Beckhoff TwinCAT
The Windows Control and Automation Technology

CP9090-S100: ActiveX Component for CP9030

Last change: 23.01.2001
Contents

CP9090-S100: ActiveX Component for CP9030

1. Overview 5
2. Installation 5
3. Integration in applications 7
   Integration in Visual Basic 7
   Integration in Visual C++ 8
4. Properties 9
   AkkuCharged 9
   AkkuCharging 10
   AkkuNotPresent 11
   AkkuVoltageOk 12
   AkkuWaiting 13
   CnfErr 14
   ComErr 15
   CycleEnabled 16
   CycleInterval 17
   DeviceAddr 18
   DeviceInfo 20
   DeviceOpen 21
   DisplayOff 22
   EnableUPS 23
   ExtVoltageOk 24
   IdentSwitch 25
   KbdOff 26
   KbusErr 27
   Led 28
   PdCycles 29
   PDLenErr 30
   PDInWLen 31
   PDOutWLen 32
   SKey 33
   UpsDelay 34
5. Methods 36
   AutoCfgPDLenWLen 36
ReadPhysData
WritePhysData
Reset
TriggerCheckOnChange

6. Events
   SKeyDown
   SkeyPress
   SkeyUp
   OnChangeInImage

7. Samples
   Overview
1. Overview
The CP9090-S100 ActiveX component serves the purpose to access the Beckhoff CP-Link card CP9030.

Characteristics
- Configuration of Beckhoff CP-Link card CP9030
- Enables access to additional control-devices (e.g. SKeys, LEDs, poti for override...)
- Enables access for locking the CP (e.g. keys, mouse, touch...)
- Enables access of USV signals

Requirements
- Microsoft Windows 9x / Win NT / Win2K
- Please work with current Service Packs.

CP9090-S200 includes...
The install version includes:
- CP9090-S100 ActiveX-component
- Documentation (PDF and HTML)
- Demo application
2. Installation
Start the "SETUP.EXE" program from disk 1/x to install the CP9090-S100 component under Windows 9x / NT.

Files and folders
Following files will be installed / updated

WINDOWS\SYSTEM(32) folder:
- CpLink9x.dll
- Atl.dll
- Ole32.dll
- Oleaut32.dll
- TcMM.sys
- TcMMHelper.dll
- TcW9xMMHelper.dll

WINDOWS\HELP:
- Cp9090-S100.chm

Help
If CP9090-S100 help files appears "empty", please install HTML-HELP-runtime-files by starting "hhupd.exe" file from the CD first.

It is recommended, that you are familiar with Beckhoff CP-card. The CD contains detailed technical information.

Support
in the event of problems/questions, please contact:

Beckhoff Industrie Elektronik
Eiserstr. 5
33415 Verl
Tel: 05246-963-157
Fax: 05246-963-199
Email: info@beckhoff.com, Internet: http://www.beckhoff.com

Please provide a detailed description of the error as well as information about the operation system (Win9x, NTx Service Packs), and the language selected (German, English), etc.

You will find the version number of the CP9090-S100 component in the 'Properties menu' of the CpLink9x.dll file.
3. Integration in applications

Integration in Visual Basic

CP9090-S100 ActiveX component can be used in Visual Basic. To do this, select the "Components.." command under the "Project' menu item in Visual Basic and mark the 'Beckhoff CpLinkW9x x.x Type Library' entry.

![Components dialog](image)

CP9090-S100 ActiveX component the appears in the Visual Basic toolbox.
Integration in Visual C++

Under construction
4. Properties

AkkuCharged

Returns the status of the USV.

```csharp
HRESULT AkkuCharged(
    [out, retval] VARIANT_BOOL* pVal
);
```

**Parameters**

- **pVal**
  
  `[out, retval]`
  
  TRUE if Akku charged. FALSE if not.

**Comments**

Property is "Read only"

**Sample VB-Syntax**

```vbnet
Dim bAkkuCharged as Boolean
bAkkuCharged = CP9030W9x1.AkkuCharged
```
AkkuCharging

Returns the status of the USV.

