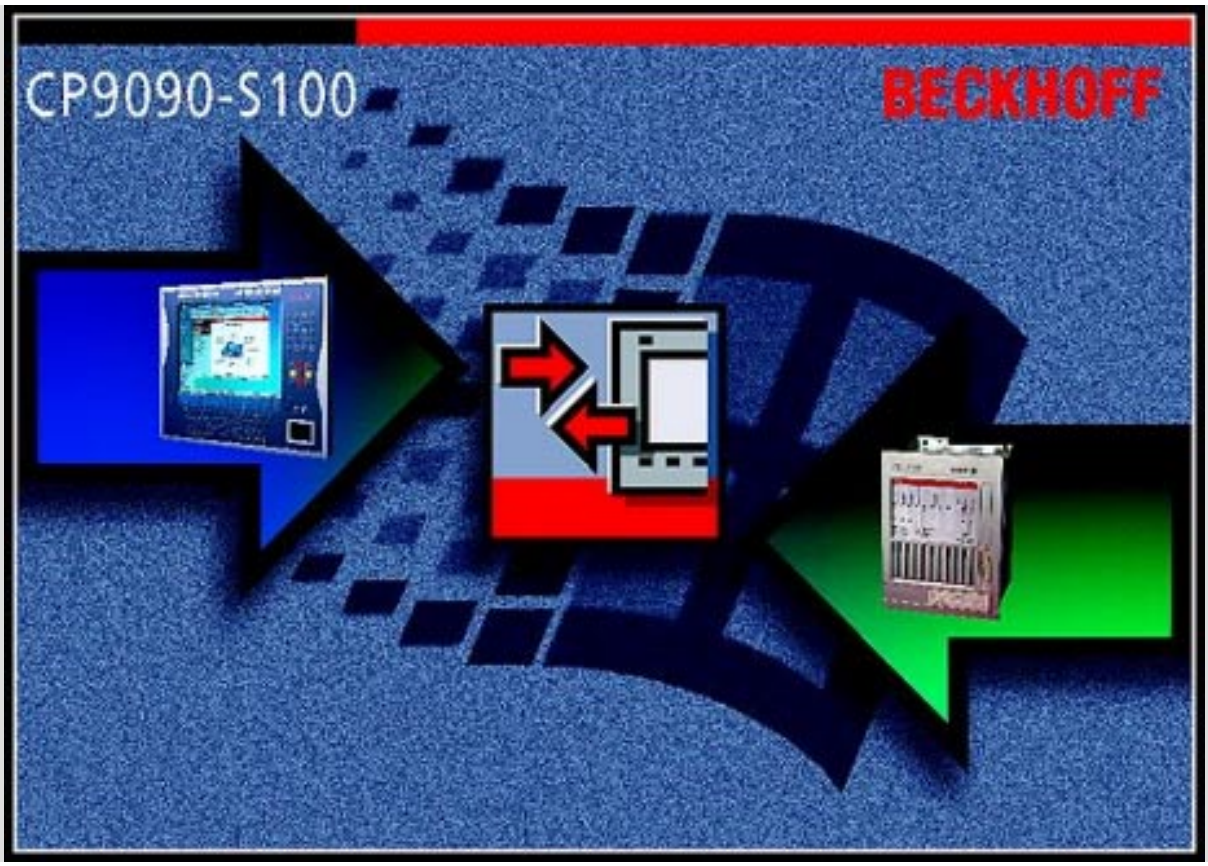



# Beckhoff® TwinCAT

The Windows Control and Automation Technology

# CP9090-S100: ActiveX Komponente für CP9030

Letzte Änderung: 18.01.2001

BECKHOFF

# Inhaltsverzeichnis

## CP9090-S100: ActiveX Komponente für CP9030

# 1. Übersicht

Die CP9090-S100 ActiveX Komponente ermöglicht den Zugriff auf die Beckhoff CP-Link Karte CP9030.

**Funktionalität**

- Konfiguration der Beckhoff CP-Link Karte CP9030
- Ansteuerung der zusätzlicher Befehlsgeräte (z.B. SKeys, LEDs, Potentiometer für Override..)
- Ansteuerung der CP-Verriegelung (z.B. Tastatur, Maus, Touch..)
- Ansteuerung der USV

**Voraussetzungen**

- Microsoft Windows 9x / Win NT / Win2K
- Bitte arbeiten Sie mit aktuellen Service Packs.

