PERFECTION IN AUTOMATION
A MEMBER OF THE ABB GROUP

# Vision Demo Application

Version 5.15
Date: 1/10/2021

## Contents

# 1 Introduction

This sample application can be used as a starting point for vision applications. Google Chrome browser must be used for the mappView project; other browser may not show the correct crosshair position.

## 1.1 System requirements

This sample was developed and tested with Automation Studio 4.9

- PLC OS system C4.90 or higher
- mappView 5.15
- Hardware files for camera (VSS112Q22.041P-000 was used in this sample)
- Chrome Browser

# 2 Project Files

The following project files are vision sensor related.

## 2.1 Logical View

All tasks starting with Vi_ should **not** be changed to make updating easier.

| | |
|---|---|
| Vision | Package with vision tasks |
| Vi_main | This task handles functions that are sensor related |
| Vi_visu | This is a helper task for the visualization |
| Vi_light | This is a helper task for the backlights and bar lights |
| Vi_nettime | This task handles the nettime calculation |
| Vi_image | This task handles everything that is image related |
| Axis | Sample task for an axis, needed for nettime handling |
| YourTask | Customer specific task |
| setRouteToCamera | Make sure to adjust the IP address in the file "\Vision_1\Logical\Vision\setRouteToCamera.bat" and execute the batch file in Windows with right click (Run as administrator). Otherwise, the sensor image does not work in the Vision Cockpit or the demo visualization. |
| mappView | mappView demo visualization for vision |
| mappRecipe | Stores the camera configuration |

## 2.2 Configuration View

| | |
|---|---|
| mappView | mappView visualization for vision |
| mappVision | mappVision configuration for vision functions |
| mappService | Configuration for recipe management |

## 2.3 Physical View

| | |
|---|---|
| Blob | Sensor for the blob function. Powerlink Node 1. |
| Measurement | Sensor for the edge measurement function. Powerlink Node 2. |
| CodeRead | Sensor for the code reader function. Powerlink Node 3. |
| Match | Sensor for the match function. Powerlink Node 4. |
| OCR | Sensor for the text recognition function. Powerlink Node 5. |
| PixelCounter | Sensor for the text pixel counter function. Powerlink Node 6. |

This is an example configuration with one camera for each vision function. You can quickly switch between the different vision functions by changing the node number.

# 3 Constants

The project provides several constants to adjust the configuration.

| Name | Default | Description |
|---|---|---|
| MAX_NUM_CAMS | 6 | Maximum number of camera's |
| MAX_NUM_LIGHTS | 5 | Maximum number of light's |
| MAX_NUM_RESULTS | 10 | Maximum number of results |
| MAX_NUM_PRODUCTS | 8 | Maximum number of products for color detection |
| MAX_IDX_VA_LIST | 19 | Maximum number of entries in the vision application list |
| DEVICE_NAME | VisionFileDevice | File device where the images and recipes are stored |
| NETTIME_DEFAULT_DELAY | 30000 | This delay is used when software trigger is used and nettime is enabled |
| MAX_NUM_CODETYPES | 69 | Maximum number of code types |
| CodeTypes | | Codes types for code reading |

# 4 Parameter structures

The sample supports multiple cameras but only one is displayed at a time. The global structures begin with a “g” for (gVisionSensor, gVisionLight, gBlob, gMT, gCodeReader, gMatch, gOCR). The global structures are arrays where the index represents the camera index. The variable “gSelectedSensor” and “gSelectedLight” map one of the global structures to the dynamic local variable in the task “Vi_visu”. If the task “Vi_visu” is not used the global select variables must be set manually.

## 4.1 Vision sensor structure (gVisionSensor)

The vision structure handles all functions and parameters that are sensor related and are independent from the vision function used.

| | | |
|---|---|---|
| CMD | | Command structure to trigger an action |
| | ImageTrigger | Start a new image aquisition |
| | ImageTriggerReset | Abort future image acquisition (with TriggerDelay) |
| | AutoSetupStartStop | Start and stop automatic camera setup |
| | VaSwitchApplication | Switch to different vision application |
| | SaveDiagData | Save diagnostic data |
| | ReadCameraInfo | Read camera information |
| CFG | | Sensor configuration |
| | VisionFunction | The type of vision function that is used with the sensor. This information is used to add the correct detail information on the main page. |
| | PowerlinkNode | The Powerlink node number for this camera. This information is used to generate the correct IP address for the camera. |
| | DataStructure | Pointer to the vision speccific function. (gBlob, gMT, gCodeReader, gMatch, gOCR) |
| | ComponentLink | Vision component name defined under mappVision in the configuration view |
| | … | All other parameters, see camera manual for details |
| DAT | | Sensor data, see manual for details |
| OPT | | Optics data, see manual for details |
| HW | | Sensor hardware information, see manual for details |

## 4.2 Functional structure

Each vision function has its own structure (gBlob, gMT, gCodeReader, gMatch, gOCR).

## 4.3 Vision light structure (gVisionLight)

The light structure handles all functions and parameters that are light related..

| | | |
|---|---|---|
| CMD | | Command structure to trigger an action |
| | FlashTrigger | Trigger flash light |
| | FlashTriggerReset | Abort flash light |
| CFG | | Sensor configuration |
| | LightType | The type of light used (None, Backlight, Lightbar). |
| | PowerlinkNode | The Powerlink node number for this light. |
| | … | All other parameters, see light manual for details |
| DAT | | Light data, see manual for details |
| HW | | Light hardware information, see manual for details |

## 4.4 Vision color structure

The color structure handles all functions and parameters that are related to the color detection function.

| | | |
|---|---|---|
| CMD | | Command structure to trigger an action |
| | Evaluate | Evaluate new product and try to determine the color |
| | Teach | Teach a new product and store color information |
| | ResetError | Reset error state |
| CFG | | Image Archiv configuration |
| | FlashColor1 | First flash color (red) |
| | FlashColor2 | Second flash color (green) |
| | FlashColor3 | Third flash color (blue) |
| | FlashColor4 | Fourth flash color (lime) |
| | ProductName | Name of the product |
| | GrayValue1 | Mean gray value for first flash color |
| | GrayValue2 | Mean gray value for second flash color |
| | GrayValue3 | Mean gray value for third flash color |
| | GrayValue4 | Mean gray value for fourth flash color |
| | TeachingIndex | Index that is currently taught |
| | MaxError | Maximum error for all gray values |
| | MinDifference | Minimum distance to next best value |
| DAT | | Sensor status information, see manual for details |
| | GrayValue1 | Current mean gray value for first flash color |
| | GrayValue2 | Current mean gray value for second flash color |
| | GrayValue3 | Current mean gray value for third flash color |
| | GrayValue4 | Current mean gray value for fourth flash color |
| | TotalError | Total error of mean gray value for all products |
| | LowError | Lowest error found |
| | LowDistance | Distance to next best value |
| | LowIndex | Index of the best value |
| | LowName | Product name of the best value |
| | Status | Status of color detection |

## 4.5 Vision image structure

The image structure handles all functions and parameters that are related to the image archive and crosshair drawing in mappView.

