. The driver will begin installing.



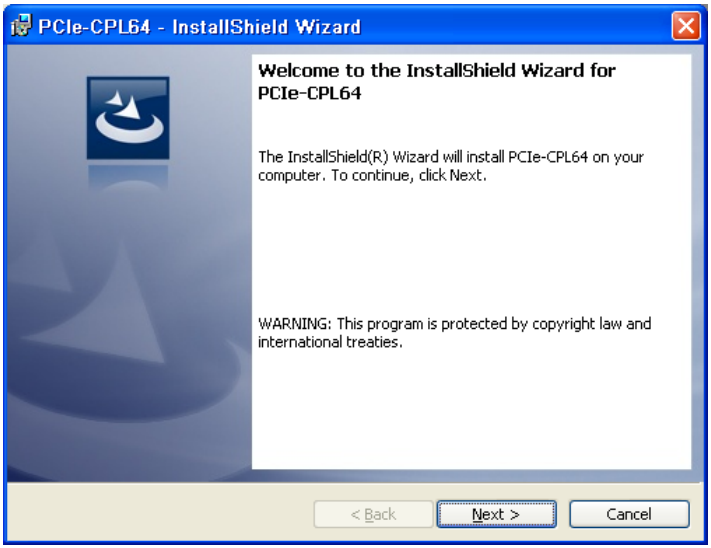
3. If there .Net 2.0 Framework is not currently installed on the system, the following window will be displayed. Click yes to install .Net 2.0 Framework.

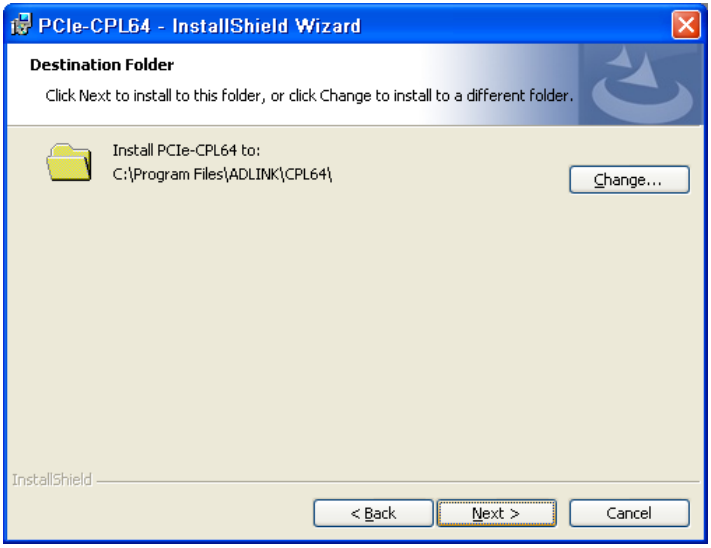


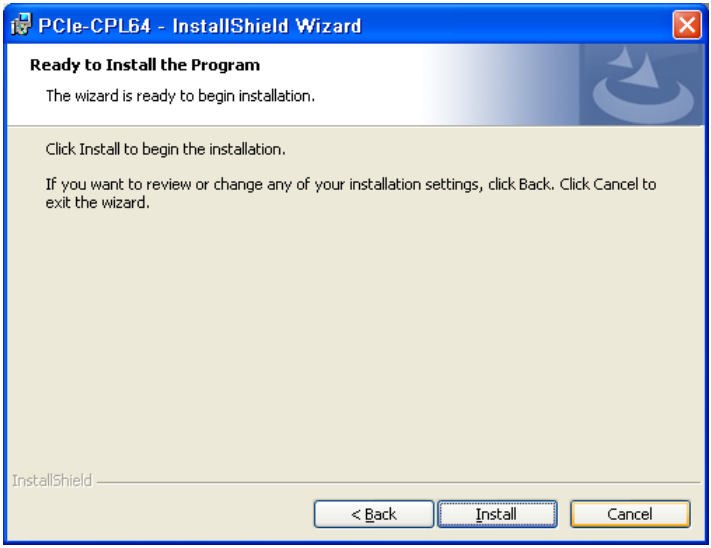




4. Click next until driver install completely.

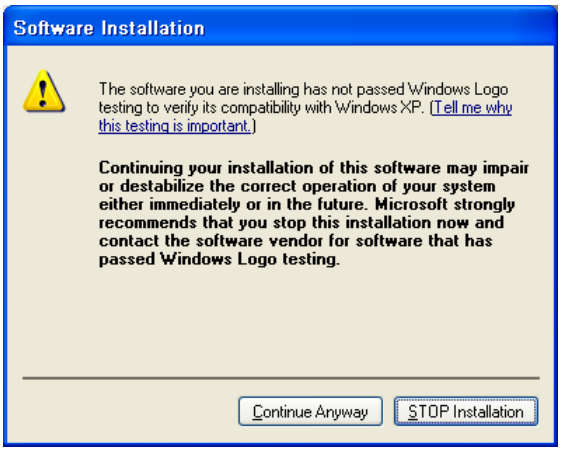




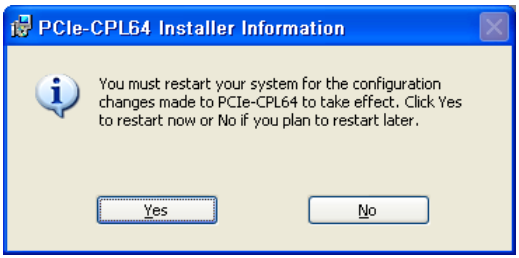

PCIe-CPL64 - InstallShield Wizard
Ready to Install the Program
The wizard is ready to begin installation.
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
InstallShield
< Back
Install
Cancel

5. When the following window is displayed, please press “Continue Anyway” to install device drivers.

Note: If a “Found New Hardware Wizard” window displays, simply ignore. After the device driver installation completes, the “Found New Hardware Wizard” window will automatically close.

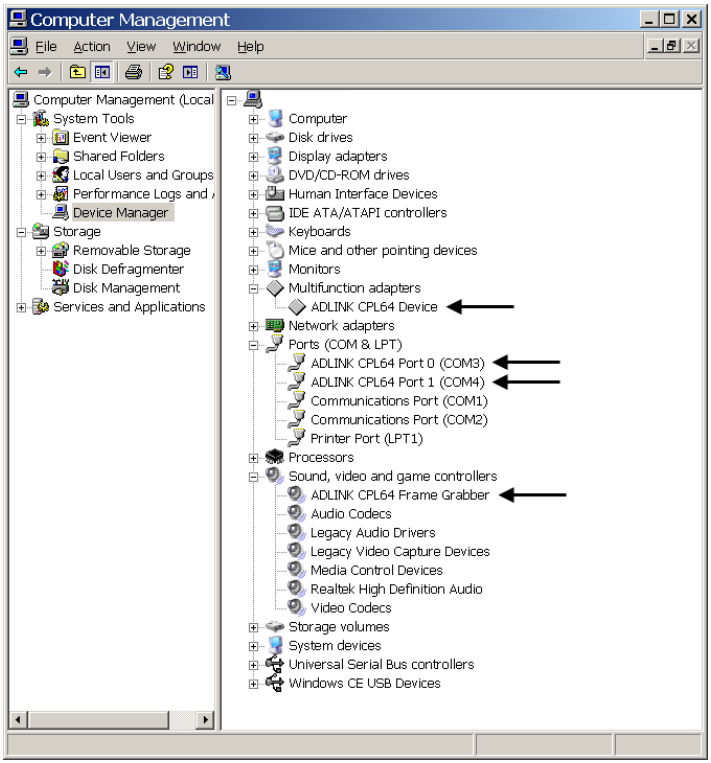


6. Click yes and restart system.

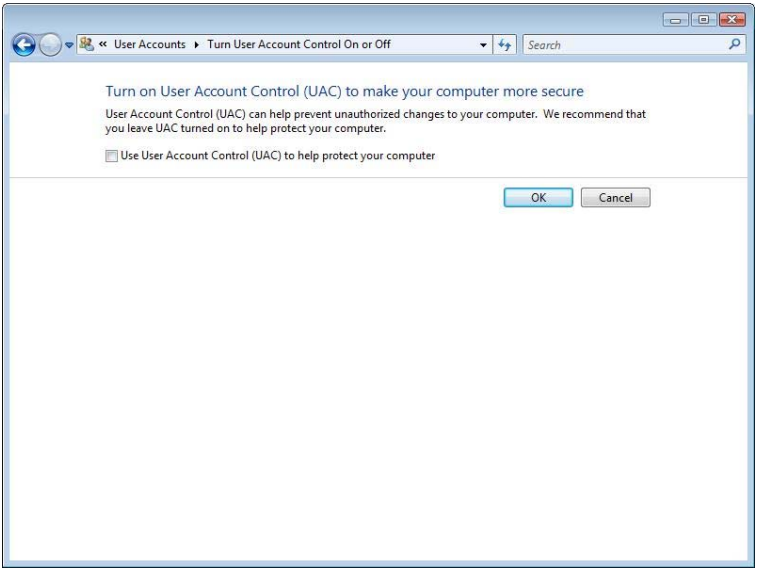


7. Go to system control panel and check for these four items:
   - ▷ ADLINK CPL64 Device
   - ▷ ADLINK CPL64 Frame Grabber
   - ▷ ADLINK CPL64 Port 0
   - ▷ ADLINK CPL64 Port 1

The system control panel should be as shown below:



Note: When using Windows Vista, the User Account Control (UAC) feature must be disabled before using the PCIe-CPL64. Click [Start] - [Settings] - [Control Panel] - [User Accounts] - [Turn User Account Control on or off]. Uncheck “Use User Account Control (UAC) to help protect your computer” for the PCIe-CPL64 to function normally under Windows Vista.



4 CamCreator Utility

Once hardware installation is complete, ensure that they are configured correctly before running the CamCreator utility. This chapter outlines how to establish a vision system and how to manually controlling PCIe-CPL64 cards to verify correct operation. CamCreator provides a simple yet powerful means to setup, configure, test, and debug the system from an easy-to-use point and click interface.

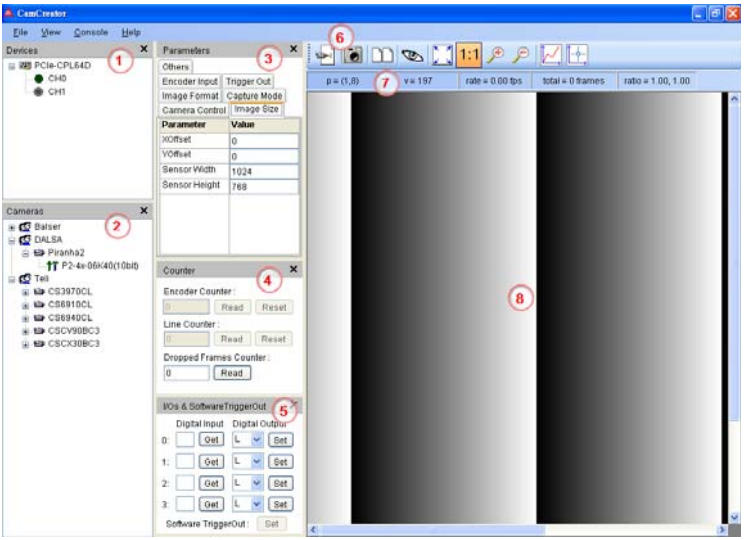
4.1 Overview

CamCreator offers the following features:

1. Automatic detection of acquisition hardware
2. Creation and modification of camera file
3. Instant modification of camera parameters
4. Serial communication
5. Opening and saving operation of still image file
6. Direct access to general purpose I/Os
7. Captured Image viewing

4.2 Component Description

Start the utility and the view should like below:



1. **Devices panel**

   The list window lists the PCIe-CPL64 cards at the local computer.

2. **Cameras panel**

   The tree browser window lists all available cameras and their camera files.

3. **Parameters panel**

   The tab window lists all adjustable parameters.

4. **Counter panel**

   A window for displaying encoder counter, line counter, and dropped frames counter.

5. **I/Os & Software Trigger panel**

   A window for reading and writing general purpose I/Os and for setting software triggerout.

6. **Toolbar**

   The toolbar simplify user's operation.

7. **Status bar**

   The status bar shows the information about captured image.

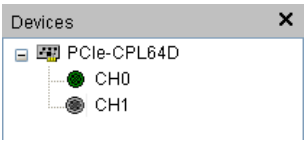
8. **Display panel**

   This window shows captured image.

4.3 Operation Theory

CamCreator provides many functions for the PCIe-CPL64 card as described below:

4.3.1 Devices Panel



Device

Current active channel

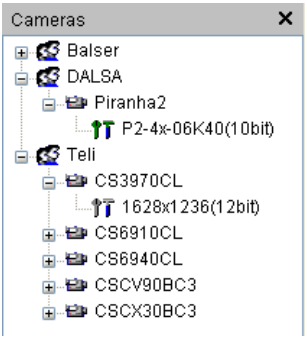
▷ All operations will apply to this channel.

Inactive channel

▷ Click the channel after this icon to activate this channel. Only one channel can be activated at one time. If you select standby channel, the current active channel will become inactive.

Close this panel

4.3.2 Cameras Panel



Camera provider

Camera type

Camera file

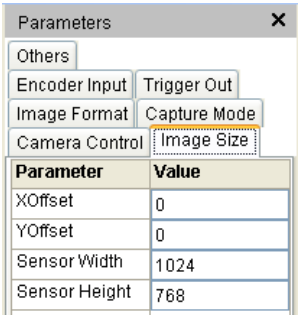
Current loaded camera file

- ▷ Select an appropriate camera file according to your camera. CamCreator will load this camera file to the device and shows its parameters in the parameters panel.

Close this panel

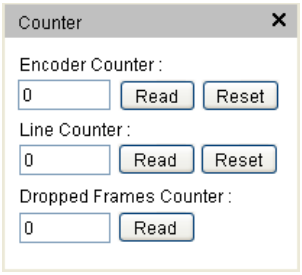
4.3.3 Parameters Panel

Parameters panel shows the parameters of camera file and other adjustable parameters what are aggregated on “Others” tab. Any modification instant applies to the current active device. Users can save the camera file with the modification parameters by “Save Cam File” command in main menu.



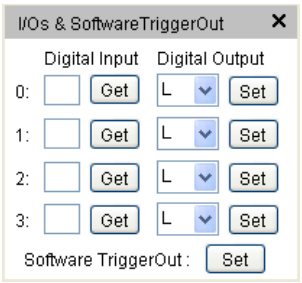
- **Close this panel**

4.3.4 Counter Panel



- **Close this panel**

4.3.5 I/Os and Software Trigger Panel



- **Close this panel**

4.3.6 Toolbar

Continuous Grab

▷ Start to grab images and display the images on display panel. Click it again to stop the grab. This is a toggle button.

