

## Benutzerhandbuch PCI-Zählerkarte ZP044

Heilig & Schwab GmbH  
Haystraße 24  
D-55566 Bad Sobernheim  
Telefon: +49 (0) 67 51 / 93 12-0  
Telefax: +49 (0) 67 51 / 62 07  
E-mail: [info@heilig-schwab.de](mailto:info@heilig-schwab.de)  
Internet: [www.heilig-schwab.de](http://www.heilig-schwab.de)

Diese Dokumentation darf weder als Ganzes noch in Auszügen vervielfältigt, an Dritte weitergegeben, in einer Datenbank gespeichert oder in eine andere Sprache übersetzt werden ohne schriftliche Genehmigung der Heilig & Schwab GmbH.

© Copyright 2004 - 2005 Heilig & Schwab GmbH. Alle Rechte vorbehalten.

Zweite Ausgabe: Bad Sobernheim, 04. Februar 2005

Die in diesem Dokument enthaltenen Informationen können ohne vorherige Mitteilung geändert werden. Die Firma Heilig & Schwab GmbH geht damit keinerlei Verpflichtungen ein.

Heilig & Schwab GmbH übernimmt keine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden, die auf den Gebrauch oder den Inhalt dieses Benutzerhandbuches zurückzuführen sind.

Weiterhin sei darauf hingewiesen, dass die Heilig & Schwab GmbH keine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Layout oder Design der Hardware können ohne vorherige Mitteilung geändert werden. Die Firma Heilig & Schwab GmbH geht damit keinerlei Verpflichtungen ein.

Alle anderen in diesem Handbuch verwendeten Warenzeichen und Produktbezeichnungen sind Eigentum der entsprechenden Firmen und Hersteller. Heilig & Schwab GmbH verzichtet auf alle Besitzrechte an den genannten Warenzeichen und Produktbezeichnungen, die nicht ihr Eigentum sind.

# Inhalt

<b>1</b>	<b>WICHTIGE HINWEISE</b>	<b>5</b>
1.1	ÜBEREINSTIMMUNGSERKLÄRUNG	5
	Hersteller- und Produktname	5
	EMV-Spezifikationen	5
1.2	VERWENDUNGSZWECK	6
1.3	LIEFERUMFANG UND SYSTEMVORAUSSETZUNG	6
	Lieferumfang	6
	Systemvoraussetzungen	6
<b>2</b>	<b>INSTALLATION</b>	<b>7</b>
2.1	EINBAU DER ZÄHLERKARTE	7
2.2	INSTALLATION DER TREIBER	7
<b>3</b>	<b>KONFIGURATION</b>	<b>8</b>
3.1	BASIS-BIBLIOTHEKSFUNKTIONEN (HS_ZP3B.DLL)	8
	Karte initialisieren	9
	Treiber-Version abfragen	9
	Zähler einzeln auslesen	10
	Alle Zähler auslesen	11
	Referenzmodus setzen	12
	Alle Zähler latchen	13
	I <sup>2</sup> C-Clockleitung setzen	13
	I <sup>2</sup> C-Datenleitung setzen	14
	I <sup>2</sup> C-Datenleitung lesen	15
	Latch-Signal auslösen	15
	Zähler-Latch-Matrix konfigurieren	16
3.2	ERWEITERTE BIBLIOTHEKSFUNKTIONEN (HS_ZP3X.DLL)	17
	Karte initialisieren	18
	Treiber-Version abfragen	18
	Referenz- und Latchmodus setzen	19
	Multiplikator für Zählerwert setzen	21
	Zähler einzeln auslesen	22
	Alle Zähler auslesen	23
	Latcheingang zurücksetzen	25
	Referenz zurücksetzen	25
	Zähler ununterbrochen einzeln auslesen	26
	Alle Zähler ununterbrochen auslesen	27
	Abstandscodierung konfigurieren	29
	Alle Zähler latchen	30
	I <sup>2</sup> C-Clockleitung setzen	30
	I <sup>2</sup> C-Datenleitung setzen	31
	I <sup>2</sup> C-Datenleitung lesen	32
	Latch-Signal auslösen	32
	Zähler-Latch-Matrix konfigurieren	33
3.3	DLLs EINBINDEN	34

<b>4</b>	<b>TECHNISCHE DATEN</b>	<b>35</b>
<b>4.1</b>	<b>STECKERANORDNUNG</b>	<b>35</b>
<b>4.2</b>	<b>STECKERBELEGUNG</b>	<b>36</b>
	Signaleingänge Achse 1 – 3 (P400 / P401 / P402 <sub>Slotblech ZP044</sub> )	36
	Slotblech-Anschluss Achse 3(P402)	37
	Signaleinspeisung / Signalabgriff (P403)	37
	Interner Latch-Ein/Ausgang (P404)	38
	Jumper (P405)	38
	Latchsignale Achse 1 – 3 (P406)	38
	Analoge Signale Achse 1 und Achse 2 (P100 / P101)	39
<b>4.3</b>	<b>PHYSIKALISCHE UND MECHANISCHE KENNWERTE</b>	<b>39</b>
<b>5</b>	<b>GARANTIEBEDINGUNGEN</b>	<b>40</b>

# 1 Wichtige Hinweise

## 1.1 Übereinstimmungserklärung

### Hersteller- und Produktname

Hersteller: Heilig & Schwab GmbH  
Haystraße 24  
D-55566 Bad Sobernheim

Produkt: kundenspezifische PCI-Zählerkarte

Modell: ZP044

Kunde: Helios Meßtechnik GmbH & Co. KG  
Bachwiesenstraße 6  
D-74676 Niedernhall

### EMV-Spezifikationen

Die Zählerkarte ZP044 darf nur in Messsystemen der Firma Helios eingesetzt werden, für welche eine Prüfung des Gesamt-Messsystems (Messsystem mit integrierter Zählerkarte ZP044) erfolgt ist.



Der Betrieb mit nicht zertifizierten Personal Computern oder nicht korrekt abgeschirmten Kabeln oder nicht ordnungsgemäß montierter Zählerkarte kann zu elektromagnetischen Störungen führen.

## 1.2 Verwendungszweck

Die Zählerkarte ZP044 ist eine PC-Einsteckkarte für den PCI-Steckplatz. Sie ist eine kundenspezifische Entwicklung auf Basis der PCI-Zählerkarte ZP051 und wurde für die Helios Meßtechnik GmbH & Co. KG, Bachwiesenstraße 6 in D-74676 Niedernhall entwickelt.

Die Zählerkarte ZP044 darf nur in Messsystemen der Firma Helios eingesetzt werden, für welche eine Prüfung des Gesamt-Messsystems (Messsystem mit integrierter Zählerkarte ZP044) erfolgt ist.

