

## User Manual PCI counter card ZP051

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)

Neither the whole nor any part of this documentation may be reproduced, passed on to third parties, stored in a database system or translated into another language without the written permission of Heilig & Schwab GmbH.

© Copyright 2001 - 2005 Heilig & Schwab GmbH. All rights reserved.

8th edition: Bad Sobernheim, May 4<sup>th</sup>, 2005

The information contained in this document is subject to change without prior notice. In doing so, Heilig & Schwab GmbH do not enter into any obligation.

Heilig & Schwab GmbH disclaim all warranties, any legal responsibility or any liability for consequential damages arising from or in connection with the content or use of this manual.

Furthermore, Heilig & Schwab GmbH hereby disclaim all warranties, any legal responsibility or any liability for consequential damages arising from the incorrect use of the hardware and/or software. The layout or design of the hardware can be changed without prior notice. In doing so, Heilig & Schwab GmbH do not enter into any obligation.

All other trademarks and product designations used in this manual are the property of the respective companies and manufacturers. Heilig & Schwab GmbH waive all property rights with regard to the named trademarks and product designations that do not belong to them.

# Contents

<b>1</b>	<b>Important Notes .....</b>	<b>5</b>
<b>1.1</b>	<b>COMPATIBILITY STATEMENT .....</b>	<b>5</b>
	Manufacturer and product name .....	5
	EMC Specifications .....	5
<b>1.2</b>	<b>PURPOSE .....</b>	<b>6</b>
<b>1.3</b>	<b>DELIVERY SCHEDULE AND SYSTEM REQUIREMENTS .....</b>	<b>7</b>
	Delivery schedule .....	7
	System requirements .....	7
<b>2</b>	<b>Installation.....</b>	<b>8</b>
<b>2.1</b>	<b>FITTING THE COUNTER CARD.....</b>	<b>8</b>
<b>2.2</b>	<b>INSTALLATION OF COUNTER CARD DRIVER AND USER SOFTWARE .....</b>	<b>9</b>
<b>3</b>	<b>Configuration .....</b>	<b>10</b>
<b>3.1</b>	<b>BASIC LIBRARY FUNCTIONS (HS_ZP3B.DLL) .....</b>	<b>10</b>
	Initialising the card .....	11
	Version query .....	12
	Reading out counters individually .....	13
	Reading out all counters .....	14
	Setting the reference mode .....	16
	Latching all counter values .....	17
	Reading out the counter referece register .....	18
	Writing the counter reference register .....	19
	Reading out the card type .....	20
	Reading out the result of probe identification .....	21
<b>3.2</b>	<b>EXTENDED LIBRARY FUNCTIONS (HS_ZP3X.DLL) .....</b>	<b>22</b>
	Initialising the card .....	23
	Version query .....	24
	Setting reference and latch modes .....	25
	Multiplier for counter values .....	27
	Reading out counters individually .....	28
	Reading out all counters .....	29
	Resetting the latch input .....	31
	Resetting the reference .....	32
	Reading out individual counters continuously .....	33
	Reading out all counters continuously .....	34
	Setting distance coded reference .....	36
	Latching all counter values .....	37
	Reading out the card type .....	38
	Reading out the result of probe identification .....	39
<b>3.3</b>	<b>LINKING THE DLLs .....</b>	<b>40</b>
<b>3.4</b>	<b>WINMY DEMO AND MEASURING PROGRAM .....</b>	<b>40</b>
	Software licence .....	40

<b>4</b>	<b>Technical data.....</b>	<b>41</b>
4.1	CONNECTOR LAYOUT .....	41
4.2	CONNECTOR ALLOCATION (AXIS 1 AND 2) .....	42
	Current inputs (standard version / ZP051-20) .....	42
	TTL inputs (optional / ZP051-20-TTL) .....	43
	Voltage inputs (optional / ZP051-20-V) .....	43
4.3	CONNECTOR ALLOCATION (AXIS 3 / LATCH INPUTS AXIS 1 – 3) .....	44
4.4	PHYSICAL AND MECHANICAL SPECIFICATIONS .....	44
<b>5</b>	<b>Guarantee terms .....</b>	<b>45</b>

# 1 Important Notes

## 1.1 Compatibility Statement

### Manufacturer and product name

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

Product: PCI counter card for 3 channels

Model: ZP051

### EMC Specifications

The counter card meets the specifications of standards EN 55022 und EN 61000-6-2.

These threshold values give sufficient protection against dangerous electromagnetic radiation for the environment. This applies as long as the product has been fitted and used in accordance with the instructions. It is also necessary for all cables to the counter card to be shielded and connected properly. The peripheral equipment must also be shielded and earthed.



Using this equipment with uncertified personal computers or incorrectly shielded cables or if the counter card has not been fitted correctly can lead to electromagnetic faults.

All changes or modifications that are not expressly approved by the manufacturer will lead to the operating licence being null and void.