**Lieferumfang**

Die Installation beinhaltet:

- CP9090-S100 ActiveX-Komponente
- Dokumentation (PDF und HTML)
- Demo Applikation

# 2. Installation

Starten Sie die Datei "SETUP.EXE" auf der Diskette 1/x um die CP9090-S100 Komponenten unter Windows 9x / NT zu installieren.

**Dateien und Verzeichnisse**

Die folgenden Dateien werden installiert bzw. aktualisiert

WINDOWS\SYSTEM(32) Verzeichnis:

- CpLink9x.dll
- Atl.dll
- Ole32.dll
- Oleaut32.dll
- TcMM.sys
- TcMMHelper.dll
- TcW9xMMHelper.dll

WINDOWS\HELP:

- Cp9090-S100.chm

**Hilfe**

Falls die HTML-Hilfe-Datei CP9090-S100.chm "leer" erscheint, installieren Sie bitte zunächst noch die HTML-HILFE-Laufzeitdateien. Dazu starten Sie die Datei "hhupd.exe" von der CD.

Es wird vorausgesetzt, dass Sie mit der Beckhoff CP-Link Karte CP9030 vertraut sind. Weitere technische Informationen sind auf der CD oder auf der Beckhoff-Web-Seite enthalten.

**Support**

Bei Fragen oder Problemen wenden Sie sich bitte an:

Beckhoff Industrie Elektronik

Eiserstr. 5

33415 Verl

Tel: 05246-963-157

Fax: 05246-963-199

Email: mailto:info@beckhoff.com Internet: http://www.beckhoff.com

Bei Problemen mit der Software hilft eine detaillierte Beschreibung des Fehlers. Des weiteren werden Informationen über das verwendete Betriebssystems (Win9x, NTx Service Packs), Betriebssystemsprache (deutsch, englisch), etc. benötigt.

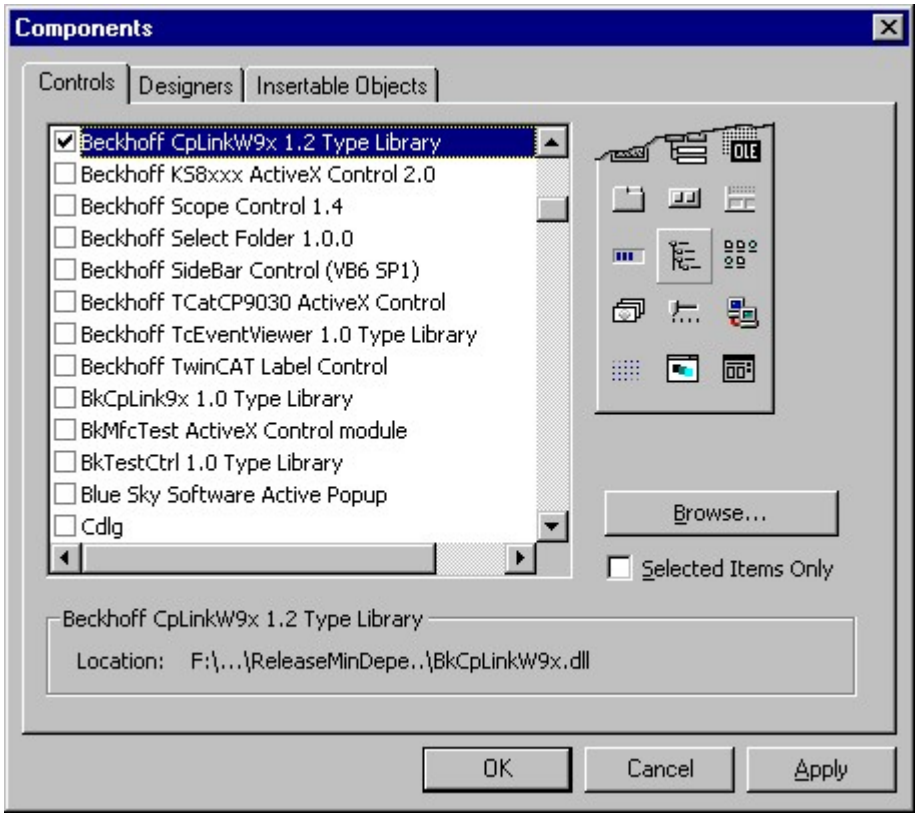
Die Versionsnummer der CP9090-S100 Komponente finden Sie in dem "Eigenschaften-Menü" der Datei CpLink9x.dll.