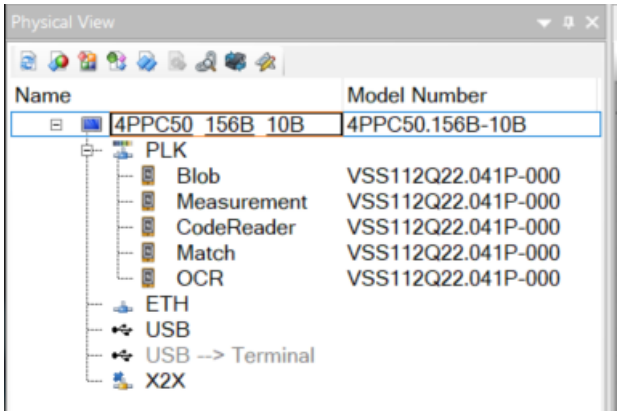
| | | |
|---|---|---|
| CMD | | Command structure to trigger an action |
| | SaveImageOnPLC | Upload an image from the sensor and store it on the flash card |
| | Refresh | Reload image list from flash card |
| | DeleteImage | Delete selected image |
| | ResetError | Acknowledge |
| | DeleteDir | Deletes the complete folder with all images |
| | CreateDir | Creates an empty folder for images |
| | DownloadImage | Downloads the image through the web browser |
| | RefreshCrosshair | Draws the crosshair information into the the last image |
| CFG | | Image Archiv configuration |
| | FileDevice | File device name where the images are stored |
| | DirName | Name of an automatically created folder on the FileDevice |
| | CameraIP | IP address of the sensor |
| | ConvertCycles | For saving the image with crosshair the image date needs to be converted with Base64. This is the number of converted bytes per TC8 cycle. So e.g. an 1.3MP bmp file has ca. 1.300.000 Bytes to convert. The defaultvalue of 10.000 needs 130 TC8 cycles. On high performance CPUs this value could be increased, maybe much more (> 6.000.000 makes all in one cycle, also with 5MP bmps). On low performace CPUs maybe this value needs to be decreased. |
| | Format | Image format (bmp (1) or jpg (0)) |
| | QualityJPG | For JPEG a quality can be defined |
| | UploadBmpJpg | If True, the bmp/jpg images will be loaded from the sensor |
| | UploadSVG | If True, the bmp/jpg will be converted to SVG and the crosshairs with data will be embedded in the new SVG file |
| DAT | | Sensor status information, see manual for details |
| | Images | Images list as data provider for connection to mappView |
| | Status | Status of the image operation |
| | Croshair | Crosshairdata, will be copied from VisionMain |

# 5 Description

## 5.1 Hardware configuration

The sensor used in this sample is VSS112Q22.041P-000. If this is not the sensor available right click on the hardware and choose “Replace Hardware Module” to select the correct hardware.

In the demo application, each sensor represents one vision function. By changing the node number, it is possible to quickly switch between different functions.




The hardware configuration uses the following Powerlink node numbers:

1: Blob
2: Measurement
3: Code Reader
4: Match
5: OCR
6: PixelCounter

## 5.2 Camera image

The mappView visualization shows the image of the camera on the main page. This is done through a web widget that points to the HTML page that points to the camera IP address. Since this IP address is not known to the outside world two options can be used to forward the communication to the camera.

### 5.2.1 Using the proxy

Starting with demo version 2.3 a proxy library can be used to access the camera image. The proxy function runs in the task Vi_image.

### 5.2.2 Using port forwarding

The image can also be aquired with port forwarding on the PC side. A predefined batch file can be executed under Windows. Make sure to adjust the IP address in the file

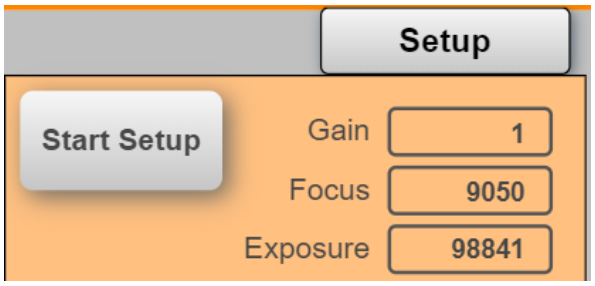
*“\ProjectName\Logical\Vision\setRouteToCamera.bat”*

and execute the batch file in Windows with right click (Run as administrator). Otherwise, the sensor image does not work in the Vision Cockpit or the demo visualization.

## 5.3 Demo application

The demo application consists of multiple pages to demonstrate the vision function. The main page is used to set up the sensor image. The bottom window shows the most important parameters and status information. The first step is to make sure that the sensor is connected and ready. All four elements at the top should be green.

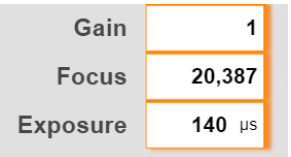

To start the auto setup process click on Setup on the bottom right corner. Click on “Start Setup” to initiate the auto setup that determines the values for gain, focus and exposure.




The sensor light should flash for about 20 seconds.

If the object is not aligned correctly use the Repetitive Mode to make continues images and allign the object.

Click on Trigger to generate a new image. In some cases it may be necessary to adjust the automatically generated values.




Use the crosshair toggle button to show additional information. Images are stored automatically when the checkbox “Auto Archiv Image” is set (see 0). Details results can also be viewed on the “Results” page. Google Chrome browser must be used for the mappView project; other browser may not show the correct crosshair position.

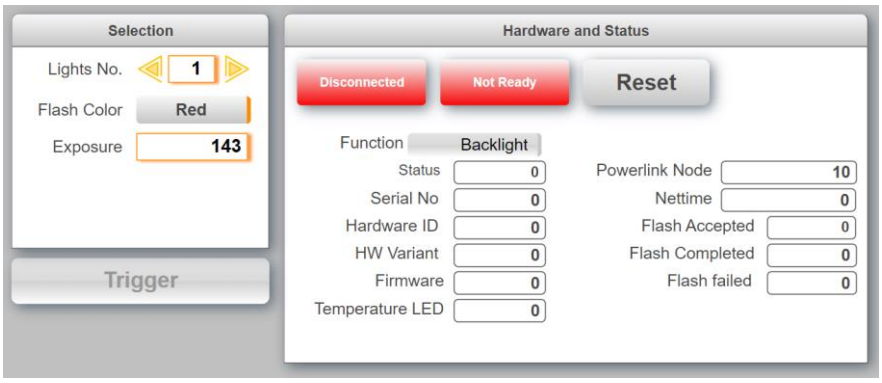
## 5.4 Changing the demo

The different tasks are designed to easily integrate into an existing application. All interactions are handle by variable structures. To allow easy update of these task they should not be changed. The demo includes a task “YourTask” that can be used to write own code.