HRESULT AkkuCharging(
    [out, retval] VARIANT_BOOL* pVal
);

Parameter

pVal
    [out, retval]
    TRUE if Akku charging. FALSE if not.

Comments

Property is "Read only"

Sample VB-Syntax

Dim bAkkuCharging as Boolean
bAkkuCharging = CP9030W9x1.AkkuCharging
AkkuNotPresent

Returns the status of the USV.

HRESULT AkkuNotPresent(
    [out, retval] VARIANT_BOOL* pVal
);

Parameters

pVal
    [out, retval]
    TRUE if not present. FALSE if present

Comments

Property is "Read only"

Sample VB-Syntax

Dim bAkkuNotPresent as Boolean
bAkkuNotPresent = CP9030W9x1.AkkuNotPresent
### AkkuVoltageOk

Returns the status of the USV.

```csharp
HRESULT AkkuVoltageOk(
    [out, retval] VARIANT_BOOL* pVal
);
```

**Parameter**

- `pVal`  
  `[out, retval]` TRUE if voltage Ok. FALSE if not.

**Comments**

Property is "Read only"

**Sample VB-Syntax**

```vbnet
Dim bAkkuVoltageOk as Boolean
bAkkuVoltageOk = CP9030W9x1.AkkuVoltageOk
```
AkkuWaiting

Returns the status of the USV charger.

HRESULT AkkuWaiting(
    [out, retval] VARIANT_BOOL* pVal
);

Parameter

pVal
    [out, retval]
    TRUE or FALSE

Comments

Property is "Read only"

Sample VB-Syntax

Dim bAkkuWaiting as Boolean
bAkkuWaiting = CP9030W9x1.AkkuWaiting
CnfErr

Returns the status of the CP configuration.

HRESULT CnfErr(
    [out, retval] VARIANT_BOOL* pVal
);

Parameters

pVal
    [out, retval]
    TRUE or FALSE

Comments

Property is "Read only"

Sample VB-Syntax

Dim bCnfErr as Boolean
bCnfErr = CP9030W9x1.CnfErr
ComErr

Returns the status of the CP communication.

```csharp
HRESULT ComErr(
    [out, retval] VARIANT_BOOL* pVal
);
```

**Parameters**

- **pVal**
  - [out, retval]
  - TRUE if error existing, otherwise FALSE.

**Comments**

Property is "Read only"

**Sample VB-Syntax**

```vbnet
Dim bComErr as Boolean
bComErr = CP9030W9x1.ComErr
```
CycleEnabled

Sets / returns the status of cycle to check status of e.g. SKeys.

CycleEnabled(
    [out, retval] VARIANT_BOOL* pVal
);

HRESULT CycleEnabled(
    [in] VARIANT_BOOL pVal
);

Parameters

pVal
TRUE or FALSE

Comment
To modify the length of interval use property "CycleInterval". Default interval is 300ms.
If "CycleEnabled" is "TRUE", CP9090-S100 will cyclic check the status of e.g. SKeys.
If status changed, an event will be fired.

Sample VB-Syntax
Dim bCycleEnabled as Boolean
CP9030W9x1.CycleEnabled = TRUE
bCycleEnabled = CP9030W9x1.CycleEnabled
CycleInterval

Sets / returns the length of cycle interval for checking the status of e.g. SKeys.

```csharp
HRESULT CycleInterval(
    [out, retval] short* pVal
);

HRESULT CycleInterval(
    [in] short pVal
);
```

**Parameters**

- **pVal**
  - Length of interval in ms

**Comment**

Activate the cycle interval with property "CycleEnabled".

**Sample VB-Syntax**

```vbnet
Dim iCycleInterval as Integer
CP9030W9x1.CycleInterval = 500
CP9030W9x1.CycleEnabled = TRUE
iCycleInterval = CP9030W9x1.CycleInterval
```
DeviceAddr
Sets/returns the address of CP9030 device.

