

CcamAPI.dll software manual for CCI₄ camera users

CCamAPI.dll

Release 7 november 2000

**Software manual for CCI₄ camera
users with PCI-LVDS interface**

C-Cam Technologies

a division of

Vector International

CcamAPI.dll software manual for CCI₄ camera users

Copyright

C-Cam Technologies division of Vector International.

This document contains proprietary and confidential information of C-Cam Technologies division of Vector International

No part of this document may be translated or reproduced in any form without prior written permission from Vector International.

All rights reserved.

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. GNU is a registered trademark of the Free Software Foundation.

Restriction

This program 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>

Table of content :

TABLE OF CONTENT :	3
1. API FUNCTIONS FOR THE PCI-LVDS INTERFACE WITH CCI₄ CAMERA	5
1.1. CCamCCLoad	5
1.2. CCamClose	8
1.3. CCamGetApiVersion	10
1.4. CCamGetVersion	12
1.5. CcamOpen	14
1.6. CCamReadWOI	16
1.7. CCamSetCC	18
1.8. CCamSetWOI	22
2. HOW TO USE THE API FUNCTIONS IN SEQUENCE	25
2.1. Find the driver	25
2.2. Send logic to camera	25
2.3. Select Window Of Interest (WOI)	26
2.4. Acquisition parameters	26
2.4.1. Exposure time	27
2.4.2. Gain and Offset	27
2.4.3. Anaval	27
2.4.4. Single shot mode	27
2.4.5. Grey values images of 8-bit or 10-bit	28
2.4.6. Test image from camera	28
2.5. Read the image into the buffer	28
2.6. Help me please...	28
3. IMAGE PROJECTION	29
3.1. The Object in real	29
3.2. Object projected on imager's optical area	29

CcamAPI.dll software manual for CCI₄ camera users

3.3. Camera output image on the monitor	30
3.4. Image in the data buffer	30
4. IMAGE IN BMP FORMAT	32
4.1. Width of BMP images	32
4.2. From Buffer to black & white BMP format	32
4.3. From buffer to RGB BMP format	33
4.4. A typical colour calculation	33
APPENDIX A : TERMINOLOGY	39
APPENDIX B : CROSS REFERENCE	40

1. API functions for the PCI-LVDS interface with CCI₄ camera

1.1. CCamCCLoad

- **Function :**

To program the camera with its logic file.

- **Inverse function :** same function with other logic file.

- **Return value :** Zero when operation was successful.

- **Syntax of definition :**

- Delphi4 :

```
FUNCTION CCamCCLoad(dpointer,bufferize:integer;
                    pBuffer:pointer):integer; stdcall;
External'CCamapi.DLL';
```

Variable	Type	Purpose
Dpointer	Integer (32-bit)	Device location
bufferize	Integer (32-bit)	Size of logic buffer
pbuffer	Pointer	Pointer to logic buffer

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamCCLoad(HANDLE Cam, unsigned long size,
                                DWORD * buffer);
```

Variable	Type	Purpose
Cam	Handle	Device location
size	Unsigned long	Size of logic buffer
buffer	DWORD	Pointer to logic buffer

- Visual Basic :

```
Declare Function CCamCCLoad Lib "CCamapi.dll"
    (ByVal Cam As Long,ByVal BufferSize As Long,
     ByRef buffer As Byte) As Long
```

CcamAPI.dll software manual for CCI₄ camera users

Variable	Type	Purpose
Cam	Val. Long	Device location
Buffersize	Val. Long	Size of logic buffer
buffer	Ref. Byte	Pointer to logic buffer

- **Example code :**

- Delphi4 :

```

if FileExists(ibis4b.ttb') then // check if file exists
begin
  AssignFile(F2,'ibis4b.ttb'); // locate file
  Reset (F2); // set pointer to front of file
  Sizefile := FileSize(F2); // what is size of file ?
  For i := 0 to SizeFile-1 do // file byte by byte
    Read(F2,buffer[i]); // get info, put it in buffer
  CloseFile(F2); // close file when finished
end;

CCamCCLoad(camhandle, // load camera location
           SizeFile, // file size
           @buffer); // location buffer

```

- Microsoft C :

```

Char CCLogic[] = "ibis4b.ttb" // assign filename to variable

If((ccf = fopen(CCLogic,"rb"))==(FILE *)0) // open file
{
  DialogBox(hInst,"CC logic file missing",hWnd,NULL);
  return FALSE; // show message
}

dst = buffer; // get position
while(fread(dst++,1,1,ccf) != 0){} // get length in dst
fclose(cff); // close file when finished

CCamCCLoad(camera, dst - buffer,(DWORD *) buffer);

```

CcamAPI.dll software manual for CCI₄ camera users

- Visual Basic :

```
Open "ibis4b.ttb" For Binary Access Read As #1
i = 1
Buffer_len = LOF(1)
Do                                     ` Loop until end of file
    MyCCLogicBuffer(i) = Asc(Input(1,#1))
    i = i + 1
Loop While i <= buffer_len
Close #1

Ret = CCamCCLoad(MyCam, buffer_len, MyCCLogicBuffer(0))

If ret = 0 Then
    Call MsgBox("Could not download Cclogic !",vbOKOnly Or
vbCritical)
    End
End If
```

Notes :

In the examples we used *ibis4b.ttb*. In practice it can be a *.ttb file with any name, depending up on the camera's functionality. Check your CDROM delivered with the camera to update your program.

CcamAPI.dll software manual for CCI₄ camera users

1.2. CCamClose

- **Function :**

To close the device and de-allocates the buffers. Must be used before your windows application ends.

- **Inverse function :** CCamOpen

- **Return value :** Zero when operation was successful.

- **Syntax of definition :**

- Delphi4 :

```
FUNCTION CCamClose(dpointer:integer):integer; stdcall;
                                External'CCamapi.DLL';
```

Variable	Type	Purpose
Dpointer	Integer (32-bit)	Device location

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamClose(HANDLE Cam) ;
```

Variable	Type	Purpose
Cam	Handle	Device location

- Visual Basic :

```
Declare Function CCamClose Lib "CCamapi.dll"
                                (ByVal Cam As Long) As Long
```

Variable	Type	Purpose
Cam	Val. Long	Device location

CcamAPI.dll software manual for CCI₄ camera users

- **Example code :**

- Delphi4 :

- ```
CCamClose(Master.camHandle);
```

- Microsoft C :

- ```
CCamClose(camera);
```

- Visual Basic :

- ```
CCamClose(MyCam);
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*
**1.3. CCamGetApiVersion**

- **Function :**

Returns version number of API in ApiVersion.

- **Return value :** ApiVersion (Integer).

- **Syntax for definition :**

- Delphi4 :