## 1.3 Lieferumfang und Systemvoraussetzung

### Lieferumfang

Im Lieferumfang zur Zählerkarte sind enthalten:

Pos.	Menge	Bezeichnung	Hinweis
1	1 St.	PC-Zählerkarte ZP044	
2	1 St.	Datenträger mit <ul style="list-style-type: none"><li>• Benutzerhandbuch</li><li>• Treiber und DLLs für Windows-Betriebssysteme</li></ul>	Wird bei Rahmenaufträgen einmalig geliefert, ist aber bei Bedarf jederzeit erhältlich.

Bitte überprüfen Sie direkt nach Erhalt der Lieferung den Inhalt des Pakets.

Falls Sie Abweichungen zum oben beschriebenen Lieferumfang feststellen, setzen Sie sich bitte umgehend mit uns in Verbindung.

Tel.: +49 (0) 67 51 / 93 12-30

### Systemvoraussetzungen

Für den Betrieb der PC-Zählerkarte ZP044 werden an Ihren Computer und Ihre Software die folgenden Systemanforderungen gestellt:

- Pentium-PC oder ein 100% kompatibles System
- Windows 3.11/Win32s und Windows 98 und höher
- Ein freier PCI-Steckplatz
- VGA-Monitor

## 2 Installation


Die Zählerkarte ZP044 darf nur von entsprechend geschultem Personal eingebaut und installiert werden.

Es wird ferner vorausgesetzt, dass vor Installation der Karte dieses Benutzerhandbuch gelesen wurde und die jeweiligen Sicherheits- und Bedienhinweise beachtet werden.

### 2.1 Einbau der Zählerkarte

Beachten Sie bitte im Umgang mit der Karte die allgemein üblichen Sicherheitsvorkehrungen bezüglich elektrostatischer Entladung. Im Zweifelsfalle entladen Sie sich durch Anfassen einer Erdverbindung, z.B. geerdete Gerätegehäuse, Heizkörper o.ä..

Beim Einsetzen der Zählerkarte in Ihren PC gehen Sie im Einzelnen wie folgt vor:

-  1. Schalten Sie den Rechner aus und ziehen Sie den Netzstecker.
2. Öffnen bzw. entfernen Sie das Gehäuse des Rechners.
3. Wählen Sie einen freien PCI-Steckplatz und entfernen Sie dessen Abdeckblech (Bracket) an der Rückseite des Rechners.
4. Setzen Sie nun die Zählerkarte ZP044 in den Steckplatz auf dem Motherboard ein. - Achten Sie darauf, dass die Karte gerade sitzt und keine benachbarte Karte berührt wird. Die Signal-Anschlussbuchsen müssen außen frei zugänglich sein.
5. Befestigen Sie nun die Zählerkarte mit einer Schraube an der hierfür vorgesehenen Bohrung in der Rückwand des Rechners.
6. Stellen Sie alle erforderlichen Anschlüsse her.
7. Schließen Sie das Rechnergehäuse ordnungsgemäß.
8. Der Rechner kann nun wieder eingeschaltet werden.

Zum Betreiben der Zählerkarte benötigen Sie jetzt die auf dem Datenträger mitgelieferte Treiber-Software für Ihr Betriebssystem.

### 2.2 Installation der Treiber

Windows 98 und höher erkennt die Zählerkarte nach deren Einbau automatisch und fordert die Installation des Treibers und der DLLs.

Nach dem Einlegen des Datenträgers erfolgt die Installation selbständig.

### 3 Konfiguration

Für den Zugriff auf den Treiber der Zählerkarte werden zwei DLLs mitgeliefert. Die DLL "HS\_ZP3B.DLL" mit Basis-Bibliotheksfunktionen und die DLL "HS\_ZP3X.DLL" mit erweiterten Bibliotheksfunktionen.

#### 3.1 Basis-Bibliotheksfunktionen (HS\_ZP3B.DLL)

Die Basis-DLL "HS\_ZP3B.DLL" enthält die folgenden Aufrufe:

DLL Funktionsaufruf	Bedeutung
HS_ZP3B_Init	Lädt Treiber und DLL und prüft, ob die Zählerkarte im PC vorhanden ist. Initialisiert die Zählerkarte.
HS_ZP3B_GetVersion	Überprüft die Versionen von Treiber und DLL.
HS_ZP3B_ReadCounter	Liest den Zählerwert und den Status einer Achse (eines Zählers) der Zählerkarte aus.
HS_ZP3B_ReadAllCounter	Liest den Zählerwert und den Status aller Achsen der Zählerkarte aus.
HS_ZP3B_SetRefMode	Setzt den Referenzmodus einer Achse.
HS_ZP3B_StopAllCounter	Latcht alle Zählerwerte aller Zählerkarten.
HS_ZP3B_I2C_SetClock	Setzt die I <sup>2</sup> C-Clockleitung einer Achse.
HS_ZP3B_I2C_SetData	Setzt die I <sup>2</sup> C -Datenleitung einer Achse.
HS_ZP3B_I2C_GetData	Liest die I <sup>2</sup> C -Datenleitung einer Achse.
HS_ZP3B_Stop	Löst ein Latch-Signal aus.
HS_ZP3B_SetStopCnf	Konfiguriert die Zähler-Latch-Matrix.



## Karte initialisieren

<b>Funktionsname:</b>	<b>HS_ZP3B_Init</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_Init (long CardNo) <b>Delphi:</b> HS_ZP3B_Init (CardNo :longint) :longint
<b>Beschreibung:</b>	Lädt Treiber und DLL und prüft, ob die Zählerkarte im PC vorhanden ist. Initialisiert die Zählerkarte.
<b>Übergabe-parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## Treiber-Version abfragen

<b>Funktionsname:</b>	<b>HS_ZP3B_GetVersion</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_GetVersion (char* DriverVersion, char* DIIVersion) <b>Delphi:</b> HS_ZP3B_GetVersion (DriverVersion :pchar, DIIVersion :pchar) :longint
<b>Beschreibung:</b>	Überprüft die Versionen von Treiber und DLL.
<b>Übergabe-parameter:</b>	<b>DriverVersion</b> Zeiger auf den String, in den die Treiberversion eingetragen wird. <b>DIIVersion</b> Zeiger auf den String, in den die DLL-Version eingetragen wird. Die Strings sollten eine Mindestgröße von 80 Zeichen haben.
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## Zähler einzeln auslesen

<b>Funktionsname:</b>	<b>HS_ZP3B_ReadCounter</b>		
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3B_ReadCounter	(long CardNo, long CounterNo, long *CountVal, long *CountStat)
	<b>Delphi:</b>	HS_ZP3B_ReadCounter	(CardNo: :longint; CounterNo :longint; var CountVal :longint; var CountStat :longint)
			:longint
<b>Beschreibung:</b>	Liest den Zählerwert und den Status einer Achse (eines Zählers) der Zählerkarte aus.		
<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartennummer (1..15)	
	<b>CounterNo</b>	Zählernummer (1..3)	
	<b>CountVal</b>	Zeiger auf Variable, die den Zählerwert aufnimmt.	
	<b>CountStat</b>	Zeiger auf Variable, die den Zählerstatus aufnimmt.	
		01h:	Zustand Latcheingang
		02h:	Fehler
		04h:	Referenzpunkt überfahren
		08h:	Änderung Zustand Latcheingang
<b>Rückgabewert:</b>	<b>0</b>	OK	
	<b>-1</b>	Fehler	