# 3. Einbindung in Applikationen

## Einbindung in Visual Basic

In Visual Basic kann die CP9090-S100 ActiveX Komponente eingesetzt werden.

Dazu müssen Sie in Visual Basic unter dem Menüpunkt ‚Projekt' den Befehl ‚Komponenten...' auswählen und den Eintrag 'Beckhoff CpLinkW9x x.x Type Library" markieren.




Anschließend erscheint die CP9090-S100 ActiveX Komponente in der Toolbox von Visual Basic (rechts unten).

# Einbindung Visual C++

## In Bearbeitung

# 4. Eigenschaften

## AkkuCharged

Liefert den Status des Ladereglers der USV.

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

**Parameter**

pVal
[out, retval]
TRUE wenn Akku geladen. FALSE wenn nicht.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

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

# AkkuCharging

Liefert den Status des Ladereglers der USV.

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

**Parameter**

pVal
[out, retval]
TRUE wenn Akku im Ladevorgang. FALSE wenn nicht.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bAkkuCharging as Boolean
bAkkuCharging = CP9030W9x1.AkkuCharging
```

# AkkuNotPresent

Liefert den Status des Ladereglers der USV.

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

**Parameter**

pVal
[out, retval]
TRUE wenn Akku nicht vorhanden. FALSE wenn vorhanden.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bAkkuNotPresent as Boolean
bAkkuNotPresent = CP9030W9x1.AkkuNotPresent
```

# AkkuVoltageOk

Liefert den Status der USV-Spannung.

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

**Parameter**

pVal
[out, retval]
TRUE wenn Spannung Ok. FALSE wenn nicht.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

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

# AkkuWaiting

Liefert den Status des Ladereglers der USV.

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

**Parameter**

pVal
[out, retval]
TRUE wenn "Wartend", sonst FALSE

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bAkkuWaiting as Boolean
bAkkuWaiting = CP9030W9x1.AkkuWaiting
```

# CnfErr

Liefert den Status der CP9030 Konfiguration.

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

**Parameter**

pVal
[out, retval]
TRUE wenn die Konfiguration fehlerhaft, sonst FALSE

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bCnfErr as Boolean
bCnfErr = CP9030W9x1.CnfErr
```

# ComErr

Liefert den Status der CP9030 Kommunikation.

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

**Parameter**

pVal
[out, retval]
TRUE wenn Fehler vorhanden, sonst FALSE.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

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

# CycleEnabled

Ist dieses Property gesetzt, dann wird der Status der SKeys in einem CycleInterval automatisch überprüft.

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


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

**Parameter**

pVal
TRUE oder FALSE

**Anmerkung**

Die Länge des Intervalls kann über die Eigenschaft "CycleInterval" eingestellt werden. Die Voreinstellung beträgt 300ms.
Wenn "CycleEnabled" gleich "TRUE" ist, prüft CP9090-S100 zyklisch den Status z.B. der SKeys.

Bei Änderung des Status wird ein Event abgefeuert.

**Beispiel VB-Syntax**

```
Dim bCycleEnabled as Boolean

CP9030W9x1.CycleEnabled = TRUE

bCycleEnabled = CP9030W9x1.CycleEnabled
```

# CycleInterval

Setzt / liefert die Dauer des Zeitintervalls mit welchem der Status der z.B. Skeys überprüft werden soll.

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

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

**Parameter**

pVal
Länge des Zeitintervals in ms

**Anmerkung**
Dies zyklische Überprüfung wird über das Property "CycleEnabled" aktiviert.

**Beispiel VB-Syntax**

```
Dim iCycleInterval as Integer

CP9030W9x1.CycleInterval = 500
CP9030W9x1.CycleEnabled = TRUE

iCycleInterval = CP9030W9x1.CycleInterval
```

# DeviceAddr

Setzt / liefert die Adresse der CP9030 Karte.

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


HRESULT DeviceAddr(
        [in] CP_DEVICEADDR pVal
);
```

**Parameter**

pVal
Adresse 0xC8000 – 0xE7800

**Anmerkung**
Die Adresse der CP9030 Karte wird über den Dip-Schalter SW400 eingestellt.

**Beispiel 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_0xE3000 = 0xE2800,
    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

Liefert Information über CP9030 Karte.

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

**Parameter**

pVal
Type und Version der CP9030 Karte.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim szInfo as String

szInfo = CP9030W9x1.DeviceInfo
```

# DeviceOpen

Setzt / liefert den Status des Kommunikationsports der CP9030-Karte (geöffnet oder geschlossen)

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


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

**Parameter**

pVal
TRUE um Kommunikationsport zu öffnen, FALSE um den Kommunikationsport zu schließen.