```
FUNCTION CCamGetApiVersion(var ApiVersion: longint): integer;
 stdcall; External 'CCamapi.DLL';
```

| Variables  | Type    | purpose     |
|------------|---------|-------------|
| ApiVersion | longint | Api version |

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamGetApiVersion
 (DWORD * pVersionLibNumber);
```

| Variables         | Type  | Purpose     |
|-------------------|-------|-------------|
| PversionApiNumber | DWORD | Api version |

- Visual Basic :

```
Declare Function CCamGetApiVersion Lib "CCamapi.dll"
 (byVal pVersionLibNumber As long)As long
```

| Variables         | Type      | Purpose     |
|-------------------|-----------|-------------|
| PversionApiNumber | Val. Long | Api version |

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- **Example code :**

- Delphi4 :

- ```
CCamGetApiVersion(ApiVersion);
```

- Microsoft C :

- ```
CCamGetApiVersion(ApiVersion);
```

- Visual Basic :

- ```
CCamGetApiVersion(ApiVersion);
```

CcamAPI.dll software manual for CCI₄ camera users

1.4. CCamGetVersion

- **Function :**

Returns version number of driver in Version.

- **Return value :** Version (Integer).

- **Syntax for definition :**

- Delphi4:

```
FUNCTION CCamGetVersion(dpointer:integer; var Version:
longint): integer; stdcall; External 'CCamapi.DLL';
```

Variables	Type	purpose
Dpointer	integer	devices location
Version	longint	Version driver

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamGetVersion(HANDLE Cam,
DWORD * pVersionNumber);
```

Variables	Type	purpose
Dpointer	integer	devices location
PversionNumber	DWORD	Version driver

- Visual Basic :

```
Declare Function CCamGetVersion Lib "CCamapi.dll"
(ByVal Cam As Long,
ByVal pVersionNumber As Long)As Long
```

Variables	Type	purpose
Cam	Val. Long	devices location
PversionNumber	Val. Long	Version driver

CcamAPI.dll software manual for CCI₄ camera users

- **Example code :**

- Delphi4 :

- ```
CCamGetVersion(Master.CamHandle, Version);
```

- Microsoft C :

- ```
CCamGetVersion(Camera, Version);
```

- Visual Basic :

- ```
CCamGetVersion(MyCam, MyVersionNumber);
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

## 1.5. CcamOpen

- **Function :**

When the function is called, the result is an identifier of opened file or opened device. This identifier must be used to access the file of the device. All other functions refer to this identifier.

- **Inverse function :** CCamClose

- **Return value :** device identifier (Integer).

- **Syntax for definition :**

- Delphi4 :

```
FUNCTION CCamOpen(s : Pchar): integer; pascal;
 External 'CCamapi.DLL';
```

| Variables | Type  | purpose       |
|-----------|-------|---------------|
| s         | Pchar | file location |

- Microsoft C :

```
DLLINOUT HANDLE WINAPI CCamOpen(LPCSTR lpFileName);
```

| Variables  | Type   | Purpose       |
|------------|--------|---------------|
| LpFileName | LPCSTR | file location |

- Visual Basic :

```
Declare Function CCamOpen Lib "CCamapi.dll"
 (ByVal lpFileName As String) As Long
```

| Variables  | Type        | Purpose       |
|------------|-------------|---------------|
| LpFileName | Val. String | file location |

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- **Example code :**

- Delphi4 :

```
Master.camHandle := CCamOpen('\\\\.\\CCRider.VXD'); // Windows 98
```

```
Master.camHandle := CCamOpen('\\\\.\\CamDev0'); // Windows NT
```

- Microsoft C :

```
char CCamName[] = "\\\\.\\CCRider.VXD" ; // Windows 98
```

```
char CCamName[] = "\\\\.\\CamDev0" ; // Windows NT
```

```
camera = CCamOpen(CCamName); // Windows 98/NT
```

- Visual Basic :

```
MyCam = CCamOpen("\\\\.\\CCRider.vxd") ' Windows 98
```

```
MyCam = CCamOpen("\\\\.\\CamDev0') ' Windows NT
```

Notes :

In the examples we used *CCRider.vxd* and *CamDev0*. In practice it can be a file with any name with the same extension, depending up on the camera's functionality. Check your CDROM delivered with the camera to update your program.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*
**1.6. CCamReadWOI**

- **Function** : Can be used to read out the image data from the PCI card.
- **Related function** : CCamSetWOI
- **Remark** : size of buffer must be exact, else the PCI card keeps waiting to put out data.
- **Syntax of definition** :

- Delphi4 :

```
FUNCTION CCamReadWOI(dpointer:integer; pBuffer:pointer;
 bufferSize: integer; var ReturnValue: integer): integer;
 stdcall; External 'CCamapi.DLL';
```

| Variables   | Type             | purpose                      |
|-------------|------------------|------------------------------|
| Dpointer    | Integer (32-bit) | device location              |
| Pbuffer     | pointer          | pointer to image buffer      |
| BufferSize  | Integer (32-bit) | size of image buffer         |
| ReturnValue | Integer (32-bit) | Zero is successful operation |

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamReadWOI(HANDLE Cam, PVOID buffer,
 ULONG bufferSize, PULONG ReturnValue);
```

| Variables   | Type   | Purpose                      |
|-------------|--------|------------------------------|
| Cam         | Handle | device location              |
| pbuffer     | PVOID  | pointer to image buffer      |
| bufferSize  | ULONG  | size of image buffer         |
| ReturnValue | PULONG | Zero is successful operation |

- Visual Basic :

```
Declare Function CCamReadWOI Lib "CCamapi.dll"
 (ByVal Cam As Long, ByRef pBuffer As Byte,
 ByVal BufferSize As Long,
 ByRef ReturnedLength As Long) As Long
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

| Variables   | Type      | Purpose                      |
|-------------|-----------|------------------------------|
| Cam         | Val. Long | device location              |
| pbuffer     | Ref. Byte | pointer to image buffer      |
| buffersize  | Val. Long | size of image buffer         |
| ReturnValue | Ref. Long | Zero is successful operation |

- **Example code :**

- Delphi4 :

```
CCamReadWOI(Master.camHandle,
 @buffer,
 sizeof(buffer),
 returnValue);
```

- Microsoft C :

```
CCamReadWOI(camera, imgbuf, sizeof(imgbuf), &ReturnValue);
```

- Visual Basic :

```
Ret = CCamReadWOI(MyCam, MyBuffer(0), Xdim * Ydim,
MyReturnedLength);

If ret = 0 Then
 Call MsgBox(Could not read window of interest !",
 VbOKOnly Or vbCritical)
End
End If
```

## 1.7. CCamSetCC

- **Function :**

This function sets the acquisition parameters of the camera.