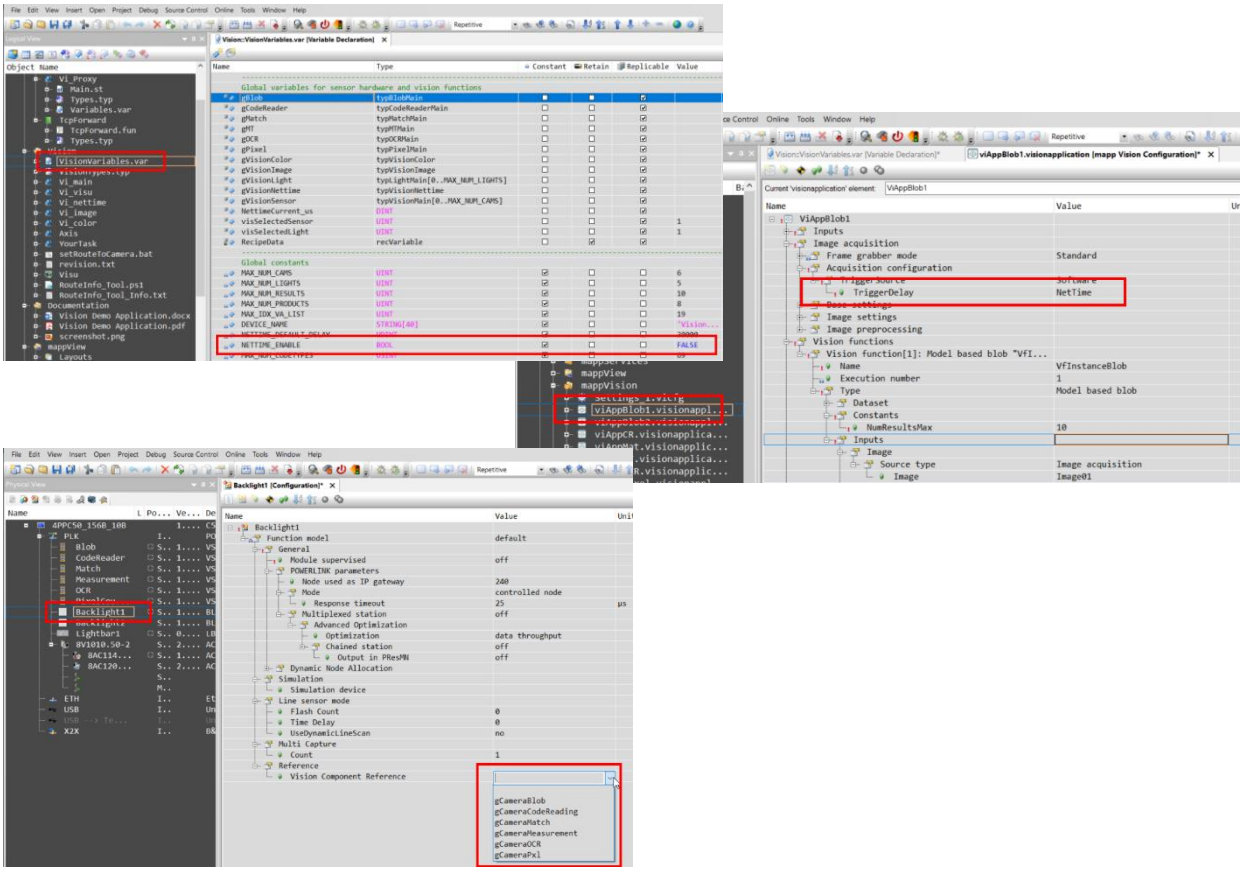
The task includes the sample configuration the different vision functions and lights. It also includes a basic state machine to connect a drive and use the nettime functionality.

## 5.5 Lights

On the lights page the backlights or lightbars can be tested.



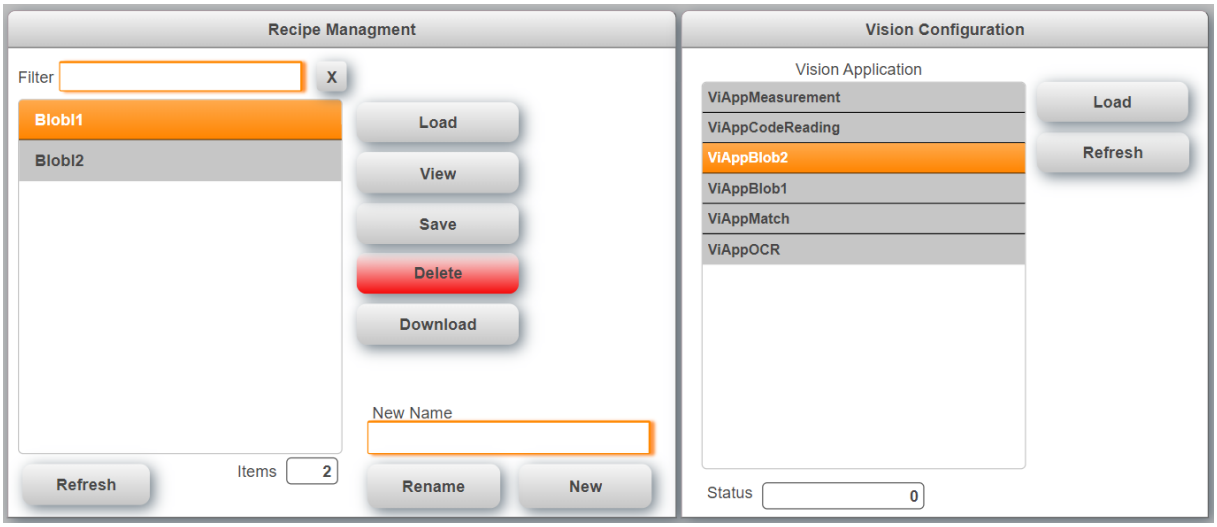

When using external lights make sure to set the constant NETTIME_ENABLE to TRUE, change the TriggerDelay to Nettime in the VisionApplication and connect the vision component under the light hardware configuration.



## 5.6 Recipe

On the recipe page, the camera settings are saved in a CSV file. The data is stored on the user partition.

On this page, it is also possible to switch between vision applications. The list contains all vision applications. The user must know what vision application is compatible with the selected vision function. This means the vision application belongs to the same type of vision function (blob, match,…) and the number of IOs points has not changed. It is not possible to switch to a dfferent vision function .
Status 328685176 indicates that the user tried to load a vision application that is not compatible with the vision component. All vision applications must be pre-defined under the configuration view in mappVision.



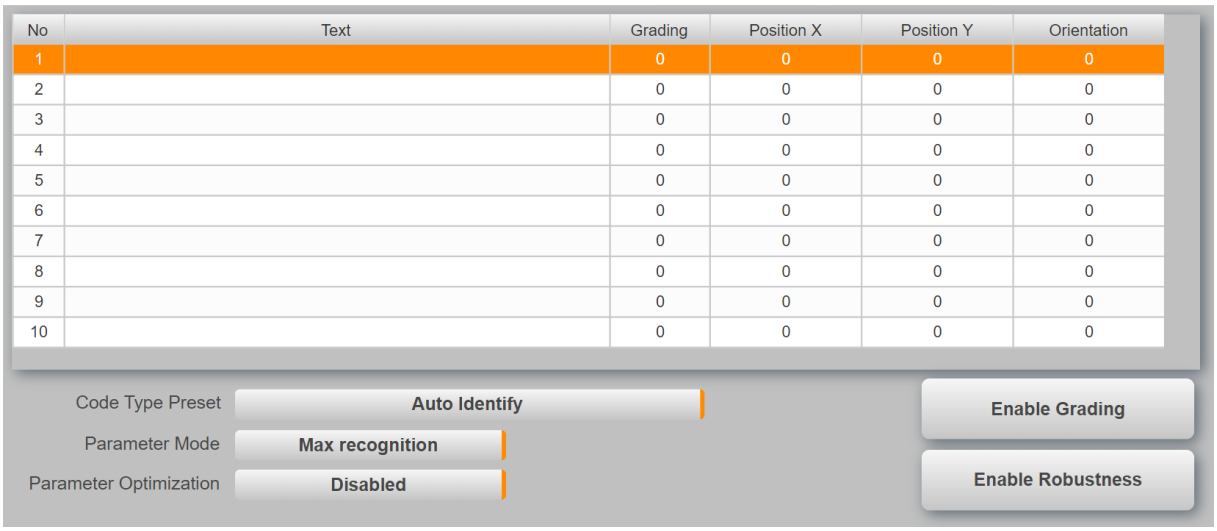

The data that is stored in a recipe is transferred in the sample task “YourTask”. By default the following data is stored:

- Vision application name
- Gain
- Exposure
- Focus
- FlashColor
- FlashSegment
- MaxItemCnt
- Timeout

When a recipe is loaded the sample task automatically switches to the vision application that is stored in the recipe.