**Anmerkung**

Bevor "DeviceOpen" aufgerufen wird, muß per Property "DeviceAddr" eine korrekte Adresse eingestellt sein. Wenn der Aufruf dieses Property fehlerhaft ist, überprüfen Sie bitte das Property "DeviceAddr".

**Beispiel VB-Syntax**

```
Dim bDeviceOpen as Boolean

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

# DisplayOff

Setzt / liefert den Status des an der CP9030 angeschlossenen Displays.

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


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

**Parameter**

pVal
TRUE oder FALSE

**Beispiel VB-Syntax**

```
Dim bDisplayOff as Boolean

CP9030W9x1.DisplayOff = True
bDisplayOff = CP9030W9x1.DisplayOff
```

# EnableUPS

Aktiviert / deaktiviert UPS Funktionalitäten der CP9030.

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


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

**Parameter**

pVal
: TRUE wenn USV aktiviert werden soll, sonst FALSE

**Beispiel VB-Syntax**

```
Dim bEnableUPS as Boolean

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

# ExtVoltageOk

Liefert des Status der externen CP9030-Spannungsversorgung.

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

**Parameter**

pVal
TRUE wenn externe Versorung Ok. FALSE wenn nicht Ok.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bExtVoltageOk as Boolean

bExtVoltageOk = CP9030W9x1.ExtVoltageOk
```

# IdentSwitch

Liefert die Einstellungen des Dip-Schalters Ident-Switch SW500 der CP9030.

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

**Parameter**

pVal
[out, retval]
Wertebereich 0..15

**Anmerkung**

Eigenschaft ist nur lesend ("Read only")

Der Ident-Switch befindet sich nur auf älteren CP9030.

**Beispiel VB-Syntax**

```
Dim iIdentSwitch as Integer

iIdentSwitch = CP9030W9x1.IdentSwitch
```

# KbdOff

Setzt / liefert den Status des an der CP9030 angeschlossenen Tastatur.

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


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

**Parameter**

pVal
TRUE = Verriegelung der Tastatur aktiviert. FALSE nicht aktiviert

**Anmerkung**
Bevor Sie diese Funktionalität nutzen können, muß auf der CP9030 Karte der Junper J300 gesetzt sein.

**Beispiel VB-Syntax**

```
Dim bKbdOff as Boolean
…
CP9030W9x1.KbdOff = True
bKbdOff = CP9030W9x1.KbdOff
```

# KbusErr

Liefert den Status der Kbus Kommunikation.

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

**Parameter**

pVal
TRUE wenn Kbus Fehler anliegt. FALSE wenn kein Fehler vorliegt

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim bKbusErr as Boolean

bKbusErr = CP9030W9x1.KbusErr
```

# Led

Setzt / liefert den Status der LED auf dem an der CP9030 angeschlossenen Control Panel.

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

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

**Parameter**

nIndex
: Nummer der LED (1..27)

pVal
: TRUE oder FALSE

**Beispiel VB-Syntax**

```
Dim bLed07 as Boolean

CP9030W9x1.Led(7) = True
bLed07 = CP9030W9x1.Led(7)
```

# PdCycles

Liefert die aktuelle Anzahl der Prozeßdaten Kommunikationszyklen der CP9030-Karte.

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

**Parameter**

pVal
0..255 (zylisch, Überlauf bei 255)

**Anmerkung**
- Eigenschaft ist nur lesend ("Read only")
- CP9030 inkrementiert diesen Wert bei jedem Prozeßdatenzyklus

**Beispiel VB-Syntax**

```
Dim iPdCycles as Integer

iPdCycles = CP9030W9x1.PdCycles
```

# PDLenErr

Liefert den Status der konfigurierten Länge des Prozeßdatenabbildes der Ein- und Ausgänge.

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

**Parameter**

pVal
TRUE wenn Fehler anliegt. FALSE wenn alles Ok.