- **Return value :** Not applicable

- **Remarks :**

- The integration time is expressed in  $\mu\text{sec.} * 40$
- The Gain is a value between 0 & 15, typical 3.
- The Offset is a value between 0 & 15, typical 2.
- The BitsPerPixel is standard 8-bits and can be 10-bit.
- The DiagnosticType can be of the following values :
  - 0 : display x address generated by camera
  - 1 : display y address generated by camera
  - 2 : display incremental pixel value generated by PCI card
  - 3 : display x XOR y address generated by camera



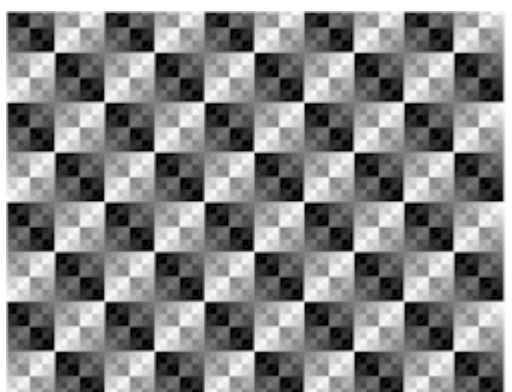
**Figure 1 :** DiagnosticType = 0



**Figure 2 :** DiagnosticType = 1



**Figure 4 :** DiagnosticType = 2



**Figure 3 :** DiagnosticType = 3

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- **Syntax of definition :**

- Delphi4 :

```

FUNCTION CCamSetCC (dpointer: integer;
 IntTime: integer;
 Gain, Offset, Anaval: Byte;
 SingleShot: boolean;
 BitsPerPixel : Byte;
 Diagnostics: Boolean;
 DiagnosticType: Byte):
 integer; stdcall; External 'CCamapi.DLL';

```

| Variables       | Type             | purpose                     |
|-----------------|------------------|-----------------------------|
| Dpointer        | Integer (32-bit) | device location             |
| IntTime         | Integer (32-bit) | Integration time            |
| Gain            | Byte             | Camera gain                 |
| Offset          | Byte             | Camera offset               |
| Anaval          | Byte             | Limited Exposure Technology |
| SingleShot      | Boolean          | Single shot or continuous   |
| BitsPerPixel    | Byte             | Grey level resolution       |
| Diagnostics     | Boolean          | Show test image             |
| DiagnosticsType | Byte             | Test image type             |

- Microsoft C :

```

DLLINOUT BOOL WINAPI CCamSetCC(HANDLE Cam,
 ULONG IntTime,
 USHORT Gain,
 USHORT Offset,
 USHORT Anaval,
 BOOL SingleShot,
 USHORT BitsPerPixel,
 BOOL Diagnostic,
 USHORT DiagnosticType);

```

| Variables  | Type   | Purpose                     |
|------------|--------|-----------------------------|
| Cam        | Handle | Device location             |
| IntTime    | ULONG  | Integration time            |
| Gain       | USHORT | Camera gain                 |
| Offset     | USHORT | Camera offset               |
| Anaval     | USHORT | Limited Exposure Technology |
| SingleShot | BOOL   | Single shot or continuous   |

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

|                |        |                       |
|----------------|--------|-----------------------|
| BitsPerPixel   | USHORT | Grey level resolution |
| Diagnostic     | BOOL   | Show test image       |
| DiagnosticType | USHORT | Test image type       |

- Visual Basic :

```

Declare Function CCamSetCC Lib "CCamapi.dll"
 (ByVal Cam As Long, ByVal IntTime As Long, ByVal Gain As
 Integer, ByVal Offset as Integer, ByRef Anaval As Byte,
 ByVal SingleShot As Long, ByVal BitsPerPixel As Integer,
 ByVal Diagnostic As Long,
 ByVal DiagnosticType As Long) As Long

```

| Variables      | Type         | Purpose                     |
|----------------|--------------|-----------------------------|
| Cam            | Val. Long    | Device location             |
| IntTime        | Val. Long    | Integration time            |
| Gain           | Val. Integer | Camera gain                 |
| Offset         | Val. Integer | Camera offset               |
| Anaval         | Ref. Byte    | Limited Exposure Technology |
| SingleShot     | Val. Long    | Single shot or continuous   |
| BitsPerPixel   | Val. Integer | Grey level resolution       |
| Diagnostic     | Val. Long    | Show test image             |
| DiagnosticType | Val. Long    | Test image type             |

- **Example code :**

- Delphi4 :

```

CCamSetCC(Master.camHandle,
 IntegrationTime*40,
 Gain,
 Offset,
 LimitedExposureTechnology,
 SingleShotMode,
 BitsPerPixel,
 ShowDiagnostics,
 DiagnosticImageType);

```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- Microsoft C :

```
CCamSetCC(camera,
 IntegrationTime*40,
 Gain,
 Offset,
 LimitedExposureTechnology,
 SingleShotMode,
 BitsPerPixel,
 ShowDiagnostics,
 DiagnosticImageType);
```

- Visual Basic :

```
CCamSetCC(MyCam, 'camera reference
 IntTime*40, 'exposure time
 Gain, 'gain
 Offset, 'offset
 MyAnaval(0), 'set well depth
 1, 'single shot
 8, '8-bits output
 0, 'no test image
 0); 'test image number zero
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

## 1.8. CCamSetWOI

- **Function :**

As the CCI<sub>4</sub> camera is able to read sub frames out the complete sensor area, we must define this frame size before we read out the pixel information. This can be done with CCamSetWOI. The information can be accessed with CCamReadWOI.

- **Related function :** CCamReadWOI

- **Return value :** Not applicable

- **Remarks :**

- The maximum value of Xend and Yend is limited with the size of the sensor. E.g. : a sensor with size 1280 x 1024 will give for a full frame : XStart = 0, Ystart = 0, Xend = 1279 and Yend = 1023.
- The Ibis4 sensor in the CCI<sub>4</sub> camera can not be used with an increment (sub sample resolution always = 1).

- **Syntax of definition :**

- Delphi4 :