HRESULT DeviceAddr(
    [out, retval] CP_DEVICEADDR* pVal
);

HRESULT DeviceAddr(
    [in] CP_DEVICEADDR pVal
);

Parameters
pVal
Address 0xC8000 – 0xE7800

Comment
The CP9030 device has to be configured via dip-switch SW400.

Sample VB-Syntax
Dim lDeviceAddr as CP_DEVICEADDR
CP9030W9x1.DeviceAddr = CPAddr_0xC8800
lDeviceAddr = CP9030W9x1.DeviceAddr

Definition CP_DEVICEADDR
typedef enum CP_DEVICEADDR
{
   CPAddr_0xC8000 = 0xC8000,
   CPAddr_0xC8800 = 0xC8800,
   CPAddr_0xC9000 = 0xC9000,
   CPAddr_0xC9800 = 0xC9800,
   CPAddr_0xCA000 = 0xCA000,
   CPAddr_0xCA800 = 0xCA800,
   CPAddr_0xCB000 = 0xCB000,
   CPAddr_0xCB800 = 0xCB800,
   CPAddr_0xCC000 = 0xCC000,
   CPAddr_0xCC800 = 0xCC800,
   CPAddr_0xCD000 = 0xCD000,
   CPAddr_0xCD800 = 0xCD800,
   CPAddr_0xCE000 = 0xCE000,
   CPAddr_0xCE800 = 0xCE800,
   CPAddr_0xCF000 = 0xCF000,
   CPAddr_0xCF800 = 0xCF800,
   CPAddr_0xD0000 = 0xD0000,
   CPAddr_0xD0800 = 0xD0800,
   CPAddr_0xD1000 = 0xD1000,
   CPAddr_0xD1800 = 0xD1800,
   CPAddr_0xD2000 = 0xD2000,
   CPAddr_0xD2800 = 0xD2800,
   CPAddr_0xD3000 = 0xD3000,
   CPAddr_0xD3800 = 0xD3800,
   CPAddr_0xD4000 = 0xD4000,
   CPAddr_0xD4800 = 0xD4800,
   CPAddr_0xD5000 = 0xD5000,
};
CPAddr_0xD5800 = 0xD5800,
CPAddr_0xD6000 = 0xD6000,
CPAddr_0xD6800 = 0xD6800,
CPAddr_0xD7000 = 0xD7000,
CPAddr_0xD7800 = 0xD7800,
CPAddr_0xD8000 = 0xD8000,
CPAddr_0xD8800 = 0xD8800,
CPAddr_0xD9000 = 0xD9000,
CPAddr_0xDA000 = 0xDA000,
CPAddr_0xDA800 = 0xDA800,
CPAddr_0xDB000 = 0xDB000,
CPAddr_0xDB800 = 0xDB800,
CPAddr_0xDC000 = 0xDC000,
CPAddr_0xDC800 = 0xDC800,
CPAddr_0xDD000 = 0xDD000,
CPAddr_0xDD800 = 0xDD800,
CPAddr_0xDE000 = 0xDE000,
CPAddr_0xDE800 = 0xDE800,
CPAddr_0xDF000 = 0xDF000,
CPAddr_0xDF800 = 0xDF800,
CPAddr_0xE0000 = 0xE0000,
CPAddr_0xE0800 = 0xE0800,
CPAddr_0xE1000 = 0xE1000,
CPAddr_0xE1800 = 0xE1800,
CPAddr_0xE2000 = 0xE2000,
CPAddr_0xE2800 = 0xE2800,
CPAddr_0xE3000 = 0xE3000,
CPAddr_0xE3800 = 0xE3800,
CPAddr_0xE4000 = 0xE4000,
CPAddr_0xE4800 = 0xE4800,
CPAddr_0xE5000 = 0xE5000,
CPAddr_0xE6000 = 0xE6000,
CPAddr_0xE6800 = 0xE6800,
CPAddr_0xE7000 = 0xE7000,
CPAddr_0xE7800 = 0xE7800

} CP_DEVICEADDR;
DeviceInfo

Returns information about CP9030 device.