## 5.7 Code Reader

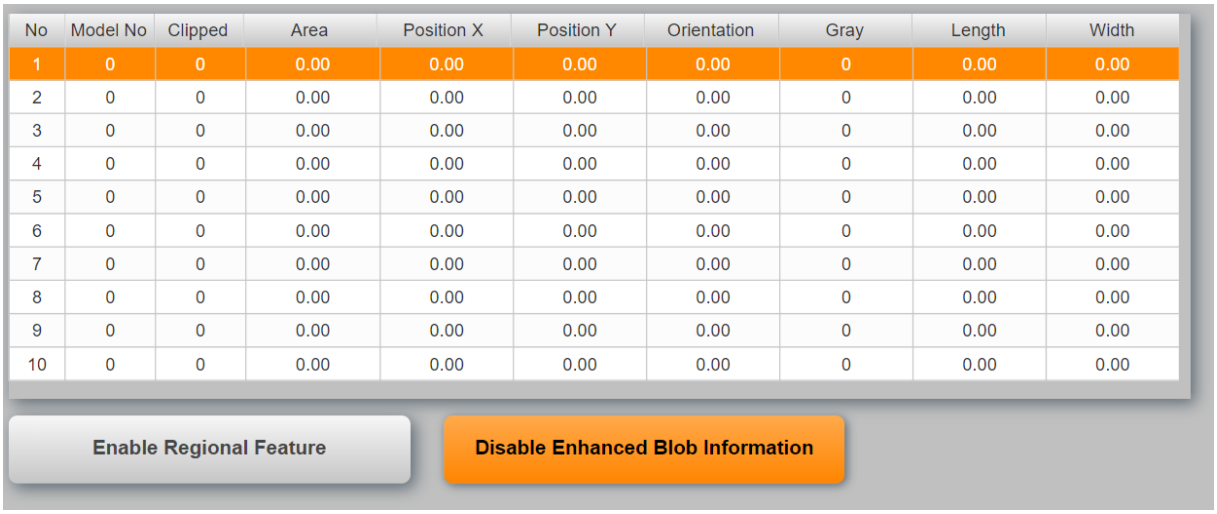
The code reader page provides the information that are specific for the code reader functions. Select the code type from the drop down menu or use “Auto Identify” to start the process that tries to identify the code automatically. The identification process can run for up to 20s.




It is possible to read multiple codes at the same time but all codes must be of the same code type.

## 5.8 Blob

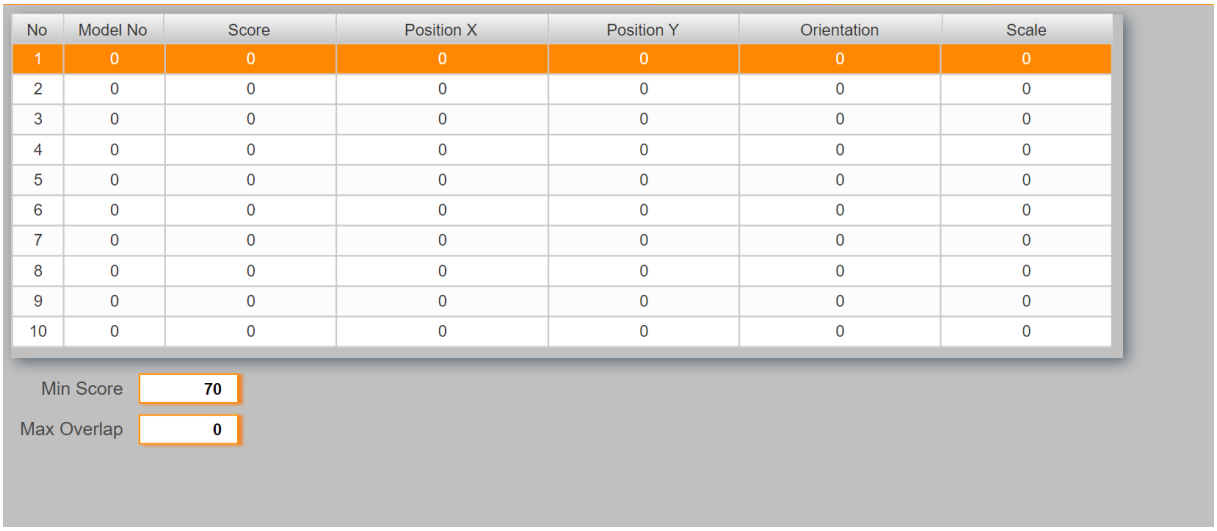
The blob page provides the information that are specific for the blob functions. The table shows the details for each blob that was detected by the sensor. Teaching must be done in the Vision Cockpit.




The blob sends only basic information like Position, Model No by default even if more data is configured. To receive all data the parameter “Enhanced Blob Information” must be enabled.

## 5.9 Match

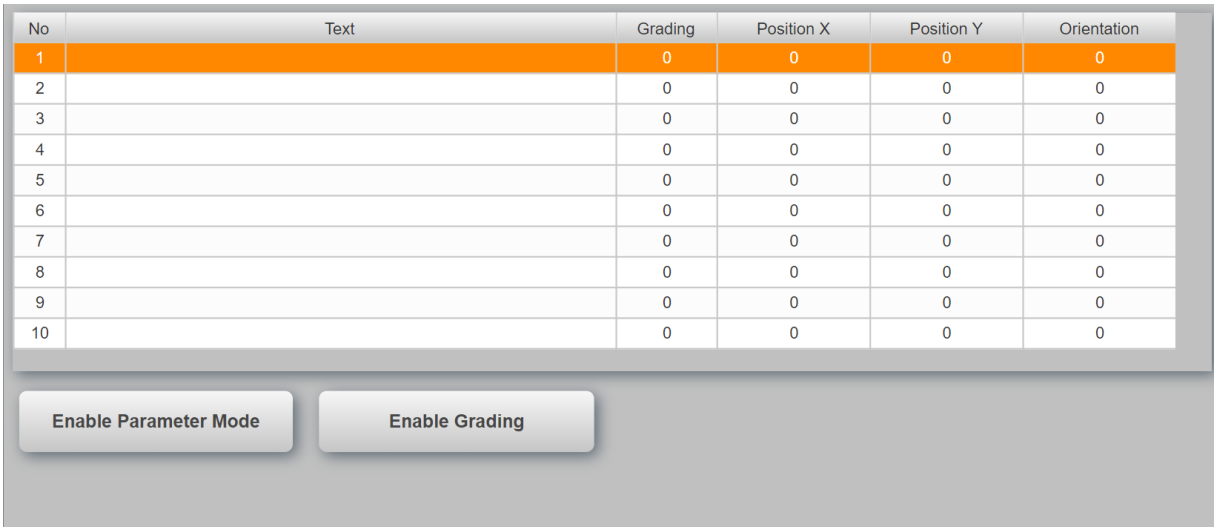
The match page provides the information that are specific for the match functions. The table shows the details for each item that was detected by the sensor. Teaching must be done in the Vision Cockpit.




