

# Basler Components



## Using the Pylon Color Conversion API

### APPLICATION NOTES

Document Number: AW000727

Version: 02 Language: 000 (English)

Release Date: 24 March 2009

## **Contacting Basler Support Worldwide**

### **Europe:**

Basler AG  
An der Strusbek 60 - 62  
22926 Ahrensburg  
Germany  
Tel.: +49-4102-463-500  
Fax.: +49-4102-463-599  
bc.support.europe@baslerweb.com

### **Americas:**

Basler, Inc.  
855 Springdale Drive, Suite 160  
Exton, PA 19341  
U.S.A.  
Tel.: +1-877-934-8472  
Fax.: +1-610-280-7608  
bc.support.usa@baslerweb.com

### **Asia:**

Basler Asia Pte. Ltd  
8 Boon Lay Way  
# 03 - 03 Tradehub 21  
Singapore 609964  
Tel.: +65-6425-0472  
Fax.: +65-6425-0473  
bc.support.asia@baslerweb.com

**[www.baslerweb.com](http://www.baslerweb.com)**

**Copyright 2009 Basler Vision Technologies.  
All material in this publication is subject to change without notice.**

# 1 Introduction

Basler GenICam compliant cameras with either an IEEE 1394 or a GigEVision interface (scout, pilot, runner series) can provide image data in different formats such as:

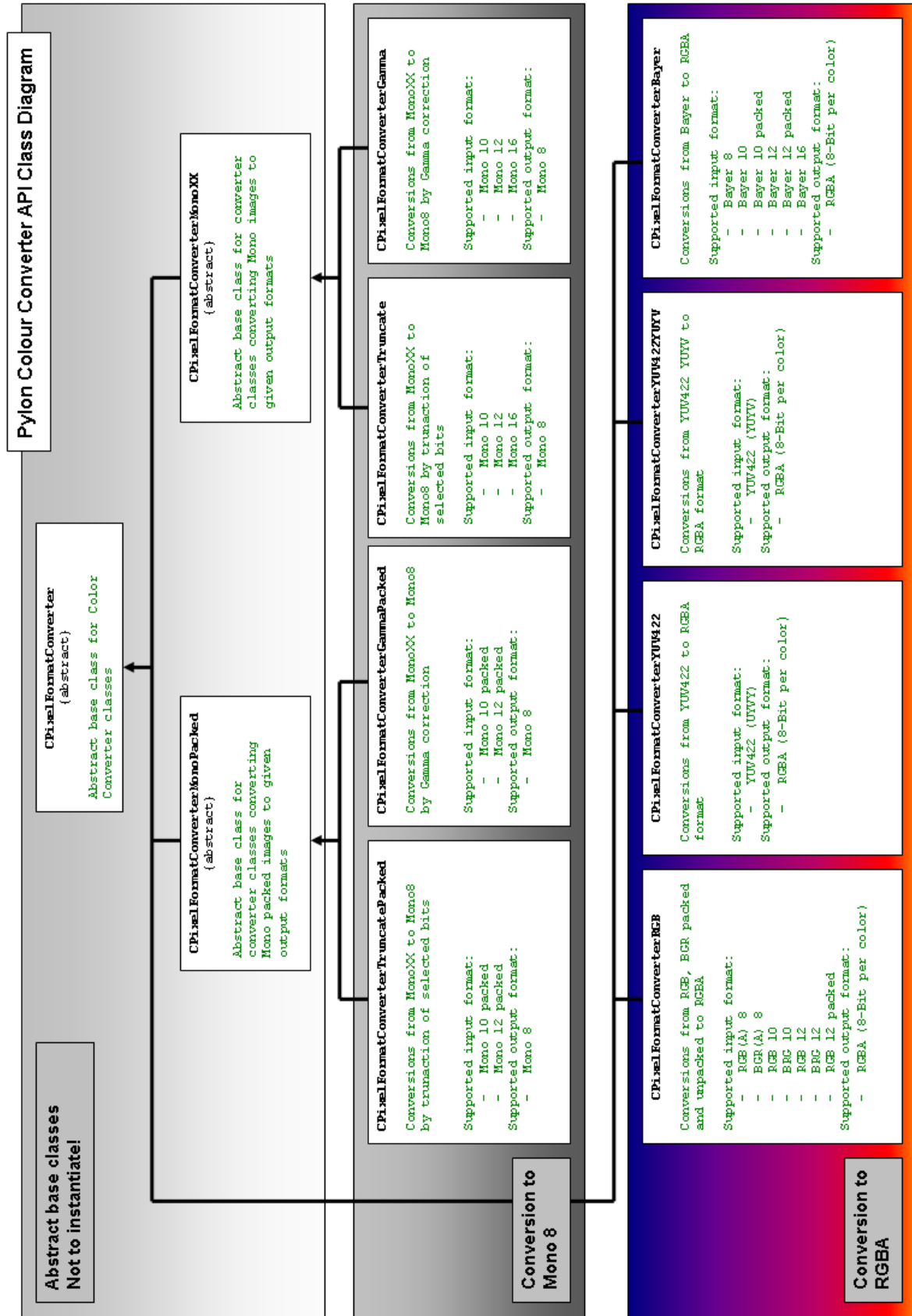
- Mono 8 (monochrome, 8 bits per pixel)
- Mono 10 (monochrome, 10 bits per pixel)
- Mono 12 (monochrome, 12 bits per pixel)
- Mono 16 (monochrome, 16 bits per pixel)
- YUV 4:2:2 packed and unpacked (16 bits per pixel)
- Various Bayer formats with 8 bits per pixel
- Various Bayer formats with 12 bits per pixel
- Various Bayer formats with 16 bits per pixel
- RGB8 (24 bits per pixel)
- RGB12 Packed (48 bits per pixel)
- RGB 12 V1 Packed (36 bits per pixel)

Not every camera can provide every pixel format. For example, the RGB formats are available on runner color cameras only.