Snap Shot

▷ Capture an image and display the image on display panel.

Dual Channel

▷ Show dual channel image on display panel. Click it again to show one channel image only.

Hide Image

▷ Hide or unhide displaying image. This is a toggle button.

Fit Size

▷ Fit the image to whole display panel.

Original Size

▷ Restore to original size.

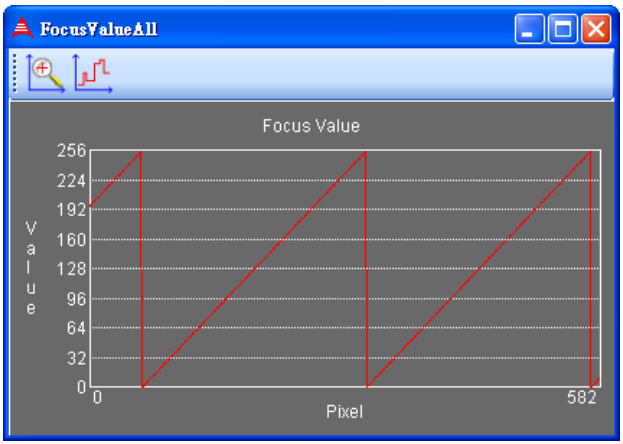
Zoom In

Zoom Out

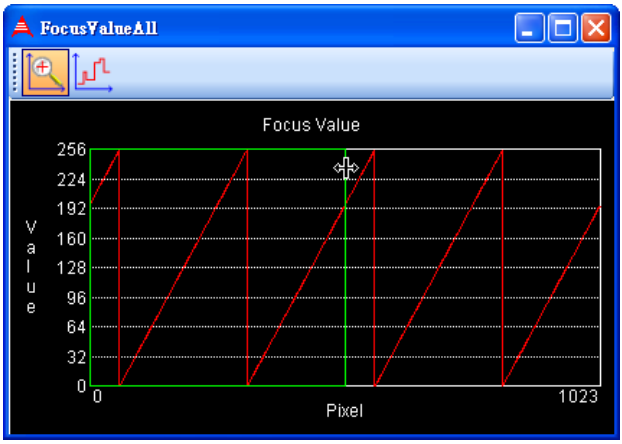
Focus Value

▷ Open a chart to see pixel values of the selected horizontal line of the image. The display image shows a red horizontal line on it. Click mouse on the display image to move the selected line.

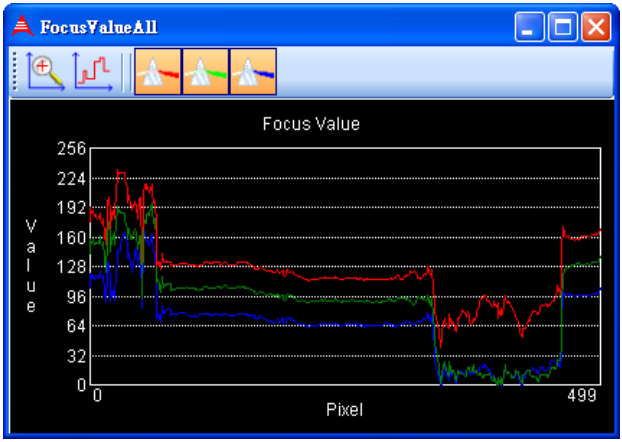
If it is grabbing images, the background color of focus value window is gray. The chart will update immediately by acquired image and the x-axis region depends on which horizontal pixels shown in display panel. The window is shown below:



After stopping grabbing, the background color of focus value window is black. The x-axis size is the width of the whole image. The window is shown below:



If the image is chromatic, there are three curves represented red, green, and blue individual in the chart. The window is shown below:



 Zoom In

▷ Open a window to zoom in the green rectangle region.

 Differential

▷ Open a window to show the slop of the line for the green rectangle region.

Drag the vertical green line to resize the green rectangle.

 Show/Hide Red Values

▷ Show or hide the red value of the pixels.

 Show/Hide Green Values

▷ Show or hide the green value of the pixels.

 Show/Hide Blue Values

▷ Show or hide the blue value of the pixels.

 Focus Cross

▷ See pixel values of the selected point of the image on toolbar. The display image shows a blue cross line on it. Click mouse on the display image to move the selected point.

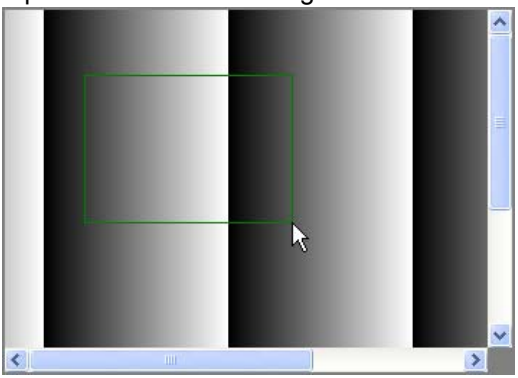
4.3.7 Status Bar

p = (1,8)	v = 197	rate = 0.00 fps	total = 0 frames	ratio = 1.00, 1.00

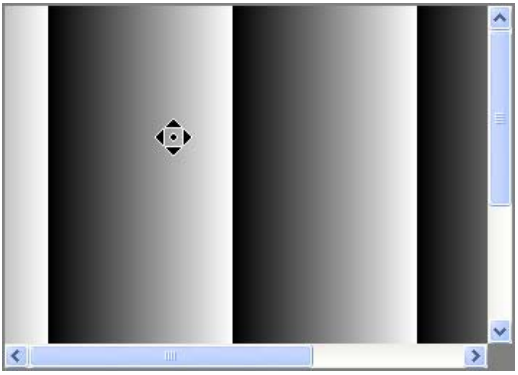
From left to right, the panel items are cursor position, pixel value, frame rate, total captured frames, and magnification (horizontal ratio, vertical ratio).

4.3.8 Display Panel

Click the left mouse button and then drag it, and the display panel will appear a green rectangle region which will be zoomed in. Keep pressing Ctrl during dragging, the image will be zoomed in at the same proportion of width and height. Shown below:



Click the right mouse button, the cursor will become a move2D icon. Then user can drag the image. Shown below:



4.3.9 Main menu

File Menu

Open Cam File

- ▷ Load camera parameters from a camera file.

Save Cam File

- ▷ Save current camera parameters to the current loaded camera file.

Save Cam File As

- ▷ Save current camera parameters to a new camera file.

Delete Cam File

- ▷ Delete the current loaded camera file.

Open Image

- ▷ Open an image from a file and display it to the display panel.

Save Image

- ▷ Save current displaying image to a bitmap file.

Exit

- ▷ Terminate CamCreator.

View menu

Devices

- ▷ Hide or unhide Devices panel.

Cameras

- ▷ Hide or unhide Cameras panel.

Parameters

- ▷ Hide or unhide Parameters panel.

Counter

- ▷ Hide or unhide Counter panel.

I/Os & Software TriggerOut

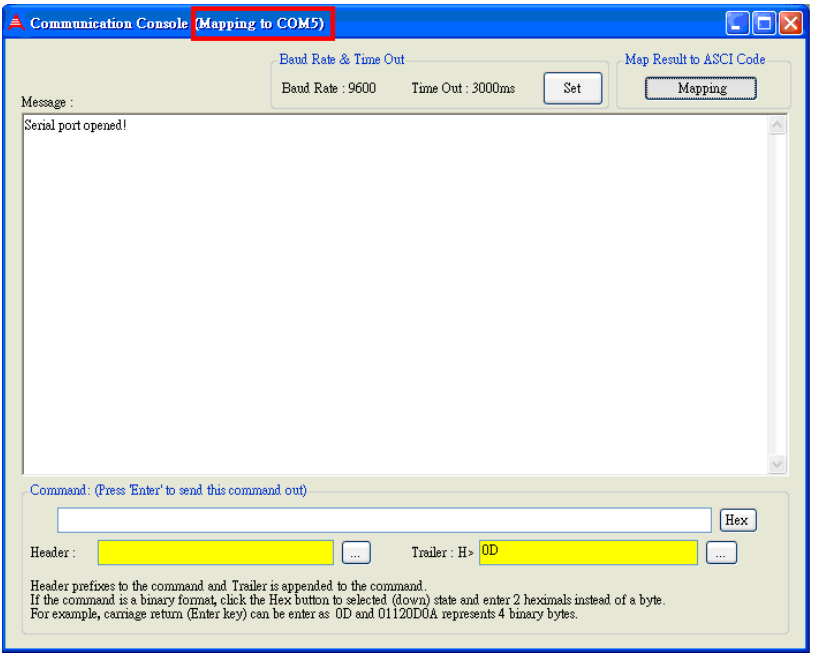
- ▷ Hide or unhide I/Os & Software TriggerOut panel.

Console Menu

Connection

Open a terminal window that allow user to communicate with camera. The title of this window will show which COM is mapped.

Press the Mapping button. The messages from camera will then be mapped to corresponding ASCI code. Or the messages will be shown as hexadecimal numerals..



5 Function Library

This chapter describes the API for PCIe-CPL64 frame grabber. Users can use these functions to develop application programs under Visual C++, C#, Visual Basic.Net, and Borland C++ Builder.

The particular functions associated with each function are presented in following table.

5.1 Function List

Table 5-1 lists all of power over camera link API functions. For the detail, please reference to the following sections.

Category	Function Name	Section
Board Information	GetDeviceCount	5.4
	GetDeviceModelName	
	GetDeviceVersion	
	GetDeviceFirmwareVersion	
	GetLibraryVersion	
Board Control	OpenDevice	5.5
	CloseDevice	
	LoadCamFile	
	SaveCamFile	
Acquisition	StartCapture	5.6
	StopCapture	
	SnapShot	
	GetFrameRate	
	GetChannelState	
	GetDroppedFrames	
	SaveImage	
Capture Mode	GetScanMode	5.7
	SetScanMode	
	GetCaptureMode	
	SetCaptureMode	

Category	Function Name	Section
Image Format	GetCameraConfiguration	5.8
	SetCameraConfiguration	
	GetSensorWidth	
	SetSensorWidth	
	GetSensorHeight	
	SetSensorHeight	
	GetXOffset	
	SetXOffset	
	GetYOffset	
	SetYOffset	
	GetSensorTap	
	SetSensorTap	
	GetPixelSize	
	SetPixelSize	
	GetLVALEnable	
	SetLVALEnable	
	GetDVALEnable	
	SetDVALEnable	
	GetDataValidDelay	
	SetDataValidDelay	
	GetPixelFormat	
	SetPixelFormat	
	GetSensorTapPlacement	
	SetSensorTapPlacement	
	GetDataBits	

Category	Function Name	Section
Camera Control Pin	GetCC1Type	5.9
	SetCC1Type	
	GetCC2Type	
	SetCC2Type	
	GetCC1PulseWidth	
	SetCC1PulseWidth	
	GetCC2PulseWidth	
	SetCC2PulseWidth	
	GetCC1NegativePulseWidth	
	SetCC1NegativePulseWidth	
	GetCC2NegativePulseWidth	
	SetCC2NegativePulseWidth	
	GetCC1Polarity	
	SetCC1Polarity	
	GetCC2Polarity	
	SetCC2Polarity	
	GetCC3Polarity	
	SetCC3Polarity	
	GetCC4Polarity	
	SetCC4Polarity	
Encoder Input	GetEncoderInputMode	5.10
	SetEncoderInputMode	
	GetEncoderDelayCount	
	SetEncoderDelayCount	
	GetEncoderCompareCount	
	SetEncoderCompareCount	
	GetEncoderInputDirection	
	SetEncoderInputDirection	
	GetEncoderCounterValue	
	EncoderCounterReset	

Category	Function Name	Section
Line Area Trigger	GetLineAreaTriggerType	5.11
	SetLineAreaTriggerType	
	GetLineAreaTriggerPolarity	
	SetLineAreaTriggerPolarity	
	GetLineAreaTriggerStartEnable	
	SetLineAreaTriggerStartEnable	
	GetLineAreaCounterValue	
	LineAreaCounterReset	
Trigger Out	GetTriggerOutState	5.12
	SetTriggerOutState	
	GetTriggerOutPolarity	
	SetTriggerOutPolarity	
	GetTriggerOutPulseWidth	
	SetTriggerOutPulseWidth	
	GetTriggerOutMode	
	SetTriggerOutMode	
	GetTriggerInPolarity	
	SetTriggerInPolarity	
	SoftwareTriggerOut	
GPIO	GetDI	5.13
	GetDO	
	SetDO	
	GetDIEvent	
	SetDIEvent	
Cable Power	GetPowerState	5.14
	SetPowerState	
Buffer	GetSystemBufferCount	5.15
	SetSystemBufferCount	
Invoke	SetCallback	5.16
Error Message	GetErrorMessage	5.17

Table 5-1: Power Over Camera Link Api Function List

Table 5-2 lists all of standard asynchronous serial communication API functions. For the detail, please reference to the following sections.

Category	Function Name	Section
Serial Communication	clFlushPort	5.18
	clGetErrorText	
	clGetManufacturerInfo	
	clGetNumBytesAvail	
	clGetNumSerialPorts	
	clGetSerialPortIdentifier	
	clGetSupportedBaudRates	
	clSerialClose	
	clSerialInit	
	clSerialRead	
	clSerialWrite	
	clSetBaudRate	

Table 5-2: Standard Serial Communication API Function List

5.2 Setting Up the Build Environment

5.2.1 Power Over Camera Link Library

Include Files

All applications using the APIs need include the file shown in the following table.

Include File	Description
Cpl64.h	The header file required for all C/C++ applications.
Cpl64.vb	The function definitions required for all VB.Net applications.
Cpl64.cs	The function definitions required for all C# applications.

Library File

All C/C++ applications using the APIs need the library file shown in the following table.

Library File	Description
Cpl64.lib	Exports API function definitions. Required for all Visual C/C++ applications.
Cpl64_bcb.lib	Exports API function definitions. Required for all Borland C++ Builder applications.

DLL Files

All applications using the APIs need the DLL file shown in the following table.

Library File	Description
Cpl64.dll	Dynamic link library. Required for all applications.

All files are located on the directory [Installed directory]\ADLINK\cpl64\Include. 'Installed directory' is the destination directory where you specified in setup program.