## Alle Zähler auslesen

**Funktionsname:** HS\_ZP3B\_ReadAllCounter

<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3B_ReadAllCounter (long CardNo, long *Count1Val, long *Count1Stat, long *Count2Val, long *Count2Stat, long *Count3Val, long *Count3Stat);
----------------	--------------------	---

[illegible]

:longint

**Beschreibung:** Liest den Zählerwert und den Status aller Achsen der Zählerkarte aus.

<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartennummer (1..15)
---------------------------------	---------------	-------------------------------

**Count1Val** Zeiger auf Variable, die den Zählerwert von Zähler 1 aufnimmt.

**Count1Stat** Zeiger auf Variable, die den Zählerstatus von Zähler 1 aufnimmt.

01h:	Zustand Latcheingang
02h:	Fehler
04h:	Referenzpunkt überfahren
08h:	Änderung Zustand Latcheingang

**Count2Val** Zeiger auf Variable, die den Zählerwert von Zähler 2 aufnimmt.

**Count2Stat** Zeiger auf Variable, die den Zählerstatus von Zähler 2 aufnimmt.

01h:	Zustand Latcheingang
02h:	Fehler
04h:	Referenzpunkt überfahren
08h:	Änderung Zustand Latcheingang

## Alle Zähler auslesen (Fortsetzung)

<b>Übergabe- parameter:</b>	<b>Count3Val</b>	Zeiger auf Variable, die den Zählerwert von Zähler 3 aufnimmt.
	<b>Count3Stat</b>	Zeiger auf Variable, die den Zählerstatus von Zähler 3 aufnimmt.
		01h: Zustand Latcheingang
		02h: Fehler
		04h: Referenzpunkt überfahren
		08h: Änderung Zustand Latcheingang
<b>Rückgabewert:</b>	<b>0</b>	OK
	<b>-1</b>	Fehler

## Referenzmodus setzen

<b>Funktionsname:</b>	<b>HS_ZP3B_SetRefMode</b>	
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3B_SetRefMode (long CardNo, long CounterNo, long RefMode);
	<b>Delphi:</b>	HS_ZP3B_SetRefMode (CardNo :longint; CounterNo :longint; RefMode :longint)  :longint
<b>Beschreibung:</b>	Setzt den Referenzmodus eines Zählers.	
<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartenummer (1..15)
	<b>CounterNo</b>	Zählernummer (1..3)
	<b>RefMode</b>	Referenzmodus
		00h: Ohne Referenz (Default) 01h: Einfach-Referenz 02h: Mehrfach-Referenz
<b>Rückgabewert:</b>	<b>0</b>	OK
	<b>-1</b>	Fehler

## Alle Zähler latchen

<b>Funktionsname:</b>	<b>HS_ZP3B_StopAllCounter</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_StopAllCounter (long CardNo); <b>Delphi:</b> HS_ZP3B_StopAllCounter (CardNo :longint) :longint
<b>Beschreibung:</b>	Latcht alle Zählerwerte aller Zählerkarten.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## I<sup>2</sup>C-Clockleitung setzen

<b>Funktionsname:</b>	<b>HS_ZP3B_I2C_SetClock</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_I2C_SetClock (long CardNo, long CounterNo, long Val); <b>Delphi:</b> HS_ZP3B_I2C_SetClock (CardNo :longint; CounterNo :longint; Val :longint) :longint
<b>Beschreibung:</b>	Setzt die I <sup>2</sup> C-Clockleitung einer Achse.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15) <b>CounterNo</b> Zählernummer (1..3) <b>Val</b> Pegel der I <sup>2</sup> C-Clockleitung (0 oder 1)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## I<sup>2</sup>C-Datenleitung setzen

<b>Funktionsname:</b>	<b>HS_ZP3B_I2C_SetData</b>		
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3B_I2C_SetData	(long CardNo, long CounterNo, long Val);
	<b>Delphi:</b>	HS_ZP3B_I2C_SetData	(CardNo :longint; CounterNo :longint; Val :longint)
			:longint
<b>Beschreibung:</b>	Setzt die I <sup>2</sup> C-Datenleitung einer Achse.		
<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartennummer (1..15)	
	<b>CounterNo</b>	Zählernummer (1..3)	
	<b>Val</b>	Pegel der I <sup>2</sup> C-Datenleitung (0 oder 1)	
<b>Rückgabewert:</b>	<b>0</b>	OK	
	<b>-1</b>	Fehler	

## I<sup>2</sup>C-Datenleitung lesen

<b>Funktionsname:</b>	<b>HS_ZP3B_I2C_GetData</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_I2C_GetData (long CardNo, long CounterNo, long *Val);  <b>Delphi:</b> HS_ZP3B_I2C_GetData (CardNo :longint; CounterNo :longint; var Val :longint)  :longint
<b>Beschreibung:</b>	Liest die I <sup>2</sup> C-Datenleitung einer Achse.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartennummer (1..15)  <b>CounterNo</b> Zählnummer (1..3)  <b>Val</b> Zeiger auf die Variable, die den Pegel der I <sup>2</sup> C-Datenleitung aufnimmt.
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler

## Latch-Signal auslösen

<b>Funktionsname:</b>	<b>HS_ZP3B_Stop</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_Stop (long CardNo);  <b>Delphi:</b> HS_ZP3B_Stop (CardNo :longint)  :longint
<b>Beschreibung:</b>	Löst ein Latch-Signal aus (Software-Latch).
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartennummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler

## Zähler-Latch-Matrix konfigurieren

**Funktionsname:** HS\_ZP3B\_SetStopCnf

**Syntax:** Visual C++: long HS\_ZP3B\_SetStopCnf (long CardNo,  
long StopOut,  
long StopIn);

**Delphi:** HS\_ZP3B\_SetStopCnf (CardNo :longint;  
StopOut :longint;  
StopIn :longint)  
:longint

**Beschreibung:** Konfiguriert die Zähler-Latch-Matrix.

**Übergabe-  
parameter:** **CardNo** Logische Kartennummer (1..15)

**StopOut** Latch-Ausgang

01h: Latch-Ausgang Achse 1  
02h: Latch-Ausgang Achse 2  
04h: Latch-Ausgang Achse 3  
08h: Latch-Ausgang für Kaskadierung

**StopIn** Latch-Eingang

01h: Latch-Eingang Achse 1  
02h: Latch-Eingang Achse 2  
04h: Latch-Eingang Achse 3  
08h: Latch-Eingang von Kaskadierung  
10h: Latch-Eingang über Softwaresteuerung