**Anmerkung**
- Eigenschaft ist nur lesend ("Read only")
- Wenn dieser Fehler anliegt, prüfen Sie die Einstellung im Property „PDOutWLen". (Siehe auch Methode „Reset" und "AutoCfgPdOutWLen" )

**Beispiel VB-Syntax**

```
Dim bPDLenErr as Boolean

bPDLenErr = CP9030W9x1.PDLenErr
```

# PDInWLen

Liefert die Länge des Prozeßdaten-Eingangsabbildes der CP9030-Karte.

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

**Parameter**

pVal
Länge in Words.

**Anmerkung**
Eigenschaft ist nur lesend ("Read only")

**Beispiel VB-Syntax**

```
Dim iPDInWLen as Integer

iPDInWLen = CP9030W9x1.PDInWLen
```

# PDOutWLen

Über dieses Property wird die Länge des Prozeß-Ausgangsabbildes der CP9030 gelesen bzw. gesetzt.

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


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

**Parameter**

pVal
Länge des Prozeßdatenausgangsabbildes in Words.

**Anmerkung**

Zeigt das Property „PDLenErr" einen Fehlerzustand der CP9030 an, so muß über das Property „PDOutWLen" die Wortlänge des Prozeßausgangsabbildes angegeben werden. Durch Aufruf der Methode „Reset" wird ein Neustart der CP9030-Firmware initiiert und die neue Konfiguration übernommen.

**Beispiel VB-Syntax**

```
Dim iPDOutWLen as Integer

iPDOutWLen = CP9030W9x1.PDOutWLen
```

# SKey

Liefert den Status der angegebenen SKey-Tasten des an die CP9030 angeschlossenen Control Panels.

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

**Parameter**

nIndex
Nummer des SKey, 1..27

pVal
TRUE wenn Taste gedrückt. FALSE wenn nicht gedrückt

**Anmerkung**

Sie können den Status der SKeys mit diesem Property "pollend" lesen.
Weiterhin haben Sie die Möglichkeit ereignisgesteuert auf die Änderung der SKeys zu reagieren. Dazu aktivieren Sie bitte "CycleEnabled" .

**Beispiel VB-Syntax**

```
Dim bSKey01 as Boolean

bSKey09 = CP9030W9x1.SKey(9)
```

# UpsDelay

Setzt / liefert die Zeit für die Ausschaltverzögerung der USV.

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

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

**Parameter**

pVal
Index mit der Verzögerungszeit für das Abschalten der USV.

| Index | Verzögerung in Sekunden |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 5 |
| 3 | 10 |
| 4 | 15 |
| 5 | 25 |
| 6 | 40 |
| 7 | 60 |
| 8 | 90 |
| 9 | 120 |
| 10 | 150 |
| 11 | 180 |
| 12 | 210 |
| 13 | 240 |
| 14 | 270 |
| 15 | 300 |

**Beispiel VB-Syntax**

```
Dim lUpsDelay as CP_UPSDELAY


CP9030W9x1.UpsDelay = CPUpsDelay_10
lUpsDelay = CP9030W9x1.UpsDelay
```

**Definition der CP_UPSDELAY-Konstanten:.**

```
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. Methoden

## AutoCfgPDOutWLen

Die Methode ermöglicht automatische Einstellung der Länge des Prozeß-Ausgangsabbildes der CP9030.

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

**Parameter**

pVal
TRUE, wenn erfolgreich, sonst FALSE

**Anmerkung**

Zeigt das Property „PDLenErr" einen Fehlerzustand der CP9030 an, so muß über das Property „PDOutWLen" die tatsächliche Wortlänge des Prozeßausgangsabbildes angegeben werden. Durch den Aufruf der Methode "AutoCfgPDOutWLen" wird die Wortlänge des Prozeßausgangsabbildes automatisch konfiguriert und gesetzt. Wird das Property "PDOutWLen" manuell gesetzt, dann muß anschließend ein "Reset" der CP9030-Firmware durchgeführt werden. Durch Aufruf der Methode „Reset" wird ein Neustart der CP9030-Firmware initiiert und die neue Konfiguration übernommen.

Die Methode AutoCfgPDOutWLen ermittelt die Länge des Prozeßausgangsabbildes eigenständig durch folgenden Zyklus

- incrementieren der Wortlänge des Prozeßausgangsabbildes (-> „PDOutWLen")
- initiieren eines Neustart der CP9030 Firmware (-> „Reset")
- prüfen des Fehlerzustand (-> „PDLenErr")

**Beispiel VB-Syntax**