Use the value “Min Score” to adjust the detection rate. A lower value is more tolerant but can also cause fault detections.

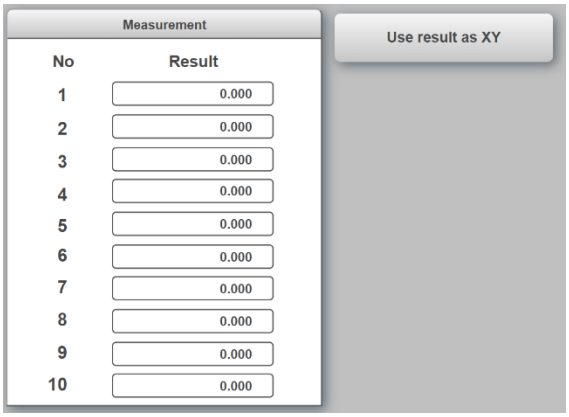
## 5.10 OCR

The OCR page provides the information that are specific for the OCR functions. The table shows the details for each text that was detected by the sensor.



## 5.11 Measurement

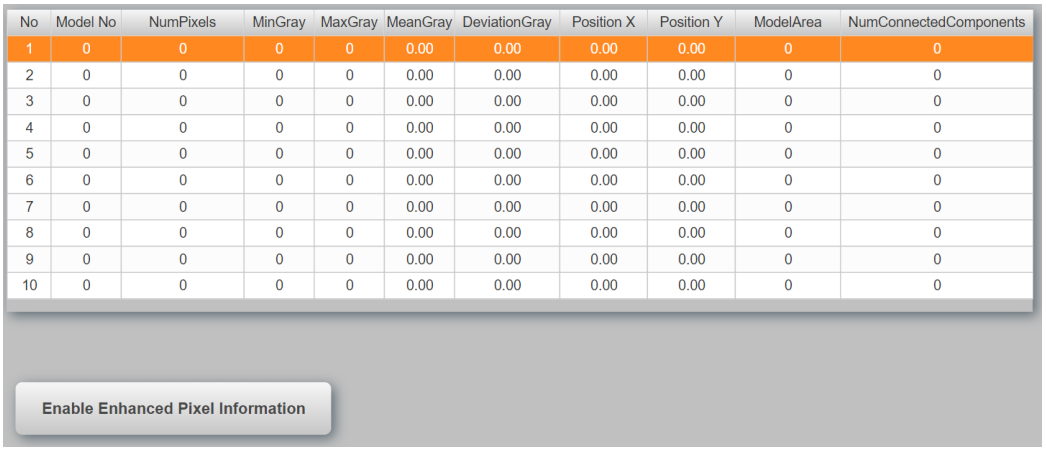
The measurement page provides the information that are specific for the edge measurement functions. This page shows the results for the different measurement functions. What is measured must be configured in the vision cockpit.




For edge detection it can be helpful to also draw crosshairs at the position where the edge was found. This can be enabled with the toggle button “Use result as XY”. In this case the first result must be defined as the X position and the second as the Y position. Repeat this pattern for all edges.

## 5.12 Pixel Counter

Vision function ModelBasedPixelCounter is a function for counting pixels and extracting features from them. ModelBasedPixelCounter makes it possible to define regions within which the pixels corresponding to a predefined grayscale value interval (ThresholdMin/ThresholdMax) are counted.

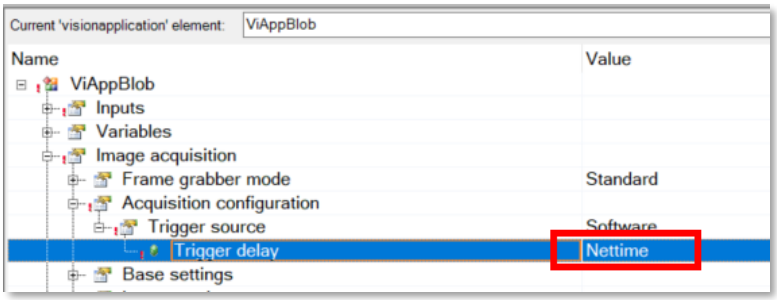



If EnhancedPixelCounterInformation is disabled, only parameters ModelNumber, NumPixels and ModelArea are cyclically filled with data. If outputs are defined that do not correspond to those described above, they will be filled with 0.

Enabling this parameter increases the size of the POWERLINK frame since the remaining cyclically reading parameters are additionally transferred.

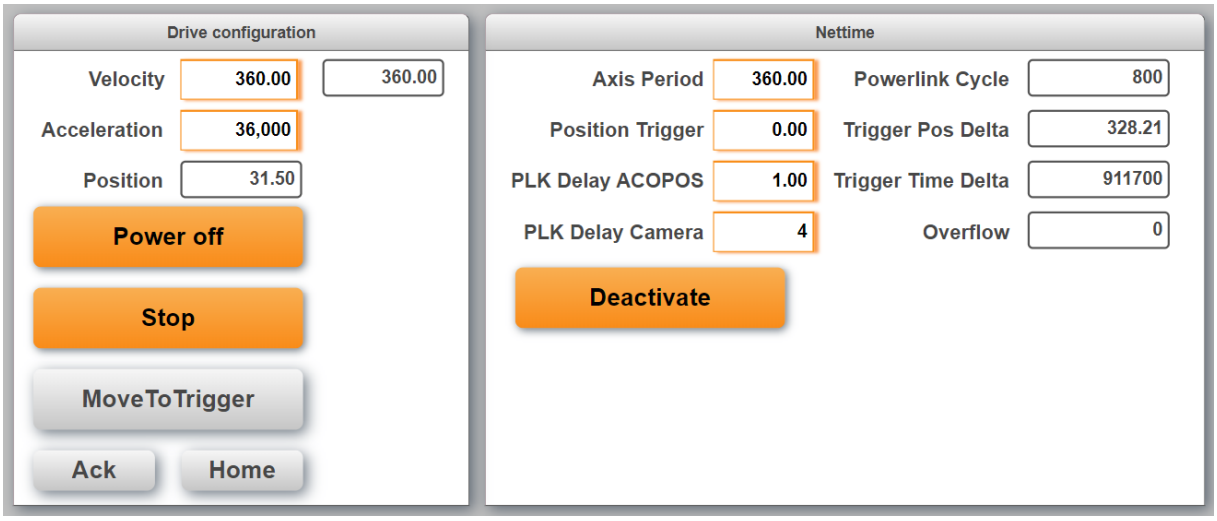
## 5.13 Using Nettime

In some applications it can be necessary to trigger the sensor periodicly depending on a drive position. This can be accomplished with nettime. To enable nettime the trigger delay has to be changed to nettime in the vision application settings.




When this feature is enabled the manual trigger gVisionSensor[].CMD.ImageTrigger only works when the parameter gVisionSensor[].CFG.NettimeDelay is set correct (Current nettime value plus offset, ex. 10ms).

The task Vi_nettime provides the necessary calculation for a motion application. It is curcial that this task runs in sync and at the same cycle as the Powerlink bus. The following page allows the configuration of the nettime function.




On the left hand side are the basic drive settings.

- Power: Switches the axis on and off. In the task “Axis” all the Axis-Handling is done. By default the setting is to use the encoder reference pulse. So when the axis is switched on and not homed it automatically searches the reference pulse
- Run: A continous movement will start with the set velocity and acceleration
- MoveToTrigger: Moves the axis to the “Position Trigger” (Nettime-settings)
- Ack: Acknowledges errors, if there are any errors
- Home: Makes again a homing, also if it was already done automatically while powering on.