5.2.2 Standard serial communication library

Include Files

All applications using the APIs need include the file shown in the following table.

Include File	Description
clallserial.h	The header file required for all C/C++ applications.
clallserial.vb	The function definitions required for all VB.Net applications.
clallserial.cs	The function definitions required for all C# applications.

Library File

All C/C++ applications using the APIs need the library file shown in the following table.

Library File	Description
clallserial.lib	Exports API function definitions. Required for all Visual C/C++ applications.
clallserial_bcb.lib	Exports API function definitions. Required for all Borland C++ Builder applications.

DLL Files

All applications using the APIs need the DLL file shown in the following table.

Library File	Description
clallserial.dll	Dynamic link library. Required for all applications.
clseradl.dll	

Registry

All applications using the APIs need to add the registry:

Registry Key:

```
HKEY_LOCAL_MACHINE\software\cameralink
      Value Name: CLSERIALPATH
      Value Type: REG_SZ
      Value data: c:\cameralink\serial
```

All files are located on the directory [Installed directory]\ADLINK\cpl64\Include. 'Installed directory' is the destination directory while you run setup disk.

5.3 Camera File

Camera file is a text file which set the predefined parameters for a specific camera. Users can use a text editor to modify it. Below describe the content of camera file.

5.3.1 CameraInfo Section

Name

Camera model name

Manufacturer

Camera manufacturer

5.3.2 CaptureMode Section

ScanMode

Indicate the type of the camera. Could be one of the following values:

- 0: LineScan
- 1: AreaScan.

CaptureMode

Indicate the capture mode of the board. Could be one of the following values:

- 0: FreeRun
- 1: LineAreaTrigger
- 2: EncoderInput (LineScan CCD only)

5.3.3 ImageFormat Section

SensorWidth

Indicate the width of the camera sensor. Possible value is:

- 1 - 16000000

SensorHeight

Indicate the height of the camera sensor. Possible value is:

- 8 - 25000

XOffset

Indicate how many pixels in a line you want to delay to capture image.

YOffset

Indicate how many lines in a frame you want to delay to capture image.

Tap

Indicate the number of taps of the camera sensor. Could be one of the following values:

- 1, 2, 3, 4

PixelSize

Indicate the total size in bits of a pixel of the image. Could be one of following value:

- 8, 10, 12, 14, 16, 24, 30, 36

TapPlcaement

Indicate the scan method and direction of the camera sensor. Could be one of the following values:

- 0, 1, 2, 3, 4, 5, 6, 7, 10, 16

PixelFormat

Indicate whether pixel data includes dummy bytes. Could be one of the following values:

- 0: Unpack
- 1: Pack

LVALEnable

Indicate whether Line Valid is enabled. Could be one of the following values:

- 0: Disable
- 1: Enable

DVALEnable

Indicate whether Data Valid is enabled. Could be one of the following values:

- 0: Disable
- 1: Enable

DVALDelay

Indicate the delay of Data Valid.

5.3.4 CameraControl Section

CC1Type

Indicate the output type of CC1 pin. Could be one of the following values:

- 0: Pulse (the CC1 output is according to CC1PulseWidth and CC1NegativePulseWidth)
- 1: DO (the CC1 output is according to CC1Polarity)

CC2Type

Indicate the output type of CC2 pin. Could be one of the following values:

- 0: Pulse (the CC2 output is according to CC2PulseWidth and CC2NegativePulseWidth)
- 1: DO (the CC2 output is according to CC2Polarity)

CC1PulseWidth

Indicate the output pulse width of CC1 pin. Possible value is:

- 1 - 6500 (unit: 1μs)

CC2PulseWidth

Indicate the output pulse width of CC2 pin. Possible value is:

- 1 - 6500 (unit: 1μs)

CC1NegativePulseWidth

Indicate the negative output pulse width of CC1 pin. Possible value is:

- 1 - 6500 (unit: 1μs)

CC2NegativePulseWidth

Indicate the negative output pulse width of CC2 pin. Possible value is:

- 1 - 6500 (unit: 1μs)

CC1Polarity

Indicate the output level of CC1 pin. Could be one of the following values:

- 0: Normal (or Low)
- 1: Inverse (or High)

CC2Polarity

Indicate the output level of CC2 pin. Could be one of the following values:

- 0: Normal (or Low)
- 1: Inverse (or High)

CC3Polarity

Indicate the output level of CC3 pin. Could be one of the following values:

- 0: Low
- 1: High

CC4Polarity

Indicate the output level of CC4 pin. Could be one of the following values:

- 0: Low
- 1: High

5.3.5 EncoderInput Section

According to external ABZ inputs, capture one line of image each trigger pulse.

EncoderInputMode

Indicate the mode of encoder input trigger. Could be one of the following values:

- 0: APhase
- 1: ABPhase
- 2: ABZPhase

EncoderInputDirection

Indicate the direction of AB phase or ABZ phase. Could be of the following values:

- 0: CW (Clockwise)
- 1: CCW (Counter Clockwise)

EncoderDelayCount

Ignore how many encoder pulses after starting capture.

EncoderCompareCount

The system will capture one line of image each EncoderComapre-Count pulses. EncoderComapreCount must be larger than 0.

5.3.6 LineAreaTrigger Section

Capture one line of image each trigger input in Line Scan mode, or one image in Area Scan mode.

LineAreaTriggerType

Indicate the external input source. Could be one of the following values:

- 0: TTL
- 1: RS422

LineAreaTriggerStartEnable

Indicate whether enable line/area trigger input. Could be one of the following values:

- 0: Disable
- 1: Enable

LineAreaTriggerPolarity

Indicate the trigger polarity. Could be one of the following values:

- 0: Rising
- 1: Falling

5.3.7 Example

The structure of Camera file is as following:

```
[SectionName1]
Key1=Value1; Comment
Key2=Value2

; This is a comment
[SectionName2]
Key3=Value3; Comment
```

In above, all names, keys, and values are case insensitive. Values can be number or string. For example, ScanMode=AreaScan is same as ScanMode=1. The characters after semicolon will be ignored. And any spaces and tabs can be inserted before, after and between keys and values.

Following is an actual camera file for Teli CSCX30BC3 camera:

```
[CameraInfo]
Name=CSCX30BC3
Manufacturer=Teli

[CaptureMode]
ScanMode=AreaScan; 0:LineScan, 1:AreaScan
CaptureMode=FreeRun; 0:FreeRun,
     1:LineAreaTrigger, 2:EncoderInput

[ImageFormat]
SensorWidth=1024; CCD width
SensorHeight=768; CCD height
XOffset=0       ; Skip pixels
YOffset=0       ; Skip lines
Tap=1; Tap/PixelSize: 1/8, 1/10, 1/12, 1/14, 1/
     16, 1/24, 2/8, 2/10, 2/12, 3/8 for Base
     configuration
PixelSize=8     ; Tap/PixelSize: 4/8, 4/10, 4/12,
     1/30, 1/36 for Medium configuration
TapPlacement=0; Tap/TapPlacement: 1/0, 2/1, 2/2,
     2/3, 2/10, 4/4, 4/5, 4/6, 4/7, 3/16
PixelFormat=Unpack; 0:Unpack, 1:Pack
LVALEnable=Enable; 0:Disable, 1:Enable
DVALEnable=Enable; 0:Disable, 1:Enable
```

```
DVALDelay=0     ; DVAL reference value when
     DVALEnable is enabled
[CameraControl]
CC1Type=Pulse   ; 0: Pulse, 1:DO
CC2Type=Pulse   ; 0: Pulse, 1:DO
CC1PulseWidth=24
CC2PulseWidth=24
CC1NegativePulseWidth=1
CC2NegativePulseWidth=1
CC1Polarity=Normal; 0:Normal (or Low), 1:Inverse
     (or Low)
CC2Polarity=Normal; 0:Normal (or Low), 1:Inverse
     (or Low)
CC3Polarity=Low ; 0:Low, 1:High
CC4Polarity=Low ; 0:Low, 1:High

[EncoderInput]
EncoderInputMode=APhase; 0:APhase(A phase),
     1:ABPhase(A/B phase), 2:ABZPhase(A/B/Z
     phase)
EncoderInputDirection=CW; 0:CW, 1:CCW
EncoderDelayCount=0
EncoderCompareCount=1

[LineAreaTrigger]
LineAreaTriggerType=TTL; 0:TTL, 1:RS422
LineAreaTriggerStartEnable=0; 0:Disable, 1:Enable
LineAreaTriggerPolarity=Rising; 0:Rising (edge),
     1:Falling (edge)
```

5.4 Board Information Functions

5.4.1 DeviceCount

Purpose

This function returns the total number of PCIe-CPL64 devices in your system.

Prototype

C/C++	int Cpl64_GetDeviceCount()
C#	int GetDeviceCount()
VB.Net	GetDeviceCount () As Integer

Return Value

Returns the total number of devices if return value >= 0; else refer to Section 5.17 for more information about return codes.

5.4.2 DeviceModelName

Purpose

This function gets or returns the model name of the device.

Prototype

C/C++	int Cpl64_GetDeviceModelName(int CardID, char *Name)
C#	string GetDeviceModelName(int CardID)
VB.Net	GetDeviceModelName (ByVal CardID As Integer) As String

Parameters

CardID	Card ID can be set by SW1. The allowed value is from 0 to 15. Please refer to the description of SW1 setting in chapter 2 Hardware Reference.
Name	Pointer to a user-allocated buffer into which the function copies the device model name. The name is NULL-terminated. It will be “PCIe-CPL64” for PCIe-CPL64 card.

Return Value

C/C++	Refer to Section 5.17 for more information about return codes.
C#	Return device model name. The name will be “PCIe-CPL64” for PCIe-CPL64 card.
VB.Net	Return device model name. The name will be “PCIe-CPL64” for PCIe-CPL64 card.

5.4.3 DeviceVersion

Purpose

This function gets or returns the version of hardware device.

Prototype

C/C++	int Cpl64_GetDeviceVersion(int CardID, char *Version)
C#	string GetDeviceVersion(int CardID)
VB.Net	GetDeviceVersion(ByVal CardID As Integer) As String

Parameters

CardID	Card ID can be set by SW1. The allowed value is from 0 to 15. Please refer to the description of SW1 setting in chapter 2 Hardware Reference.
Version	Pointer to a user-allocated buffer into which the function copies the version string. The version is NULL-terminated.

Return Value

C/C++	Refer to Section 5.17 for more information about return codes.
C#	Return hardware version.
VB.Net	Return hardware version.

5.4.4 DeviceFirmwareVersion

Purpose

This function gets or returns the version of firmware.

Prototype

C/C++	int Cpl64_GetDeviceFirmwareVersion(int CardID, char *Version)
C#	string GetDeviceFirmwareVersion(int CardID)
VB.Net	GetDeviceFirmwareVersion(ByVal CardID As Integer) As String

Parameters

CardID	Card ID can be set by SW1. The allowed value is from 0 to 15. Please refer to the description of SW1 setting in chapter 2 Hardware Reference.
Version	Pointer to a user-allocated buffer into which the function copies the version string. The version is NULL-terminated.

Return Value

C/C++	Refer to Section 5.17 for more information about return codes.
C#	Return firmware version.
VB.Net	Return firmware version.

5.4.5 LibraryVersion

Purpose

This function gets or returns the version of DLL library.

Prototype

C/C++	int Cpl64_GetLibraryVersion(char *Version)
C#	string GetLibraryVersion()
VB.Net	GetLibraryVersion() As String

Parameters

CardID	Card ID can be set by SW1. The allowed value is from 0 to 15. Please refer to the description of SW1 setting in chapter 2 Hardware Reference.
Version	Pointer to a user-allocated buffer into which the function copies the version string. The version is NULL-terminated.

Return Value

C/C++	Refer to Section 5.17 for more information about return codes.
C#	Return library version.
VB.Net	Return library version.

5.5 Board Control Functions

5.5.1 OpenDevice

Purpose

This function initializes the device referred to by CardID and returns a handle. You should call this function before you call other functions except board information functions.

Prototype

C/C++	int Cpl64_OpenDevice(int CardID, int Channel)
C#	int OpenDevice(int CardID, int Channel)
VB.Net	OpenDevice (ByVal CardID As Integer, ByVal Channel As Integer) As Integer

Parameters

CardID	Card ID can be set by SW1. The allowed value is from 0 to 15. Please refer to the description of SW1 setting in chapter 2 Hardware Reference.
Channel	Which channel you want to open. For PCIe-CPL64, this value should be 0 or 1.

Return Value

Returns handle if the return value >= 0. Otherwise refer to Section 5.17 for more information about return codes.

5.5.2 CloseDevice

Purpose

This function closes a channel and release all allocated resources. Application should call this function before you terminate your application.

Prototype

C/C++	int Cpl64_CloseDevice(int Handle)
C#	int CloseDevice(int Handle)
VB.Net	CloseDevice (ByVal Handle As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Return Value

Refer to Section 5.17 for more information about return codes.

5.5.3 LoadCamFile

Purpose

This function loads a predefined camera file and set parameters according to this camera file.

Prototype

C/C++	int Cpl64_LoadCamFile(int Handle, char *FileName)
C#	int LoadCamFile(int Handle, string FileName)
VB.Net	LoadCamFile (ByVal Handle As Integer, ByVal FileName As String) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
FileName	The file name of the camera file you want to load.

Return Value

Refer to Section 5.17 for more information about return codes.

5.5.4 SaveCamFile

Purpose

This function saves a camera file from current parameters.

Prototype

C/C++	int Cpl64_SaveCamFile(int Handle, char *FileName)
C#	int SaveCamFile(int Handle, string FileName)
VB.Net	SaveCamFile (ByVal Handle As Integer, ByVal FileName As String) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
FileName	The file name of the camera file you want to save.

Return Value

Refer to Section 5.17 for more information about return codes.

5.6 Acquisition Functions

5.6.1 StartCapture

Purpose

This function starts to capture frame images. Before this function is called, user should call SetCallback function to register user's callback routine. The user's callback routine will be called when frames are ready.

Prototype

C/C++	int Cpl64_StartCapture(int Handle, int TotalFrameCount=0)
C#	int StartCapture(int Handle, int TotalFrameCount)
VB.Net	StartCapture (ByVal Handle As Integer, ByVal TotalFrameCount As Integer) As Integer

Parameters

Handle
: The value returned by the OpenDevice function.

TotalFrameCount
: Indicates how many frames you want to capture. When the specific count is reached, the system stops capturing. If you set the count to 0, the system will never stop until StopCapture function is called.

Return Value

Refer to Section 5.17 for more information about return codes.