```
Dim bResult as Boolean


bResult = CP9030W9x1.AutoCfgPDOutWLen
```

# ReadPhysData

Liest einen Wert vom DPRam der CP9030-Karte.
Benutzen Sie diese Methode für den Zugriff auf zusätzliche Befehlsgeräte z.B. das Potentiometer.

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

**Parameter**

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

pData
[in, out]
Datenpuffer, als Variant wird akzeptiert:
(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)

**Anmerkung**
Sie sollten mit dem Aufbau des DP-Ram vertraut sein !!

**Beispiel 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

Setzt den Wert im DP-Ram der CP9030.
Benutzen Sie diese Methode zum Zugriff auf zusätzliche Befehlsgeräte z.B. das Potentiometer.

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

**Parameter**

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

pData
[in, out]
Datenpuffer, als Variant wird akzeptiert:
(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)

**Anmerkung**
Sie sollten mit dem Aufbau des DP-Ram vertraut sein !!

**Beispiel VB-Syntax**

```
Dim iData as Integer
Dim lData as Long
Dim bData as Byte
call CP9030W9x1.WritePhysData(&H000&, lData)' Write 4 bytes
call CP9030W9x1.WritePhysData(&H000&, iData)' Write 2 bytes
call CP9030W9x1.WritePhysData(&H000&, bData)' Write 1 bytes
```

# Reset

Führt einen Neustart der CP9030-Firmware durch.

```
HRESULT Reset();
```

**Anmerkung**

Siehe auch: Property „PDLenErr"

**Beispiel VB-Syntax**

```
CP9030W9x1.Reset
```

# TriggerCheckOnChange

Mit der Methode können Sie die zyklische Überprüfung des Status der SKeys bzw. des Prozessdaten-Images (z.B. mit einem Multimedia-Timer) triggern.

```
HRESULT TriggerCheckOnChange();
```

**Parameter**
Keine

**Anmerkung**
Jedes Mal, wenn die Trigger-Methode aufgerufen wurde, werden beim geänderten Status des Prozessdaten-eingangs-Images bzw. der SKeys die entsprechenden Events SKeyDown, SKeyPress, SKeyUp aufgerufen.

**Beispiel VB-Syntax**

```
Private Sub Timer1_Timer()

    Call BkCp9030W9x1.TriggerCheckOnChange

End Sub
```

# 6. Events

## SKeyDown

Wird einmal abgefeuert, wenn der Bediener eine SKey-Taste drückt und das Property "CycleEnabled" aktiviert wurde.

```
HRESULT SkeyDown(
        [in] short Index
);
```

**Parameter**

Index
Nummer des SKey, 1..27

Beispiel VB-Syntax

```
Sub CP9030W9x1_SkeyDown(ByVal Index As Integer)
    ' add code here

End Sub
```

# SKeyPress

Wird solange abgefeuert, wie der Bediener eine SKey-Taste gedrückt hält. Aktiv nur dann, wenn das Property "CycleEnabled" aktiviert wurde.

```
HRESULT SkeyPress(
        [in] short Index
);
```

**Parameter**

Index
Nummer des SKey, 1..27

**Beispiel VB-Syntax**

```
Sub CP9030W9x1_SkeyPress(ByVal Index As Integer)
    ' add code here

End Sub
```

# SKeyUp

Wird einmal abgefeuert, wenn der Bediener eine SKey losläßt und das Property "CycleEnabled" aktiviert wurde.

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

**Parameter**

Index
Nummer des SKey, 1..27

**Beispiel VB-Syntax**

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

End Sub
```

# OnChangeInImage

Wird jedesmal abgefeuert, wenn sich das Eingangsabbild der CP9030 geändert hat.

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

**Parameter**

PhysOffset
Byte-Offset im Eingangsabbild der CP9030 (0x0000 … 0x01FF) welches sich verändert hat

**Anmerkung**
Um unnötiges Pollen des Status der z.B. SKey-Tasten zu vermeiden, können Sie Ihre Applikation ereignisgesteuert implementieren.

**Beispiel VB-Syntax**

```
Sub CP9030W9x1_OnChangeInImage(ByVal PhysOffset as Long)
    ' add code here

End Sub
```

# 7. Beispiel

## Übersicht

| Beschreibung | Quelltexte |
|---|---|
| Beispiel 1: Einbindung in Visual Basic | Sample01.exe |