## 1.2 Purpose

The ZP051 counter card is a PC plug-in card for the PCI slot. It is used to connect rotary encoders, linear encoders and length gauges directly to the computer.

The card has 3 separate counter channels (axis 1, axis 2 and axis 3). Two of these counter channels can be connected directly to the counter card by means of two 9 pin sub-D socket connectors. The third axis and three latch inputs can be connected by means of a 20 pin shrouded header.

The integral interpolation electronic system divides the measuring system signal periods. The counting signals thus produced will be totalled in three counting registers depending on the preceding mathematical sign and stored in the appropriate register by means of a counter access control system. It will then be read using the PC software.

The integral analysis of reference signals separately for each axis enables analyses relating to any point on the encoder to be reproduced.

To improve measuring and operating security the measuring system amplitude and input frequency input signals will be monitored and an error signal triggered if a correct signal analysis can no longer be guaranteed. This will be stored as an error bit and can be read or deleted with the software.

The card can quickly be integrated into your application using the driver software for the various Windows operating systems that is included in the package. These drivers will allow you to read out the counter values easily and access the card's various parameters and operating modes.

## 1.3 Delivery s schedule and system requirements

### Delivery schedule

The following is included in the counter card package:

Item	Amount	Name	Notes
1	1	ZP051-20 PCI counter card	
2	1	Data carrier (CD ROM) with <ul style="list-style-type: none"><li>• user manual</li><li>• Driver and DLLs for Windows operating systems</li><li>• WinMy demo and measuring program</li></ul>	One will be supplied with the basic orders but can be re-ordered at any time if necessary.

Please check the contents of the package directly after delivery.

Please contact us immediately if the contents as described above are not correct.

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

### System requirements

You will need the following system requirements on your computer and your software to run the ZP051 PC counter card:

- Pentium PC or a fully compatible system
- Windows 95/98 or later
- A free PCI slot
- VGA monitor

Please note that to use the “WinMy” software you will need at least 8 MB of memory and approximately 5 MB of hard disc space.

## 2 Installation


The ZP051 counter card may only be fitted and installed by appropriately trained personnel.

It is also assumed that before installing this card you will have read this user manual and observed the relevant safety and operating instructions.

### 2.1 Fitting the counter card

When handling the card please observe the normal safety precautions regarding electrostatic discharge. If you are in doubt, discharge any static electricity by holding on to an earth connection such as earthed equipment housings, radiators etc.

Proceed as follows to insert the counter card in your PC:

-  1. Switch the computer off and unplug it.
2. Open or remove the computer housing.
3. Select a free PCI slot and remove the bracket on the back of the computer.
4. Now insert the ZP051 counter card into the slot on the mother board. - Ensure that the card is straight and not in contact with any neighbouring card. The signal connector sockets must be freely accessible from outside.
5. Now fix the counter card by screwing it into the hole in the back of the computer.
6. If necessary, connect the third axis or the latch inputs using the shrouded header on the ZP051 circuit card.
7. Close the computer housing properly.
8. Connect the encoders (first and second axes) to the ZP051 counter card's signal connector sockets.

The counter card is now fitted and ready for operation. The computer can be switched on.

To use the counter card you now only need the driver software for your operating system supplied on the data carrier. Installing the software will be described in the chapter "Installation of counter card driver and user software".



## **2.2 Installation of counter card driver and user software**

The driver and DLLs for Windows operating systems are on the data carrier supplied.

Windows 95/98 or later operating systems recognise the counter card after it has been fitted and start to install the corresponding drivers using the device driver or hardware wizards. After inserting the data carrier and selecting the drivers they will be installed automatically.

With Windows NT the driver software will be installed by running the set up program on the data carrier.

## 3 Configuration

Two DLLs are included to access the counter card drivers. The DLL "HS\_ZP3B.DLL" with basic library functions and the DLL "HS\_ZP3X.DLL" with extended library functions.

### 3.1 Basic library functions (HS\_ZP3B.DLL)

The basic DLL "HS\_ZP3B.DLL" contains the following calls:

DLL function call	Meaning
HS_ZP3B_Init	Loads drivers and checks whether the card is in the PC and initialises it.
HS_ZP3B_GetVersion	Checks the versions of drivers and the DLL.
HS_ZP3B_ReadCounter	Reads out the counter value with status information of one counter of the counter card.
HS_ZP3B_ReadAllCounter	Reads out the counter value with status information of all counters of the counter card.
HS_ZP3B_SetRefMode	Sets the counter reference mode.
HS_ZP3B_StopAllCounter	Latches all counter values of all counter cards.
HS_ZP3B_ReadRefReg	Reads out the counter reference register of one counter.
HS_ZP3B_WriteRefReg	Writes into the counter reference register of one counter.
HS_ZP3B_ReadCardType	Reads out the type of the counter card.
HS_ZP3B_ProbeAvail	Reads out the result of probe identification.

