CCutil Manual

Release 25 February 2003

C-Cam Technologies

a division of

Vector International

Copyright

Disclaimer

The information contained within this document has been carefully checked and is believed to be entirely reliable and consistent with the product that it describes. However, no responsibility is assumed for inaccuracies. C-Cam Technologies division of Vector International assumes no liability arising from of the application or use of any product or circuit described herein. C-Cam Technologies reserves the right to make changes to any product and product documentation in an effort to improve performance, reliability or design.

Trademarks

IBM, PC/AT, VGA and SVGA are registered trademarks of International Business Machine Corporation. MS-DOS is a registered trademark of Microsoft Corporation.

Restriction

This code is restricted in reproduction, use and transfer. See the Vector International conditions of use. The license is granted for use of the software on a single computer. By using the software, the user implies agreement to the conditions of use and agrees to settle all disputes through the court in Leuven Belgium.

Distribution

Distribution is only allowed through registered representatives. A list of these representatives can be found on our web site.

Contact address

C-Cam Technologies
division of
Vector International

Interleuvenlaan 46,
B-3001 Leuven
Belgium

Tel. +32 (0)16 40 20 16
Fax +32 (0)16 40 03 23

email info@vector-international.be
http://www.vector-international.be

1 Introduction 4
2 CCutil functions 5
2.1 Display functions 5
2.1.1 CCU_InitDisplay 5
2.1.2 CCU_DisplayBuffer 6
2.1.3 CCU_DisplayBufferFit 8
2.1.4 CCU_FreeDisplay 9
2.1.5 CCU_ConvertBuffer16To8 10
2.2 Histogram functions 11
2.2.1 CCU_CalculateDrawHistogram 11
2.2.2 CCU_CalculateDrawHistogramColor 12
2.3 Color functions 13
2.3.1 CCU_Colorize 13
2.3.2 CCU_CalculateWhiteBalance 15
2.4 Correction functions 16
2.4.1 CCU_FPNCorrection 16
2.5 File operation functions 17
2.5.1 CCU_SaveBufferAsBMP_BW 17
2.5.2 CCU_SaveBufferAsBMP_Color 18
2.5.3 CCU_SaveBufferAsBinary 19
3 Structures 20
3.1 CCU_ColorParms 20

1 Introduction

This document describes the Application Program Interface of the functions in the utility library called CCUTIL.DLL.
This utility library was first intended to be of use for only internally purposes. But as we deliver sample programs for controlling our cameras that make use of functions in this library, we decided to publish the functions in this library.
The functions in this library have nothing to do with camera acquisition, but are very useful tools on data from cameras already acquired by the functions in CCAPI.DLL. The number of functions in this library will grow in the future as soon as there is need of new functionality. The functions in this library are not optimized for speed, their only purpose is to quickly write demo programs. Also the algorithms used may not be perfect and can always be improved in the future. The interface of the functions described in this manual will not change anymore so programs using these functions need not to be rewritten when new updates of the library become available.
There are more functions available in this library then described in this manual, but they are still under development and their interface might still change. Therefor we advise you not to use these functions until they appear in this manual.

If you have questions regarding this document, please e-mail to c-cam@vector-international.be. We will be glad to help you.

The engineering team of C-Cam hopes you enjoy their effort in enhancing the industrial digital camera revolution.

C-Cam Technologies

2 CCutil functions

All functions in the ccutil.dll library have the same prefix: `CCU_`
Functions that can be found in the header file but are not described in this manual are still under development. We advise not to use these functions as their interface may still change or become obsolete.

2.1 Display functions

2.1.1 CCU_InitDisplay

- **Function:**

  Initializes the structures and reserves memory needed for displaying, this function has to be called only once before any other display function.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_InitDisplay( void ) ;
```

2.1.2 CCU_DisplayBuffer

- **Function:**

  Displays a buffer in an hDC of an object with the given dimensions. The buffers contents will be displayed as 8 bit data in either gray values or color. `CCU_InitDisplay` must be called before using this function.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_DisplayBuffer( HDC hDC,
                                   PVOID Buffer,
                                   USHORT XStart,
                                   USHORT YStart,
                                   USHORT XSize,
                                   USHORT YSize,
                                   USHORT XPan,
                                   USHORT YPan,
                                   float Zoom,
                                   BOOL Color ) ;
```