HRESULT DeviceInfo(
    [out, retval] BSTR* pVal
);

Parameters

pVal
Type and Version of CP9030 device

Comments
Property is "Read only"

Sample VB-Syntax

Dim szInfo as String

szInfo = CP9030W9x1.DeviceInfo
DeviceOpen

Sets / returns the status of CP9030 communication port (open or close)

```csharp
HRESULT DeviceOpen(
    [out, retval] VARIANT_BOOL* pVal
);

HRESULT DeviceOpen(
    [in] VARIANT_BOOL pVal
);
```

**Parameters**

- **pVal**
  
  TRUE if open, FALSE if not

**Comment**

You have to set the property "DeviceAddr" before setting "DeviceOpen" to TRUE. If this function failed, check the property "DeviceAddr".

**Sample VB-Syntax**

```vbnet
Dim bDeviceOpen as Boolean

CP9030W9x1.DeviceOpen = True
bDeviceOpen = CP9030W9x1.DeviceOpen
```
DisplayOff

Sets / returns the status of the display connected to CP9030.

HR\text{RESULT} \text{DisplayOff}(
    \begin{align*}
    &\text{[\textit{out, retval}] VARIANT\_BOOL*} \ pVal \\
    &{} \text{);}
\end{align*}
)

HR\text{RESULT} \text{DisplayOff}(
    \begin{align*}
    &\text{[\textit{in}] VARIANT\_BOOL} \ pVal \\
    &{} \text{);}
\end{align*}
)

Parameter

\textit{pVal}

TRUE or FALSE

Sample VB-Syntax

\textit{Dim bDisplayOff as Boolean}

\textit{CP9030W9x1.DisplayOff = True}
\textit{bDisplayOff = CP9030W9x1.DisplayOff}
EnableUPS

Enables / disables UPS features of CP9030.

```c
HRESULT EnableUPS(
    [out, retval] VARIANT_BOOL* pVal
);

HRESULT EnableUPS(
    [in] VARIANT_BOOL pVal
);
```

**Parameters**

*pVal*

TRUE or FALSE

**Sample VB-Syntax**

```vbnet
Dim bEnableUPS as Boolean

CP9030W9x1.bEnableUPS = True
bEnableUPS = CP9030W9x1.EnableUPS
```
ExtVoltageOk

Returns the status of the external CP9030-voltage.

```c
HRESULT ExtVoltageOk(
    [out, retval] VARIANT_BOOL* pVal
);
```

Parameters

- **pVal**
  TRUE if external voltage is OK. FALSE if not OK.

Comments

Property is "Read only"

Sample VB-Syntax

```vb
Dim bExtVoltageOk as Boolean

bExtVoltageOk = CP9030W9x1.ExtVoltageOk
```
IdentSwitch

Returns the settings of Ident-Switch SW500 of CP9030.

HRESULT IdentSwitch(
    [out, retval] short* pVal
);

Parameters

pVal
  [out, retval]
  Range 0..15

Comments

Property is "Read only"

Sample VB-Syntax

Dim iIdentSwitch as Integer

iIdentSwitch = CP9030W9x1.IdentSwitch
KbdOff

Sets / returns the status of keyboard connected to CP9030.

```c
HRESULT KbdOff(
    [out, retval] VARIANT_BOOL* pVal
);

HRESULT KbdOff(
    [in] VARIANT_BOOL pVal
);
```

**Parameters**

pVal

- TRUE = hidden keyboard.
- FALSE enables keyboard

**Comment**

To use this feature of locking the keyboard you have to close jumper J300 on CP9030 device first.

**Sample VB-Syntax**

```vbnet
Dim bKbdOff as Boolean
...
CP9030W9x1.KbdOff = True
bKbdOff = CP9030W9x1.KbdOff
```
KbusErr

Returns the error status of Kbus communication.