**Rückgabewert:** 0 OK

-1 Fehler

**Funktions-  
erläuterungen**

Eingänge	Achse 1	●			
	Achse 2		▲		
	Achse 3				
	Kaskadierung	●	▲		
	Software				
		Achse 1	Achse 2	Achse 3	Kaskadierung
		Ausgänge			

**Beispiel ●:**

StopOut 01h  
StopIn 09h (01h+08h)

**Beispiel ▲:**

StopOut 02h  
StopIn 0Ah (02h+08h)



### 3.2 Erweiterte Bibliotheksfunktionen (HS\_ZP3X.DLL)

Die erweiterten Bibliotheksfunktionen der DLL "HS\_ZP3X.DLL" enthalten die folgenden Aufrufe:

DLL Funktionsaufruf	Bedeutung
HS_ZP3X_Initialize	Lädt den Treiber und prüft, ob die Zählerkarte im PC vorhanden ist. Initialisiert die Zählerkarte.
HS_ZP3X_GetVersion	Überprüft die Versionen von Treiber und erweiterter DLL.
HS_ZP3X_SetMode	Setzt den Referenz- und den Latchmodus einer Achse (eines Zählers).
HS_ZP3X_SetCounterMul	Setzt den Multiplikator für eine Achse.
HS_ZP3X_GetCounter	Liest den Zählerwert und den Status einer Achse der Zählerkarte aus.
HS_ZP3X_GetAllCounter	Liest den Zählerwert und den Status aller Achsen der Zählerkarte aus.
HS_ZP3X_ResetCounterStop	Setzt den Steuereingang einer Achse der Zählerkarte zurück.
HS_ZP3X_ResetRef	Setzt die Referenz der Zählerkarte zurück.
HS_ZP3X_GetCounterCurrent	Liest den Zählerwert und den Status einer Achse ununterbrochen aus.
HS_ZP3X_GetAllCounterCurrent	Liest den Zählerwert und den Status aller Achsen einer Zählerkarte ununterbrochen aus.
HS_ZP3X_SetDistCode	Konfiguriert die Parameter für abstandscodierte Maßstäbe.
HS_ZP3X_StopAllCounter	Latcht alle Zählerwerte aller Zählerkarten.
HS_ZP3X_I2C_SetClock	Setzt die I <sup>2</sup> C-Clockleitung einer Achse.
HS_ZP3X_I2C_SetData	Setzt die I <sup>2</sup> C-Datenleitung einer Achse.
HS_ZP3X_I2C_GetData	Liest die I <sup>2</sup> C-Datenleitung einer Achse.
HS_ZP3X_Stop	Löst ein Latch-Signal aus.
HS_ZP3X_SetStopCnf	Konfiguriert die Zähler-Latch-Matrix.

## Karte initialisieren

<b>Funktionsname:</b>	<b>HS_ZP3X_Initialize</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_Initialize (long CardNo) <b>Delphi:</b> HS_ZP3X_Initialize (CardNo:longint) :longint
<b>Beschreibung:</b>	Lädt den Treiber und prüft, ob die Zählerkarte im PC vorhanden ist. Initialisiert die Karte und setzt die Referenz- und Latch-Parameter auf die <b>Defaulteinstellung</b> .
<b>Übergabe-parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## Treiber-Version abfragen

<b>Funktionsname:</b>	<b>HS_ZP3X_GetVersion</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_GetVersion (char* DriverVersion, char* DIIVersion) <b>Delphi:</b> HS_ZP3X_GetVersion (DriverVersion :pchar; DIIVersion :pchar) :longint
<b>Beschreibung:</b>	Überprüft die Versionen von erweitertem Treiber und DLL.
<b>Übergabe-parameter:</b>	<b>DriverVersion</b> Zeiger auf den String in den die Treiberversion eingetragen wird. <b>DIIVersion</b> Zeiger auf den String in den die DLL-Version eingetragen wird. Die Strings sollten eine Mindestgröße von 80 Zeichen haben.
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## Referenz- und Latchmodus setzen

**Funktionsname:** HS\_ZP3X\_SetMode

**Syntax:**           **Visual C++:** long HS\_ZP3X\_SetMode (long CardNo,  
long CounterNo,  
long RefMode,  
long StopMode);

**Delphi:**           HS\_ZP3X\_SetMode (CardNo :longint;  
CounterNo :longint;  
RefMode :longint;  
StopMode :longint)

:longint

**Beschreibung:** Setzt den Referenz- und den Latchmodus einer Achse.  
Bitte beachten Sie auch die untenstehende Funktionserläuterung.

**Übergabe-  
parameter:**

**CardNo**      Logische Kartennummer (1..15)

**CounterNo**   Zählernummer (1..3)

**RefMode**      Referenzmodus

00h:      Ohne Referenz (Default)  
01h:      Einfach-Referenz  
02h:      Mehrfach-Referenz  
03h:      Abstandscodierte Referenz

**StopMode**    Latchmodus

00h:      Ohne Latch (Default)  
01h:      Manueller Latch  
02h:      Automatischer Latch  
00h:      Latch bei steigender Flanke (Default)  
10h:      Latch bei fallender Flanke  
20h:      Latch bei steigender und fallender Flanke

**Rückgabewert:**   **0**    OK

**-1**   Fehler

## Referenz- und Latchmodus setzen (Fortsetzung)

### Funktions- erläuterungen

Für die praktische Anwendung ist es notwendig, die Funktionalität der Referenzauswertung und der Zähler-Latch-Funktion zu erläutern.

#### Referenzauswertung

- Bei **zyklischer (mehrfacher) Referenzauswertung** wird der Zählerwert bei jedem Überfahren des Referenzpunktes wieder auf 0 gesetzt. Dies wird häufig bei Drehgebern verwendet, wodurch automatisch die Winkel auf 0-360° genormt werden.
- Die **einmalige Referenzauswertung** setzt den Zähler lediglich beim erstmaligen Überfahren des Referenzpunktes auf 0. Weitere Referenzsignale werden dann von der Karte ignoriert. Solange in diesem Modus kein Referenzsignal gesetzt ist, stellt der Zähler keinen definierten Wert zur Verfügung.
- Bei **abstandscodierter Referenzauswertung** wird die Referenz nach kurzem Fahrweg erkannt und ausgewertet. Weitere Referenzsignale werden dann von der Karte ignoriert. Solange in diesem Modus kein Referenzsignal gesetzt ist, stellt der Zähler keinen definierten Wert zur Verfügung.