Variable	C Type	Purpose
HDC	Handle to Device Context	Destination where contents of Buffer will be drawn
Buffer	PVOID	Points to the data to be displayed, the data will be interpreted as 8 bit data. This can be the data that is acquired from a camera. If the data is in 16-bit format, use the function `CCU_ConvertBuffer16To8` to convert to the correct format.
XStart	USHORT	Specifies the left edge in the device context where the image will be drawn.
YStart	USHORT	Specifies the top edge in the device context where the image will be drawn.
XSize	USHORT	Specifies the width of the part of the image in the buffer you want to display.
YSize	USHORT	Specifies the height of the part of the image in the buffer you want to display.
XPan	USHORT	Determines the left edge position of the part of the image in the buffer you want to display.
YPan	USHORT	Determines the top edge position of the part of the image in the buffer you want to display.

Zoom	float	Specifies the magnification factor. 1 is actual pixel size, a value higher then 1 magnifies the image, a value lower then 1 makes the image smaller.
Color	BOOL	If FALSE, the image will be displayed as gray values. If TRUE, the image will be displayed as a color image. For a color image, the contents of the buffer should be in the correct format, see the function `CCU_Colorize` for more information on color.

2.1.3 CCU_DisplayBufferFit

- **Function:**

  Displays a buffer in an hDC of an object with the given dimensions and fits it in the given destination window. The buffers contents will be displayed as 8 bit data in either gray values or colour. `CCU_InitDisplay` must be called before using this function.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_DisplayBufferFit(  HDC hDC,
                                       PVOID Buffer,
                                       USHORT XDest,
                                       USHORT YDest,
                                       USHORT XDestSize,
                                       USHORT YDestSize,
                                       USHORT XSrcSize,
                                       USHORT YSrcSize,
                                       BOOL Color ) ;
```

Variable	C Type	Purpose
HDC	Handle to Device Context	Destination where contents of Buffer will be drawn
Buffer	PVOID	Points to the data to be displayed, the data will be interpreted as 8 bit data. This can be the data which is acquired from a camera. If the data is in 16 bit format, use the function `CCU_ConvertBuffer16To8` to convert to the correct format.
XDest	USHORT	Specifies the left edge in the device context where the image will be drawn.
YDest	USHORT	Specifies the top edge in the device context where the image will be drawn.
XDestSize	USHORT	Specifies the width of the image in the device context you want to be drawn.
YDestSize	USHORT	Specifies the height of the image in the device context you want to be drawn.
XSrcSize	USHORT	Specifies the width of the image in the buffer you want to display.
YSrcSize	USHORT	Specifies the height of the image in the buffer you want to display.
Color	BOOL	If FALSE, the image will be displayed as gray values. If TRUE, the image will be displayed as a color image. For a color image, the contents of the buffer should be in the correct format, see the function `CCU_Colorize` for more information on color.

2.1.4 CCU_FreeDisplay

- **Function:**

  Frees any allocated memory used for displaying. `CCU_InitDisplay` must be called before using this function.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_FreeDisplay( void ) ;
```

2.1.5 CCU_ConvertBuffer16To8

- **Function:**

  When an image is acquired from a camera with a bit-depth of more than 8, the contents of this buffer cannot be displayed correctly by the display functions. This function transforms the data in the buffer from a 16-bit format to an 8-bit format. Use this function before you call any of the display functions.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_ConvertBuffer16To8( PVOID InBuffer,
                                        PVOID OutBuffer,
                                        ULONG PixelCount,
                                        UCHAR BitShift ) ;
```

Variable	C Type	Purpose
InBuffer	PVOID	Pointer to the buffer which contains the 16 bit data.
OutBuffer	PVOID	Pointer to the buffer where the converted data should be written to. The size of OutBuffer may be half the size of InBuffer. OutBuffer may be the same pointer as InBuffer but be aware that the original values will be lost.
PixelCount	ULONG	Specifies the number of pixels which should be converted.
BitShift	UCHAR	Specifies the number bits the pixels should be shifted to the right (divide by 2). For example InBuffer contains 12 bit data, the upper 4 bits are not used. If we want to display this data, we can only select 8 bits out of these 12. In most cases you will want to display the upper 8 bits out of these 12, then BitShift should be 4. If you don't select the upper 8 bits of the image data, then you get some sort of digital amplification but be aware that wrap around of the data may occur.

2.2 Histogram functions

2.2.1 CCU_CalculateDrawHistogram

- **Function:**

  Given a buffer with data acquired from a camera, this functions returns the histogram table and can also draw the histogram in a given hDC. The drawn histogram will always be 256 pixels wide and 192 pixels high, independent if the data is 8 bit wide or more. For higher bit-depths, the drawn histogram will average all values that fall in between 2 pixels. The function can also return the gravity-point of the histogram.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int CCU_CalculateDrawHistogram(  HWND hWnd,
                                            const void * imagebuffer,
                                            ULONG pixelcount,
                                            USHORT pixelwidth,
                                            unsigned long * histogram,
                                            long * gravity ) ;
```