```
FUNCTION CCamSetWOI(dpointer, YStart, YEnd, YInc, XStart, XEnd,
 XInc:integer):INTEGER; stdcall; External 'CCamapi.DLL';
```

| Variables | Type             | Purpose                  |
|-----------|------------------|--------------------------|
| dpointer  | Integer (32-bit) | device location          |
| Ystart    | Integer (32-bit) | y of first pixel         |
| Yend      | Integer (32-bit) | y of last pixel          |
| Yinc      | Integer (32-bit) | y increment (resolution) |
| Xstart    | Integer (32-bit) | x of first pixel         |
| Xend      | Integer (32-bit) | x of last pixel          |
| Xinc      | Integer (32-bit) | x increment (resolution) |

Note: Yinc and Xinc are not implemented for the CCI<sub>4</sub> camera. Set these values equal to 1.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- Microsoft C :

```
DLLINOUT BOOL WINAPI CCamSetWOI(HANDLE Cam, USHORT YStart,
USHORT YEnd, USHORT YInc, USHORT XStart, USHORT XEnd, USHORT
XInc);
```

| Variables | Type   | purpose                  |
|-----------|--------|--------------------------|
| Cam       | HANDLE | device location          |
| Ystart    | USHORT | y of first pixel         |
| Yend      | USHORT | y of last pixel          |
| Yinc      | USHORT | y increment (resolution) |
| Xstart    | USHORT | x of first pixel         |
| Xend      | USHORT | x of last pixel          |
| Xinc      | USHORT | x increment (resolution) |

Note: Yinc and Xinc are not implemented for the CCI<sub>4</sub> camera. Set these values equal to 1.

- Visual Basic :

```
Declare Function CCamSetWOI Lib "CCamapi.dll" (ByVal Cam As Long,
ByVal Ystart As Integer, ByVal Yend As Integer, ByVal Yinc As
Integer, ByVal Xstart As Integer, ByVal Xend As Integer, ByVal
Xinc As Integer) As Long
```

| Variables | Type         | purpose                  |
|-----------|--------------|--------------------------|
| Cam       | Val. Long    | device location          |
| Ystart    | Val. Integer | y of first pixel         |
| Yend      | Val. Integer | y of last pixel          |
| Yinc      | Val. Integer | y increment (resolution) |
| Xstart    | Val. Integer | x of first pixel         |
| Xend      | Val. Integer | x of last pixel          |
| Xinc      | Val. Integer | x increment (resolution) |

Note: Yinc and Xinc are not implemented for the CCI<sub>4</sub> camera. Set these values equal to 1.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

- **Example code :**

- Delphi4 :

```
CCamSetWOI(Master.camhandle, // device location
 y_start, // y of first pixel
 y_end, // y of last pixel
 1, // sub sample resolution y
 x_start, // x of first pixel
 x_end, // x of last pixel
 1); // sub sample resolution x
```

- Microsoft C :

```
CCamSetWOI(camera, // device location
 YStart, // y of first pixel
 YEnd, // y of last pixel
 1, // sub sample resolution y
 XStart, // x of first pixel
 XEnd, // x of last pixel
 1); // sub sample resolution x
```

- Visual Basic :

```
Ret = CCamSetWOI(MyCam, ` device location
 YStart, ` y of first pixel
 YEnd, ` y of last pixel
 1, ` sub sample resolution y
 XStart, ` x of first pixel
 XEnd, ` x of last pixel
 1); ` sub sample resolution x

if ret = 0 then
 Call MsgBox("Could not set window of interest !",
 vbOKOnly Or vbCritical)
 End
End If
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

## 2. How to use the API functions in sequence

Before you start using the API, make sure a camera is connected to your computer. If not, the API will start looking for the camera and never stops doing this with as result a blocked PC locked in a busy loop.

You also must make sure the driver is installed. If not, blue Windows screens will appear on the screen as soon as you make use of any API function.

Below, we will explain the sequence with an example for Windows 98 and Delphi. The sequence is identical for other programming languages.

### 2.1. Find the driver

First of all, check of the *CCRider.vxd* is available in the *c:\Windows\system* directory. If it is available, use the function *CCamOpen* to get the camera handle, which is needed to use the camera with all the other API functions.

```
CamHandle := 0;
camHandle := CCamOpen('\.\CCRider.vxd');
```

If the *camHandle* remains zero, no connection with the camera was made. In that case, check the driver. Look in your Windows Control Panel under System. Under Device Manager, you will find under other devices the driver for the camera.

### 2.2. Send logic to camera

Secondly, we got to send the logic to the camera. This is the internal program of the camera. Every time the power of the camera was disconnected, the camera loses its internal logic. B

This is done with the function *CCamCCLoad* from the API. The logic can be found in the \*.ttb file. To do this, we read the \*.ttb file into a buffer. E.g.:

```
if FileExists('ibis4b.ttb') then // check if file exists
begin
 AssignFile(F2,'ibis4b.ttb'); // locate file
 Reset (F2); // bring pointer to front of file
 Sizefile := FileSize(F2); // what is the size of the file ?
 For i := 0 to Sizefile-1 do // file byte by byte
 Read(F2,buffer[i]); // get the info and put it in buffer
 CloseFile(F2); // close file when finished
end;
```

This is done, knowing that *F2* is a File of Byte, and *buffer* is a linear array of byte with a length more than 1280 x 1024. If the buffer contains the information, it can be loaded into the camera. E.g.:

### *CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

```
CCamCCLoad(camhandle, // load camera location
 SizeFile, // file size
 @buffer); // information of location buffer
```

When this is done, the camera should be ready to use.

## 2.3. Select Window Of Interest (WOI)

Thirdly, we should tell the camera which part of the image is needed. This is done with the API function `CCamSetWOI`. e.g.:

```
CCamSetWOI(camhandle, // location camera
 ystart, // y of the first pixel
 yend, // y of the last pixel
 1, // must be 1 for Ibis sensor
 xstart, // x of first pixel
 xend, // x of last pixel
 1); // must be 1 for Ibis sensor
```

As the Ibis 4 sensor has 1280 x 1024 pixel, the x-value must be between 0 and 1279. The y-value must be between 0 and 1023.

Another thing must be taken in consideration. If your image display is using a BMP-like format, the x-length of the window must be a multiple of 4. If not, the BMP format will show an image in italic. If you are using an RGB sensor, multiples of 3 must be taken in account. For this reason, RGB BMP images are most easy to use in multiples of 12. The programmer must also check the co-ordinates are given in the correct order.