On the right hand side are the nettime settings:

- Axis period: This is the number of units for one cycle (360 for a rotating axis). It should be the same value as in the axis settings
- Position Trigger: This is the position where the image will be made. Should be in the period

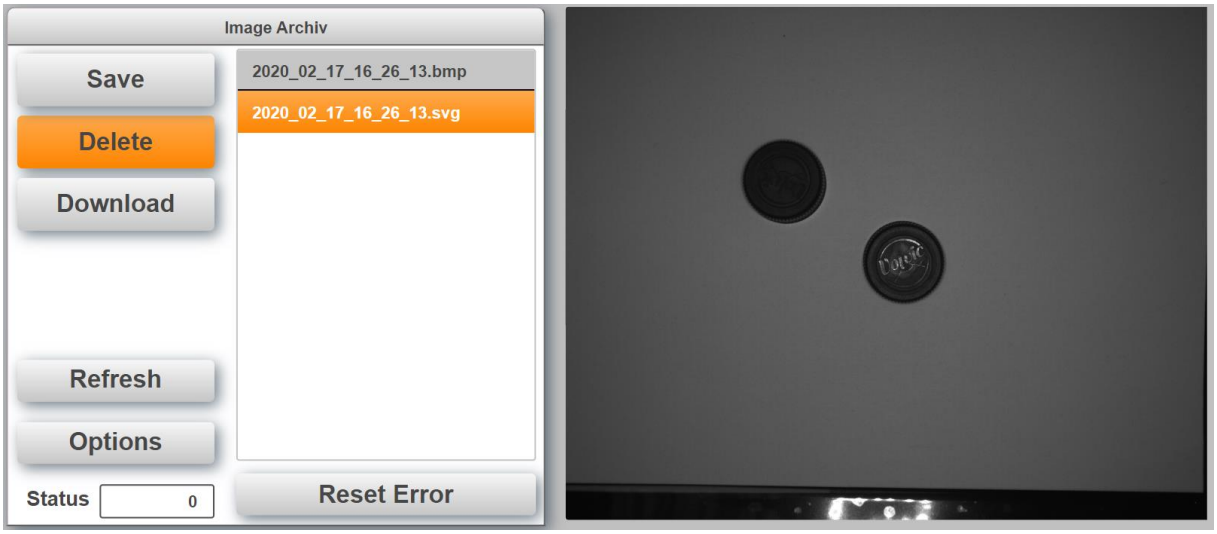
- PLK Delay ACOPOS (number of PLK cycles): When the PLC reads the motor position, this position is some time old, e.g. 1 oder 2 PLK cycles. This delay will be compensated. The value is the number of PLK cycles for the “age” of the motorposition. It is possible to use a value with fraction digits. This makes sense because of not only the Powerlink has a delay. E.g. also the encoder could have a small delay. So it is possible to adjust the value very precise
- PLK Delay Camera (number of PLK cycles): This is used to calculate the time when the nettime value must be set at the latest to make it to the sensor in time.The camera needs to get the nettime for the trigger some time before the trigger. A good value is 4. If the value is too small, the camera gets the nettime too late and can’t make the image any more. If the value is too high, the camera gets the nettime earlier. A speed-change will then no more be calculated.
- Powerlink Cycle: Powerlink cycletime (in microseconds)
- Trigger Pos Delta: This is the remaining position delta to the next trigger (in units)
- Trigger Time Delta: This is the remaining time delta to the next trigger (in microseconds)
- Overflow: If the nettime handling wants to send the next trigger to the camera, but the camera is not ready, this value will be increased by 1.

### 5.13.1 Precision

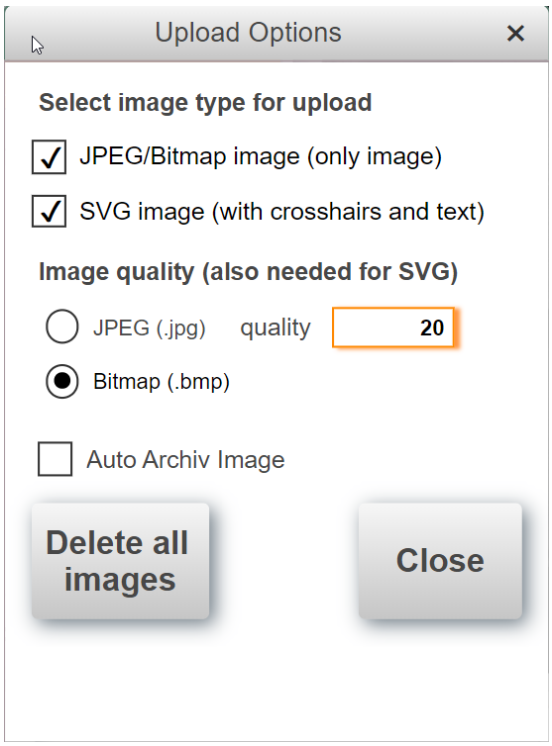
If the nettime seems not to be precise, it makes sense to check the lag error of the axis. To get very good results, a well tuned controller is necessaray.

## 5.14 Image Archive

The image archive is used to store sensor images on the PLC flash card. This can be necessary to inspect ‘bad’ products later in the process. The image archive is controlled by its own task “VisionImage” and structure (see 0).




The number of images that are stored depends on the size of array *VisionImage.DATA.Images*. The default size is 20. When the list is full and a new images are uploaded the oldest images will automatically be deleted. The task will also highlight and load the newest image after upload or refresh. Images are stored automatically when the checkbox “Auto Archiv Image” is set on the main page.