#### Zähler-Latch-Funktion

- **Automatische Zähler-Latch-Freigabe** bedeutet, dass die Zählerwertausgabe solange blockiert wird, wie das Latchsignal anliegt. Sobald dieses weggeschaltet wird, ist die Ausgabe wieder automatisch freigegeben. Der Zustand des Latch-Registers wird beim Auslesen der Zählerwerte übermittelt und kann so für die weitere Auswertung verwendet werden. Eine typische Anwendung hierfür ist die Kantenantastung mittels eines Kantensensors.
- Bei der **manuellen Zähler-Latch-Freigabe** bleibt die Zählerwertausgabe auch dann noch blockiert, wenn das Latchsignal nicht mehr anliegt. Diese muss dann explizit mit dem "HS\_ZP3X\_ResetCounterStop-Befehl" für jede Achse getrennt freigegeben werden.

## Multiplikator für Zählerwert setzen

**Funktionsname:** HS\_ZP3X\_SetCounterMul

**Syntax:**           **Visual C++:** long HS\_ZP3X\_SetCounterMul (long CardNo,  
long CounterNo,  
long Multiplier)

**Delphi:**           HS\_ZP3X\_SetCounterMul (CardNo :longint;  
CounterNo :longint;  
Multiplier :longint)  
  
:longint

**Beschreibung:**   Setzt den Multiplikator für einen Zähler. Mit Hilfe des Multiplikators kann auch einfach die Zählrichtung gedreht werden. Der Multiplikator ist dann auf -1 zu setzen.

**Übergabe-  
parameter:**       **CardNo**     Logische Kartenummer (1..15)

**CounterNo**   Zählernummer (1..3)

**Multiplier**   Multiplikator

**Rückgabewert:**   **0**    OK  
  
                  **-1**   Fehler

## Zähler einzeln auslesen

**Funktionsname:** HS\_ZP3X\_GetCounter

**Syntax:**           **Visual C++:** long HS\_ZP3X\_GetCounter (long CardNo,  
long CounterNo,  
long \*Count,  
long \*Status)

**Delphi:**           HS\_ZP3X\_GetCounter (CardNo :longint;  
CounterNo :longint;  
var Count :longint;  
var Status :longint)  
  
:longint;

**Beschreibung:** Liest den Zählerwert und den Status eines Zählers aus. Der Zählerwert wird mit dem Referenzwert und dem Multiplikator verrechnet.

Hinweis: Wenn der Latcheingang aktiv ist (siehe auch Funktion "Referenz- und Latchmodus setzen"), wird der Zählerwert eingefroren.

**Übergabeparameter:**

**CardNo**       Logische Kartennummer (1..15)

**CounterNo**   Zählernummer (1..3)

**Count**       Zeiger auf Variable, die den Zählerwert aufnimmt.

**Status**       Zeiger auf Variable, die den Zählerstatus aufnimmt.

01h:   Fehler  
02h:   Referenzpunkt bereits überfahren  
04h:   Zähler gelatcht  
08h:   Zustand Latcheingang  
10h:   Änderung Zustand Latcheingang

**Rückgabewert:**   **0**   OK

**-1**   Fehler

## Alle Zähler auslesen

**Funktionsname:** HS\_ZP3X\_GetAllCounter

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_GetAllCounter (long CardNo, long *Count1, long *Status1, long *Count2, long *Status2, long *Count3, long *Status3)
----------------	--

```
Delphi:      HS_ZP3X_GetAllCounter (CardNo    :longint;  
                                     var Count1 :longint;  
                                     var Status1 :longint;  
                                     var Count2 :longint;  
                                     var Status2 :longint;  
                                     var Count3 :longint;  
                                     var Status3 :longint)
```

```
:longint;
```

<b>Beschreibung:</b>	Liest den Zählerwert und den Status aller Zähler einer Zählerkarte aus. Der Zählerwert wird mit dem Referenzwert und dem Multiplikator verrechnet.
----------------------	--

Hinweis: Wenn der Latcheingang aktiv ist (siehe auch Funktion "Referenz- und Latchmodus setzen"), wird der Zählerwert eingefroren.

<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartennummer (1..15)
---------------------------------	---------------	-------------------------------

<b>Count1</b>	Zeiger auf Variable, die den Zählerwert von Zähler 1 aufnimmt.
---------------	--

**Status1** Zeiger auf Variable, die den Zählerstatus von Zähler 1 aufnimmt.

01h:	Fehler
02h:	Referenzpunkt bereits überfahren
04h:	Zähler gelatcht
08h:	Zustand Latcheingang
10h:	Änderung Zustand Latcheingang

**Alle Zähler auslesen (Fortsetzung)**

<b>Übergabe- parameter:</b>	<b>Count2</b>	Zeiger auf Variable, die den Zählerwert von Zähler 2 aufnimmt.
	<b>Status2</b>	Zeiger auf Variable, die den Zählerstatus von Zähler 2 aufnimmt.
		01h: Fehler
		02h: Referenzpunkt bereits überfahren
		04h: Zähler gelatcht
		08h: Zustand Latcheingang
		10h: Änderung Zustand Latcheingang
	<b>Count3</b>	Zeiger auf Variable, die den Zählerwert von Zähler 3 aufnimmt.
	<b>Status3</b>	Zeiger auf Variable, die den Zählerstatus von Zähler 3 aufnimmt.
		01h: Fehler
		02h: Referenzpunkt bereits überfahren
		04h: Zähler gelatcht
		08h: Zustand Latcheingang
		10h: Änderung Zustand Latcheingang
<b>Rückgabewert:</b>	<b>0</b>	OK
	<b>-1</b>	Fehler



## Latcheingang zurücksetzen

<b>Funktionsname:</b>	<b>HS_ZP3X_ResetCounterStop</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_ResetCounterStop (long CardNo, long CounterNo)  <b>Delphi:</b> HS_ZP3X_ResetCounterStop (CardNo :longint; CounterNo :longint)  :longint;
<b>Beschreibung:</b>	Setzt den Steuereingang eines Zählers der Zählerkarte zurück. Der Zähler zählt danach sofort wieder weiter.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)  <b>CounterNo</b> Zählernummer (1..3)
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler
<b>Hinweis</b>	Die Freigabe muss getrennt für jede Achse erfolgen.

## Referenz zurücksetzen

<b>Funktionsname:</b>	<b>HS_ZP3X_ResetRef</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_ResetRef (long CardNo, long CounterNo);  <b>Delphi:</b> HS_ZP3X_ResetRef (CardNo :longint; CounterNo :longint)  :longint;
<b>Beschreibung:</b>	Setzt die Referenz der Zählerkarte zurück. Der Referenzpunkt muss erneut überfahren werden.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)  <b>CounterNo</b> Zählernummer (1..3)
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler

## Zähler ununterbrochen einzeln auslesen

**Funktionsname:** HS\_ZP3X\_GetCounterCurrent

**Syntax:**           **Visual C++:** long HS\_ZP3X\_GetCounterCurrent (long CardNo,  
long CounterNo,  
long \*Count,  
long \*Status);

**Delphi:**           HS\_ZP3X\_GetCounterCurrent (CardNo :longint;  
CounterNo :longint;  
var Count :longint;  
var Status :longint)

:longint;

**Beschreibung:** Liest den Zählerwert und den Status eines Zählers ununterbrochen aus. Der Zählerwert wird mit dem Referenzwert und dem Multiplikator verrechnet.