Function name:	HS_ZP3B_Init
----------------	--------------

**Syntax:**                    **Visual C++:** long HS\_ZP3B\_Init (long CardNo)

```
Delphi:      HS_ZP3B_Init   (CardNo      :longint)
              :longint
```

<b>Description:</b>	Loads drivers and checks whether the card is in the PC and initialises it.
---------------------	--

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Version query

<b>Function name:</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>Description:</b>	Checks the versions of drivers and the DLL.
<b>Delivery parameter:</b>	<b>DriverVersion:</b> Pointer to the string where the driver version is entered.
	<b>DIIVersion:</b> Pointer to the string where the DLL version is entered. The strings should have at least 80 characters.
<b>Return value:</b>	<b>0</b> OK  <b>-1</b> Error

## Reading out counters individually

**Function name:** HS\_ZP3B\_ReadCounter

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_ReadCounter (long CardNo, long CounterNo, long *CountVal, long *CountStat)
----------------	---

```
Delphi:      HS_ZP3B_ReadCounter  (CardNo:      :longint;  
                                CounterNo    :longint;  
                                var CountVal  :longint;  
                                var CountStat :longint)
```

```
:longint
```

<b>Description:</b>	Reads out the counter value with status information of one counter of the counter card.
---------------------	---

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CounterNo** Counter number (1...3)

**CountVal**    Pointer to the counter value variable.

**CountStat** Pointer to the counter status variable.

01h:	Latch input status
02h:	Error
04h:	Reference already detected
08h:	Latch input status change

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Reading out all counters

**Function name:** HS\_ZP3B\_ReadAllCounter

**Syntax:**

**Visual C++:** long HS\_ZP3B\_ReadAllCounter (long CardNo,  
long \*Count1Val,  
long \*Count1Stat,  
long \*Count2Val,  
long \*Count2Stat,  
long \*Count3Val,  
long \*Count3Stat);

**Delphi:** HS\_ZP3B\_ReadAllCounter (CardNo :longint;  
var Count1Val :longint;  
var Count1Stat :longint;  
var Count2Val :longint;  
var Count2Stat :longint;  
var Count3Val :longint;  
var Count3Stat :longint)  
:longint

**Description:** Reads out the counter value with status information of all counters of the counter card.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**Count1Val** Pointer to the counter value variable of counter 1.

**Count1Stat** Pointer to the counter status variable of counter 1.

01h: Latch input status  
02h: Error  
04h: Reference already detected  
08h: Latch input status change

**Count2Val** Pointer to the counter value variable of counter 2.

**Count2Stat** Pointer to the counter status variable of counter 2.

01h: Latch input status  
02h: Error  
04h: Reference already detected  
08h: Latch input status change

**Reading out all counters (continued)**

**Delivery parameter:** **Count3Val** Pointer to the counter value variable of counter 3.

**Count3Stat** Pointer to the counter status variable of counter 3.

01h: Latch input status

02h: Error

04h: Reference already detected

08h: Latch input status change

**Return value:**     **0**   OK

**-1**   Error

## Setting the reference mode

**Function name:** HS\_ZP3B\_SetRefMode

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_SetRefMode (long CardNo, long CounterNo, long RefMode);
----------------	---

```

Delphi:      HS_ZP3B_SetMode (CardNo    :longint;
                          CounterNo  :longint;
                          RefMode    :longint)
                          :longint

```

<b>Description:</b>	Sets the counter reference mode.
---------------------	----------------------------------

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CounterNo** Counter number (1...3)

**RefMode**      Reference mode

00h:	No reference detection (default)
01h:	Single reference detection
02h:	Cyclical reference detection

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error



## Latching all counter values

**Function name:** HS\_ZP3B\_StopAllCounter

**Syntax:**           **Visual C++:** long HS\_ZP3B\_StopAllCounter (long CardNo);

**Delphi:** HS\_ZP3B\_StopAllCounter (CardNo :longint)  
:longint

<b>Description:</b>	Latches all counter values of all counter cards.
---------------------	--

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Reading out the counter referece register

**Function name:** HS\_ZP3B\_ReadRefReg

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_ReadRefReg (long CardNo, long CounterNo, long *RefVal);
----------------	---

```
Delphi:      HS_ZP3B_ReadRefReg      (CardNo      :longint;  
                                      CounterNo    :longint;  
                                      var RefVal    :longint)  
  
            :longint
```

<b>Description:</b>	Reads out the counter reference register of one counter.
---------------------	--

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CounterNo** Counter number (1...3)

<b>RefVal</b>	Pointer to the reference value variable.
---------------	--

<b>Return value:</b>	<b>0</b>	OK
----------------------	----------	----

**-1 Error**

## Writing the counter reference register

**Function name:** HS\_ZP3B\_WriteRefReg

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_WriteRefReg (long CardNo, long CounterNo, long RefVal);
----------------	---