HRESULT KbusErr(
    [out, retval] VARIANT_BOOL* pVal
);

Parameters

pVal
    TRUE if KBus error existing. FALSE if no error

Comment

Property ist "Read only"

Sample VB-Syntax

Dim bKbusErr as Boolean

bKbusErr = CP9030W9x1.KbusErr
Led

Sets / returns the status of LED on control panel connected to CP9030.

```c
HRESULT Led(
    [in] short nIndex,
    [out, retval] VARIANT_BOOL* pVal
);

HRESULT Led(
    [in] short nIndex,
    [in] VARIANT_BOOL pVal
);
```

**Parameters**

- **nIndex**
  - Number of LED (1..27)

- **pVal**
  - TRUE or FALSE

**Sample VB-Syntax**

```vbnet
Dim bLed07 as Boolean
CP9030W9x1.Led(7) = True
bLed07 = CP9030W9x1.Led(7)
```
PdCycles

Returns the number of process-data communications of CP9030.

HRESULT PdCycles(
    [out, retval] short* pVal
);

Parameters

pVal
    0..255 (cyclic)

Comment
- Property is "Read only"
- CP9030 increments this value for each process data-cycle.

Sample VB-Syntax
Dim iPdCycles as Integer
iPdCycles = CP9030W9x1.PdCycles
PDLenErr

Returns the error status for length of output-process-data.

HRESULT PDLenErr(
    [out, retval] VARIANT_BOOL* pVal
);

Parameters

pVal
    TRUE if error, FALSE if no error

Comment
- Property is "Read only"
- If this error is existing, check the property "PDOutWLen" (see although method .Reset)

Sample VB-Syntax

Dim bPDLenErr as Boolean

bPDLenErr = CP9030W9x1.PDLenErr
PDInWLen
Returns the length of process-input-image of CP9030.

HRESULT PDInWLen(
    [out, retval] short* pVal
);

Parameters
pVal   Number of words.

Comment
Property ist "Read only"

Sample VB-Syntax
Dim iPDIInWLen as Integer
iPDIInWLen = CP9030W9x1.PDInWLen
**PDOutWLen**

Sets / returns the length of process output-image of CP9030.

```c
HRESULT PDOutWLen(
    [out, retval] short* pVal
);
```

```c
HRESULT PDOutWLen(
    [in] short pVal
);
```

**Parameters**

pVal  
Number of words.

**Comment**

Property `PDOutWLen` signals an error of CP9030, so you have to set property `PDOutWLen` to correct word-length of process-output-image. Calling the method `Reset` will initiate a reset of CP9030-firmware.

**Sample VB-Syntax**

```vbnet
Dim iPDOOutWLen as Integer
iPDOOutWLen = CP9030W9x1.PDOOutWLen
```
SKey

Returns the status of specified SKey on control panel connected to CP9030.

```c
HRESULT SKey(
    [in] short nIndex,
    [out, retval] VARIANT_BOOL* pVal
);
```

**Parameters**

- `nIndex`: Number of SKey, 1..27
- `pVal`: TRUE if SKey pressed. FALSE if not pressed

**Comment**

You can poll the state of an SKey by reading the property "SKey". If you set "CycleEnabled" to TRUE, SKey-events will be fired if state of an SKey changed.

**Sample VB-Syntax**

```vbnet
Dim bSKey01 as Boolean
bSKey09 = CP9030W9x1.SKey(9)
```
## UpsDelay

Sets / returns the time for the delay before the UPS switches off.

```c
HRESULT UpsDelay(
    [out, retval] CP_UPSDELAY* pVal
);

HRESULT UpsDelay(
    [in] CP_UPSDELAY pVal
);
```

### Parameters

- **pVal**
  - Index with the delay time for the UPS to switch off.