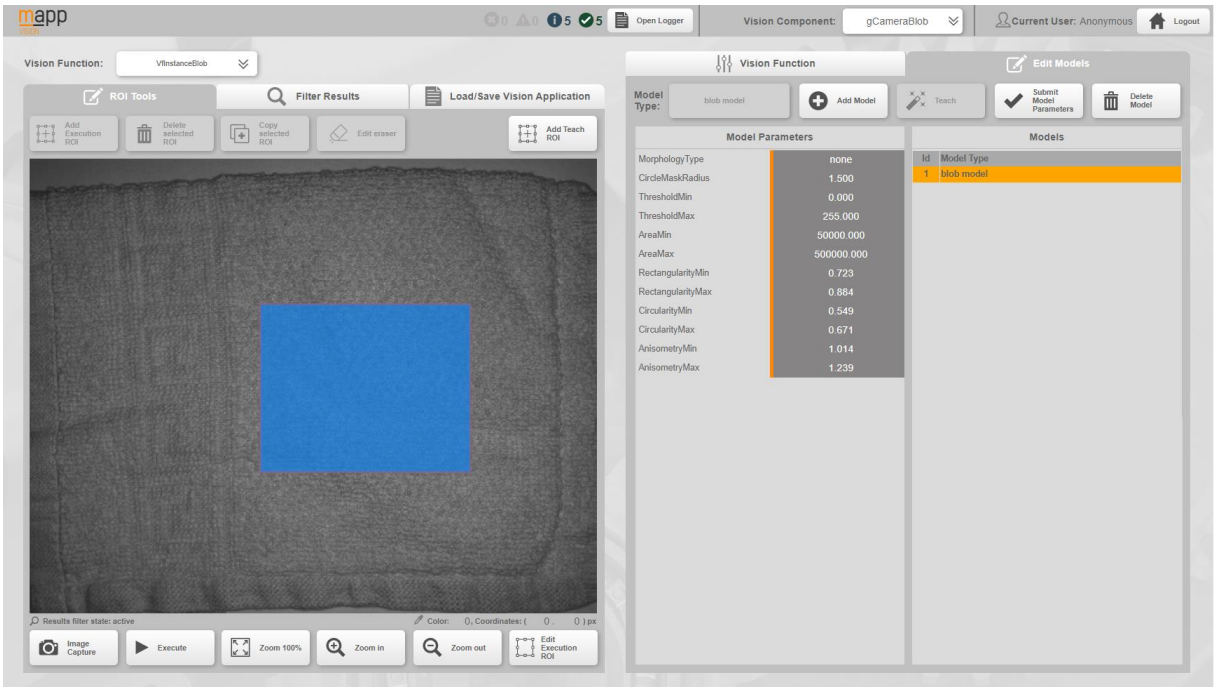



In the Options Dialog it can be selected, if the sensor creates a BMP or JPEG image. For JPEG images the quality can be selected. Also 100% is possible. It can be selected if the BMP or JPG will be saved as it is and/or if a SVG with crosshairs will be created. All Options are possible, so only SVG Upload is possible or also both or only BMP/JPEG. “Reset” resets e.g. FileIO Erros, you can find in the “Status” information on image archive. “Delete all images” deletes the complete folder with all images and creates the new empty folder.

The PLC has the FTP server enabled to check the images remotely. The user name and password is “bundr”.

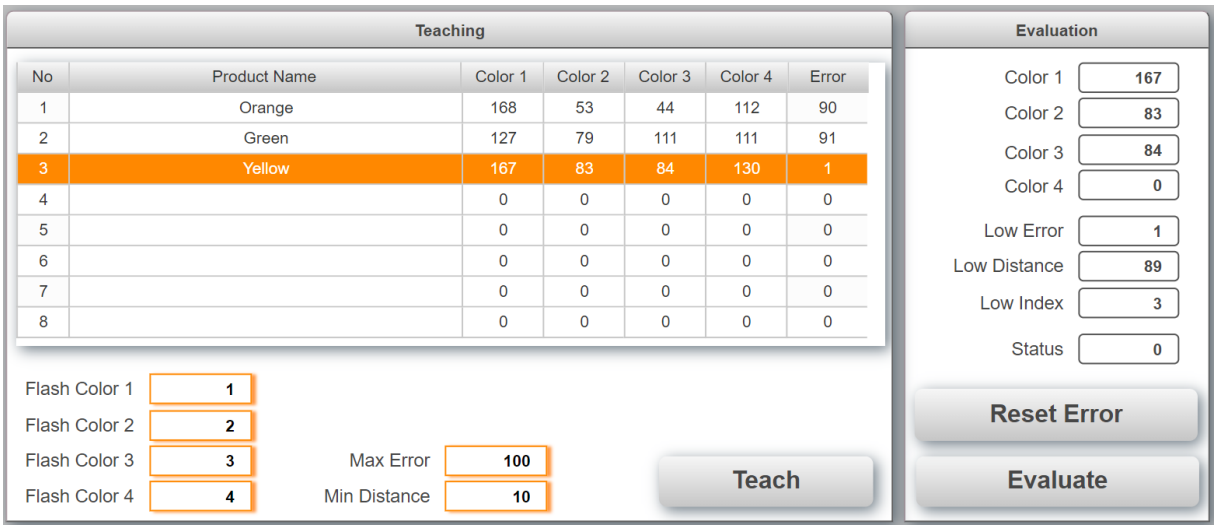
## 5.15 Color detection

The task “VI_color” can be used to detect different colors. The color detection uses the blob function. To detect any color a model with the following parameters should be generated.




The area can be limited by defining a ROI. A larger ROI will increase the color detection performance. Ideally the ROI should only cover one color. Of course the number of colors and variants that can be detected has its limits. If the colors are very different from each other one flash color can be good enough to distinguish between them. For more variants two or more flash colors are necessary and there can be cases where even four flash colors are not enough to detect a product consistently.
Depending on the number of flash colors needed the time to detect the product will increase. A typical value for detecting a blob is around 50ms. Using 4 colors will therefore at least take 200ms plus some overhead.

The color page has a teaching part on the left and evaluation on the right. To teach a new product click on the product name in the table and enter a new name. Then click on “Teach” to start the process. The camera will flash with all configured colors and create a finger print for the new product with the mean gray value. Up to 4 colors can be defined for the process.




To detect a product use the button “Evaluate” on the bottom right. The camera will flash with all colors again and calculate the difference for each product in the column “Error”. The product with the lowest distance is then selected as best value.
If the error for the best value exceeds “Max Error” then no product was found and an error is generated. The task also calculates the second best value. If the difference between the best and second best value is smaller than “Min Distance” the two are two close together to be differentiated and an error is generated.
The value “Low error” shows the difference between the teached product and the detected modell. This indicates how good the detected product matches the teched one. The value “Low distance” shows the distance to the next best hit. This indicates how the solid the detected product is distinguished from the next best match. If only one product was teached this value will always show 65535. The value “Low Index” is the detected best match.

## 5.16 VC4 Visualization

In the project is a VC4 visualization included which can show the camera image in this VC4 visualization. To test it the VC4 visualization can be opened with a VNC viewer. The “Refresh” button loads the image from the camera. Note: Therefore an image should exist in the camera, so first make an image by the known ways and then load it to the VC4 visualization.


### 5.16.1 Background

There are some limitations which will be described here. The camera provides only bmp (8 bit) and jpg images. The VC4 visualization supports only png or bmp (24 bit) images. Because of png and jpg include complex compression algorithms, it is not easily possible to convert these images on the PLC. So there is only the way to convert the 8 bit bmp from the camera to a 24 bit bmp for the VC4 visualization. Therefore the “raw” bmp needs to transferred from the camera to the PLC and then it needs manually to be converted from 8 bit bmp to 24 bit bmp. This takes some time and CPU load, please keep this in mind.
Depending of the CPU load etc. a refresh takes 5 to 15 seconds.

### 5.16.2 Usage

There is a FBK called “ViShowImgOnVC4” implemented. It is called in the task “Vi_image”.
Also a simple VC4 visualization is included. The FBK loads the image from the camera, converts it from 8 bit bmp to 24 bit bmp, saves it on the PLC and creates a HTML stream which is connected to a HTML View element in the VC4 visualization. The FBK “ViShowImgOnVC4” needs some configuration information (CFG) and the Powerlink node number of the camera. It has an input “RefreshImage” to load the new picture from the camera and finally show it on the VC4 visualization. The input “ImgWidthInVC4_px” needs the width of the image in VC4 (auto resize).

#### 5.16.2.1 Implementation in VC4

In the VC4 visualization it is only necessary to add a HTML View element. Only the property “HTML Stream” needs to be connected to the variable “ViShowImgOnVC4_0.HTMLStreamContent”. That’s all. The size of the HTML View should be some pixels bigger as the width set in “ImgWidthInVC4_px”. The height should be matching the image proportions.

