

# BAUMER AG

## VeriSens Provider

Version 1.0.2

### User's guide

June 24, 2013

NOTES:

**[ Revision history ]**

Version	Date	Contents
1.0.0	2012-8-02	First edition.
1.0.2	2013-6-24	TimeOut option included

**[ Hardware ]**

Model	Version	Notes
VeriSens ID Series		
VeriSens CS Series		
VeriSens XF Series		
VeriSens XC Series		

## Contents

1. Introduction .....	4
2. Outline of provider .....	5
2.1. Outline.....	5
2.2. Method and property .....	6
2.3. CaoWorkspace::AddController method.....	6
2.4. CaoController::Execute method .....	7
3. Command reference .....	8
3.1. Controller class .....	8
3.2. CaoController::Execute("IsConnected") command .....	9
3.3. CaoController::Execute("IsSetupMode") command .....	9
3.4. CaoController::Execute("IsRunMode") command .....	9
3.5. CaoController::Execute("RebootDevice") command.....	9
3.6. CaoController::Execute("GetCurrentJob") command.....	10
3.7. CaoController::Execute("ClearStatistics") command .....	10
3.8. CaoController::Execute("GetData") command .....	10
3.9. CaoController::Execute("GetDataXYR") command.....	10
3.10. CaoController::Execute("GetImage") command.....	11
3.11. CaoController::Execute("DoTrigger") command.....	11
3.12. CaoController::Execute("DoTriggerAndGetData") command .....	11
3.13. CaoController::Execute("SwitchToSetupMode") command .....	12
3.14. CaoController::Execute("SwitchToRunMode") command .....	12
4. Sample program.....	13
5. Preparing the Baumer VeriSens Camera for the GetDataXYR function	14
5.1. Prepare the device settings.....	14
5.2. Set point position of the detected part for the GetDataXYR function.....	15
5.3. Set up the output process interface for the GetDataXYR function .....	17

## 1. Introduction

This documentation describes how to use the VeriSens Provider for an easily communication with a VeriSens Smart Camera from BAUMER. The Provider can be used with a PC and high level languages or directly using the RC8 controller with the DENSO PacScript language.

The connection with the VeriSens is based on the TCP/IP Ethernet Communication Interface. The Provider is acting as a Client and the VeriSens camera as a Server waiting for commands.

## 2. Outline of provider

### 2.1. Outline

The VeriSens Provider offers one execution method by CaoController::Execute to send parameters and receive current data values using the VeriSens TCP/IP Ethernet Communication Interface.

**Table 2-1 Baumer VeriSense provider**

File name	BaumerVeriSens.dll
ProgID	CaoProv.BAUMER.VeriSens
Registry registration	regsvr32 [PATH]/ BaumerVeriSens.dll
Registry un-registration	regsvr32 /u [PATH]/ BaumerVeriSens.dll

## 2.2. Method and property

The VeriSens provider connects at the time when the method AddController is executed

## 2.3. CaoWorkspace::AddController method

### Syntax

```
AddController ( < bstrCtrlName:VT_BSTR > and < bstrProvName:VT_BSTR >
                <bstrPcName:VT_BSTR > [,<bstrOption:VT_BSTR>] )
```

bstrCtrlName :[in] Controller name arbitrary.

bstrProvName : [in] Provider name. (Fixed to " BAUMER.Verisens")

bstrPcName : [in] Execution machine name of provider

bstrOption : [in] Option character string

Following is a list of option string items.

Option	Meaning
Conn=< connection parameter >	Set the communication form and connection parameters
TimeOut[=<Timeout period>]	Specify time-out time (millisecond) for sending and receiving (default:infinite)

### Conn option

The following describes the connection parameter string for the Conn option for Ethernet.

"eth:<IP Address>[:<Port No>]"

<IP Address> : Mandatory. Specify IP address.  
Example:"192.168.0.1"

<Port No> : Specify the port number to communicate:  
Example:"192.168.0.1:23"  
The port number is 23.(fixed)

## 2.4. CaoController::Execute method

The command name is specified in the first argument and the parameter of the command is specified for the second argument. Please refer to Chapter 3 Command reference for details of each command.