## 2.4. Acquisition parameters

Fourthly, we must give the camera its acquisition parameters. This is done with the function `CCamSetCC`. E.g.:

```
CCamSetCC(camhandle, // location camera
 ExposureTimeInMiliseconds * 40, // exposure time
 Gain, // gain between 0 and 5, advised: 3
 Offset, // offset between 0 and 5, advised: 3
 Addr(Anaval), // full well parameters of pixel
 True, // always single shot mode
 8, // give a 8 or 10-bit image
 true, // true if a test image is needed,
 // false if a camera image is needed
 3); // type of test image between 0 and 3
```

Now we will explain in more detail the input parameters.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

#### 2.4.1. Exposure time

The exposure time is the integration time of the pixel, expressed in  $\mu$ seconds. It is the time between the first reset and opening of the row and the readout or closing of the row.

For a 10 MHz camera, we multiply the value with 40 to be correct.

The camera reads the sensor from top to bottom, in the selected area.

#### 2.4.2. Gain and Offset

The Gain parameter must be taken in relation with the offset parameter. It is used to regulate the analogue signal going into the ADC converter to get the digital signal. The ADC needs an input signal between 2 and 4 Volts. The Offset is used to push the signal from 0 to 2 Volts. With an offset value of 3, the 2 Volts are reached as lower threshold of 2 Volts, which is also the advised value. The Gain parameter is used to stretch the signal between 2 and 4 Volts. With a Gain value of 3, the 4 Volts upper threshold is reached. For this reason, a Gain value of 3 is the advised figure.

The user is free to set those values differently for one reason or another. In that case values between 0 and 5 are acceptable, but higher values are allowed. If a gain of 5 is needed, it is better to enlarge the integration time instead or to use more light on the scene to be visualised. The higher the gain value, the more noise will be visible in the image.

#### 2.4.3. Anaval

Only Anaval 0 and 1 are currently in use. 2 and 3 are meant for future use. They regulate the full well depth of the sensor.

The pixel must have an operation voltage between 2.5 Volts and 5.7 Volts to have an acceptable well depth. With a value of 64 about 5 Volts or a well depth of 60 000 is reached.

The value can be chosen by the user to see an effect in the image, depending up on the scene to be visualised.

#### 2.4.4. Single shot mode

The single shot mode is mean to receive image by image. Every time you call the function CCamReadWOI, one image is placed in the buffer. For this, the boolean value must be set to true.

A continues mode is meant for future use.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users***2.4.5. Grey values images of 8-bit or 10-bit**

To display in Windows BMP format, 8-bit images are required. The interface card will convert the ADC into an 8-bit format. Due to this, parts of the signal are not shown. Only the highest 8-bits of the 10-bits are given. The image buffer will contain 8-bit data.

For more detailed images, a 10-bit mode is available. Only in this case the full data is given. The 10-bit data is stored in a 16-bit right aligned format.

**2.4.6. Test image from camera**

The camera has a feature, which can help programmers to test the software and the correct operation of the camera: a set of test images.

If the test image parameter is set, all other parameters of this function are overruled. If the parameter is set to false, the sensor image is shown. Only the type of test image must be selected, a value between 0 and 3.

**2.5. Read the image into the buffer**

Fifthly, we should get the image from the camera. Once the function CCamReadWOI is activated, an image is take and placed into the buffer. E.g.:

```
CCamReadWOI(CamHandle, // camera location
 @buffer, // where to find the image to process
 ImageSize, // height * width
 Return); // how many bytes were transferred.
```

At the moment the CCamReadWOI is called, an image is made with the preset parameters and placed into the buffer. All the other API functions do not need to be repeated when no change in acquisition parameters or Window Of Interest is to be made.

For a 8-bit black & white image, the serial buffer can be defined as :

```
Buffer : array [0..1310720] of byte;
```

**2.6. Help me please...**

Free help is only an email away. If your application does not work, we might be able to help you. If you encounter a problem, we might have seen it before you and found a solution for it. Your question will help us updating the documentation.

If you need help or you got remarks, contact us between 9 and 15 hours GMT for telephone support on the number +32 (0)16 40 20 16. Send us a Fax on the number +32 (0)16 40 03 23. Support will be given on the following email address: [dirk@vector-international.be](mailto:dirk@vector-international.be)

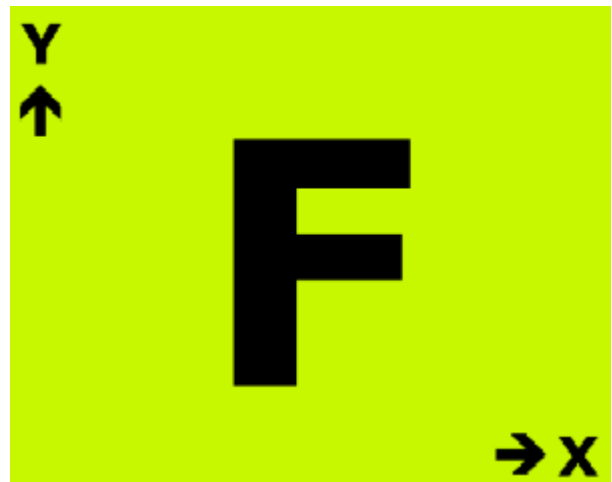
### 3. Image projection

The co-ordinate system is relative, if no reference is defined. Below we explain how an image is positioned as it is processed from the optical plain into the image buffer.

#### 3.1. The Object in real

If an object is standing in front of the camera, a man standing next to the camera would see the object as shown in this paragraph.

In real, we use a positive co-ordinate system. Due to the processing of the data, this co-ordinate system is changed through the process. This is the result of the optics, the point of view and the software.

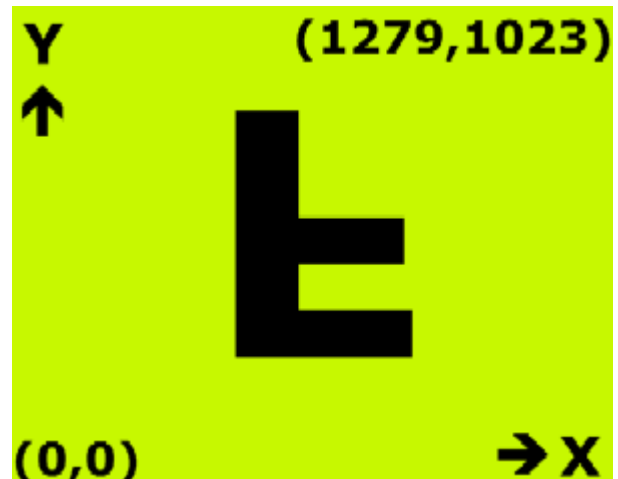


#### 3.2. Object projected on imager's optical area

The optics project the image differently on the imager. If the person standing next to the camera turns his back and looks at the front of the optical area, he would see the image projected as shown in this paragraph.