**Delphi:** HS\_ZP3B\_WriteRefReg (CardNo :longint;  
CounterNo :longint;  
RefVal :longint)

:longint

**Description:** Writes into the counter reference register of one counter.

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CounterNo** Counter number (1...3)

**RefVal**      Reference value

<b>Return value:</b>	<b>0</b>	OK
----------------------	----------	----

**-1 Error**

## Reading out the card type

**Function name:** HS\_ZP3B\_ReadCardType

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3B_ReadCardType (long CardNo long *CardType);
----------------	---

```

Delphi:      HS_ZP3B_ReadCardType  (CardNo      :longint
                                     var CardType  :longint)
                                     :longint

```

<b>Description:</b>	Reads out the type of the counter card.
---------------------	---

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CardType** Pointer to the card type variable.

01h: 11  $\mu A_{SS}$  inputs  
02h: 1  $V_{SS}$  inputs  
03h: TTL inputs

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Reading out the result of probe identification

<b>Function name:</b>	<b>HS_ZP3B_ProbeAvail</b>		
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3B_ProbeAvail	(long CardNo long *ProbeAvail);
	<b>Delphi:</b>	HS_ZP3B_ProbeAvail	(CardNo :longint var ProbeAvail :longint)
		:longint	
<b>Description:</b>	Reads out the result of probe identification.		
<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)	
	<b>ProbeAvail</b>	Pointer to the variable taking up the result of probe identification.	
		00h:	No probe connected
		01h:	Probe connected
<b>Return value:</b>	<b>0</b>	OK	
	<b>-1</b>	Error	

### 3.2 Extended library functions (HS\_ZP3X.DLL)

The DLL "HS\_ZP3X.DLL" extended library functions contain the following calls:

DLL Funktionsaufruf	Bedeutung
HS_ZP3X_Initialize	Loads drivers and checks whether the card is in the PC and initialises it.
HS_ZP3X_GetVersion	Checks the versions of drivers and the extended DLL.
HS_ZP3X_SetMode	Sets a counter's reference and latch modes.
HS_ZP3X_SetCounterMul	Sets the multiplier for one counter.
HS_ZP3X_GetCounter	Reads out the counter value and status of one counter.
HS_ZP3X_GetAllCounter	Reads out the counter value and status of all counters on a card.
HS_ZP3X_ResetCounterStop	Resets the latch status of a counter.
HS_ZP3X_ResetRef	Resets the counter card reference.
HS_ZP3X_GetCounterCurrent	Reads out the counter value and status of one counter continuously.
HS_ZP3X_GetAllCounterCurrent	Reads out the counter value and status of all counters on a card continuously.
HS_ZP3X_SetDistCode	Sets the distance coded reference parameters of linear encoders.
HS_ZP3X_StopAllCounter	Latches all counter values of all counter cards.
HS_ZP3X_ReadCardType	Reads out the type of the counter card.
HS_ZP3X_ProbeAvail	Reads out the result of probe identification.



## Version query

<b>Function name:</b>	<b>HS_ZP3X_GetVersion</b>		
<b>Syntax:</b>	<b>Visual C++:</b>	long HS_ZP3X_GetVersion (char* DriverVersion, char* DllVersion)	
	<b>Delphi:</b>	HS_ZP3X_GetVersion (DriverVersion :pchar; DllVersion :pchar)	
		:longint	
<b>Description:</b>	Checks the versions of drivers and the extended DLL.		
<b>Delivery parameter:</b>	<b>DriverVersion</b>	Pointer to the string where the driver version is entered.	
	<b>DllVersion</b>	Pointer to the string where the DLL version is entered. The strings should have at least 80 characters.	
<b>Return value:</b>	<b>0</b>	OK	
	<b>-1</b>	Error	



## Setting reference and latch modes

**Function name:** 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

**Description:**       Sets a counter's reference and latch modes.  
Please also note the explanation of the function below.

**Delivery parameter:** **CardNo**     Logical card number (1...15)

**CounterNo** Counter number (1...3)

**RefMode**     Reference mode

00h:     No reference detection (default)

01h:     Single reference detection

02h:     Cyclical reference detection

**StopMode**    Latch mode

00h:     No latch detection (default)

01h:     Manual latch

02h:     Automatic latch

00h:     Latch in increasing flank (default)

10h:     Latch in falling flank

20h:     Latch in increasing and in falling flank

**Return value:**       **0**    OK

**-1**   Error

## Setting reference and latch modes (continued)

### Function explanations

To use the application it is necessary to explain the functionality of the reference analysis and the counter latch function.