**Syntax** Execute ( <bstrCommandName:VT\_BSTR>, [<vntParam: VT\_VARIANT> ] )

bstrCommandName : [in] Command name (VT\_BSTR)

vntParam : [in] Parameter

Return value : [out] Return value of command (VT\_R4 | VT\_ARRAY or VT\_BSTR)

## 3. Command reference

### 3.1. Controller class

**Table 3-1 CaoController::Execute command list**

Command	Function	Page
IsConnected	Get info about the connection status	9
IsSetupMode		9
IsRunMode		9
RebootDevice	Restart the camera	9
GetCurrentJob	Get the current job number	10
ClearStatistics	Clear the statistics of the active job	10
GetData	Get the latest datas from the camera	10
GetDataXYR	Get the latest detection datas X, Y, and Rotation <b>(Refer Chapter 5)</b>	10
GetImage	Get the live image	11
DoTrigger	Trigger a new image	11
DoTriggerAndGetData	Trigger an image and receive datas directly	11
SwitchToSetupMode		12
SwitchToRunMode		12



### 3.2. CaoController::Execute("IsConnected") command

Get the current connection status.

**Syntax**      IsConnected ()  
                  Argument                 :   None  
                  Return Value            :   VT\_BOOL

**Example**      bool connected = false  
                  connected = CaoCtrl.Execute("IsConnected")

### 3.3. CaoController::Execute("IsSetupMode") command

**Syntax**      IsSetupMode ()  
                  Argument                 :   None  
                  Return Value            :   VT\_BOOL

**Example**      bool mode = false  
                  mode = CaoCtrl.Execute("IsSetupMode")

### 3.4. CaoController::Execute("IsRunMode") command

**Syntax**      IsRunMode ()  
                  Argument                 :   None  
                  Return Value            :   VT\_BOOL

**Example**      bool mode = false;  
                  mode = CaoCtrl.Execute("IsRunMode")

### 3.5. CaoController::Execute("RebootDevice") command

Reboot the camera system.

**Syntax**      RebootDevice()  
                  Argument                 :   None  
                  Return Value            :   None

**Example**      caoCtrl.Execute("RebootDevice")

### 3.6. CaoController::Execute("GetCurrentJob") command

Returns the current job number.

**Syntax**      GetCurrentJob()  
Argument                   : None  
Return Value               : VT\_I2

**Example**      short jobNumber = CaoCtrl.Execute("GetCurrentJob")

### 3.7. CaoController::Execute("ClearStatistics") command

Clears the statistics of the current job.

**Syntax**      ClearStatistics ()  
Argument                   : None  
Return Value               : None

**Example**      CaoCtrl.Execute("ClearStatistics")

### 3.8. CaoController::Execute("GetData") command

Get the current data defined in the "Output process interface" in the VeriSens Application Suite software.

**Syntax**      GetData ()  
Argument                   : None  
Return Value               : VT\_BSTR

**Example**      string data = string.empty;  
data = CaoCtrl.Execute("GetData")

### 3.9. CaoController::Execute("GetDataXYR") command

Get the position and rotation data of a detected item. If no item is detected, the return value would be (0,0,0)

**Syntax**      GetDataXYR ()  
Argument                   : None  
Return Value               : Detection values (VT\_R4 | VT\_ARRAY)

**Example**      ReceiveValues = CreateArray(3, VT\_R4)

```

ReceiveValues(0) = 0
ReceiveValues(1) = 0
ReceiveValues(2) = 0
ReceiveValues = BaumerController.Execute("GetDataXYR", "")
ebug.Print ReceiveValues(0) ' x position
Debug.Print ReceiveValues(1) ' y position
Debug.Print ReceiveValues(2) ' rotation

```

### 3.10. CaoController::Execute("GetImage") command

Get the live image

<b>Syntax</b>	GetImage ()	
	Argument	: None
	Return Value	: Detection values (VT_R4   VT_ARRAY)

**Example**

### 3.11. CaoController::Execute("DoTrigger") command

Trigger a new image. Notice that the camera has to set in trigger mode.

<b>Syntax</b>	DoTrigger ()	
	Argument	: None
	Return Value	: None

**Example**    caoCtrl.Execute("DoTrigger")

### 3.12. CaoController::Execute("DoTriggerAndGetData") command

Trigger a new image and receive the datas immediately. Notice that the camera has to set in trigger mode.