5.6.2 StopCapture

Purpose

This function stops capturing. User need to call this function after start capturing even TotalFrameCount was reached.

Prototype

C/C++	int Cpl64_StopCapture(int Handle)
C#	int StopCapture(int Handle)
VB.Net	StopCapture (ByVal Handle As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Return Value

Refer to Section 5.17 for more information about return codes.

5.6.3 SnapShot

Purpose

This function grabs one frame within a specified time out.

Prototype

C/C++	int Cpl64_SnapShot(int Handle, void **Buffer, int Timeout)
C#	int SnapShot(int Handle, out IntPtr Buffer, int Timeout)
VB.Net	SnapShot (ByVal Handle As Integer, ByRef Buffer As IntPtr, ByVal Timeout as Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Buffer	Points to a frame buffer. Upon a successful call, contains the data of the frame image.
Timeout	Indicates the timeout, in milliseconds.

Return Value

Refer to Section 5.17 for more information about return codes.

5.6.4 FrameRate

Purpose

This function gets frame rate.

Prototype

C/C++	int Cpl64_GetFrameRate(int Handle, float& Rate)
C#	int GetFrameRate(int Handle, out float Rate)
VB.Net	GetFrameRate (ByVal Handle As Integer, ByRef Rate As Single) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Rate	Indicates the frame rate in fps.

Return Value

Refer to Section 5.17 for more information about return codes.

5.6.5 ChannelState

Purpose

This function gets the state of the channel: stopped or running.

Prototype

C/C++	int Cpl64_GetChannelState(int Handle, int& State)
C#	int GetChannelState(int Handle, out int State)
VB.Net	GetChannelState (ByVal Handle As Integer, ByRef State As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
State	0: stopped; 1: running

Return Value

Refer to Section 5.17 for more information about return codes.

5.6.6 DroppedFrames

Purpose

This function gets dropped frames after start capturing. The dropped frames will be reset when call StartCapture function.

The system capture frames as soon as it can. But sometimes user's callback routine spends too much time that the next frame will be dropped because the frame buffer queue is full. To solve this problem, user needs to shorten the process time of the call-back routine, or enlarge the frame buffer queue.

Prototype

C/C++	int Cpl64_GetDroppedFrames(int Handle, int& Frames)
C#	int GetDroppedFrames(int Handle, out int Frames)
VB.Net	GetDroppedFrames (ByVal Handle As Integer, ByRef Frames As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Frames	Indicates how many frames were dropped.

Return Value

Refer to Section 5.17 for more information about return codes.

Example

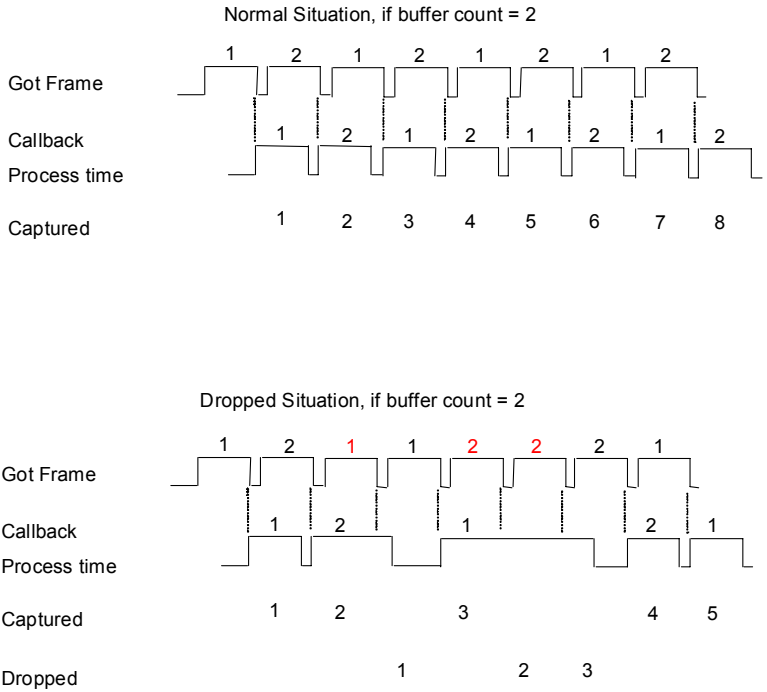


Figure 5-1: Dropped Frames

5.6.7 SaveImage

Purpose

This function saves frame buffer as an image file or as an raw data file depending on the extension of the file name.

Prototype

C/C++	int Cpl64_SaveImage(int Handle, char *FileName, void *Buffer)
C#	int SaveImage(int Handle, string FileName, IntPtr Buffer)
VB.Net	SaveImage (ByVal Handle As Integer, ByVal FileName As String, ByVal Buffer() As Byte) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
FileName	The file name of the image you want to save. The library supports following type of images: BMP: if FileName is *.bmp JPEG: if FileNAME is *.jpg or *.jpeg JIFF: if FileName is *.tif PNG: if FileName is *.png Raw data: other file type
Buffer	Pointer of the frame buffer you got from callback routine or from the Buffer parameter of SnapShot function. Be careful, this buffer will be release when call StartCapture function. Don't pass it to SaveImage function in such situation.

Return Value

Refer to Section 5.17 for more information about return codes.

5.7 Capture Mode Functions

5.7.1 ScanMode

Purpose

This function gets or sets scan mode. Scan mode is a camera specific parameter.

Prototype

C/C++	int Cpl64_GetScanMode(int Handle, int& Mode)
	Int Cpl64_SetScanMode(int Handle, int Mode)
C#	int GetScanMode(int Handle, out int Mode)
	int SetScanMode(int Handle, int Mode)
VB.Net	GetScanMode (ByVal Handle As Integer, ByRef Mode As Integer) As Integer
	SetScanMode (ByVal Handle As Integer, ByVal Mode As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Mode	0: Line scan CCD; 1: Area scan CCD

Return Value

Refer to Section 5.17 for more information about return codes.

5.7.2 CaptureMode

Purpose

This function gets or sets capture mode.

Prototype

C/C++	int Cpl64_GetCaptureMode(int Handle, int& Mode)
	Int Cpl64_SetCaptureMode(int Handle, int Mode)
C#	int GetCaptureMode(int Handle, out int Mode)
	int SetCaptureMode(int Handle, int Mode)
VB.Net	GetCaptureMode (ByVal Handle As Integer, ByRef Mode As Integer) As Integer
	SetCaptureMode (ByVal Handle As Integer, ByVal Mode As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Mode	Indicates how and when to capture the frame. There are 4 modes: 0: Free Run; Continue grab frames without waiting for external trigger 1: Line Area Trigger; Grabs a line each external trigger in (Line scan CCD) or grabs a frame each external trigger in (Area scan CCD). Please refer to chapter 5.11. 2: Encoder Input; Grabs a line each pulse of ABZ signal in (Line scan CCD only). Please refer to chapter 5.10.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8 Image Format Functions

Image format functions are camera specific parameters. Most of they can be set from camera file.

5.8.1 CameraConfiguration

Purpose

This function gets or sets the configuration of camera.

Prototype

C/C++	int Cpl64_GetCameraConfiguration(int Handle, int& Configuration)
	Int Cpl64_SetCameraConfiguration(int Handle, int Configuration)
C#	int GetCameraConfiguration(int Handle, out int Configuration)
	int SetCameraConfiguration(int Handle, int Configuration)
VB.Net	GetCameraConfiguration (ByVal Handle As Integer, ByRef Configuration As Integer) As Integer
	SetCameraConfiguration (ByVal Handle As Integer, ByVal Configuration As Integer) As Integer

Parameters

Handle — The value returned by the OpenDevice function.

Configuration

0: Base configuration camera, 1: Medium configuration camera

The PCIe-CPL64 supports dual base configuration camera or one medium configuration camera. Only channel 0 can select medium configuration. Channel 1 can not select this mode.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.2 SensorWidth

Purpose

This function gets or sets the width of camera sensor. Sensor width is actual width of captured frame.

Prototype

C/C++	int Cpl64_GetSensorWidth(int Handle, int& Width)
	Int Cpl64_SetSensorWidth(int Handle, int Width)
C#	int GetSensorWidth(int Handle, out int Width)
	int SetSensorWidth(int Handle, int Width)
VB.Net	GetSensorWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetSensorWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Width	Indicates the width of camera sensor.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.3 SensorHeight

Purpose

This function gets or sets the height of camera sensor. Sensor height is actual height of captured frame.

Prototype

C/C++	int Cpl64_GetSensorHeight(int Handle, int& Height)
	Int Cpl64_SetSensorHeight(int Handle, int Height)
C#	int GetSensorHeight(int Handle, out int Height)
	int SetSensorHeight(int Handle, int Height)
VB.Net	GetSensorHeight (ByVal Handle As Integer, ByRef Height As Integer) As Integer
	SetSensorHeight (ByVal Handle As Integer, ByVal Height As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Height	Indicates the height of camera sensor.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.4 XOffset

Purpose

This function gets or sets the offset of pixels in each line.

Prototype

C/C++	int Cpl64_GetXOffset (int Handle, int& Offset)
	Int Cpl64_SetXOffset (int Handle, int Offset)
C#	int GetXOffset (int Handle, out int Offset)
	int SetXOffset (int Handle, int Offset)
VB.Net	GetXOffset (ByVal Handle As Integer, ByRef Offset As Integer) As Integer
	SetXOffset (ByVal Handle As Integer, ByVal Offset As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Offset	Indicates how many pixels in each scan line to be skipped.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.5 YOffset

Purpose

This function gets or sets the offset of lines in each frame.

Prototype

C/C++	int Cpl64_GetYOffset (int Handle, int& Offset)
	Cpl64_SetYOffset (int Handle, int Offset)
C#	int GetYOffset (int Handle, out int Offset)
	int SetYOffset (int Handle, int Offset)
VB.Net	GetYOffset (ByVal Handle As Integer, ByRef Offset As Integer) As Integer
	SetYOffset (ByVal Handle As Integer, ByVal Offset As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Offset	Indicates how many lines in each frames to be skipped.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.6 SensorTap

Purpose

This function gets or sets the size of tap of camera.

Prototype

C/C++	int Cpl64_GetSensorTap(int Handle, int& Tap)
	Int Cpl64_SetSensorTap(int Handle, int Tap)
C#	int GetSensorTap (int Handle, out int Tap)
	int SetSensorTap (int Handle, int Tap)
VB.Net	GetSensorTap (ByVal Handle As Integer, ByRef Tap As Integer) As Integer
	SetSensorTap (ByVal Handle As Integer, ByVal Tap As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Tap	Indicates the size of tap.

Return Value

Refer to Section 5.17 for more information about return codes.

Reference

Please see chapter 5.8.11 for the relation with PixelSize and chapter 5.8.12 for the relation with TapPlacement.

5.8.7 PixelSize

Purpose

This function gets or sets valid bits of a pixel of camera.

Prototype

C/C++	int Cpl64_GetPixelSize(int Handle, int& Size)
	Int Cpl64_SetPixelSize(int Handle, int Size)
C#	int GetPixelSize (int Handle, out int Size)
	int SetPixelSize (int Handle, int Size)
VB.Net	GetPixelSize (ByVal Handle As Integer, ByRef Size As Integer) As Integer
	SetPixelSize (ByVal Handle As Integer, ByVal Size As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Size	Indicated the valid bits of a pixel except if Tap =3 and PixelSize = 8, the valid bits are 24 which is RGB24 color sensor. The valid bits are arranged from highest bit (MSB).

Return Value

Refer to Section 5.17 for more information about return codes.

Reference

Please see chapter 5.8.11 for the supported pixel size.

Example

If SensorTap=1 and PixelSize=12, the frame data is:

Pixel	0		1		2		3	
Byte	0	1	2	3	4	5	6	7
Data, in hexadecimal	10	11	f0	ff	00	00	30	10

In this example, the value of pixels is, in hexadecimal:

Pixel 0 = 111, Pixel 1 = fff, Pixel 2 = 0, Pixel 3 = 103

If SensorTap=1 and PixelSize=36 (RGB121212), PixelFormat=0 (Unpack), the frame data is:

Pixel	0								1							
RGB	dummy		R		G		B		dummy		R		G		B	
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data, in hexadecimal	00	00	10	11	10	11	10	11	00	00	f0	ff	00	00	00	00

In this example, the value of pixels is, in hexadecimal:

Pixel 0 = 111 (R) + 111 (G) + 111 (B) = gray color, Pixel 1 = 0 (R) + 0 (G) + fff (B) = blue color

5.8.8 LVALEnable

Purpose

This function gets or sets the state of Line Valid. Line Valid (LVAL) is defined HIGH for valid pixel.

Prototype

C/C++	int Cpl64_GetLVALEnable(int Handle, int& Enable)
	Int Cpl64_SetLVALEnable(int Handle, int Enable)
C#	int GetLVALEnable (int Handle, out int Enable)
	int SetLVALEnable (int Handle, int Enable)
VB.Net	GetLVALEnable (ByVal Handle As Integer, ByRef Enable As Integer) As Integer
	SetLVALEnable (ByVal Handle As Integer, ByVal Enable As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Enable	0: Disable LVAL, 1: Enable LVAL

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.9 DVALEnable

Purpose

This function gets or sets the state of Data Valid. Data Valid (DVAL) is defined HIGH when data is valid.

Prototype

C/C++	int Cpl64_GetDVALEnable(int Handle, int& Enable)
	Int Cpl64_SetDVALEnable(int Handle, int Enable)
C#	int GetDVALEnable (int Handle, out int Enable)
	int SetDVALEnable (int Handle, int Enable)
VB.Net	GetDVALEnable (ByVal Handle As Integer, ByRef Enable As Integer) As Integer
	SetDVALEnable (ByVal Handle As Integer, ByVal Enable As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Enable	0: Disable DVAL, 1: Enable DVAL

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.10 DataValidDelay

Purpose

This function gets or sets the delay of Data Valid. The delay is the clocks between the rising edge of Data Valid and the time when data is valid.

Prototype

C/C++	int Cpl64_GetDataValidDelay(int Handle, int& Delay)
	Int Cpl64_SetDataValidDelay(int Handle, int Delay)
C#	int GetDataValidDelay (int Handle, out int Delay)
	int SetDataValidDelay (int Handle, int Delay)
VB.Net	GetDataValidDelay (ByVal Handle As Integer, ByRef Delay As Integer) As Integer
	SetDataValidDelay (ByVal Handle As Integer, ByVal Delay As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Delay	Indicates the delay of Data Valid, in clock.