Two rotations are made here. First round the X and the Y as due to the optics, secondly round the Y again due to the rotation of the person.

If he takes a closer look at the sensor, he will see the ADC converter of the chip below the sensitive area on the right.

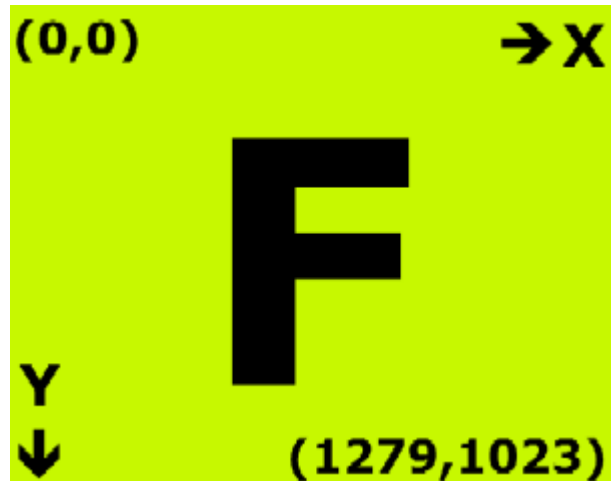


*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

### 3.3. Camera output image on the monitor

A programmer in a Windows environment would see the image on the monitor as shown in this paragraph.

Notice that BMP files must be rotated round X not to get an image in the upside down position. For more information about the BMP format, we suggest to read Chapter 4.



### 3.4. Image in the data buffer

Once the image is made, the data is dumped in a buffer defined by the programmer. According to the programmer's language, a buffer definition could look like this:

- Delphi4:

*Type*

*MonochromeBuffer = **array**[0..1310720] of byte;*

*Var*

*Buffer : MonochromeBuffer;*

- Microsoft C:

*Unsigned char buffer[1024 \* 1280 + 1280];*

- Visual Basic:

*Const Xdim As Long = 1280*

*Const Ydim As Long = 1024*

*Dim MyBuffer(Xdim \* Ydim) As Byte*

The buffer is a sequential array in one dimension with a minimum size of 1024 x 1280. It is allowed to make it larger if additional calculations require doing so.

This means pixel (0,0) is the first pixel and pixel (1279,1023) is the last pixel, in case full size images are processed.

|     |     |     |     |     |     |           |           |           |           |
|-----|-----|-----|-----|-----|-----|-----------|-----------|-----------|-----------|
| 0,0 | 1,0 | 2,0 | 3,0 | 4,0 | ... | 1276,1023 | 1277,1023 | 1278,1023 | 1279,1023 |
|-----|-----|-----|-----|-----|-----|-----------|-----------|-----------|-----------|

The numbers listed in the cells above are the (x,y) pixel positions on the sensor. We start counting from zero for the first pixel.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

A sequential buffer of a full resolution image would look like this:

|                      |                      |                      |                      |                      |     |                         |                         |                         |                         |
|----------------------|----------------------|----------------------|----------------------|----------------------|-----|-------------------------|-------------------------|-------------------------|-------------------------|
| 0<br>(0,0)           | 1<br>(1,0)           | 2<br>(2,0)           | 3<br>(3,0)           | 4<br>(4,0)           | ... | 1276<br>(1276,0)        | 1277<br>(1277,0)        | 1278<br>(1278,0)        | 1279<br>(1279,0)        |
| 1280<br>(0,1)        | 1281<br>(1,1)        | 1282<br>(2,1)        | 1283<br>(3,1)        | 1284<br>(4,1)        | ... | 2556<br>(1276,1)        | 2557<br>(1277,1)        | 2558<br>(1278,1)        | 2559<br>(1279,1)        |
| 2560<br>(0,2)        | 2561<br>(1,2)        | 2562<br>(2,2)        | 2563<br>(3,2)        | 2564<br>(4,2)        | ... | 3836<br>(1276,2)        | 3837<br>(1277,2)        | 3838<br>(1278,2)        | 3839<br>(1279,2)        |
| ...                  | ...                  | ...                  | ...                  | ...                  | ... | ...                     | ...                     | ...                     | ...                     |
| 1308 160<br>(0,1022) | 1308 161<br>(1,1022) | 1308 162<br>(2,1022) | 1308 163<br>(3,1022) | 1308 164<br>(4,1022) | ... | 1309 336<br>(1277,1022) | 1309 337<br>(1277,1022) | 1309 338<br>(1278,1022) | 1309 339<br>(1279,1022) |
| 1309 440<br>(0,1023) | 1309 441<br>(1,1023) | 1309 442<br>(2,1023) | 1309 443<br>(3,1023) | 1309 444<br>(4,1023) | ... | 1310 716<br>(1276,1023) | 1310 717<br>(1277,1023) | 1310 718<br>(1278,1023) | 1310 719<br>(1279,1023) |

The black numbers are the buffer cell numbers, the smaller blue numbers are the pixel positions on the sensor.

If a window of interest is taken, e.g. (100,100) to (200,200) the pixels are listed sequentially the first pixel being (100,100) and the last pixel being (200,200). All other pixels not within the window of interest are not presented in the buffer. This means the length of the buffer can be made dynamic.

|                |                |                |                |                |     |                  |                  |                  |                  |
|----------------|----------------|----------------|----------------|----------------|-----|------------------|------------------|------------------|------------------|
| 0<br>(100,100) | 1<br>(101,100) | 2<br>(102,100) | 3<br>(103,100) | 4<br>(104,100) | ... | 396<br>(200,197) | 397<br>(200,198) | 398<br>(200,199) | 399<br>(200,200) |
|----------------|----------------|----------------|----------------|----------------|-----|------------------|------------------|------------------|------------------|

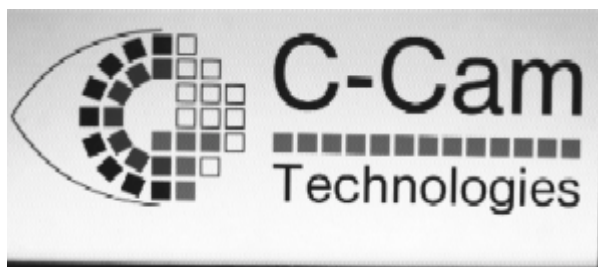
As can be noticed, the API will always place the window of interest in the beginning of the buffer, which will speed-up the process.

## 4. Image in BMP format

As the image is in the buffer, the work of the camera, driver and API are finished. The buffer can be use as main work area. But it might be interesting to show the results on screen.