#### Reference analysis

- With **cyclical (repeated) reference analysis** the counter value is set to 0 again when the reference point is passed. This is often used with rotary encoders where the angle is automatically standardised to 0 - 360°.
- The **single reference analysis** only sets the counter to 0 the first time the reference mark is passed. The card will then ignore any other reference signals. As long as no reference signal is set in this mode the counter has no defined value.
- The **distance coded reference** sets the counter to 0 after a short moving. The card will then ignore any other reference signals. As long as no reference signal is set in this mode the counter has no defined value.

#### Counter latch function

- **Automatic counter latch release** means that the counter value output will be blocked as long as there is a latch signal. As soon as this is disconnected the output is released automatically again. The status of the latch register is determined by reading out the counter values and can thus be used for further analyses. A typical use for this is edge scanning using an edge sensor.
- With **manual counter latch release** the counter value output is also blocked until the latch signal disappears. This must then be released explicitly with the "HS\_ZP3X\_ResetCounterStop" command for each axis separately.

### Multiplier for counter values

<b>Function name:</b>	<b>HS_ZP3X_SetCounterMul</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_SetCounterMul (long CardNo, long CounterNo, long Multiplier)  <b>Delphi:</b> HS_ZP3X_SetCounterMul (CardNo :longint; CounterNo :longint; Multiplier :longint)  :longint
<b>Description:</b>	Sets the multiplier for one counter. The counter direction can be reversed easily using the multiplier. The multiplier must then be set to -1.
<b>Delivery parameter:</b>	<b>CardNo</b> Logical card number (1...15)  <b>CounterNo</b> Counter number (1...3)  <b>Multiplier</b> Multiplier
<b>Return value:</b>	<b>0</b> OK  <b>-1</b> Error

## Reading out counters individually

**Function name:** 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;

**Description:** Reads out the counter value and status of one counter. The counter value is included with the reference value and the multiplier.

Note: If the latch input is active (also see the "Setting reference and latch modes" function), the counter value is latched.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**CounterNo** Counter number (1...3)

**Count** Pointer to the counter value variable.

**Status** Pointer to the counter status variable.

01h: Error  
02h: Reference already detected  
04h: Counter latched  
08h: Latch input status  
10h: Latch input status change

**Return value:**   **0** OK

**-1** Error

## Reading out all counters

**Function name:** 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>Description:</b>	Reads out the counter value and status of all counters on a card. The counter value is included with the reference value and the multiplier.
---------------------	--

Note: If the latch input is active (also see the “Setting reference and latch modes” function), the counter reading is latched.

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**Count1**      Pointer to the counter value variable of counter 1.

**Status1**      Pointer to the counter status variable of counter 1.

01h:	Error
02h:	Reference already detected
04h:	Counter latched
08h:	Latch input status
10h:	Latch input status change

**Count2**      Pointer to the counter value variable of counter 2.

**Status2**      Pointer to the counter status variable of counter 2.

01h:	Error
02h:	Reference already detected
04h:	Counter latched
08h:	Latch input status
10h:	Latch input status change

**Reading out all counters (continued)**

<b>Delivery parameter:</b>	<b>Count3</b>	Pointer to the counter value variable of counter 3.
	<b>Status3</b>	Pointer to the counter status variable of counter 3.
		01h: Error
		02h: Reference already detected
		04h: Counter latched
		08h: Latch input status
		10h: Latch input status change
<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Resetting the latch input

<b>Function name:</b>	<b>HS_ZP3X_ResetCounterStop</b>
<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_ResetCounterStop (long CardNo, <span style="margin-left: 150px;">long CounterNo)</span>  <b>Delphi:</b> HS_ZP3X_ResetCounterStop (CardNo        :longint; <span style="margin-left: 150px;">CounterNo        :longint)</span>  <span style="margin-left: 150px;">:longint;</span>
<b>Description:</b>	Resets the latch status of a counter. The counter then immediately continues to count again.
<b>Delivery parameter:</b>	<b>CardNo</b> Logical card number (1...15)  <b>CounterNo</b> Counter number (1...3)
<b>Return value:</b>	<b>0</b> OK  <b>-1</b> Error
<b>Note:</b>	It must be released separately for each axis.

## Resetting the reference

**Function name:** HS\_ZP3X\_ResetRef

<b>Syntax:</b>	<b>Visual C++:</b> long HS_ZP3X_ResetRef (long CardNo, long CounterNo);
----------------	--

```

Delphi:      HS_ZP3X_ResetRef  (CardNo    :longint;
                                CounterNo  :longint)

                                :longint;

```

<b>Description:</b>	Resets the counter card reference. The reference point must be passed once again.
---------------------	--

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

**CounterNo** Counter number (1...3)

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error



## Reading out individual counters continuously

**Function name:** 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;

**Description:** Reads out the counter value and status of one counter continuously. The counter value is included with the reference value and the multiplier.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**CounterNo** Counter number (1...3)

**Count** Pointer to the counter value variable.

**Status** Pointer to the counter status variable.

01h: Error  
02h: Reference already detected  
04h: Counter latched  
08h: Latch input status  
10h: Latch input status change

**Return value:**   **0** OK  
                  **-1** Error

## Reading out all counters continuously

**Function name:** 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;

**Description:** Reads out the counter value and status of all counters on a card continuously. The counter value is included with the reference value and the multiplier.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**Count1** Pointer to the counter value variable of counter 1.