Variable	C Type	Purpose
HWnd	Handle to window	Window or object in a window where the histogram should be drawn. If this value is NULL, no histogram will be drawn, only calculated.
Imagebuffer	PVOID	Pointer to the buffer which contains the image data where the histogram should be calculated over.
PixelCount	ULONG	Specifies the number of pixels in imagebuffer.
Pixelwidth	USHORT	Specifies the bit depth of the pixels in imagebuffer.
Histogram	PULONG	Pointer to the buffer where the calculated histogram will be returned in. The size of this buffer must be 2^pixelwidth.
Gravity	PLONG	Pointer to a long variable where the gravity of the histogram will be returned in. If NULL, no gravity will be returned.

2.2.2 CCU_CalculateDrawHistogramColor

- **Function:**

  Same as `CCU_CalculateDrawHistogram`. But the histogram can be calculated for a certain color component in the buffer. The data in the buffer must be formatted in color for this function to be meaningful. See color functions for more information on color formatting.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int CCU_CalculateDrawHistogram(  HWND hWnd,
                                            const void * imagebuffer,
                                            ULONG pixelcount,
                                            USHORT pixelwidth,
                                            USHORT color,
                                            unsigned long * histogram,
                                            long * gravity ) ;
```

Variable	C Type	Purpose
hWnd	Handle to window	Window or object in a window where the histogram should be drawn. If this value is NULL, no histogram will be drawn, only calculated.
imagebuffer	PVOID	Pointer to the buffer which contains the image data where the histogram should be calculated over.
PixelCount	ULONG	Specifies the number of pixels in imagebuffer.
pixelwidth	USHORT	Specifies the bit depth of the pixels in imagebuffer.
color	USHORT	The color component where the histogram should be calculated over. Possible values are : 0 for red, 1 for green, 2 for blue and 3 for luminance. The luminance is a weigted average of all color components. L = 0.3*R + 0.59*G + 0.11*B
histogram	PULONG	Pointer to the buffer where the calculated histogram will be returned in. The size of this buffer must be 2^pixelwidth.
gravity	PLONG	Pointer to a long variable where the gravity of the histogram will be returned in. If NULL, no gravity will be returned.

2.3 Color functions

2.3.1 CCU_Colorize

- **Function:**

  Converts a buffer which contains image data acquired from a camera with a color filter array (CFA) on the sensor, to another buffer and recombines the color information. The output buffer will be three times bigger then the input buffer because each pixel will be represented by the three main color values, which are gathered by the neighboring pixels. The format of the output buffer is compatible with the bitmap format, which means that for each pixel the first word represents blue, the second word represents green and the third word represents red. The colorize function supports several algorithms for color recombination depending on the used CFA pattern.

  Bayer pattern :

  $R_{11}G_{12}R_{13}G_{14}R_{15}G_{16}$ …
  $G_{21}B_{22}G_{23}B_{24}G_{25}B_{26}$
  $R_{31}G_{32}R_{33}G_{34}R_{35}G_{36}$
  $G_{41}B_{42}G_{43}B_{44}G_{45}B_{46}$
  …

  Blue pixel 22 = Avg(R11, R13, R31, R33) ; Avg(G12, G21, G23, G32) ; B22
  Green pixel 23 = Avg(R13, R33) ; Avg(G12, G14, G23, G32, G34) ; Avg(B22, B24)
  Green pixel 32 = Avg(R31, R33) ; Avg(G21, G23, G32, G41, G43) ; Avg(B22, B42)
  Red pixel 33 = R33 ; Avg(G23, G32, G34, G43) ; Avg(B22, B24, B42, B44)

  Diagonal pattern :

  $R_{11}G_{12}B_{13}R_{14}G_{15}B_{16}$ …
  $G_{21}B_{22}R_{23}G_{24}B_{25}R_{26}$
  $B_{31}R_{32}G_{33}B_{34}R_{35}G_{36}$
  …

  Blue pixel 22 = Avg(R11, R23, R32) ; Avg(G12, G21, G33) ; Avg(B13, B22, B31)
  Red pixel 23 = Avg(R14, R23, R32) ; Avg(G12, G24, G33) ; Avg(B13, B22, B34)
  Green pixel 24 = Avg(R14, R23, R35) ; Avg(G15, G24, G33) ; Avg(B13, B25, B34)

  Vertical pattern :

  $R_{11}G_{12}B_{13}R_{14}G_{15}B_{16}$ …
  $R_{21}G_{22}B_{23}R_{24}G_{25}B_{26}$ …
  $R_{31}G_{32}B_{33}R_{34}G_{35}B_{36}$ …
  …

  Green pixel 22 = Avg(R11, R21, R31) ; Avg(G12, G22, G33) ; Avg(B13, B23, B33)
  Blue pixel 23 = Avg(R14, R24, R34) ; Avg(G12, G22, G33) ; Avg(B13, B23, B33)
  Red pixel 24 = Avg(R14, R24, R34) ; Avg(G15, G25, G35) ; Avg(B13, B23, B33)

  For speed reasons, the outer circumference (1 pixel wide) of the image will not be converted and will therefor not contain valid color information. No edge exceptions are built into the algorithms.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int CCU_Colorize(  PVOID InBuffer,
                              PVOID OutBuffer,
                              USHORT PixelWidth,
                              USHORT Width,
                              USHORT Height,
                              struct CCU_ColorParms * cp ) ;
```