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.11 PixelFormat

Purpose

This function gets or sets the packing method of frame data.

Prototype

C/C++	int Cpl64_GetPixelFormat(int Handle, int& Format)
	Int Cpl64_SetPixelFormat(int Handle, int Format)
C#	int GetPixelFormat (int Handle, out int Format)
	int SetPixelFormat (int Handle, int Format)
VB.Net	GetPixelFormat (ByVal Handle As Integer, ByRef Format As Integer) As Integer
	SetPixelFormat (ByVal Handle As Integer, ByVal Format As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Format	0: Unpack, 1: Pack
	Unpack format is the raw data coming from hardware. Pack format is the data eliminating dummy bytes of unpack data.
	If user selects pack format, the library consumes more time and more CPU resource to rearrange frame data than unpack format.

The following table is the of the relation between SensorTap, PixelSize, and bytes of a pixel:

SensorTap	PixelSize	Bytes of pixel in unpack	Bytes of pixel in pack
1	8	1	1
1	10, 12, 14, 16	2	2
1	24	4	3
1	30, 36	8	6
2	8	1	1
2	10, 12,	2	2
3	8	4	3
4	8	1	1
4	10, 12	2	2

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.12 SensorTapPlacement

Purpose

This function gets the scan method and direction of sensor tap.

Prototype

C/C++	int Cpl64_GetSensorTapPlacement(int Handle, int& Value)
	Int Cpl64_SetSensorTapPlacement(int Handle, int Value)
C#	int GetSensorTapPlacement (int Handle, out int Value)
	int SetSensorTapPlacement (int Handle, int Value)
VB.Net	GetSensorTapPlacement (ByVal Handle As Integer, ByRef Value As Integer) As Integer
	SetSensorTapPlacement (ByVal Handle As Integer, ByVal Value As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Value	0: single tap, 1: dual tap – even-odd sequence, 2: dual tap – start & middle position, 3: dual tap – start & end position, 10: dual tap – two ways from middle position, 4: quad tap – progress scan sequence, 5: quad tap – 4 start position, 6: quad tap – even-odd pair, start & middle position, 7: quad tap – even-odd pair, start & end position, 16 triple tap

Below are the diagrams:

0: Single Tap

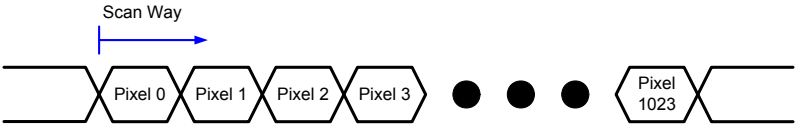


Figure 5-2: Single Tap Placement

1: Dual Tap – Even-Odd sequence

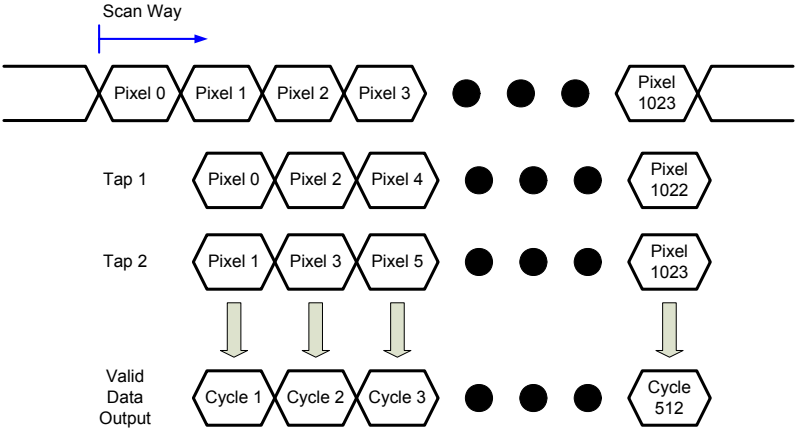


Figure 5-3: Dual Tap- Even-odd Sequence Placement

2: Dual Tap – Start & Middle position

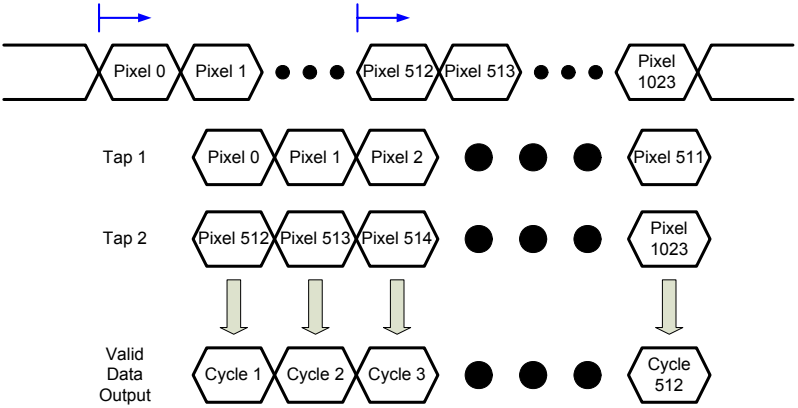


Figure 5-4: Dual Tap- Start & Middle Position Placement

3: Dual Tap – Start & End position

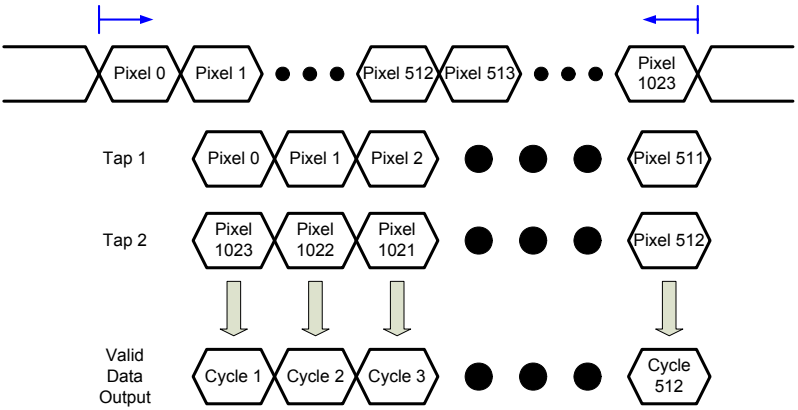


Figure 5-5: Dual Tap- Start & End Position Placement

4: Quad Tap – Progress Scan sequence

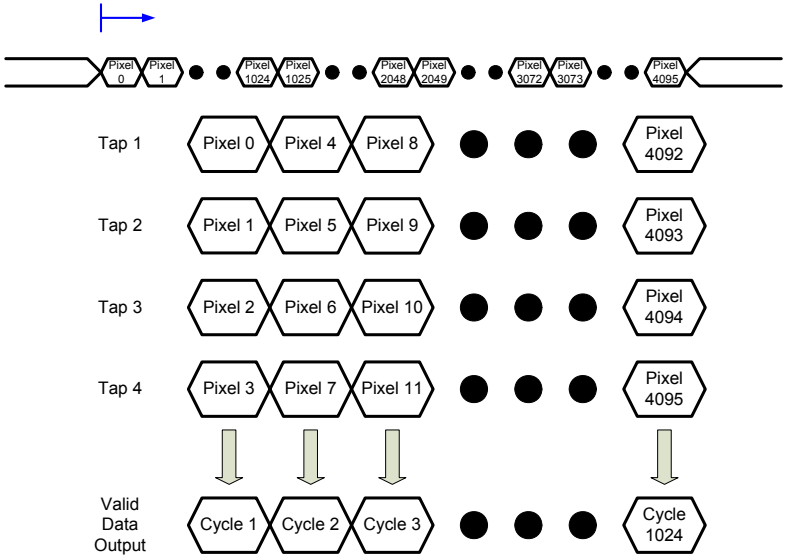


Figure 5-6: Quad Tap- Progress Scan Sequence Placement

5: Quad Tap – 4 Start position

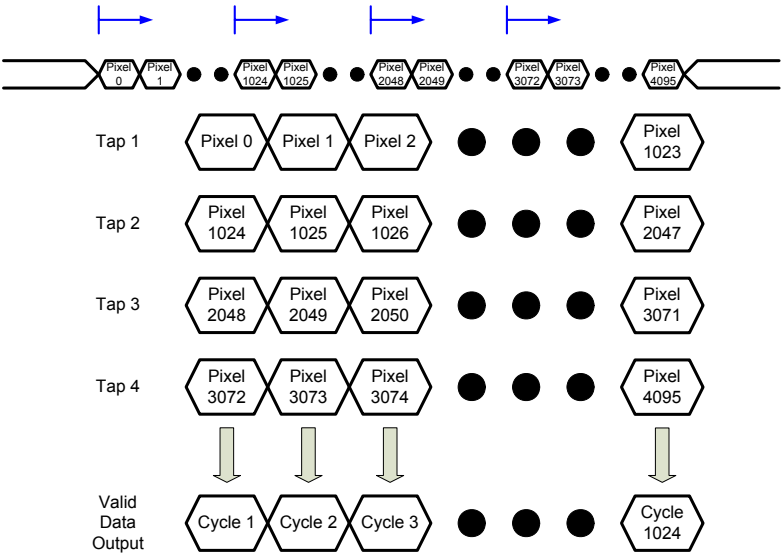


Figure 5-7: Quad Tap- 4 Start Position Placement

6: Quad Tap – Even-Odd pair, Start & Middle position

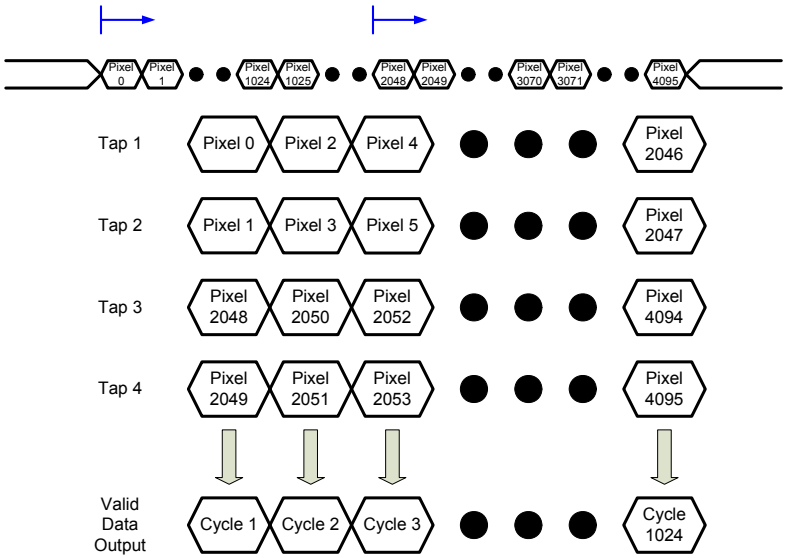


Figure 5-8: Quad Tap- Even-odd Pair, Start & Middle Position Placement

7: Quad tap – Even-Odd pair, Start & End position

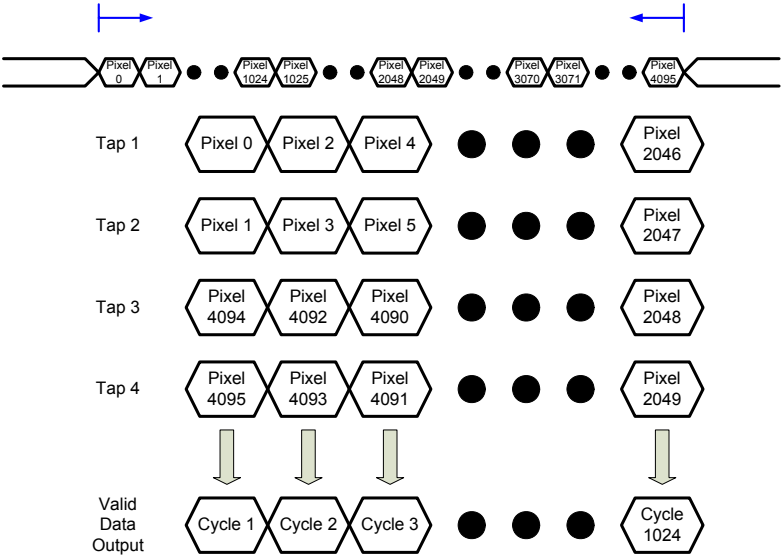


Figure 5-9: Quad Tap- Even-odd Pair, Start & End Position Placement

Below is the table of relation between SensorTap and SensorTap-Placement:

SensorTap	SensorTapPlacement
1	0
2	1, 2, 3, or 10
3	16
4	4, 5, 6, or 7

Return Value

Refer to Section 5.17 for more information about return codes.

5.8.13 DataBits

Purpose

This function gets the total bits of a pixel.

Prototype

C/C++	int Cpl64_GetDataBits(int Handle, int& DataBits)
C#	int GetDataBits (int Handle, out int DataBits)
VB.Net	GetDataBits (ByVal Handle As Integer, ByRef DataBits As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
DataBits	Indicates the total bits of a pixel.

Following is a table of bits of a pixel:

SensorTap	PixelSize	Bits of pixel in unpack	Bits of pixel in pack
1	8	8	8
1	10, 12, 14, 16	16	16
1	24	32	24
1	30, 36	64	48
2	8	8	8
2	10, 12,	16	16
3	8	32	24
4	8	8	8
4	10, 12	16	16

Return Value

Refer to Section 5.17 for more information about return codes.

5.9 Camera Control Functions

The camera control pins are LVDS pairs in standard Camera Link interface. They are reserved for general purpose camera control. In general, CC1 can be set as EXSYNC signal which controls the line period of camera. CC2 can be set as PRIN signal which controls the exposure time of camera. But this is not a general principle. You need to set them to conform to the camera specification.

5.9.1 CC1Type, CC2Type

Purpose

This function gets or sets the type of CC1 or CC2. CC1 and CC2 can be pulse output or general digital output.

Prototype

C/C++	int Cpl64_GetCC1Type(int Handle, int& Type)
	Int Cpl64_SetCC1Type(int Handle, int Type)
	Int Cpl64_GetCC2Type(int Handle, int& Type)
	Int Cpl64_SetCC2Type(int Handle, int Type)
C#	int GetCC1Type (int Handle, out int Type)
	int SetCC1Type (int Handle, int Type)
	int GetCC2Type (int Handle, out int Type)
	int SetCC2Type (int Handle, int Type)
VB.Net	GetCC1Type (ByVal Handle As Integer, ByRef Type As Integer) As Integer
	SetCC1Type (ByVal Handle As Integer, ByVal Type As Integer) As Integer
	GetCC2Type (ByVal Handle As Integer, ByRef Type As Integer) As Integer
	SetCC2Type (ByVal Handle As Integer, ByVal Type As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Type	0: Pulse ouput, 1: Digital output type. If the type is pulse output, CC1 is base on the settings of polarity, pulse width, and negative width. If the type is digital output, CC1 is based on the setting of polarity.