**Übergabe-  
parameter:**       **CardNo**     Logische Kartennummer (1..15)

**CounterNo**   Zählernummer (1..3)

**Count**        Zeiger auf Variable, die den Zählerwert aufnimmt.

**Status**        Zeiger auf Variable, die den Zählerstatus aufnimmt.

01h:     Fehler  
02h:     Referenzpunkt bereits überfahren  
04h:     Zähler gelatcht  
08h:     Zustand Latcheingang  
10h:     Änderung Zustand Latcheingang

**Rückgabewert:**   **0**    OK  
                      **-1**   Fehler

## Alle Zähler ununterbrochen auslesen

**Funktionsname:** HS\_ZP3X\_GetAllCounterCurrent

**Syntax:**           **Visual C++:** long HS\_ZP3X\_GetAllCounterCurrent (long CardNo,  
long \*Count1,  
long \*Status1,  
long \*Count2,  
long \*Status2,  
long \*Count3,  
long \*Status3);

**Delphi:**           HS\_ZP3X\_GetAllCounterCurrent (CardNo :longint;  
var Count1 :longint;  
var Status1 :longint;  
var Count2 :longint;  
var Status2 :longint;  
var Count3 :longint;  
var Status3 :longint)

:longint;

**Beschreibung:** Liest den Zählerwert und den Status aller Achsen einer Zählerkarte ununterbrochen aus. Der Zählerwert wird mit dem Referenzwert und dem Multiplikator verrechnet.

**Übergabe-  
parameter:**

**CardNo**      Logische Kartenummer (1..15)

**Count1**      Zeiger auf Variable, die den Zählerwert von Zähler 1 aufnimmt.

**Status1**      Zeiger auf Variable, die den Zählerstatus von Zähler 1 aufnimmt.

01h:      Fehler  
02h:      Referenzpunkt bereits überfahren  
04h:      Zähler gelatcht  
08h:      Zustand Latcheingang  
10h:      Änderung Zustand Latcheingang

**Count2**      Zeiger auf Variable, die den Zählerwert von Zähler 2 aufnimmt.

**Status2**      Zeiger auf Variable, die den Zählerstatus von Zähler 2 aufnimmt.

01h:      Fehler  
02h:      Referenzpunkt bereits überfahren  
04h:      Zähler gelatcht  
08h:      Zustand Latcheingang  
10h:      Änderung Zustand Latcheingang

**Alle Zähler ununterbrochen auslesen (Fortsetzung)**

<b>Übergabe- parameter:</b>	<b>Count3</b>	Zeiger auf Variable, die den Zählerwert von Zähler 3 aufnimmt.
	<b>Status3</b>	Zeiger auf Variable, die den Zählerstatus von Zähler 3 aufnimmt.  01h: Fehler 02h: Referenzpunkt bereits überfahren 04h: Zähler gelatcht 08h: Zustand Latcheingang 10h: Änderung Zustand Latcheingang
<b>Rückgabewert:</b>	<b>0</b>	OK
	<b>-1</b>	Fehler

## Abstandscodierung konfigurieren

**Funktionsname:** HS\_ZP3X\_SetDistCode

**Syntax:** **Visual C++:** long HS\_ZP3X\_SetDistCode (long CardNo,  
long CounterNo,  
long Distance,  
long Pitch,  
long Offset,  
long Direction);

**Delphi:** HS\_ZP3X\_SetDistCode (CardNo :longint;  
CounterNo :longint;  
Distance :longint;  
Pitch :longint;  
Offset :longint;  
Direction :longint)  
:longint

**Beschreibung:** Konfiguriert die Parameter für abstandscodierte Maßstäbe.

**Übergabe-  
parameter:** **CardNo** Logische Kartenummer (1..15)

**CounterNo** Zählernummer (1..3)

**Distance** Grundabstand der Referenzmarken in mm

**Pitch** Signalperiode in  $\mu\text{m}$

**Offset** Maßstabsanfang in mm

**Direction** Grundzählrichtung des Maßstabs

00h: positiv (Default), z. B. Acu-Rite  
01h: negativ, z. B. Heidenhain

**Rückgabewert:** **0** OK

**-1** Fehler

**Hinweis:** Folgende Reihenfolge der Funktionsaufrufe muss eingehalten werden:

HS\_ZP3X\_Initialize(...)  
HS\_ZP3X\_SetDistCode(...)  
HS\_ZP3X\_SetMode(...)

Soll mit Default-Werten (Distance=20mm; Pitch=20 $\mu\text{m}$ ; Offset=0mm) gearbeitet werden, kann der Aufruf "HS\_ZP3X\_SetDistCode(...)" entfallen.

## Alle Zähler latchen

<b>Funktionsname:</b>	<b>HS_ZP3X_StopAllCounter</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_StopAllCounter (long CardNo); <b>Delphi:</b> HS_ZP3X_StopAllCounter (CardNo :longint) :longint
<b>Beschreibung:</b>	Latcht alle Zählerwert aller Zählerkarten.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## I<sup>2</sup>C-Clockleitung setzen

<b>Funktionsname:</b>	<b>HS_ZP3X_I2C_SetClock</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_I2C_SetClock (long CardNo, long CounterNo, long Val); <b>Delphi:</b> HS_ZP3X_I2C_SetClock (CardNo :longint; CounterNo :longint; Val :longint) :longint
<b>Beschreibung:</b>	Setzt die I <sup>2</sup> C-Clockleitung einer Achse.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartenummer (1..15) <b>CounterNo</b> Zählernummer (1..3) <b>Val</b> Pegel der I <sup>2</sup> C-Clockleitung (0 oder 1)
<b>Rückgabewert:</b>	<b>0</b> OK <b>-1</b> Fehler

## I<sup>2</sup>C-Datenleitung setzen

<b>Funktionsname:</b>	<b>HS_ZP3X_I2C_SetData</b>		
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3X_I2C_SetData	(long CardNo, long CounterNo, long Val);
	<b>Delphi:</b>	HS_ZP3X_I2C_SetData	(CardNo :longint; CounterNo :longint; Val :longint)
			:longint
<b>Beschreibung:</b>	Setzt die I <sup>2</sup> C-Datenleitung einer Achse.		
<b>Übergabe- parameter:</b>	<b>CardNo</b>	Logische Kartennummer (1..15)	
	<b>CounterNo</b>	Zählernummer (1..3)	
	<b>Val</b>	Pegel der I <sup>2</sup> C-Datenleitung (0 oder 1)	
<b>Rückgabewert:</b>	<b>0</b>	OK	
	<b>-1</b>	Fehler	