Variable	C Type	Purpose
InBuffer	PVOID	Pointer to the buffer that contains the gray values from an image acquired by a camera with a sensor with a CFA pattern.
OutBuffer	PVOID	Pointer to the buffer where the recombined color values will be written to. The size of this buffer should be 3 times the size of InBuffer. For each word containing a pixel in InBuffer, 3 words containing blue, green and red values will be written in OutBuffer.
PixelWidth	USHORT	Speciefies the bit depth of the pixels in InBuffer.
Width	USHORT	Specifies the width of the image in InBuffer in pixels.
Height	USHORT	Specifies the height of the image in InBuffer in pixels.
Cp	Struct CCU_ColorParms *	Structure containing parameters for recombining the color values. See definition of `struct CCU_ColorParms`.

2.3.2 CCU_CalculateWhiteBalance

- **Function:**

  Calculates the white balance parameters given a rectangular area in the image. The algorithm tries to calculate the coefficients for red and blue that make the average value of all pixels in that area a gray level. Be sure that the area marked should actually be a shade of gray. When the color temperature of the source light is unknown, this function can be used to "auto white balance" your image. Call this function before you call `CCU_Colorize`.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int CCU_CalculateWhiteBalance( PVOID ColBuffer,
                                          USHORT PixelWidth,
                                          USHORT Width,
                                          USHORT XStart,
                                          USHORT XEnd,
                                          USHORT YStart,
                                          USHORT YEnd,
                                          struct CCU_ColorParms * cp);
```

Variable	C Type	Purpose
ColBuffer	PVOID	Pointer to the buffer that contains the gray values from an image acquired by a camera with a sensor with a CFA pattern.
PixelWidth	USHORT	Speciefies the bit depth of the pixels in ColBuffer.
Width	USHORT	Specifies the width of the image in ColBuffer in pixels.
XStart	USHORT	Specifies the left edge of the rectangular area in the image.
XEnd	USHORT	Specifies the right edge of the rectangular area in the image.
YStart	USHORT	Specifies the top edge of the rectangular area in the image.
YEnd	USHORT	Specifies the bottom edge of the rectangular area in the image.
cp	Struct CCU_ColorParms *	Structure containing parameters for recombining the color values. See definition of `struct CCU_ColorParms`.

2.4 Correction functions

2.4.1 CCU_FPNCorrection

- **Function:**

  Corrects an image in a buffer that contains Fixed Pattern Noise (FPN) by subtracting the amount of FPN stored in another buffer.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_FPNCorrection( PVOID InBuffer,
                                   PVOID OutBuffer,
                                   USHORT PixelWidth,
                                   USHORT XSize,
                                   USHORT XStart,
                                   USHORT YStart,
                                   USHORT XEnd,
                                   USHORT YEnd,
                                   USHORT XInc,
                                   USHORT YInc,
                                   PVOID CorrBuffer ) ;
```