Return Value

Refer to Section 5.17 for more information about return codes.

5.9.2 CC1Polarity, CC2Polarity, CC3Polarity, CC4Polarity

Purpose

This function gets or sets the polarity of CC1, CC2, CC3, or CC4.

Prototype

C/C++	int Cpl64_GetCC1Polarity(int Handle, int& Polarity)
	Int Cpl64_SetCC1Polarity(int Handle, int Polarity)
	Int Cpl64_GetCC2Polarity(int Handle, int& Polarity)
	Int Cpl64_SetCC2Polarity(int Handle, int Polarity)
	Int Cpl64_GetCC3Polarity(int Handle, int& Polarity)
	Int Cpl64_SetCC3Polarity(int Handle, int Polarity)
	Int Cpl64_GetCC4Polarity(int Handle, int& Polarity)
	Int Cpl64_SetCC4Polarity(int Handle, int Polarity)
C#	int GetCC1Polarity (int Handle, out int Polarity)
	int SetCC1Polarity (int Handle, int Polarity)
	int GetCC2Polarity (int Handle, out int Polarity)
	int SetCC2Polarity (int Handle, int Polarity)
	int GetCC3Polarity (int Handle, out int Polarity)
	int SetCC3Polarity (int Handle, int Polarity)
	int GetCC4Polarity (int Handle, out int Polarity)
	int SetCC4Polarity (int Handle, int Polarity)
VB.Net	GetCC1Polarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer

SetCC1Polarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

GetCC2Polarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer

SetCC2Polarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

GetCC3Polarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer

SetCC3Polarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

GetCC4Polarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer

SetCC4Polarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Polarity	0: Low, 1: High, or 0: Normal, 1: Inverse for CC1 and CC2 when their type is pulse output

Return Value

Refer to Section 5.17 for more information about return codes.

5.9.3 CC1PulseWidth, CC2PulseWidth

Purpose

This function gets or sets the width of pulse output.

Prototype

C/C++	int Cpl64_GetCC1PulseWidth(int Handle, int& Width)
	Int Cpl64_SetCC1PulseWidth(int Handle, int Width)
	Int Cpl64_GetCC2PulseWidth(int Handle, int& Width)
	Int Cpl64_SetCC2PulseWidth(int Handle, int Width)
C#	int GetCC1PulseWidth (int Handle, out int Width)
	int SetCC1PulseWidth (int Handle, int Width)
	int GetCC2PulseWidth (int Handle, out int Width)
	int SetCC2PulseWidth (int Handle, int Width)
VB.Net	GetCC1PulseWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetCC1PulseWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer
	GetCC2PulseWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetCC2PulseWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Width	Indicates the width of pulse width.

Return Value

Refer to Section 5.17 for more information about return codes.

5.9.4 CC1NegativePulseWidth, CC2NegativePulseWidth

Purpose

This function gets or sets the negative part of width of pulse output.

Prototype

C/C++	int Cpl64_GetCC1NegativePulseWidth(int Handle, int& Width)
	Int Cpl64_SetCC1NegativePulseWidth(int Handle, int Width)
	Int Cpl64_GetCC2NegativePulseWidth(int Handle, int& Width)
	Int Cpl64_SetCC2NegativePulseWidth(int Handle, int Width)
C#	int GetCC1NegativePulseWidth (int Handle, out int Width)
	int SetCC1NegativePulseWidth (int Handle, int Width)
	int GetCC2NegativePulseWidth (int Handle, out int Width)
	int SetCC2NegativePulseWidth (int Handle, int Width)
VB.Net	GetCC1NegativePulseWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetCC1NegativePulseWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer
	GetCC2NegativePulseWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetCC2NegativePulseWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Width	Indicates the negative part of width of pulse output.

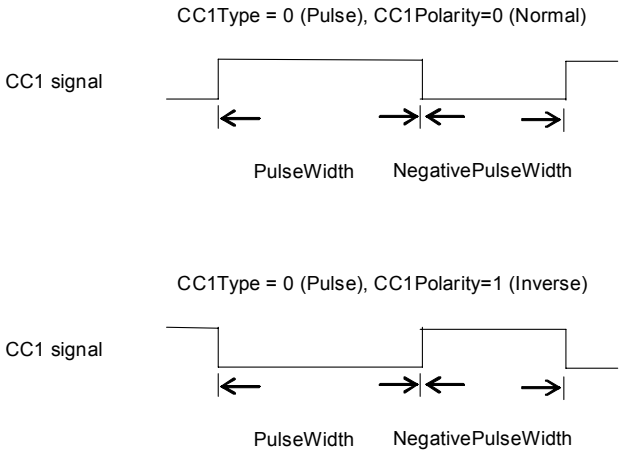


Figure 5-10: CC1, CC2 Pulse Output

Return Value

Refer to Section 5.17 for more information about return codes.

5.10 Encoder Input Functions

Encoder trigger is a kind of external trigger. This trigger can be used only for line scan CCD. System grabs a scan line from camera when it receives a trigger signal. User can call SetScanMode function to use this trigger.

One PCIe-CPL64 card owns one encoder input. If user connects 2 base configuration cameras to both channels of the card, the two cameras share one encoder input.

The following sections describe the parameters of encoder trigger.

5.10.1 EncoderInputMode

Purpose

This function gets or sets the mode of encoder input. This mode indicates the type of trigger input sources. Encoder input usually connects to optics ruler or motor encoder.

Prototype

C/C++	int Cpl64_GetEncoderInputMode(int Handle, int& Mode)
	Int Cpl64_SetEncoderInputMode(int Handle, int Mode)
C#	int GetEncoderInputMode (int Handle, out int Mode)
	int SetEncoderInputMode (int Handle, int Mode)
VB.Net	GetEncoderInputMode (ByVal Handle As Integer, ByRef Mode As Integer) As Integer
	SetEncoderInputMode (ByVal Handle As Integer, ByVal Mode As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Mode 0: A phase, 1: A/B phase, 2: A/B/Z phase

A phase and B phase are square pulses. Z is a start signal. For example,

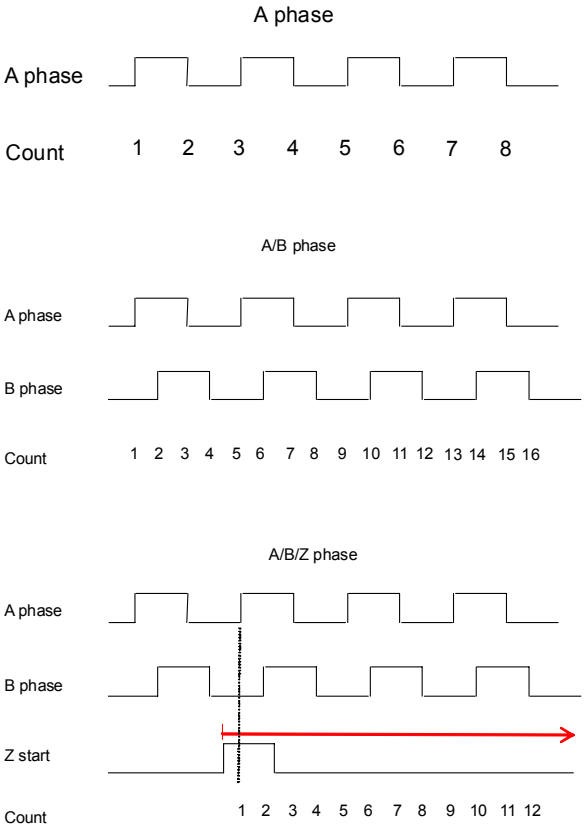


Figure 5-11: Encoder Input Signals

Return Value

Refer to Section 5.17 for more information about return codes.

5.10.2 EncoderDelayCount

Purpose

This function gets or sets the count of encoder input signal to be ignored after start capturing.

Prototype

C/C++	int Cpl64_GetEncoderDelayCount(int Handle, int& Count)
	Int Cpl64_SetEncoderDelayCount(int Handle, int Count)
C#	int GetEncoderDelayCount (int Handle, out int Count)
	int SetEncoderDelayCount (int Handle, int Count)
VB.Net	GetEncoderDelayCount (ByVal Handle As Integer, ByRef Count As Integer) As Integer
	SetEncoderDelayCount (ByVal Handle As Integer, ByVal Count As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Count	Indicates the delay count of encoder input signal.

Return Value

Refer to Section 5.17 for more information about return codes.

5.10.3 EncoderCompareCount

Purpose

This function gets or sets the compare count of encoder input signal. That is how many signals to be treated as a valid signal.

Prototype

C/C++	int Cpl64_GetEncoderCompareCount(int Handle, int& Count)
	Int Cpl64_SetEncoderCompareCount(int Handle, int Count)
C#	int GetEncoderCompareCount (int Handle, out int Count)
	int SetEncoderCompareCount (int Handle, int Count)
VB.Net	GetEncoderCompareCount (ByVal Handle As Integer, ByRef Count As Integer) As Integer
	SetEncoderCompareCount (ByVal Handle As Integer, ByVal Count As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Count	Indicates the number of the compare count. This value must be larger than 0.

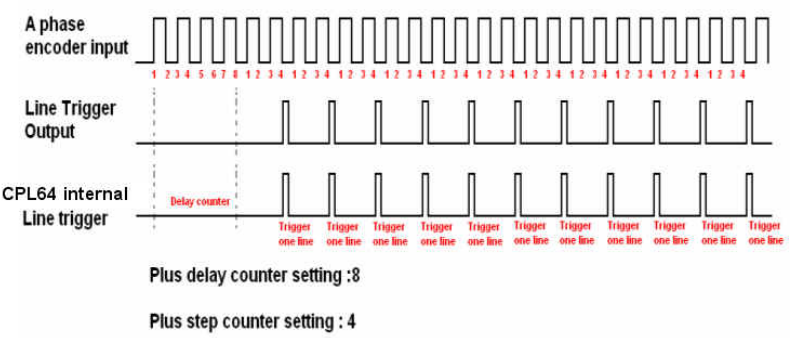


Figure 5-12: Delay Count and Compare Count Signals

Return Value

Refer to Section 5.17 for more information about return codes.

5.10.4 EncoderInputDirection

Purpose

This function gets or sets the direction of encoder input signal when encoder input mode is A/B phase or A/B/Z phase.

Prototype

C/C++	int Cpl64_GetEncoderInputDirection(int Handle, int& Direction)
	Int Cpl64_SetEncoderInputDirection(int Handle, int Direction)
C#	int GetEncoderInputDirection (int Handle, out int Direction)
	int SetEncoderInputDirection (int Handle, int Direction)
VB.Net	GetEncoderInputDirection (ByVal Handle As Integer, ByRef Direction As Integer) As Integer
	SetEncoderInputDirection (ByVal Handle As Integer, ByVal Direction As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Direction0: Clockwise rotation (CW), 1: Counter Clockwise rotation (CCW)

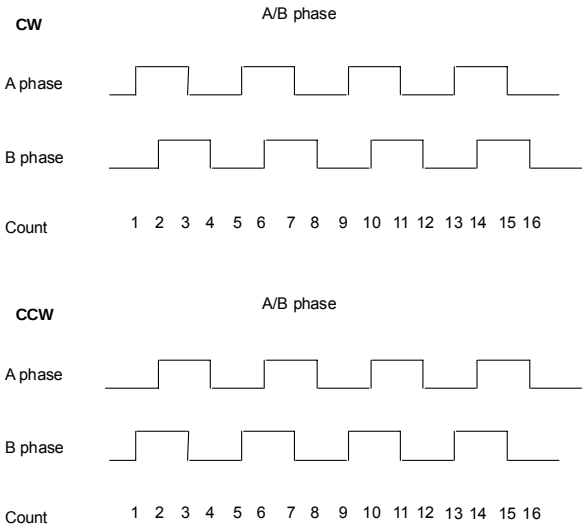


Figure 5-13: Encoder Input Direction Signals

Return Value

Refer to Section 5.17 for more information about return codes.

5.10.5 EncoderCounterValue

Purpose

This function gets the counter value of valid encoder trigger in. This value is incremental or decremented until you reset it by calling EncoderCounterReset function.

Prototype

C/C++	int Cpl64_GetEncoderCounterValue(int Handle, int& Count)
C#	int GetEncoderCounterValue (int Handle, out int Count)
VB.Net	GetEncoderCounterValue (ByVal Handle As Integer, ByRef Count As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Count	Indicates the counter value.

Return Value

Refer to Section 5.17 for more information about return codes.

5.10.6 EncoderCounterReset

Purpose

This function resets encoder counter.

Prototype

C/C++	int Cpl64_EncoderCounterReset(int Handle)
C#	int EncoderCounterReset (int Handle)
VB.Net	EncoderCounterReset (ByVal Handle As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Return Value

Refer to Section 5.17 for more information about return codes.

5.11 Line Area Trigger Functions

Line area trigger is a kind of external trigger. This trigger can be used with both of line scan CCD and area scan CCD. System grabs a scan line from line scan CCD or grabs a frame for area scan CCD when it receives a trigger signal. User can call SetScanMode function to use this trigger. Following sections describe the parameters of line area trigger.

5.11.1 LineAreaTriggerType

Purpose

This function gets or sets the type of line trigger in. Line trigger in is an external trigger input.

Prototype

C/C++	int Cpl64_GetLineAreaTriggerType(int Handle, int& Type)
	Int Cpl64_SetLineAreaTriggerType(int Handle, int Type)
C#	int GetLineAreaTriggerType (int Handle, out int Type)
	int SetLineAreaTriggerType (int Handle, int Type)
VB.Net	GetLineAreaTriggerType (ByVal Handle As Integer, ByRef Type As Integer) As Integer
	SetLineAreaTriggerType (ByVal Handle As Integer, ByVal Type As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Type	0: TTL type, 1: RS422 type If type is TTL, trigger signal comes from pin9 (channel 0) or pin10 (channel 1) of CN3. If type is RS422, trigger signal is a differential signal coming from pin1 (channel 0+) and pin2 (channel 0-) pair or pin3 (channel 1+) and pin4 (channel 1-) pair.

Return Value

Refer to Section 5.17 for more information about return codes.

5.11.2 LineAreaTriggerPolarity

Purpose

This function gets or sets the polarity of line trigger in.