<b>Syntax</b>	DoTriggerAndGetData ()	
	Argument	: None
	Return Value	: VT_BSTR

**Example**    string data = string.empty;  
                  data = caoCtrl.Execute("DoTriggerAndGetData")

### 3.13. CaoController::Execute(“SwitchToSetupMode”) command

Switch to the setup mode to configure the camera with the VeriSens Application Suite software.

**Syntax**      SwitchToSetupMode ()  
Argument                    : None  
Return Value                : None

**Example**      caoCtrl.Execute(“SwitchToSetupMode”)

### 3.14. CaoController::Execute(“SwitchToRunMode”) command

Switch from the Setup Mode to the Run Mode

**Syntax**      SwitchToRunMode ()  
Argument                    : None  
Return Value                : None

**Example**      caoCtrl.Execute(“SwitchToRunMode”)

## 4. Sample program

### List 4-1 BaumerExample.pcs

```
!TITLE "Denso robot program"
#define VT_R4          4
Sub Main
    Takearm Keep = 1
    Dim SendValues As Variant
    Dim ReceiveValues As Variant
    Dim BaumerController As Object
    BaumerController = CAO.AddController("Baumer","CaoProv. BAUMER.VeriSens", "",
"Conn=eth:172.20.57.42:23")

    ReceiveValues = CreateArray(3, VT_R4)
    ReceiveValues(0) = 0
    ReceiveValues(1) = 0
    ReceiveValues(2) = 0

    Do
        ReceiveValues = BaumerController.Execute("GetDataXYR", "")

        Debug.Print "Received Values"
        Debug.Print ReceiveValues(0)
        Debug.Print ReceiveValues(1)
        Debug.Print ReceiveValues(2)
    Loop
End Sub
```

## 5. Preparing the Baumer VeriSens Camera for the GetDataXYR function

### 5.1. Prepare the device settings

Make sure that the results of the image processing are sent with an GD command process. You will find the settings under “Device” -> “Device Settings”. Please be sure that you have the following settings:

#### Protocol setting

Type : TCP

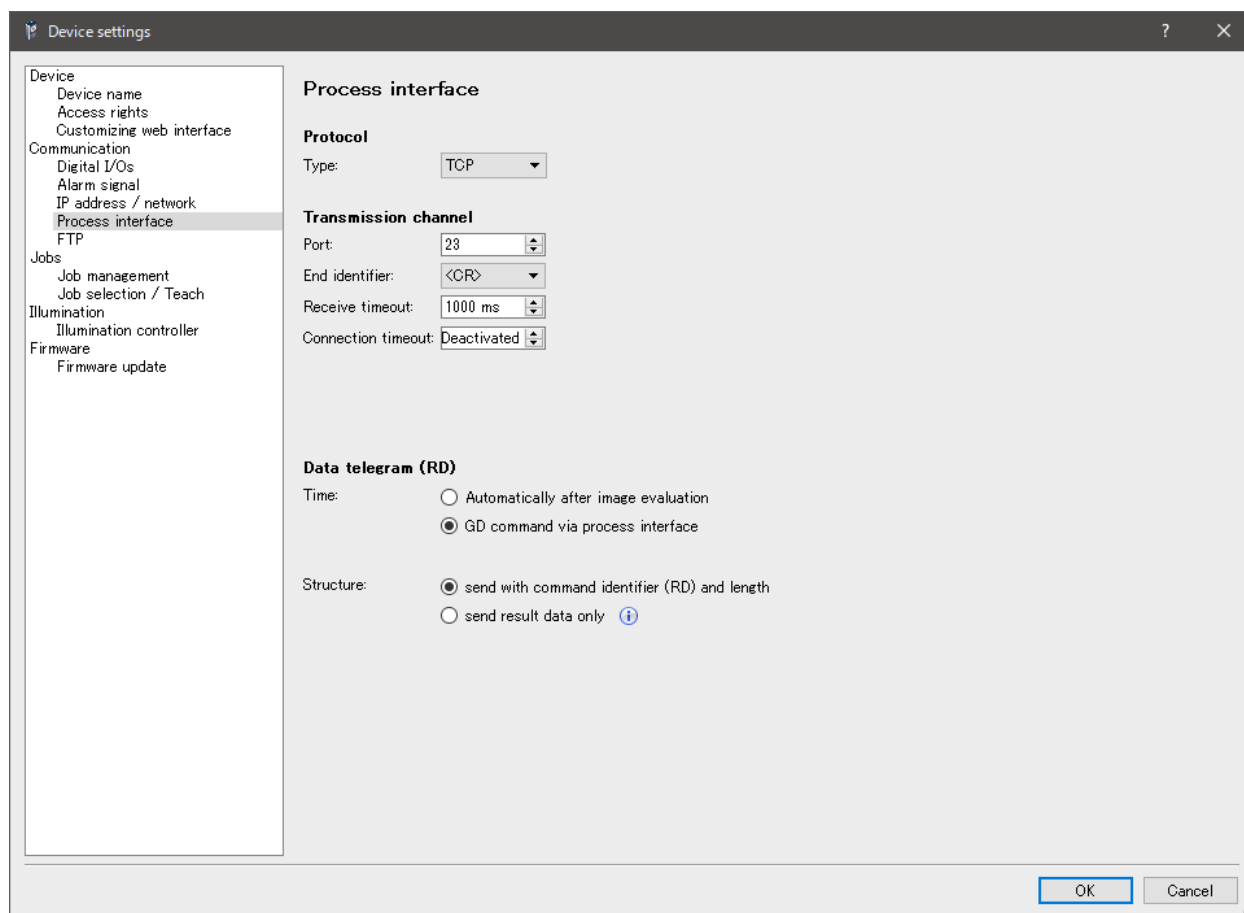
#### Transmission channel setting

Port : 23

End identifier: <CR>

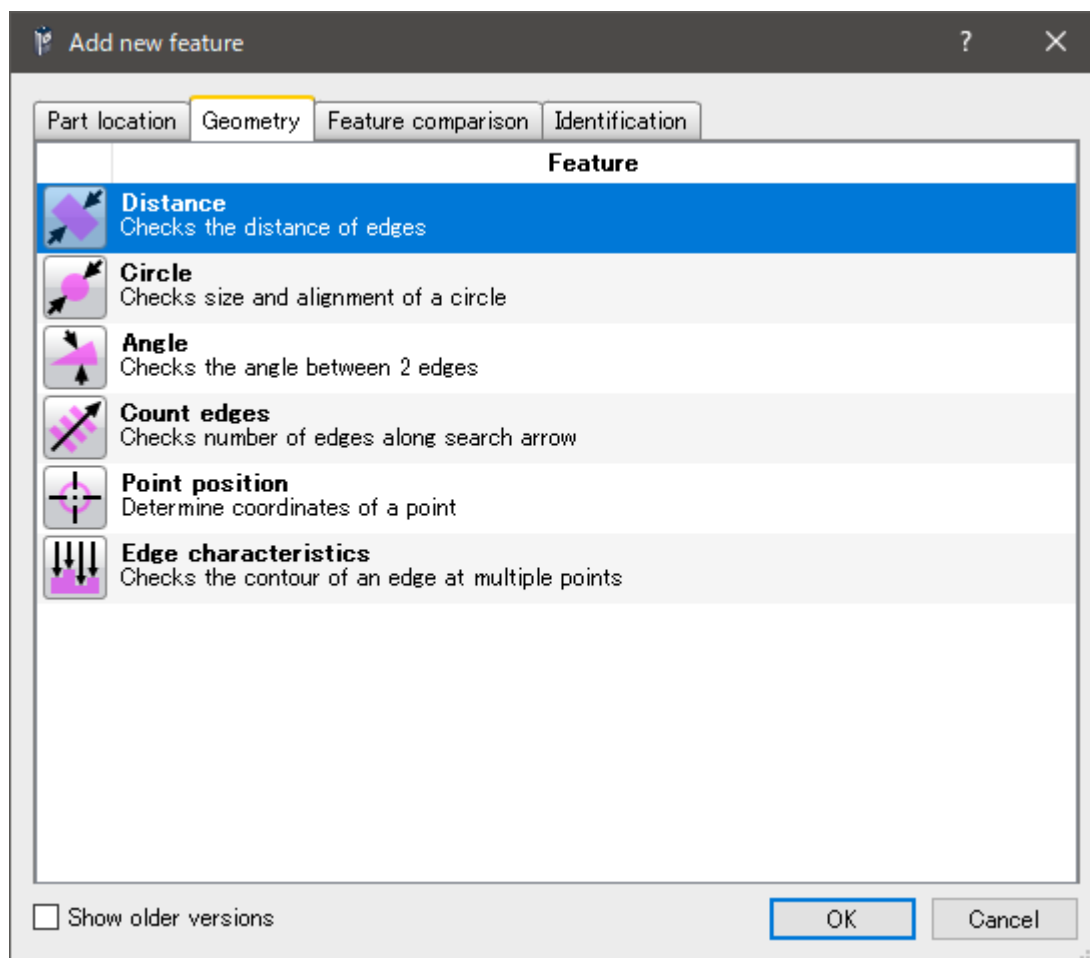
#### Data telegram (RD) setting

Time : GD command via process interface



## 5.2. Set point position of the detected part for the GetDataXYR function

Make sure you have to determine the coordinates of a point. For example the middle point of a detected part. To do this, you have to include the “Point position” feature to your current job.



The screenshot displays the Baumer VeriSens Application Suite v2.8.2 interface. The main window shows a live image of a white part with several black circular features. A green crosshair is positioned on one of the circles. The configuration window on the right shows the 'Point position' settings, including rotation and position coordinates. The bottom panel shows the '2. Check features' step and the 'Edit Point position 1' configuration options.

**Point position**

This feature check determines the position and rotational position of a point, as an absolute in the picture or relative to a reference. It is therefore advisable that it is only used with part location. This feature check can, e.g. be used to determine the grasp position for robots (pick and place).

This feature check supports external Teach. The switching points are adjusted as an absolute to the current measured value.

• Set the point on the position to be determined. You may need to rotate it with the lever.

Rotation 170,34 °  
 Position X 414 Pixel  
 Position Y 283 Pixel

**Baumer**

1. Adjust image | **2. Check features** | 3. Configure interfaces | 4. Activate device

VS XC200M20X00EP-51741117

2: Edit "Point position 1"

Rotation -169.44 °  
 Position X 355 Pixel  
 Position Y 241 Pixel

Calculation method: Absolute position in image

Form: + Point  
 Track position on: Part location on contours 1  