When the captured image data must be displayed on a screen or stored as a bitmap to a file, most of the available image formats must be converted into a format that the operating system can handle and display:

- Mono8 for all monochrome formats (Mono10, Mono12, or Mono16)
- RGBA for all color formats (YUV, Bayer, and RGB formats).

Thus the only format that does not need to be converted is Mono8. To convert any of the other formats, you can use the pylon color conversion API, which has been part of the Pylon SDK since version 2.0.



## 2 API Description

The color conversion API provides a set of classes that can convert any monochrome input format into Mono8 and any color input format into RGBA.

All classes are derived from the abstract base class `CPixelFormatConverter` as illustrated in the class diagram on [page 2](#).

As you can see in the class diagram, there are two kinds of color converters - those which convert any kind of mono format into Mono8 and those which convert any kind of color format into RGBA.

When you would like to convert a camera's output format, you must first determine which `ColorConverter` class you'll need.

The usage of the `ColorConverter` consists of the following steps (in this example we convert a BayerBG8 image to RGBA):

1. Instantiate the required `ColorConverter` object:

```
CPixelFormatConverterBayer MyColorConverterBayer;
```

2. Create an instance of the struct `SImageFormat`, which describes the format of the image buffer to be converted (INPUT for `ColorConverter`):

```
SImageFormat MyInputformat;  
// Image width in pixels  
MyInputformat.Width = 640;  
// Image height in pixels  
MyInputformat.Height = 480;  
// Size of one image line in bytes (= Width * bytes per pixel)  
MyInputformat.LinePitch = 640;  
// enum of type "PixelFormat" declared in pixeltype.h  
MyInputformat.PixelFormat = PixelType_BayerBG8;
```

**!! The line pitch must ONLY be set if the input format is NOT a packet format !!**

**!! For packet formats (e.g., BayerGB12Packed), the line pitch must be set to zero !!**

3. Initialize the ColorConverter with the input format:

```
MyColorConverterBayer.Init (MyInputFormat);
```

4. Create a buffer that is big enough to hold the converted (OUTPUT) image.

A Mono8 image needs one byte per pixel, an RGBA image needs 4 bytes per pixel.

```
// An RGBA image needs 4 bytes per pixel
int BytePerPixel = 4
unsigned char *MyOutputBuffer = new unsigned char [640*480* BytePerPixel]
```

5. Create an instance of the struct SOutputImageFormat, which describes the format of the image buffer AFTER conversion (OUTPUT for ColorConverter):

```
SOutputImageFormat MyOutputFormat;
// Size of one image line in bytes (= width * bytes per pixel)
MyOutputFormat.LinePitch = 640 * BytePerPixel;
// enum of type "PixelFormat" declared in pixeltype.h
MyOutputFormat.PixelFormat = PixelType_RGBA8packed;
```

To display the image output buffer as a Windows bitmap, you must make sure that the width of the output buffer (and thus the LinePitch!) is DWORD aligned! This could be most easily achieved if the width of the INPUT buffer in PIXEL (= AOI Width) is already DWORD aligned.

6. Perform the actual conversion

```
MyColorConverterBayer ->Convert (
    MyOutputBuffer,           // Pointer to destination buffer
    640 * 480 * BytePerPixel, // Size of destination buffer in bytes
    MyInputBuffer,           // void* Pointer pointing to INPUT buffer
    640 * 480,               // Size of INPUT buffer
    MyInputFormat,           // Image format of the input image
    MyOutputFormat           // Image format of the output image
);
```

## Some additional remarks concerning conversion to Mono 8

The conversion to Mono 8 always implies that an input format with 10 to 16 bits per pixel must be converted into an output format with 8 bits per pixel (Mono 8). The pylon color conversion API provides two approaches for this conversion:

- Truncation (CPixelFormatConverterTruncate, CPixelFormatConverterTruncatePacked)  
For 10 bit input formats, 2 bits will be cut away to get an 8 bit output format. For 12 bit input formats, 4 bits will be cut away to get an 8 bit output format. For 16 bit input formats, 8 bits will be cut away to get an 8 bit output format. By default, the least significant bits will be truncated. With the SetHighOffsetBit method, the point where to truncate can be applied to any other bit.
- Gamma (CPixelFormatConverterGamma, CPixelFormatConverterGammaPacked)  
When using one of the Gamma color Converters, any more-than-8-bit input format will be mapped to an 8 bit output format using a lookup table that expresses a Gamma correction. The default Gamma value is 1.0, which is a linear mapping from any more-than-8-bit input format to 8 bits.  
With the SetGamma method you can apply any other gamma factor.



---

## Revision History

Doc. ID Number	Date	Changes
AW00072701000	29 Aug 2008	Initial release of this document.
AW00072702000	24 Mar 2009	Corrected an error in step 5 on <a href="#">page 4</a> regarding the size of one image line in bytes.