Variable	C Type	Purpose
InBuffer	PVOID	Pointer to a buffer that contains the image with the FPN.
OutBuffer	PVOID	Pointer to the buffer where the corrected data will be written to. This buffer may be the same as InBuffer, but be aware that the original data will be lost.
PixelWidth	USHORT	Speciefies the bit depth of the pixels in InBuffer and CorrBuffer.
XSize	USHORT	Specifies the width of the image in CorrBuffer.
XStart	USHORT	Specifies the left edge of the rectangular area in the image. This area will be corrected using the same area in CorrBuffer.
YStart	USHORT	Specifies the top edge of the rectangular area in the image. This area will be corrected using the same area in CorrBuffer.
XEnd	USHORT	Specifies the right edge of the rectangular area in the image. This area will be corrected using the same area in CorrBuffer.
YEnd	USHORT	Specifies the bottom edge of the rectangular area in the image. This area will be corrected using the same area in CorrBuffer.
XInc	USHORT	Specifies the increment value in the x direction.
YInc	USHORT	Specifies the increment value in the y direction.
CorrBuffer	PVOID	Pointer to the buffer that contains only the FPN which is used to subtract from InBuffer.

2.5 File operation functions

2.5.1 CCU_SaveBufferAsBMP_BW

- **Function:**

  Saves an image in the buffer as a black & white Bitmap™ file.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_SaveBufferAsBMP_BW( char * FileSpec,
                                        PVOID buffer,
                                        USHORT xsize,
                                        USHORT ysize,
                                        int flipvertical ) ;
```

Variable	C Type	Purpose
FileSpec	Char *	Pointer to a string that contains the filename with extension. A pathname may also be included.
buffer	PVOID	Pointer to the buffer containing the image to be saved.
xsize	USHORT	Width of the image.
ysize	USHORT	Height of the image.
flipvertical	int	If zero, the image is stored in the same order as in the buffer but will be displayed upside-down by Windows™. If nonzero, the image is stored upside-down but will be displayed correctly.

2.5.2 CCU_SaveBufferAsBMP_Color

- **Function:**

  Saves an image in the buffer as a color Bitmap™ file. The image data in the buffer must have the correct format. See color functions for more information on color.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_SaveBufferAsBMP_BW( char * FileSpec,
                                        PVOID buffer,
                                        USHORT xsize,
                                        USHORT ysize,
                                        int flipvertical ) ;
```

Variable	C Type	Purpose
FileSpec	Char *	Pointer to a string that contains the filename with extension. A pathname may also be included.
buffer	PVOID	Pointer to the buffer containing the image to be saved.
xsize	USHORT	Width of the image.
ysize	USHORT	Height of the image.
flipvertical	int	If zero, the image is stored in the same order as in the buffer but will be displayed upside-down by Windows™. If nonzero, the image is stored upside-down but will be displayed correctly.

2.5.3 CCU_SaveBufferAsBinary

- **Function:**

  Saves an image in the buffer in a binary file.

- **Return value:**

  Zero when operation was not successful. Nonzero if successful.

- **Syntax definition:**

```
CCUTIL_API int  CCU_SaveBufferAsBinary( char * FileSpec,
                                        PVOID buffer,
                                        ULONG size ) ;
```

Variable	C Type	Purpose
FileSpec	Char *	Pointer to a string that contains the filename with extension. A pathname may also be included.
buffer	PVOID	Pointer to the buffer containing the image to be saved.
size	ULONG	Number of bytes to be saved.

3 Structures

3.1 CCU_ColorParms

```
struct CCU_ColorParms {
            CFA             CFAType ;
            START_COLOR     StartColor ;
            double          RedLuminance, GreenLuminance, BlueLuminance ;
            double          Saturation, Brightness, Contrast ;
            } ;
```

Variable	C Type	Purpose
CFAType	Enumeration type	Determines the color filter array (CFA) used.
StartColor	Enumeration type	Determines what start color should be used when applying the color recombination algorithm. This can be useful when a WOI is used that is smaller then the size of the sensor.
RedLuminance GreenLuminance BlueLuminance	double	The coefficients used for multiplying the corresponding color components. These are also called the white balance parameters.
Saturation	double	Determines color saturation. A value of 0 lets the colors unmodified. A value between 0 and -1 makes the colors degrade to a gray level. -1 is a black and white image. A positive value amplifies the colors.
Brightness	double	Offset value that will be added to the color components when recombining color. A value of zero lets the image unmodified. A positive value makes the image brighter, a negative value makes the image darker.
Contrast	double	Value which is used to multiply all color components. A value of 1 lets the image unmodified. A value between 0 and 1 decreases the contrast, a value above 1 increases the contrast. Negative values are not meaningful.