**Status1** Pointer to the counter status variable of counter 1.

01h: Error  
02h: Reference already detected  
04h: Counter latched  
08h: Latch input status  
10h: Latch input status change

**Count2** Pointer to the counter value variable of counter 2.

**Status2** Pointer to the counter status variable of counter 2.

01h: Error  
02h: Reference already detected  
04h: Counter latched  
08h: Latch input status  
10h: Latch input status change

**Reading out all counters continuously (continued)**

<b>Delivery parameter:</b>	<b>Count3</b>	Pointer to the counter value variable of counter 3.
	<b>Status3</b>	Pointer to the counter status variable of counter 3.
		01h: Error
		02h: Reference already detected
		04h: Counter latched
		08h: Latch input status
		10h: Latch input status change
<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Setting distance coded reference

<b>Function name</b>	<b>HS_ZP3X_SetDistCode</b>
<b>Syntax:</b>	<p><b>Visual C++:</b> long HS_ZP3X_SetCistCode (long CardNo, long CounterNo, long Distance, long Pitch, long Offset, long Direction);</p> <p><b>Delphi:</b> HS_ZP3X_SetDistCode (CardNo :longint; CounterNo :longint; Distance :longint; Pitch :longint; Offset :longint; Direction :longint)</p> <p>:longint</p>
<b>Description:</b>	Sets the distance coded reference parameters of linear encoders.
<b>Delivery parameter:</b>	<p><b>CardNo</b> Logical card number (1...15)</p> <p><b>CounterNo</b> Counter number (1...3)</p> <p><b>Distance</b> Distance between reference marks (mm)</p> <p><b>Pitch</b> Grating pitch (µm)</p> <p><b>Offset</b> Encoder offset (mm)</p> <p><b>Direction</b> Basic counter direction of the linear encoder</p> <p>00h: positive (Default), for example Acu-Rite 01h: negative, for example Heidenhain</p>
<b>Return value:</b>	<p><b>0</b> OK</p> <p><b>-1</b> Error</p>
<b>Note:</b>	<p>The calls of the extended library functions have to be in order of:</p> <p>HS_ZP3X_Initialize(...) HS_ZP3X_SetDistCode(...) HS_ZP3X_SetMode(...)</p> <p>If you want to work with the default values (Distance=20mm; Pitch=20µm; Offset=0mm), the call "HS_ZP3X_SetDistCode(...)" can be dropped.</p>

## Latching all counter values

**Function name:** HS\_ZP3X\_StopAllCounter

**Syntax:**           **Visual C++:** long HS\_ZP3X\_StopAllCounter (long CardNo);

**Delphi:** HS\_ZP3X\_StopAllCounter (CardNo :longint)  
:longint

<b>Description:</b>	Latches all counter values of all counter cards.
---------------------	--

<b>Delivery parameter:</b>	<b>CardNo</b>	Logical card number (1...15)
----------------------------	---------------	------------------------------

<b>Return value:</b>	<b>0</b>	OK
	<b>-1</b>	Error

## Reading out the card type

**Function name:** HS\_ZP3X\_ReadCardType

**Syntax:**

**Visual C++:** long HS\_ZP3X\_ReadCardType (long CardNo  
long \*CardType);

**Delphi:** HS\_ZP3X\_ReadCardType (CardNo :longint  
var CardType :longint)  
:longint

**Description:** Reads out the type of the counter card.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**CardType** Pointer to the card type variable.