<table>
<thead>
<tr>
<th>Index</th>
<th>Delay in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>7</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>120</td>
</tr>
<tr>
<td>10</td>
<td>150</td>
</tr>
<tr>
<td>11</td>
<td>180</td>
</tr>
<tr>
<td>12</td>
<td>210</td>
</tr>
<tr>
<td>13</td>
<td>240</td>
</tr>
<tr>
<td>14</td>
<td>270</td>
</tr>
<tr>
<td>15</td>
<td>300</td>
</tr>
</tbody>
</table>

### Example in VB syntax

```vbnet
Dim lUpsDelay as CP_UPSDELAY

CP9030W9x1.UpsDelay = CPUpsDelay_10
lUpsDelay = CP9030W9x1.UpsDelay
```
Definition of the CP_UPSDELAY constants:

```c
typedef
enum CP_UPSDELAY
{
    CPUpsDelay_0 = 0,
    CPUpsDelay_2 = 0x1,
    CPUpsDelay_5 = 0x2,
    CPUpsDelay_10 = 0x3,
    CPUpsDelay_15 = 0x4,
    CPUpsDelay_25 = 0x5,
    CPUpsDelay_40 = 0x6,
    CPUpsDelay_60 = 0x7,
    CPUpsDelay_90 = 0x8,
    CPUpsDelay_120 = 0x9,
    CPUpsDelay_150 = 0xa,
    CPUpsDelay_180 = 0xb,
    CPUpsDelay_210 = 0xc,
    CPUpsDelay_240 = 0xd,
    CPUpsDelay_270 = 0xe,
    CPUpsDelay_300 = 0xf
} CP_UPSDELAY;
```
5. Methods

AutoCfgPDOutWLen

The method permits the length of the process output image of the CP9030 to be set automatically.

```csharp
HRESULT AutoCfgPDOutWLen(
    [out, retval] VARIANT_BOOL* pVal
);
```

**Parameters**

- **pVal**: TRUE if successful, otherwise FALSE

**Remark**

If the property "PDLenErr" displays an error condition of the CP9030, it is necessary to enter the actual word length of the process output image via the property "PDOutWLen". By calling the method "AutoCfgPDOutWLen", the word length of the process output image is automatically configured and set. If the property "PDOutWLen" is set manually, it is necessary subsequently to perform a "Reset" of the CP9030 firmware. Calling the "Reset" method will initiate a restart of the CP9030 firmware, and the new configuration will be adopted.

The AutoCfgPDOutWLen method autonomously determines the length of the process output image by executing the following cycle of actions:

- incrementing the word length of the process output image (-> "PDOutWLen")
- initiating a restart of the CP9030 firmware (-> "Reset")
- checking the error state (-> "PDLenErr")

**Example in VB syntax**

```vbnet
Dim bResult as Boolean

bResult = CP9030W9x1.AutoCfgPDOutWLen
```
ReadPhysData

Returns the value from DP-Ram of CP9030.
Use this method to access additional devices e.g. Poti.

HRESULT ReadPhysData(
    [in] long PhysOffset,
    [in, out] VARIANT* pVal
);

Parameter

PhysOffset
    [in]
    Offset in DPram (Input: 0x0000 .. 0x03FF)

pData
    [in, out]
    Databuffer, following variant-types are implemented:
            (VT_BYREF und VT_UI1)
            (VT_BYREF und VT_I2)
            (VT_BYREF und VT_I4)
            (VT_BYREF und VT_R4)
            (VT_BYREF und VT_R8)

Comment

You should be familiar with DP-Ram of CP9030 !!

Sample VB-Syntax

Dim iData as Integer
Dim lData as Long
Dim bData as Byte

call CP9030W9x1.ReadPhysData(&H104&, lData)' Read 4 bytes

call CP9030W9x1.ReadPhysData(&H104&, iData)' Read 2 bytes

call CP9030W9x1.ReadPhysData(&H104&, bData)' Read 1 bytes
WritePhysData

Sets the value in DP-Ram of CP9030.
Use this method to access additional devices e.g. Poti.