## I<sup>2</sup>C-Datenleitung lesen

<b>Funktionsname:</b>	<b>HS_ZP3X_I2C_GetData</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_I2C_GetData (long CardNo, long CounterNo, long *Val);  <b>Delphi:</b> HS_ZP3X_I2C_GetData (CardNo :longint; CounterNo :longint; var Val :longint)  :longint
<b>Beschreibung:</b>	Liest die I <sup>2</sup> C-Datenleitung einer Achse.
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartennummer (1..15)  <b>CounterNo</b> Zählnummer (1..3)  <b>Val</b> Zeiger auf die Variable, die den Pegel der I <sup>2</sup> C-Datenleitung aufnimmt.
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler

## Latch-Signal auslösen

<b>Funktionsname:</b>	<b>HS_ZP3X_Stop</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_Stop (long CardNo);  <b>Delphi:</b> HS_ZP3X_Stop (CardNo :longint)  :longint
<b>Beschreibung:</b>	Löst ein Latch-Signal aus (Software-Latch).
<b>Übergabe- parameter:</b>	<b>CardNo</b> Logische Kartennummer (1..15)
<b>Rückgabewert:</b>	<b>0</b> OK  <b>-1</b> Fehler



## Zähler-Latch-Matrix konfigurieren

**Funktionsname:** HS\_ZP3X\_SetStopCnf

**Syntax:** Visual C++: long HS\_ZP3X\_SetStopCnf (long CardNo,  
long StopOut,  
long StopIn);

**Delphi:** HS\_ZP3X\_SetStopCnf (CardNo :longint;  
StopOut :longint;  
StopIn :longint)  
:longint

**Beschreibung:** Konfiguriert die Zähler-Latch-Matrix.

**Übergabe-  
parameter:** **CardNo** Logische Kartennummer (1..15)

**StopOut** Latch-Ausgang

01h: Latch-Ausgang Achse 1  
02h: Latch-Ausgang Achse 2  
04h: Latch-Ausgang Achse 3  
08h: Latch-Ausgang für Kaskadierung

**StopIn** Latch-Eingang

01h: Latch-Eingang Achse 1  
02h: Latch-Eingang Achse 2  
04h: Latch-Eingang Achse 3  
08h: Latch-Eingang von Kaskadierung  
10h: Latch-Eingang über Softwaresteuerung

**Rückgabewert:** 0 OK

-1 Fehler

**Funktions-  
erläuterungen**

Eingänge	Achse 1	●			
	Achse 2		▲		
	Achse 3				
	Kaskadierung	●	▲		
	Software				
		Achse 1	Achse 2	Achse 3	Kaskadierung
		Ausgänge			

**Beispiel ●:**

StopOut 01h  
StopIn 09h (01h+08h)

**Beispiel ▲:**

StopOut 02h  
StopIn 0Ah (02h+08h)

### **3.3 DLLs einbinden**

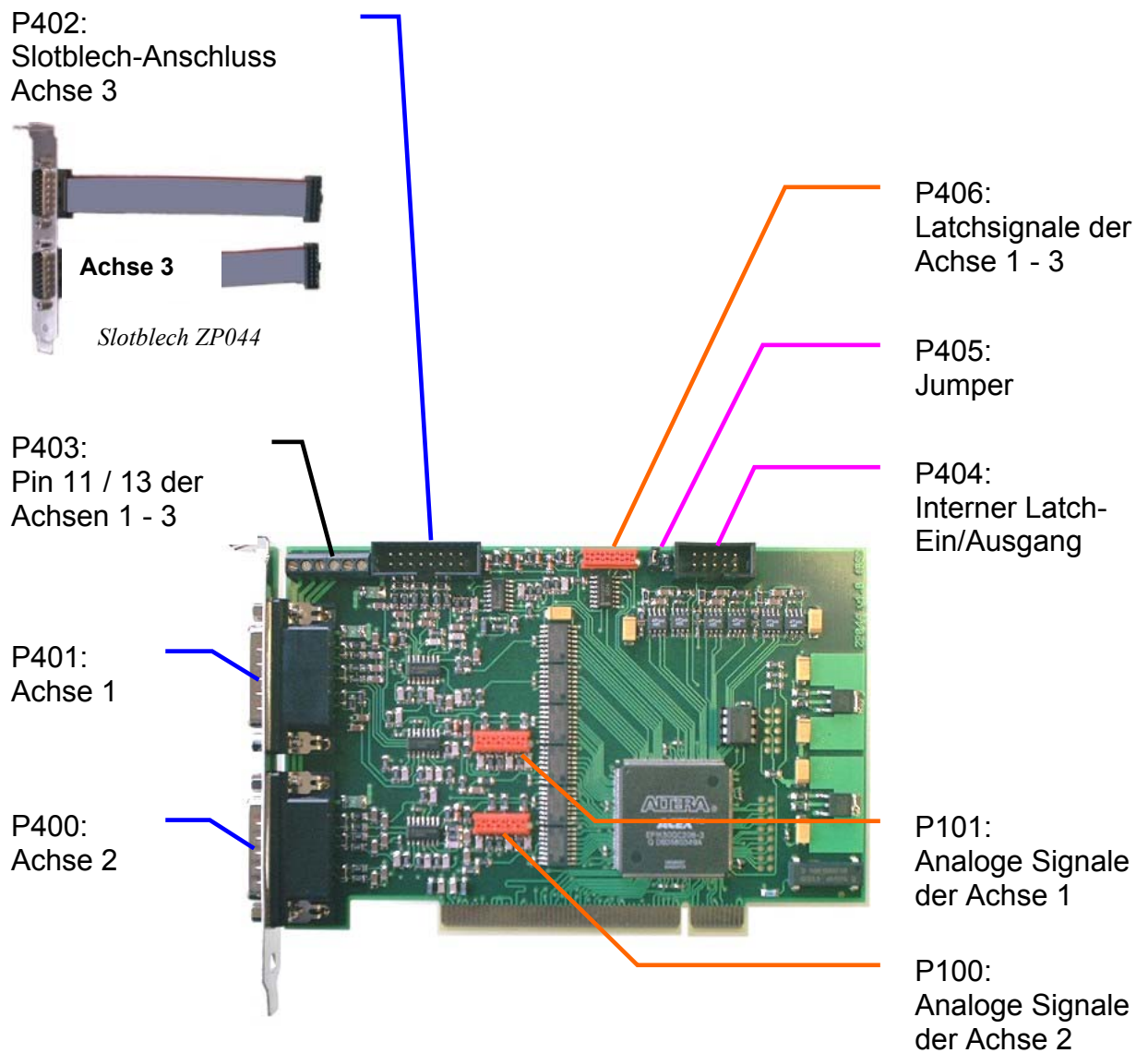
Die DLLs werden über ein Interface-Modul der jeweiligen Programmiersprache eingebunden, welches die notwendigen Deklarationen enthält, um auf die Bibliotheksfunktionen zuzugreifen.