01h: 11  $\mu A_{SS}$  inputs  
02h: 1  $V_{SS}$  inputs  
03h: TTL inputs

**Return value:** **0** OK  
**-1** Error

## Reading out the result of probe identification

**Function name:** HS\_ZP3X\_ProbeAvail

**Syntax:**

**Visual C++:** long HS\_ZP3X\_ProbeAvail (long CardNo  
long \*ProbeAvail);

**Delphi:** HS\_ZP3X\_ProbeAvail (CardNo :longint  
var ProbeAvail :longint)  
:longint

**Description:** Reads out the result of probe identification.

**Delivery parameter:** **CardNo** Logical card number (1...15)

**ProbeAvail** Pointer to the variable taking up the  
result of probe identification.

00h: No probe connected  
01h: Probe connected

**Return value:** **0** OK  
**-1** Error

### 3.3 Linking the DLLs

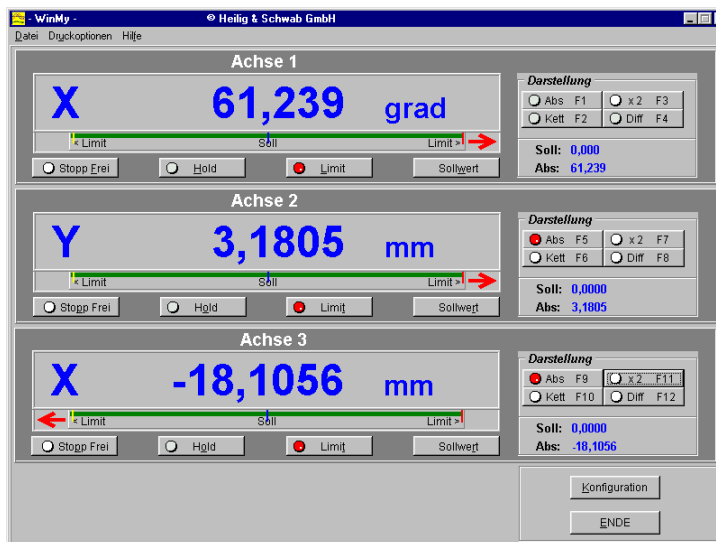
The DLLs are linked by means of an interface module in the appropriate programming language. This module contains the necessary declarations to access the library functions.

This interface module is available on the data carrier in the package in Visual C++ and Borland Delphi.

You must create this module yourself for other programming languages such as Visual Basic.

### 3.4 WinMy demo and measuring program

The Windows WinMy software is used to test the counter card and enables you to access measuring systems with incremental linear and rotary encoders directly. The program supports counter results displayed as absolute, differential or incremental measurements.



*WinMy demo program interface*

### Software licence

This agreement sets out the licence conditions for the software supplied. This software is intended solely for running the counter card and may only be installed for this purpose. This software or even parts of it may not be used in other applications or made available to third parties.



## 4 Technical data

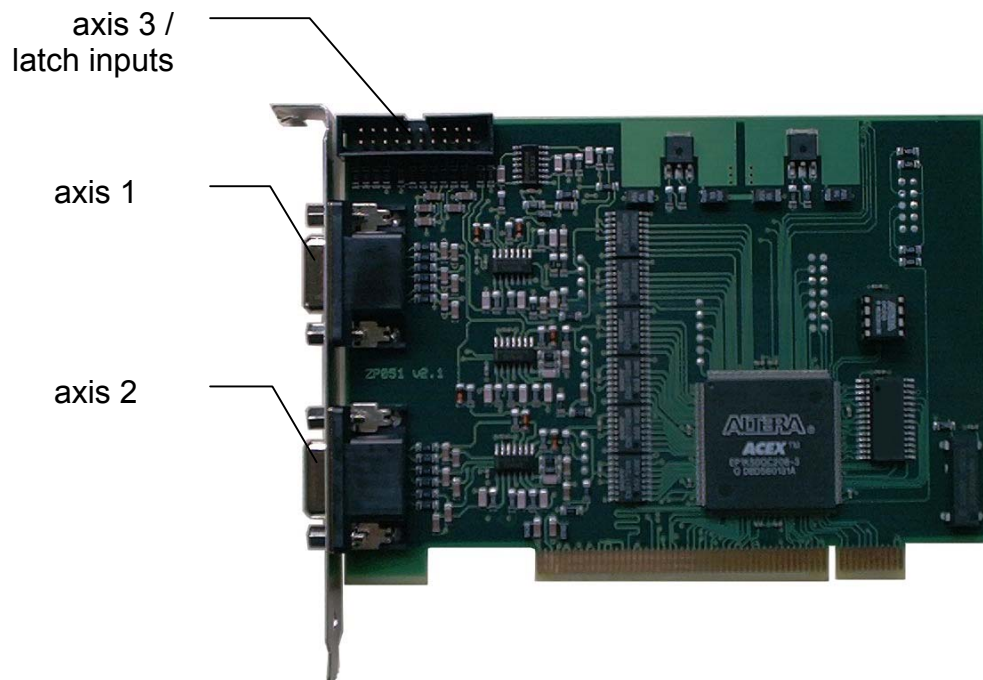
### 4.1 Connector layout

Two of the three measuring system inputs (axis 1 and axis 2) are connected directly to the counter card using two 9 pin sub-D socket connectors that are located in the counter card bracket.

The third axis can be connected by means of a 20 pin shrouded header on the ZP051 circuit card (see picture “ZP051-20 connector layout”).

If latch inputs are needed up to three latch inputs can be connected using the 20 pin shrouded header, too.

The technical input specifications are described in the following sections.



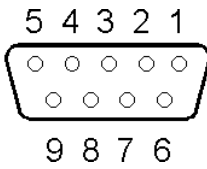
*ZP051-20 connector layout*

## 4.2 Connector allocation (axis 1 and 2)



If axis 3 is also connected to a 9 pin socket connector by using the ZP051/1 bracket or the ZA101 counter adapter card, the following information is valid for all three axis.