HRESULT WritePhysData(
    [in] long PhysOffset,
    [in, out] VARIANT* pData
);

Parameters

PhysOffset
[in]
Offset in DPRam (Input: 0x0000 .. 0x00FF)

pData
[in, out]
Databuffer, following variant-types are implemented:
(VT_BYREF und VT_UI1)
(VT_BYREF und VT_I2)
(VT_BYREF und VT_I4)
(VT_BYREF und VT_R4)
(VT_BYREF und VT_R8)

Comment
You should be familiar with DP-Ram of CP9030 !!

Sample VB-Syntax

Dim iData as Integer
Dim lData as Long
Dim bData as Byte
Call CP9030W9x1.WritePhysData(&H0&, lData)' Write 4 bytes
Call CP9030W9x1.WritePhysData(&H0&, iData)' Write 2 bytes
Call CP9030W9x1.WritePhysData(&H0&, bData)' Write 1 bytes
Reset

Executes a reset of CP9030-firmware.

HRESULT Reset();

Comment
See: Property „PDLenErr“

Sample VB-Syntax
CP9030W9x1.Reset
TriggerCheckOnChange

With this method you can trigger cyclic checking of the status of the SKeys or of the process data image (e.g. with a multimedia timer).

HRESULT TriggerCheckOnChange();

Parameters

None

Remark

Every time the trigger method is called, then if the status of the process data input image or of the SKeys has changed, the corresponding event - SKeyDown, SkeyPress or SKeyUp - is called.

Example in VB syntax

Private Sub Timer1_Timer()
    Call BkCp9030W9x1.TriggerCheckOnChange
End Sub
6. Events

SKeyDown

Occurs once when the user presses a SKey and the property "CycleEnabled" is TRUE.

HRESULT SkeyDown(
    [in] short Index
);

Parameters

Index
Number of SKey, 1..27

Comment

You can poll the state of an SKey by reading the property "SKey". If you set "CycleEnabled" to TRUE, SKey-
events will be fired if state of an SKey changed.

Sample VB-Syntax

Sub CP903GW9x1_SkeyDown(ByVal Index As Integer)
    ' add code here
End Sub
SKeyPress

Occurs while the user presses a SKey and the property "CycleEnabled" is TRUE.

HRESULT SkeyPress(
    [in] short Index
);

Parameters

Index
Number des SKey, 1..27

Comment
You can poll the state of an SKey by reading the property "SKey". If you set "CycleEnabled" to TRUE, SKey-
events will be fired if state of an SKey changed.

Sample VB-Syntax

Sub CP9030W9x1_SkeyPress(ByVal Index As Integer)
    ' add code here
End Sub
**SKeyUp**

Occurs once when the user releases a SKey and the property "CycleEnabled" is TRUE.

```csharp
HRESULT SKeyUp(
    [in] short Index
);
```

**Parameters**

Index  
Number des SKey, 1..27

**Comment**

You can poll the state of an SKey by reading the property "SKey". If you set "CycleEnabled" to TRUE, SKey-events will be fired if state of an SKey changed.

**Sample VB-Syntax**

```vb
Sub CP9030W9x1_SkeyUp(ByVal Index As Integer)
    ' add code here
End Sub
```
OnChangeInImage

Occurs once when the process input image changes.

```csharp
HRESULT OnChangeInImage(
    [in] long PhysOffset
);
```

**Parameters**

PhysOffset

Byte-Offset of input image of CP9030 (0x0000 ... 0x01FF) which changed

**Comment**

To avoid unnecessary cyclic polling of e.g. SKey-status, you can implement event-driven code in this event.

**Sample VB-Syntax**

```vbnet
Sub CP9030W9x1_OnChangeInImage(ByVal PhysOffset as Long)
    ' add code here
End Sub
```
7. Samples

Overview

<table>
<thead>
<tr>
<th>Description</th>
<th>Sourcecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1: Integration into Visual Basic</td>
<td>Sample01.exe</td>
</tr>
</tbody>
</table>