Prototype

C/C++	int Cpl64_GetLineAreaTriggerPolarity(int Handle, int& Polarity)
	Int Cpl64_SetLineAreaTriggerPolarity(int Handle, int Polarity)
C#	int GetLineAreaTriggerPolarity (int Handle, out int Polarity)
	int SetLineAreaTriggerPolarity (int Handle, int Polarity)
VB.Net	GetLineAreaTriggerPolarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer
	SetLineAreaTriggerPolarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Polarity	0: Rising edge, 1: Falling edge

Return Value

Refer to Section 5.17 for more information about return codes.

5.11.3 LineAreaTriggerStartEnable

Purpose

This function gets or sets whether line area trigger is referred to the state of the line trigger start input.

Prototype

C/C++	int Cpl64_GetLineAreaTriggerStartEnable(int Handle, int& Enable)
	Int Cpl64_SetLineAreaTriggerStartEnable(int Handle, int Enable)
C#	int GetLineAreaTriggerStartEnable (int Handle, out int Enable)
	int SetLineAreaTriggerStartEnable (int Handle, int Enable)
VB.Net	GetLineAreaTriggerStartEnable (ByVal Handle As Integer, ByRef Enable As Integer) As Integer
	SetLineAreaTriggerStartEnable (ByVal Handle As Integer, ByVal Enable As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Enable	0: Disable, 1: Enable
	If disabled, the system grabs a scan line (line scan CCD) or a frame (area scan CCD) according directly to the signal of the line trigger in.
	If enabled, the system starts to grab only while the line trigger start input is in high level. For example:

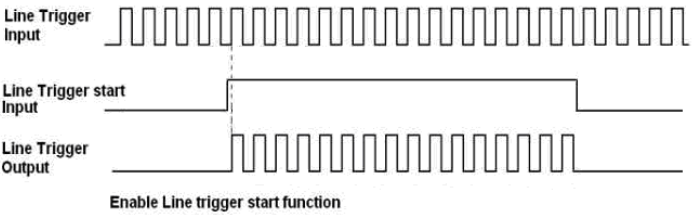


Figure 5-14: Line/area Start Enable Signal

Return Value

Refer to Section 5.17 for more information about return codes.

5.11.4 LineAreaCounterValue

Purpose

This function gets the counter value of valid line trigger in signal. This value is incremental until you reset it by calling LineAreaCounterReset function.

Prototype

C/C++	int Cpl64_GetLineAreaCounterValue(int Handle, int& Count)
C#	int GetLineAreaCounterValue (int Handle, out int Count)
VB.Net	GetLineAreaCounterValue (ByVal Handle As Integer, ByRef Count As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Count	Indicates the counter value.

Return Value

Refer to Section 5.17 for more information about return codes.

5.11.5 LineAreaCounterReset

Purpose

This function resets the counter value.

Prototype

C/C++	int Cpl64_LineAreaCounterReset(int Handle)
C#	int LineAreaCounterReset (int Handle)
VB.Net	LineAreaCounterReset (ByVal Handle As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Return Value

Refer to Section 5.17 for more information about return codes.

5.12 Trigger Out Functions

Trigger out is a kind of digital output. Usually we use it to control the light source of camera.

5.12.1 TriggerOutState

Purpose

This function gets or sets the state of trigger out.

Prototype

C/C++	int Cpl64_GetTriggerOutState(int Handle, int& Enable)
	Int Cpl64_SetTriggerOutState(int Handle, int Enable)
C#	int GetTriggerOutState (int Handle, out int Enable)
	int SetTriggerOutState (int Handle, int Enable)
VB.Net	GetTriggerOutState (ByVal Handle As Integer, ByRef Enable As Integer) As Integer
	SetTriggerOutState (ByVal Handle As Integer, ByVal Enable As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Enable	0: Disable trigger out, 1: Enable trigger out

Return Value

Refer to Section 5.17 for more information about return codes.

5.12.2 TriggerOutPolarity

Purpose

This function gets or sets the polarity of trigger out.

Prototype

C/C++	int Cpl64_GetTriggerOutPolarity(int Handle, int& Polarity)
	Int Cpl64_SetTriggerOutPolarity(int Handle, int Polarity)
C#	int GetTriggerOutPolarity (int Handle, out int Polarity)
	int SetTriggerOutPolarity (int Handle, int Polarity)
VB.Net	GetTriggerOutPolarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer
	SetTriggerOutPolarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Polarity	0: Low active, 1: High active

Return Value

Refer to Section 5.17 for more information about return codes.

5.12.3 TriggerOutPulseWidth

Purpose

This function gets or sets the pulse width of trigger out.

Prototype

C/C++	int Cpl64_GetTriggerOutPulseWidth(int Handle, int& Width)
	Int Cpl64_SetTriggerOutPulseWidth(int Handle, int Width)
C#	int GetTriggerOutPulseWidth (int Handle, out int Width)
	int SetTriggerOutPulseWidth (int Handle, int Width)
VB.Net	GetTriggerOutPulseWidth (ByVal Handle As Integer, ByRef Width As Integer) As Integer
	SetTriggerOutPulseWidth (ByVal Handle As Integer, ByVal Width As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Width	The pulse width, in 8 nanoseconds

Return Value

Refer to Section 5.17 for more information about return codes.

5.12.4 TriggerOutMode

Purpose

This function gets or sets the mode of trigger out.

Prototype

C/C++	int Cpl64_GetTriggerOutMode(int Handle, int& Mode)
	Int Cpl64_SetTriggerOutMode(int Handle, int Mode)
C#	int GetTriggerOutMode (int Handle, out int Mode)
	int SetTriggerOutMode (int Handle, int Mode)
VB.Net	GetTriggerOutMode (ByVal Handle As Integer, ByRef Mode As Integer) As Integer
	SetTriggerOutMode (ByVal Handle As Integer, ByVal Mode As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Mode	0: External trigger (trigger from trigger in signal), 1: Software trigger (trigger by software; SoftwareTriggerOut function)

Return Value

Refer to Section 5.17 for more information about return codes.

5.12.5 TriggerInPolarity

Purpose

This function gets or sets the polarity of trigger in.

Prototype

C/C++	int Cpl64_GetTriggerInPolarity(int Handle, int& Polarity)
	Int Cpl64_SetTriggerInPolarity(int Handle, int Polarity)
C#	int GetTriggerInPolarity (int Handle, out int Polarity)
	int SetTriggerInPolarity (int Handle, int Polarity)
VB.Net	GetTriggerInPolarity (ByVal Handle As Integer, ByRef Polarity As Integer) As Integer
	SetTriggerInPolarity (ByVal Handle As Integer, ByVal Polarity As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Polarity	0: Rising, 1: Falling

Return Value

Refer to Section 5.17 for more information about return codes.

Example

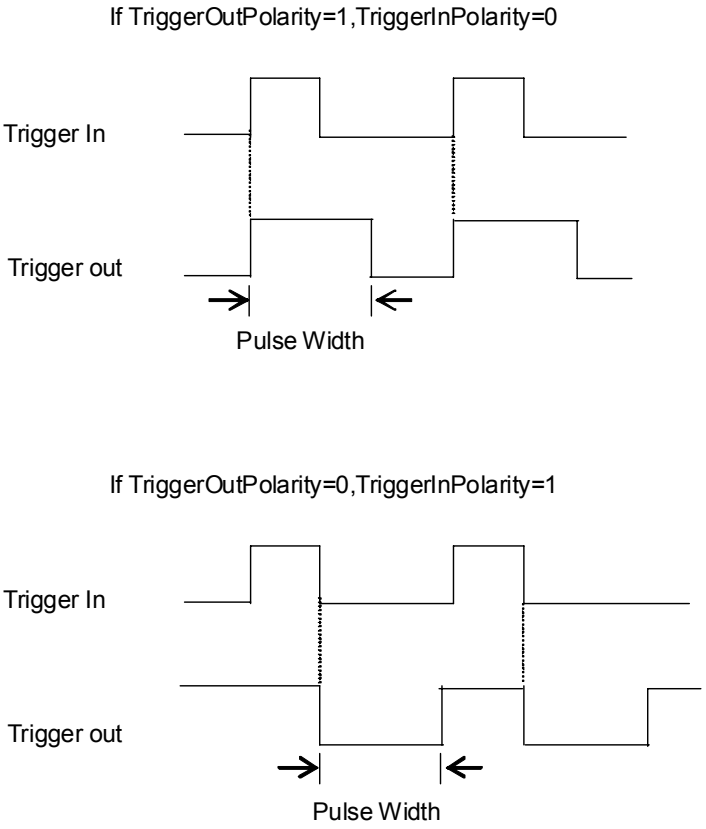


Figure 5-15: Timing Chart of Trigger Out and Trigger In

5.12.6 SoftwareTriggerOut

Purpose

This function outputs a pulse signal to trigger out I/O when the Mode of trigger out mode is 1 (SetTriggerOutMode function) and enable trigger out state (SetTriggerOutState function).

Prototype

C/C++	int Cpl64_SoftwareTriggerOut(int Handle)
C#	int SoftwareTriggerOut (int Handle)
VB.Net	SoftwareTriggerOut (ByVal Handle As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.

Return Value

Refer to Section 5.17 for more information about return codes.

5.13 GPIO Functions

GPIO is an abbreviation of General Purpose Input Output. A PCIe-CPL64 has 4 digital inputs and 4 digital outputs. You can control them from both channel 0 and channel 1 of camera port, but you are accessing the same I/Os.

5.13.1 DI

Purpose

This function gets the state of digital input. Please refer to chapter 2 Hardware Reference for the contacts of digital inputs

Prototype

C/C++	int Cpl64_GetDI(int Handle, int Channel, int& State)
C#	int GetDI(int Handle, int Channel, out int State)
VB.Net	GetDI (ByVal Handle As Integer, ByVal Channel As Integer, ByRef State As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Channel	Indicates the channel of digital input. This value can be from 0 to 3.
State	0: Low, 1: High

Return Value

Refer to Section 5.17 for more information about return codes.

5.13.2 DO

Purpose

This function gets or sets the state of digital output. Please refer to chapter 2 Hardware Reference for the contact of digital outputs.

Prototype

C/C++	int Cpl64_GetDO(int Handle, int Channel, int& State)
	Int Cpl64_SetDO(int Handle, int Channel, int State)
C#	int GetDO (int Handle, out int State)
	int SetDO (int Handle, int State)
VB.Net	GetDO (ByVal Handle As Integer, ByRef State As Integer) As Integer
	SetDO (ByVal Handle As Integer, ByVal State As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Channel	Indicates the channel of digital input. This value can be from 0 to 3.
State	0: Low, 1: High

Return Value

Refer to Section 5.17 for more information about return codes.

5.13.3 DIEvent

Purpose

This function gets or sets the event of digital input. If user enables the event, the system will raise an event signal when digital input changes its state. Users can set a callback routine by calling Set-Callback function to receive the event.

Prototype

C/C++	int Cpl64_GetDIEvent(int Handle, int Channel, int& Enable)
	Int Cpl64_SetDIEvent(int Handle, int Channel, int Enable)
C#	int GetDIEvent (int Handle, out int Enable)
	int SetDIEvent (int Handle, int Enable)
VB.Net	GetDIEvent (ByVal Handle As Integer, ByRef Enable As Integer) As Integer
	SetDIEvent (ByVal Handle As Integer, ByVal Enable As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
Channel	Indicates the channel of digital input. This value can be from 0 to 3.
Enable	0: Disable event, 1: Enable event

Return Value

Refer to Section 5.17 for more information about return codes.

5.14 Cable Power Functions

5.14.1 PowerState

Purpose

This function gets or sets the state of cable power.

Prototype

C/C++	int Cpl64_GetPowerState(int Handle, int& State)
	Int Cpl64_SetPowerState(int Handle, int State)
C#	int GetPowerState (int Handle, out int State)
	int SetPowerState (int Handle, int State)
VB.Net	GetPowerState (ByVal Handle As Integer, ByRef State As Integer) As Integer
	SetPowerState (ByVal Handle As Integer, ByVal State As Integer) As Integer

Parameters

Handle	The value returned by the OpenDevice function.
State	Could be: 0: Auto detect; Auto detect the type of camera. Provides power to camera if the camera is a power over camera link camera; otherwise provides no power. 1: Short ground; This mode is used for camera link camera. 2: Short +12 V; This mode is used for the power over camera link camera. Be careful; do not connect cable to camera link camera in this mode.

Return Value

Refer to Section 5.17 for more information about return codes.

5.15 Buffer Functions

5.15.1 SystemBufferCount

Purpose

This function gets or sets the size of frame buffer queue. Frame buffers are used to store frame data coming from camera. The queue will be allocated when StartCapture function is called.

Prototype

C/C++	int Cpl64_GetSystemBufferCount(int Handle, int& Count)
	Int Cpl64_SetSystemBufferCount(int Handle, int Count)
C#	int GetSystemBufferCount (int Handle, out int Count)
	int SetSystemBufferCount (int Handle, int Count)
VB.Net	GetSystemBufferCount (ByVal Handle As Integer, ByRef Count As Integer) As Integer
	SetSystemBufferCount (ByVal Handle As Integer, ByVal Count As Integer) As Integer

Parameters

Handle — The value returned by the OpenDevice function.

Count — Indicates the size of frame buffer queue. Allowed value is from 2 to 100. Default value is 4. More buffers consume more system memory. Below is the formula of required memory in byte:

Required memory = BufferCount * SensorWidth * SensorHeight * UnpackPixelByte

In this example, UnpackPixelByte lists in following table:

SensorTap	PixelSize	UnpackPixelByte
1	8	1
1	10, 12, 14, 16	2
1	24	4
1	30, 36	8
2	8	1
2	10, 12,	2
3	8	4
4	8	1
4	10, 12	2

For example, BufferCount = 4, SensoeWidth=8192, SensorHeight=512, SensorTap=3, and PixelSize=8, required memory = 4 * 8192 * 512 * 4 = 67108864 (bytes)

Return Value

Refer to Section 5.17 for more information about return codes.

5.16 Invoke Functions

5.16.1 Callback

Purpose

This function set user's defined callback routine. Callback routine will be called when event is ready.

Prototype

C/C++	int Cpl64_SetCallback(int Handle, int EventType, void (__stdcall *fun)(int Handle, int EventType, void *Buffer, int Size))
C#	int SetCallback(int Handle, int EventType, Callback CallbackProc)
VB.Net	SetCallback (ByVal Handle As Integer, ByVal EventType As Integer, ByVal CallbackProc As Callback) As Integer

Parameters

EventType	Indicates what kind of event this callback routine will belong to. May be one of the following: 0: frame image event 1: DI0 event 2: DI1 event 3: DI2 event 4: DI3 event
fun	Address of callback routine.
Callback	Pointer of callback routine.