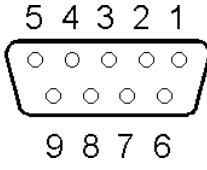
### Current inputs (standard version / ZP051-20)

Pin	Signal	Notes
1	- $\varphi_0$	 <p>9 pin sub-D socket connectors</p>
2	0 V	
3	- $\varphi_{90}$	
4	Shield	
5	- REF	
6	+ $\varphi_0$	
7	+ 5 V	
8	+ $\varphi_{90}$	
9	+ REF	

### Signal features

Signal:	7 - 15 $\mu A_{ss}$ , typ. 11 $\mu A_{ss}$ (sinusoid)
Signal spacing:	256 x interpolation
Reference signals:	3.5 - 8 $\mu A_{ss}$ , typ. 5 $\mu A_{ss}$
Counter width:	28 bits
Phase angle $\varphi_0$ / $\varphi_{90}$ :	$90^\circ \pm 10^\circ$
Input frequency:	0 - 75 kHz
Display step:	free choice

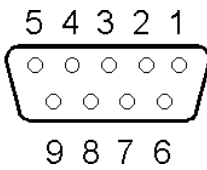
**TTL inputs (optional / ZP051-20-TTL)**

Pin	Signal	Notes
1	/ $U_{a1}$	 9 pin sub-D socket connectors
2	0 V	
3	/ $U_{a2}$	
4	Shield	
5	/ $U_{a0}$	
6	$U_{a1}$	
7	+ 5 V	
8	$U_{a2}$	
9	$U_{a0}$	

**Signal features**

Signal:	$U_H \geq 2 \text{ V}$ , $U_L \leq 0,8 \text{ V}$ (EIA Standard RS422)
Signal spacing:	4 x interpolation
Reference signals:	$U_H \geq 2 \text{ V}$ , $U_L \leq 0,8 \text{ V}$ (EIA Standard RS422)
Counter width:	28 bits
Input frequency:	0 - 500 kHz
Display step:	free choice

**Voltage inputs (optional / ZP051-20-V)**

Pin	Signal	Notes
1	A -	 9 pin sub-D socket connectors
2	0 V	
3	B -	
4	Shield	
5	R -	
6	A +	
7	+ 5 V	
8	B +	
9	R +	

**Signal features**

Signal:	0,8 - 1,2 $V_{ss}$ , typ. 1 $V_{ss}$ (sinusoid)
Signal spacing:	256 x interpolation
Reference signals:	0,2 - 0,85 $V_{ss}$
Counter width:	28 bits
Phase angle A / B:	$90^\circ \pm 10^\circ$
Input frequency:	0 - 75 kHz
Display step:	free choice

### 4.3 Connector allocation (axis 3 / latch inputs axis 1 – 3)

Pin	Signal			Hinweis
	Current	(optional)		
		TTL	Voltage	
1	- $\varphi_0$	/ $U_{a1}$	A -	<div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div>1</div><div>2</div><p>20 pin shrouded header on board</p></div>
2	+ $\varphi_0$	$U_{a1}$	A +	
3	GND	0 V	0 V	
4	+ 5V	+ 5 V	+ 5 V	
5	- $\varphi_{90}$	/ $U_{a2}$	B -	
6	+ $\varphi_{90}$	$U_{a2}$	B +	
7	GND	0 V	0 V	
8	+ REF	$U_{a0}$	R +	
9	- REF	/ $U_{a0}$	R -	
10	+ 5V	+ 5 V	+ 5 V	
11	GND	0 V	0 V	
12				
13				
14				
15	GND	GND	GND	
16				
17	Latch 1	Latch 1	Latch 1	
18	Latch 2	Latch 2	Latch 2	
19	Latch 3	Latch 3	Latch 3	
20	+ 5V	+ 5V	+ 5V	

### 4.4 Physical and mechanical specifications

Dimensions:	140 x 100 mm
Weight:	about 140 g (without cable)
Storage temperature:	- 30° to + 70° C
Operating temperature:	0° to + 45°C
Rel. humidity:	< 75 %

## 5 Guarantee terms

The manufacturer guarantees their hardware and software products for a period of one year from delivery date. During this guarantee period the manufacturer will either repair in the manufacturing firm or replace products that prove to be faulty.

This applies only if the card has been operated by appropriately trained personnel.

**The guarantee terms do not apply if:**

- damage has been caused by faulty or inappropriate repairs by the customer.
- damage has been caused by the customer's software.
- damage has been caused by incorrect use of the software.
- the connections to the customer's measuring system have been damaged.
- damage has been caused by unauthorised modifications.
- damage has been caused by not observing the storage and operating conditions.
- units have had the serial number removed.
- damage has been caused by extra high voltage or electrostatic discharge.