### 4.1. Width of BMP images

The buffer will list exact the requested pixels of the window of interest. Notice that a BMP format needs in the x dimension a multiple of 4. If neglected, the image will be displayed in an italic shape. At the right, we show a BMP image with an original length of 1271 pixels in the x dimension.



If we ad on column to the window of interest we have 1272 pixels, which can be divided by 4. The result is the image on the left. Now the image is displayed as it should. Windows BMP wants multiples of 4 for RGB and on blank (for future use). Programmers must mind for these effects.

When RGB images are processed, the length must be checked for two reasons: multiples of 4 for the BMP format and multiples of 3 for the RGB calculations. The last one can be avoided, but in that case the routines will become more complex and slower. For this reason, we advice to choose for multiples of 12, starting at a certain colour for which the RGB routines are written.



### 4.2. From Buffer to black & white BMP format

To make sure the full grey-level resolution is available from the video card, make sure your computer is working with 24-bit true colour. If not, the image will be showing contours around objects at places not to be expected.

To read image data in a black & White BMP format with 256 grey-levels, the windows function *setbitmapbits* can be used. This function can be called in Delphi, Visual C and Visual Basic. Below, we give an example for Delphi.

```
Setbitmapbits(image1.Picture.Bitmap.handle, image_size, @buffer);
```

Make sure your BMP image is defined as a black & white 8-bit. If not, a totally wrong image will be visible on the screen. We advice you to look deeper into our example programs.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

### 4.3. From buffer to RGB BMP format

To make sure the full grey-level resolution is available from the video card, make sure your computer is working with 24-bit true colour. If not, the image will be showing contours in wrong colours around objects at places not to be expected.

To read image data in a RGB BMP format with 3x256 grey-levels, the windows function *StretchDIBits* can be used. This function can be called in Delphi, Visual C and Visual Basic. Below, we give an example for Delphi.

```

StretchDIBits(Image1.canvas.Handle, // image reference
 0,0, // destination origin
 Xend + 1 - XStart, // destination end
 Yend + 1 - Xstart, // destination end
 0,0, // source origin
 Xend + 1 - XStart, // destination end
 Yend + 1 - Xstart, // destination end
 @RGBbuffer, // data buffer
 TbitmapInfo(BitmapHeader^), // BMP header info
 DIB_RGB_COLORS, // RGB settings
 SRCCOPY); // function

```

In this case, the RGBbuffer is also a linear buffer, minimum 3 times the size of the black & white version.

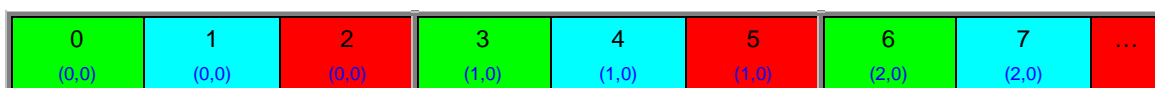
```

Type
 ColourBuffer = array [0..3932160] of byte;

Var
 RGBbuffer : ColourBuffer;

```

Notice the colours are to be placed in reverse RGB-order and each byte is only 1/3 of a pixel.



### 4.4. A typical colour calculation

The CCI<sub>4</sub> camera can be equipped with two kinds of RGB patterns: diagonal and Bayer. As much information is available for the classic Bayer pattern, we explain in this example the diagonal pattern.

Although we say the sensor has a resolution of 1280 x 1024, 3 extra rows and/or columns are available at each side of the sensor. C-Cam has chosen to start with a green pixel as first pixel. All other available pixels are defined by this colour.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

The diagonal pattern looks like this:

|               |               |               |               |               |               |               |               |     |
|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|-----|
| 0<br>(0,0)    | 1<br>(1,0)    | 2<br>(2,0)    | 3<br>(3,0)    | 4<br>(4,0)    | 5<br>(5,0)    | 6<br>(6,0)    | 7<br>(7,0)    | ... |
| 1280<br>(0,1) | 1281<br>(1,1) | 1282<br>(2,1) | 1283<br>(3,1) | 1284<br>(4,1) | 1285<br>(5,1) | 1286<br>(6,1) | 1287<br>(7,1) | ... |
| 2560<br>(0,2) | 2561<br>(1,2) | 2562<br>(2,2) | 2563<br>(3,2) | 2564<br>(4,2) | 3836<br>(5,2) | 3837<br>(6,2) | 3838<br>(7,2) | ... |
| 3840<br>(0,3) | 3841<br>(1,3) | 3842<br>(2,3) | 3843<br>(3,3) | 3844<br>(4,3) | 3845<br>(5,3) | 3846<br>(6,3) | 3846<br>(7,3) | ... |
| ...           | ...           | ...           | ...           | ...           | ...           | ...           | ...           | ... |

The colour of a pixel can be defined with the following formulae (example in delphi4):

```
Position := (3 + x mod 3 - y mod 3) mod 3;
```

```
Case position of
```

```
0 : ColorOfPixel := Green;
```

```
2 : ColorOfPixel := Red;
```

```
else ColorOfPixel := Blue;
```

```
end;
```

There are several methods to reconstruct the RGB out of the grey-level data. We give you one simple example, which is reasonable fast and gives a relative good image quality.

Lets look at the first 3 x 3 pixels. We see that all colours are represented equal in amount. We can assume this group of pixels is a good representation of the area. Lets call this group of pixels a pixel cell.

|               |               |               |
|---------------|---------------|---------------|
| 0<br>(0,0)    | 1<br>(1,0)    | 2<br>(2,0)    |
| 1280<br>(0,1) | 1281<br>(1,1) | 1282<br>(2,1) |
| 2560<br>(0,2) | 2561<br>(1,2) | 2562<br>(2,2) |

Out of this pixel cell, we will calculate the RGB value of one pixel.

This can be done in two ways. Or we calculate the mean value of each colour, or we pick the median value out of 3 equal colours. We might expect a better representation if we took the mean value, but a test will show that the median value gives a sharper image with smaller edge errors. The median value also has one advantage. If one of the 3 pixels is a hot or cold pixel (defect), it is filtered out which is an advantage. With a mean value the defect stays visible.