# 6 Tips and Hints

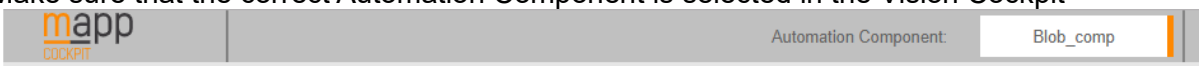
## 6.1 Sensor is connected and ready but the image on the main page is not refreshed

Make sure to adjust the IP address in the file “\ProjectName\Logical\Vision\setRouteToCamera.bat” and execute the batch file in Windows with right click (Run as administrator).

## 6.2 The Vision Cockpit does not work correct and/or does not show the sensor image when the sensor is connected and ready.

Make sure to adjust the IP address in the file “\Vision_1\Logical\Vision\setRouteToCamera.bat” and execute the batch file in Windows with right click (Run as administrator).

Make sure that the correct Automation Component is selected in the Vision Cockpit




## 6.3 How to setup a T50 and C50 to use demo?

The T50/C50 must use the PLC as Gateway for the camera image on the main page to work. Assuming that the PLC has the IP address: 192.168.1.100.

**T50**
Go into the T50 and change the following settings

Web:
http://192.168.1.100:81/index.html?visuId=visVision

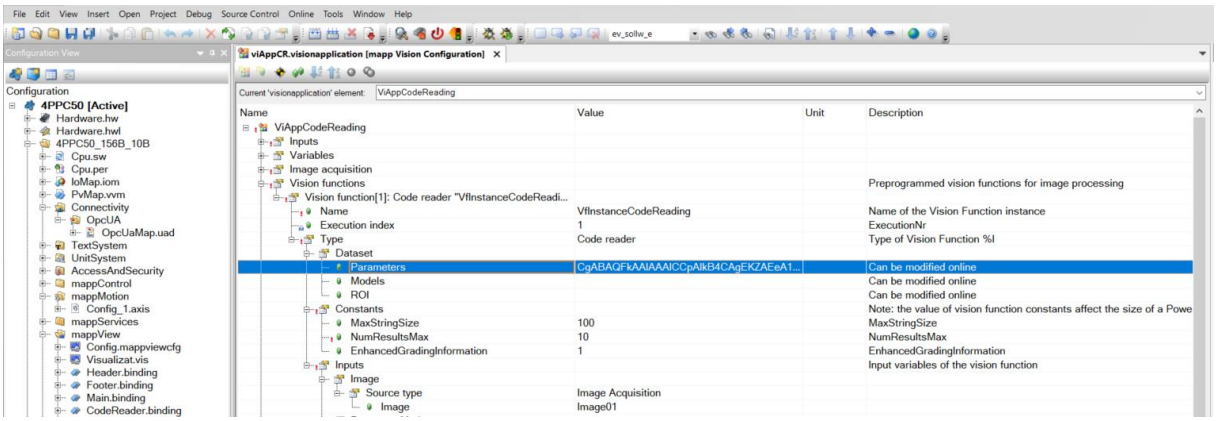
Network:
IP address: 192.168.1.98
Subnet mask: 255.255.255.0
Gateway: 192.168.1.100

**C50**
Go into the CPU configuration under Terminal configuration change the network settings

Network:
IP address: 192.168.1.98
Subnet mask: 255.255.255.0
Gateway: 192.168.1.100

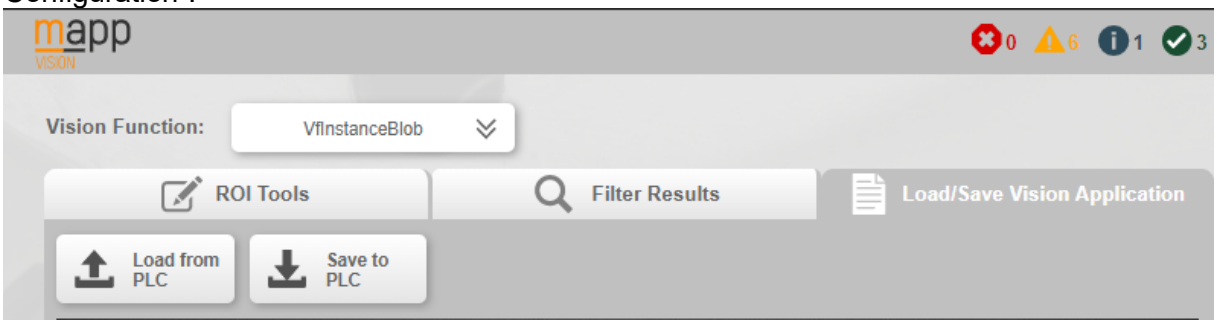
## 6.4 How is the sensor configuration selected?

The default configuration is defined in the Automation Studio project under mappVision->…visionapplication.

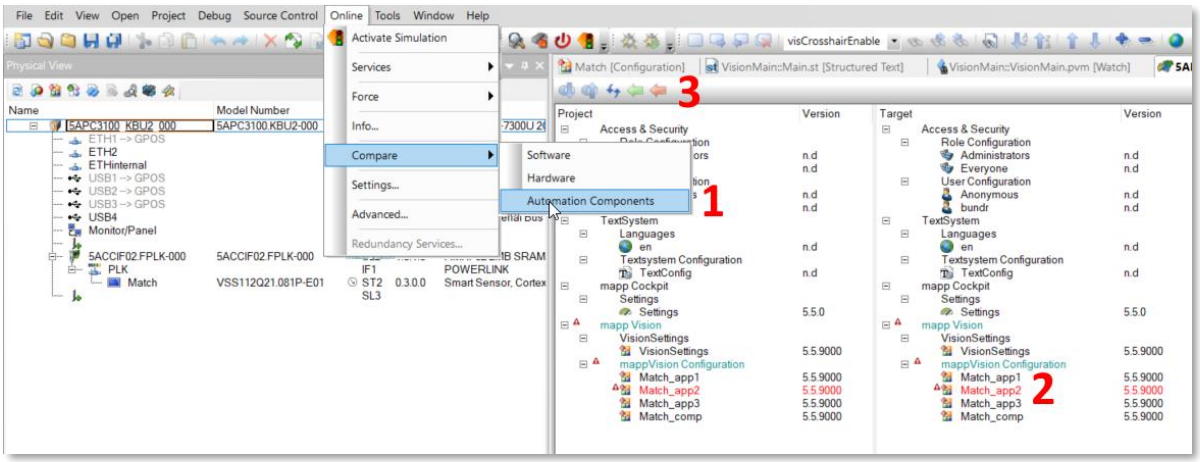


## 6.5 How to store a configuration taught in the vision Cockpit in the Automation Studio project.

Teach the configuration in the vision Cockpit and use the button “Save Vision Function Configuration”.




Go back into Automation Studio and select Online->Compare->Automation Components (1).




Select the vision application highlighted in red (2). Select orange arrow (3) at the top to transfer the sensor configuration back to Automation Studio.

# 7 Error numbers

| Error number | Task | Description |
|---|---|---|
| 3000 | Vi_main | Number of applications exceeds array size. Increase MAX_IDX_VA_LIST |
| 50200 | Vi_image | Buffer for creating SVG file is too small |
| 50201 | Vi_image | Buffer for creating BMP file is too small |
| 50202 | Vi_image | BMP data is incorrect |
| 50210 | Vi_image | The camera does not have image to transfer |
| 50211 | Vi_image | Download response timed out |
| 50212 | Vi_image | Size of download file exceeds maximum |
| 50300 | Vi_image | Powerlink node number is 0 |
| 50301 | Vi_image | No image type for selected (JPG or SVG) |

# 8 Revision History

➔ You can find the revision history also in the project (folder “Vision”/revision.txt)