 External teach: - Do not use -

Permission Rights: Deactivated

- Live - EN



### 5.3. Set up the output process interface for the GetDataXYR function

To make sure the provider can read the output of the image processing results from the camera you have to create the following output interface.

The output string has to start with a “S” and end with a “E”. Between two coordinates there have to be a semicolon (;) as separator.

Example for an output string: **S355.00;241.00;-169.44E**

The screenshot shows the 'Output process interface' configuration window in the VeriSens Application Suite. The window title is 'VeriSens Application Suite v2.8.2 - VS XC200M20X00EP-51741117 - MotionSkipTest'. The main area shows a live image of a white component with several black circular features. A green crosshair is positioned over one of the features. The configuration window on the right shows a table for defining data output parameters.

Parameters	Meaning
Start	Character string as a start sequence preceding the data block.
End	Character string as an end sequence concluding the data block.
Separator	Character string included as a delimiter between each individual result of the feature check.

NOTE  
To enter binary characters, you may use the \ symbol in the text. The value can then be specified in hexadecimal format. To add a backslash, enter \\. The character \00 cannot be used.

The configuration window also shows a table for defining data output parameters:

Active	Feature	Value	Start	Format	Min. length
<input checked="" type="checkbox"/>	Point position 1	Position X		ASCII (2 decimal places)	0
<input checked="" type="checkbox"/>	Point position 1	Position Y		ASCII (2 decimal places)	0
<input checked="" type="checkbox"/>	Point position 1	Rotation		ASCII (2 decimal places)	0

The configuration window also shows a preview of the output string: S355.00;241.00;-169.44E. The length of the data telegram is 23 Byte.