Dieses Interface-Modul ist für Visual C++ und für Borland Delphi auf dem im Lieferumfang enthaltenen Datenträger vorhanden.

Für andere Programmiersprachen, wie z.B. Visual Basic, müssen Sie sich dieses Modul selbst erstellen.

## 4 Technische Daten

### 4.1 Steckeranordnung

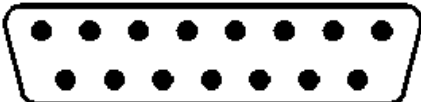


#### Steckeranordnung der ZP044

Die technische Spezifikationen der Ein- und Ausgänge wird in den nachfolgenden Abschnitten beschrieben.

## 4.2 Steckerbelegung

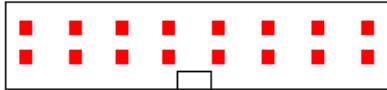
### Signaleingänge Achse 1 – 3 (P400 / P401 / P402 Slotblech ZP044)

Pin	Signal	Hinweis
1	+ 5 V	 <p>15-polige Sub-D-Stiftleisten</p>
2	GND	
3	+ $\varphi_0$	
4	- $\varphi_0$	
5	+ Daten I <sup>2</sup> C	
6	+ $\varphi_{90}$	
7	- $\varphi_{90}$	
8	- Daten I <sup>2</sup> C	
9	Latch	MicroMaTch P406
10	+ REF	
11	max. 24V/1A	Federklemme P403
12	- REF	
13	max. 24V/1A	Federklemme P403
14	+ Takt I <sup>2</sup> C	
15	- Takt I <sup>2</sup> C	


### Signalkennndaten

Signal:	0,6 - 1,2 V <sub>ss</sub> , typ. 1 V <sub>ss</sub> (sinusförmig)
Signalteilung:	256-fache Interpolation
Referenzsignale:	typ. 0,5 V <sub>ss</sub> Nutzanteil
Zählerbreite:	28 Bit
Phasenwinkel $\varphi_0$ / $\varphi_{90}$ :	90° ± 10°
Eingangsfrequenz:	0 - 78 kHz

**Slotblech-Anschluss Achse 3(P402)**

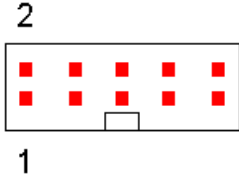
Pin	Signal	Hinweis		
1	+ 5 V	<div><div>2</div><div></div><div>1</div><div>16-polige Wannenstiftleiste</div></div>		
2	Latch			MicroMatch P406
3	GND			
4	+ REF			
5	+ $\varphi_0$			
6	max. 24V/1A			Federklemme P403
7	- $\varphi_0$			
8	- REF			
9	+ Daten I <sup>2</sup> C			
10	max. 24V/1A			Federklemme P403
11	+ $\varphi_{90}$			
12	+ Takt I <sup>2</sup> C			
13	- $\varphi_{90}$			
14	- Takt I <sup>2</sup> C			
15	- Daten I <sup>2</sup> C			
16	n. c.	frei		

**Signaleinspeisung / Signalabgriff (P403)**

Pin	Signal	Hinweis	
1	PIN 13	 <p>6-polige Federklemme</p>	
2	PIN 11		
3	PIN 13		
4	PIN 11		
5	PIN 13		
6	PIN 11		

### Interner Latch-Ein/Ausgang (P404)

Über diesen Anschluss steht ein interner Latch-Ein/Ausgang zur Verfügung, der es ermöglicht, einzelne oder alle Achsen mehrerer Zählerkarten gemeinsam zu verriegeln (parametrierbar über Software-Konfiguration).

Pin	Signal	Hinweis	
1	Latch int	 10-polige Wannenstiftleiste	
2	GND		
3	Latch int		
4	GND		
5	Latch int		
6	GND		
7	Latch int		
8	GND		
9	Latch int		
10	GND		

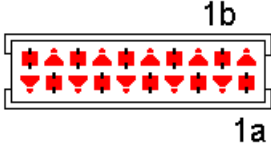
Wegen besserer Schirmung und erhöhter Kontaktsicherheit sind die Signale parallel geführt.

### Jumper (P405)

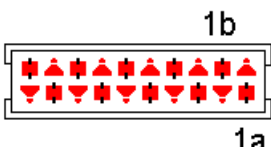


Werden mehrere Zählerkarten über den Internen Latch-Ein/Ausgang (P404) kaskadiert, sollte der Jumper P405 auf einer der Zählerkarten gesteckt sein!

### Latchsignale Achse 1 – 3 (P406)

Pin	Signal	Hinweis		
1a	Latch Achse 1	<div></div> <p>10-polige MicroMatch-Buchse</p>		
1b	+ 5 V			
2a	Latch Achse 2			
2b	GND			
3a	Latch Achse 3			
3b	n. c.			frei
4a	GND			
4b	Latch intern			
5a	GND			
5b	n. c.			frei

**Analoge Signale Achse 1 und Achse 2 (P100 / P101)**

Pin	Signal	Hinweis	
1a	- $\varphi_0$		
1b	+ $\varphi_0$		
2a	GND		
2b	n. c.		frei
3a	- $\varphi_{90}$		
3b	+ $\varphi_{90}$		
4a	n. c.		frei
4b	+ REF		
5a	- REF		
5b	n. c.		frei

**4.3 Physikalische und mechanische Kennwerte**

Abmessungen: ca. 160 x 100 mm (B x H)

Gewicht: ca. 110g (ohne Kabel)

Lagertemperatur: - 30° bis + 70° C

Betriebstemperatur: 0° bis + 45°C

Rel. Luftfeuchte: < 75 %

## 5 Garantiebedingungen

Der Hersteller garantiert die Funktion ihrer Hard - und Softwareprodukte für die Dauer von einem Jahr nach Lieferdatum. Während dieser Garantiefrist erklärt sich der Hersteller bereit, Produkte, die sich als fehlerhaft erwiesen haben, wahlweise im Herstellerwerk zu reparieren oder zu ersetzen.

Es wird weiterhin vorausgesetzt, dass die Karte nur von entsprechend geschultem und ausgebildeten Personal bedient wird.

### **Ausgenommen von Garantieleistungen sind:**

- Schäden durch unsachgemäße oder unangemessene Reparatur durch den Kunden.
- Schäden durch Software des Kunden.
- Schäden durch nicht korrekten Einsatz der Software.
- Schäden an Verbindungen zu Messsystemen des Kunden.
- Schäden durch nicht genehmigte Veränderungen.
- Schäden durch Nichteinhaltung der Lager - und Betriebsbedingungen.
- Geräte, bei denen die Seriennummer entfernt worden ist.
- Schäden, die durch Hochspannung oder elektrostatische Entladung verursacht worden sind.