Return Value

Refer to Section 5.17 for more information about return codes.

5.17 Error Message Functions

5.17.1 ErrorMessage

Purpose

Obtain the error message.

Prototype

C/C++	int Pro_GetErrorInfo(int Code, char *Message)
C#	string GetErrorMessage(int code)
VB.Net	GetErrorMessage(ByVal code As Integer) As String

Parameters

Code	The code returned by other power over camera link functions.
Message	The error message of Code.

Return Value

C/C++	Always return 0.
C#	Return message
VB.Net	Return message

Remarks

Below are the definition of error codes:

Error Code	Definition Name	Meaning
0	ERROR_NoError	Success
-1	ERROR_InvalidCardID	Invalid card ID
-2	ERROR_InvalidChannel	Invalid channel number
-3	ERROR_InvalidHandle	Invalid handle
-4	ERROR_CardNotExist	The card could not be found
-5	ERROR_SetupapiNotFound	setupapi.dll is missing

Error Code	Definition Name	Meaning
-6	ERROR_SetupapiCorrupt	Incorrect setupapi.dll
-7	ERROR_DeviceNotOpened	The device need to be opened before further operations are attempted
-8	ERROR_DeviceOpened	The device was already opened. Do not allowed open twice!
-9	ERROR_InsufficientMemory	Out of memory
-10	ERROR_CallDriverFailed	Driver is corrupt or its version doesn't match
-11	ERROR_InvalidArgument	The input argument(s) exceed allowed range
-12	ERROR_TooManyBuffers	The number of buffers exceed allowed range
-13	ERROR_DeviceRunning	An operation is invalid while device is still running
-14	ERROR_DeviceNotRunning	An operation is invalid while device is not running
-15	ERROR_NullPointer	The pointer could not be null
-16	ERROR_WaitTimeout	There is no response
-17	ERROR_InvalidCamFile	Some parameters of cam file are missing or their value are incorrect
-18	ERROR_FileCannotBeSave	The file is read-only or is locked by other process
-19	ERROR_FileNotExist	The file doesn't exist

5.18 Serial Communication Functions

Serial communication functions are frame grabber-independent API. Allow users to write a camera-specific configuration utility. The following API offers standard asynchronous serial communication API followed by camera link specification.

5.18.1 Defined Data Type

Defined Data Type	C++ Type	C# Type	VB.Net Type
hSerRef	void*	IntPtr	IntPtr
INT32	int	int	integer
UINT32	unsigned int	uint	UInt32
INT8	char	char	Byte

5.18.2 clFlushPort

Purpose

This function discards any bytes that are available in the input buffer.

Prototype

C/C++	INT32 clFlushPort(hSerRef serialRef)
C#	int clFlushPort(IntPtr serialRef)
VB.Net	clFlushPort(ByVal serialRef As IntPtr) As Integer

Parameters

serialRef	The value obtained by the clSerialInit function.

Return Value

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCE

Refer to Section 5.18.13 for more information about status codes.

5.18.3 clGetErrorText

Purpose

This function converts an error code to error text for display in a dialog box or in a standard I/O window.

Prototype

C/C++ INT32 clGetErrorText(const INT8* manuName, INT32 errorCode, INT8* errorText, UINT32* errorTextSize)

C# int clGetErrorText(string manuName, int errorCode, byte[] errorText, ref uint errorTextSize)

VB.Net clGetErrorText(ByVal manuName As String, ByVal errorCode As Integer, ByVal errorText() As Byte, ByRef errorTextSize As UInt32) As Integer

Parameters

manufacturerName
: The manufacturer name in a NULL-terminated buffer. Manufacturer name is returned from clGetPortInfo.

errorCode
: The error code used to find the appropriate error text.

errorText
: A caller-allocated buffer which contains the NULL-terminated error text on function return.

errorTextSize
: On success, contains the number of bytes written into the buffer, including the NULL-termination character. This value should be the size in bytes of the error text buffer passed in. On CL_ERR_BUFFER_TOO_SMALL, contains the size of the buffer needed to write the error text.

Return Value

CL_ERR_NO_ERR

CL_ERR_MANU_DOES_NOT_EXIST

CL_ERR_BUFFER_TOO_SMALL

CL_ERR_ERROR_NOT_FOUND

Refer to Section 5.18.13 for more information about status codes.

5.18.4 clGetNumBytesAvail

Purpose

This function outputs the number of bytes that are received at the port specified by serialRef but are not yet read out.

Prototype

C/C++	INT32 clGetNumBytesAvail(hSerRef serialRef, UINT32* numBytes)
C#	int clGetNumBytesAvail(IntPtr serialRef, ref uint numBytes)
VB.Net	clGetNumBytesAvail (ByVal serialRef As IntPtr, ByRef numBytes As UInt32) As Integer

Parameters

serialRef	The value obtained by the clSerialInit function.
numBytes	The number of bytes currently available to be read from the port.

Return Value

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCE

Refer to Section 5.18.13 for more information about status codes.

5.18.5 clGetNumPorts

Purpose

This function returns the total number of Camera Link serial ports in your system.

Prototype

C/C++	INT32 clGetNumPorts(UINT32* Ports)
C#	int clGetNumPorts(ref uint Ports)
VB.Net	clGetNumPorts (ByRef Ports As UInt32) As Integer

Parameters

numPorts	The number of Camera Link serial ports in your system.

Return Value

CL_ERR_NO_ERR

Refer to Section 5.18.13 for more information about status codes.

5.18.6 clGetPortInfo

Purpose

This function provides information about the port specified by the index.

Prototype

C/C++ INT32 clGetPortInfo(UINT32 serialIndex, INT8* manufacturerName, UINT32* nameBytes, INT8* portID, UINT32* IDBytes, UINT32* version)

C# int clGetPortInfo(uint serialIndex, byte[] manufacturerName, ref uint nameBytes, byte[] portID, byte[] IDBytes, ref uint version)

VB.Net clGetPortInfo (ByVal serialIndex As UInt32, ByVal manufacturerName() As Byte, ByRef nameBytes As UInt32, ByVal portID() As Byte, ByVal IDBytes() As Byte, ByRef version As UInt32) As Integer

Parameters

serialIndex The index of the port for which you want information. The valid range for this index is 0 to (n-1) where n is the value of numPorts returned by clGetNumPorts.

manufacturerName
Pointer to a user-allocated buffer into which the function copies the manufacturer name. The returned name is NULL-terminated. In the case that the DLL conforms to the October 2000 specification, this parameter will contain the file name of the DLL rather than the manufacturer name.

nameBytes As an input, this value should be the size of the buffer that is passed. On successful return, this parameter contains the number of bytes written

	into the buffer, including the NULL termination character. On CL_ERR_BUFFER_TOO_SMALL, this parameter contains the size of the buffer needed to write the data text..
portID	A manufacturer-specific identifier for the serial port. In the case that the manufacturer DLL conforms to the October 2000 specification, on return this parameter will be Port n, where n is a unique index for the port.
IDBytes	As an input, this value should be the size of the buffer that is passed. On successful return, this parameter contains the number of bytes written into the buffer, including the NULL-termination character. On CL_ERR_BUFFER_TOO_SAMLL, this parameter contains the size of the buffer needed to write the data text..
version	The version of the Camera Link specifications with which this frame grabber software complies.

Return Value

CL_ERR_NO_ERR

CL_ERR_BUFFER_TOO_SMALL

CL_ERR_INVALID_INDEX

Refer to Section 5.18.13 for more information about status codes.

5.18.7 clGetSupportedBaudRates

Purpose

This function returns the valid baud rates of the interface.

Prototype

C/C++	INT32 clGetSupportedBaudRates(hSerRef serialRef, UINT32* baudRates)
C#	int clGetSupportedBaudRates(IntPtr serialRef, ref uint baudRates)
VB.Net	clGetSupportedBaudRates (ByVal serialRef As IntPtr, ByRef baudRates As UInt32) As Integer

Parameters

serialRef	The value obtained from the clSerialInit function, which describes the port being queried for baud rates.
baudRates	Bitfield that describes all supported baud rates of the serial port as described by serialRef. The supported baud rates are defined below: CL_BAUDRATE_96001 CL_BAUDRATE_192002 CL_BAUDRATE_384004 CL_BAUDRATE_576008 CL_BAUDRATE_11520016

Return Value

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCEL

CL_ERR_FUNCTION_NOT_FOUND

Refer to Section 5.18.13 for more information about status codes.

5.18.8 clSerialClose

Purpose

This function closes the serial device and cleans up resources associated with serialRef. Upon return, serialRef is no longer usable.

Prototype

C/C++	void clSerialClose(hSerRef serialRef)
C#	void clSerialClose(IntPtr serialRef)
VB.Net	clSerialClose (ByVal serialRef As IntPtr)

Parameters

serialRef	The value obtained from the clSerialInit function, which describes the port being queried for baud rates.

5.18.9 clSerialInit

Purpose

This function initializes the device referred to by serialIndex, and returns a pointer to an internal serial reference structure.

Prototype

C/C++	INT32 clSerialInit (UINT32 serialIndex, hSerRef* serialRefPtr)
C#	int clSerialInit (uint serialIndex, ref IntPtr serialRefPtr)
VB.Net	clSerialInit (ByVal serialIndex As UInt32, ByRef serialRefPtr As IntPtr) As Integer

Parameters

serialIndex	A zero-based index value. If there are n serial devices in the system that is supported by this library, the range of serialIndex is 0 to (n-1). The order of the serial devices is vendor-specific. The number of serial ports supported by this DLL is output by the clGetNumSerialPorts function.
serialRefPtr	Upon a successful call, a pointer to the vendor-specific reference to the current serial session will be put into the value pointed to by serialRefPtr.

Return Value

CL_ERR_NO_ERR

CL_ERR_PORT_IN_USE

CL_ERR_INVALID_INDEX

CL_ERR_UNABLE_TO_OPEN_PORT

Refer to Section 5.18.13 for more information about status codes.

5.18.10clSerialRead

Purpose

This function reads from the serial device referenced by serialRef.

Prototype

C/C++	INT32 clSerialRead(hSerRef serialRef, INT8* buffer, UINT32* bufferSize, UINT32 serialTimeout)
C#	int clSerialRead(IntPtr serialRef, byte[] buffer, ref uint bufferSize, uint serialTimeout)
VB.Net	clSerialRead (ByVal serialRef As IntPtr, ByVal buffer() As Byte, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer

Parameters

serialRef	The value obtained from clSerialInit function.
buffer	Points to a user-allocated buffer. Upon a successful call, contains the data read from the serial device.
bufferSize	As an input parameter, bufferSize contains the buffer size to indicate the maximum number of bytes that the buffer can accomodate. Upon a successful call, bufferSize is the number of bytes that were read successfully from the serial device.
serialTimeout	Indicates the timeout, in milliseconds.

Return Value

CL_ERR_NO_ERR

CL_ERR_TIMEOUT

CL_ERR_INVALID_REFERENCE

Refer to Section 5.18.13 for more information about status codes.

5.18.11clSerialWrite

Purpose

This function writes data in buffer to the serial device referenced by serialRef.

Prototype

C/C++	INT32 clSerialWrite(hSerRef serialRef, INT8* buffer, UINT32* bufferSize, UINT32 serialTimeout)
C#	int clSerialWrite(IntPtr serialRef, string buffer, ref uint bufferSize, uint serialTimeout) or int clSerialWrite(IntPtr serialRef, byte[] buffer, ref uint bufferSize, uint serialTimeout)
VB.Net	clSerialWrite (ByVal serialRef As IntPtr, ByVal buffer As String, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer Or clSerialWrite (ByVal serialRef As IntPtr, ByVal buffer() As Byte, ByRef bufferSize As UInt32, ByVal serialTimeout As UInt32) As Integer

Parameters

serialRef	The value obtained from clSerialInit function.
buffer	Contains data to write to the serial port.
bufferSize	As an input parameter, bufferSize contains the number of bytes of data in the buffer to write to the serial device. Upon a successful call, bufferSize contains the number of bytes that was written successfully to the serial device.
serialTimeout	Indicates the timeout, in milliseconds.

Return Value

CL_ERR_NO_ERR

CL_ERR_TIMEOUT

CL_ERR_INVALID_REFERENCE

Refer to Section 5.18.13 for more information about status codes.

5.18.12clSetBaudRate

Purpose

This function sets the Baud Rate for the serial port of the selected port. Use clGetSupportedBaudRates to determine supported baud rates.

Prototype

C/C++	INT32 clSetBaudRate(hSerRef serialRef, UINT32 baudRate)
C#	int clSetBaudRate(IntPtr serialRef, uint baudRate)
VB.Net	clSetBaudRate (ByVal serialRef As IntPtr, ByVal baudRate As UInt32) As Integer

Parameters

serialRef	The value obtained from clSerialInit function.
baudRate	The baud rate you want to use. This input expects the values represented by the CL_BAUDRATE constants.

Return Value

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCE

CL_ERR_BAUD_RATE_NOT_SUPPORTED

Refer to Section 5.18.13 for more information about status codes.

5.18.13Status Codes

Error Code	Definition Name	Meaning
0	CL_ERR_NO_ERR	Function returned successfully.
-10001	CL_ERR_BUFFER_TOO_SMALL	User buffer not large enough to hold data.
-10002	CL_ERR_MANU_DOES_NOT_EXIST	The requested manufacturer's DLL does not exist on your system.
-10003	CL_ERR_PORT_IN_USE	Port is valid but cannot be opened because it is in use.
-10004	CL_ERR_TIMEOUT	Operation not completed within specified timeout period.
-10005	CL_ERR_INVALID_INDEX	Not a valid index
-10006	CL_ERR_INVALID_REFERENCE	The serial reference is not valid.
-10007	CL_ERR_ERROR_NOT_FOUND	Could not find the error description for this error code.
-10008	CL_ERR_BAUD_RATE_NOT_SUPPORTED	Requested baud rate is not supported by this interface.
-10009	CL_ERR_OUT_OF_MEMORY	System is out of memory and could not perform required action.
-10010	CL_ERR_REGISTRY_KEY_NOT_FOUND	Could not found cameralink registry key in system registry.
-10098	CL_ERR_UNABLE_TO_LOAD_DLL	The DLL was unable to load due to a lack of memory or because it does not export all required functions.
-10099	CL_ERR_FUNCTION_NOT_FOUND	Function does not exist in the manufacturer's library.