A mean filter would simply look like:

```
Function me3(a,b,c:integer):integer;
```

```
Begin
```

```
Me3 := trunc((a+b+c)/3);
```

```
End;
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

A median filter can be defined as follows:

```
Function M3(a,b,c : integer):integer;
Var m : integer;
Begin
 If a > b then
 Begin
 m := b; b := a; a := m;
 End;
 If a > c then
 Begin
 m := c; c := a; a := m;
 End;
 If b > c then m3 := c else m3 := b;
End;
```

The result of the first pixel cell can be calculated as follow:

```
RGBbuffer[j] := m3(buffer[i+1],
 buffer[i+width+2],
 buffer[i+2*width]) ; // Blue
RGBbuffer[j+1] := m3(buffer[i],
 buffer[i+width+1],
 buffer[i+2*width+2]) ; // Green
RGBbuffer[j+2] := m3(buffer[i+2],
 buffer[i+width],
 buffer[i+2*width+1]) ; // Red
```

Now we move one pixel cell to the right (still taking the first pixel as reference) for the calculation of the next RGB pixel (1,0). Blue has now taken the place of green, red has taken the place of blue and green has taken the place of red.

|               |               |               |
|---------------|---------------|---------------|
| 1<br>(1,0)    | 2<br>(2,0)    | 3<br>(3,0)    |
| 1281<br>(1,1) | 1282<br>(2,1) | 1283<br>(3,1) |
| 2561<br>(1,2) | 2562<br>(2,2) | 2563<br>(3,2) |

The result of the second pixel cell can be calculated as follow:

```
RGBbuffer[j+3] := m3(buffer[i+1],
 buffer[i+width+2],
 buffer[i+2*width+3]) ; // Blue
RGBbuffer[j+4] := m3(buffer[i+3],
 buffer[i+width+1],
 buffer[i+2*width+2]) ; // Green
RGBbuffer[j+5] := m3(buffer[i+2],
 buffer[i+width+3],
 buffer[i+2*width+1]) ; // Red
```

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

Now we moved to the third pixel cell to the right (still taking the first pixel as reference) for the calculation of the next RGB pixel (2,0). Red has now taken the place of green, green has taken the place of blue and blue has taken the place of red.

The result of the second pixel cell can be calculated as follow:

```

RGBbuffer[j+6] := m3(buffer[i+4],
 buffer[i+width+2],
 buffer[i+2*width+3] ; // Blue
RGBbuffer[j+7] := m3(buffer[i+3],
 buffer[i+width+4],
 buffer[i+2*width+2]) ; // Green
RGBbuffer[j+8] := m3(buffer[i+2],
 buffer[i+width+3],
 buffer[i+2*width+4]) ; // Red

```

|               |               |               |
|---------------|---------------|---------------|
| 2<br>(2,0)    | 3<br>(3,0)    | 4<br>(4,0)    |
| 1282<br>(2,1) | 1283<br>(3,1) | 1284<br>(4,1) |
| 2562<br>(2,2) | 2563<br>(3,2) | 2564<br>(4,2) |

If we would take the next pixel cell to calculate pixel (3,0), we notice we have the same structure of the first pixel cell. We have now a loop that can be repeated over the complete sensor area. If you enlarge the *i*-parameter with 3 and the *j*-parameter with 9, the loop can run until the *i*-parameter is larger than the total amount of pixels minus the 3 last lines.

#### Advantages:

- Relatively simple and fast routine.
- Bad pixels are corrected.
- Edges are corrected relatively smooth.
- The routine is useful for smaller windows of interest if multiples of 12 pixels in width are read out. In height one must make sure the first pixel in the array has the correct starting colour.

#### Optimisation:

- This example shows an 8-bit image RGB calculation. If the RGB calculations are made in 10-bit and then a histogram stretch is made on the image to 8-bit, less saturated zones are to be expected.
- The colours are to be corrected in function of the surrounding light other parameters like offset, gain and anaval. Luminance and chrominance can be enhanced to get more vivid colours.
- This routine loses some of its resolution. About 2/3 of the resolution is still available.
- In this routine the right edge is not corrected. Separate routines with the same philosophy can ad this right column.
- Ad similar routines for another starting colour: blue and red.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

The image above was made with the routines described above. It was taken with office TL-lights. A small colour enhancement was made : red + 19%, green + 4% and blue + 25%. Acquisition parameters : exposure time = 80000, Anaval 0&1 = 0, Gain = 3 and offset = 3.

Notice the nearly saturation of the white area below. This can be enhanced with higher Anaval values and a better regulation of the gain and offset.

Although the colours are real, the yellow could be made better. For this a gamma correction is advised.



*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

If we over-enlarge the image and look in more detail to the edges, the diagonal pattern stays slightly visible in slanting areas. At vertical and horizontal edges, the change is perfect. Depending up on the kind of algorithm and the post-processing, this effect could be enhanced.

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

## **Appendix A : Terminology**

- WOI : Window of interest. The part of the sensor we read
- Sub sample resolution : (= increment) the resolution of the image. E.g . : 1 = full resolution, 2 = half resolution, 3 = one third resolution,... This is not to be compared to binning with a CCD. Sub sample resolution only reads out single pixels, no binning of pixels.
- RGB : Red, Green, Blue
- Pixel cell : group of pixels, e.g.: 3 x 3 pixels

## Appendix B : Cross reference

### A

acquisition, 26  
ADC, 28  
Anaval, 27

### B

binning, 39  
BitsPerPixel, 18  
BMP, 32

### C

CCamCCLoad, 5, 25  
CCamClose, 5, 8, 14  
CCamGetLibVersion, 10  
CCamGetVersion, 12  
CCamOpen, 5, 8, 14, 25  
CCamReadWOI, 16, 22, 27, 28  
CCamSetCC, 18, 26  
CCamSetWOI, 16, 22, 26  
close, 8  
colour calculation, 33  
colour of a pixel, 34  
Contact address, 2  
Copyright, 2

### D

data buffer, 30  
device identifier, 14  
DiagnosticType, 18  
diagonal pattern, 33  
Disclaimer, 2  
Distribution, 2

### E

exposure time, 27

### F

first pixel, 30

### G

gain, 18, 27

### H

help, 28

### I

increment, 22, 39

### L

last pixel, 30

### M

maximum value, 22  
mean, 34  
median, 34

### O

offset, 27

### P

pixel cell, 34

### R

read image, 16  
Restriction, 2  
RGB calculations, 32  
RGBbuffer, 33

### S

sequence, 25  
setbitmapbits, 32  
single shot mode, 27  
size of buffer, 16  
StretchDIBits, 33  
sub frames, 22  
Sub sample resolution, 39

### T

test images, 28  
Trademarks, 2  
ttb, 6

### V

version number, 10, 12  
VXD, 15

*CcamAPI.dll software manual for CCI<sub>4</sub> camera users*

**W**

Window of interest, 39

**X**

Xend, 22  
XStart, 22

**Y**

Yend, 22  
Ystart, 22