DIN EN ISO 9001:2000
zertifiziert

ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER
+49 (0)7223 / 9493 – 0

# Technisches Referenzhandbuch

# ADDICOUNT APCI-1710

## Funktionsprogrammierbare Zählerkarte für den PCI- und CompactPCI-Bus

Ausgabe: 09.02-09/2005

Produktinformation

Dieses Handbuch enthält die technischen Anlagen, wichtige Anleitungen zur korrekten Inbetriebnahme und Nutzung sowie Produktinformation entsprechend dem aktuellen Stand vor der Drucklegung.
Der Inhalt dieses Handbuchs und die technischen Daten des Produkts können ohne vorherige Ankündigung geändert werden. Die ADDI-DATA GmbH behält sich das Recht vor, Änderungen bzgl. der technischen Daten und der hierin enthaltenen Materialien vorzunehmen.

Gewährleistung und Haftung

Der Nutzer ist nicht berechtigt, über die vorgesehene Nutzung der Karte hinaus Änderungen des Werks vorzunehmen sowie in sonstiger Form in das Werk einzugreifen.
ADDI-DATA übernimmt keine Haftung bei offensichtlichen Druck- und Satzfehlern. Darüber hinaus übernimmt ADDI-DATA, soweit gesetzlich zulässig, weiterhin keine Haftung für Personen- und Sachschäden, die darauf zurückzuführen sind, dass der Nutzer die Karte unsachgemäß installiert und/oder in Betrieb genommen oder bestimmungswidrig verwendet hat, etwa indem die Karte trotz nicht funktionsfähiger Sicherheits- und Schutzvorrichtungen betrieben wird oder Hinweise in der Betriebsanleitung bzgl. Transport, Lagerung, Einbau, Inbetriebnahme, Betrieb, Grenzwerte usw. nicht beachtet werden. Die Haftung ist ferner ausgeschlossen, wenn der Betreiber die Karte oder die Quellcode-Dateien unbefugt verändert und/oder die ständige Funktionsbereitschaft von Verschleißteilen vorwerfbar nicht überwacht wurde und dies zu einem Schaden geführt hat.

Urheberrecht

Dieses Handbuch, das nur für den Betreiber und dessen Personal bestimmt ist, ist urheberrechtlich geschützt. Die in der Betriebsanleitung und der sonstigen Produktinformation enthaltenen Hinweise dürfen vom Nutzer des Handbuchs weder vervielfältigt noch verbreitet und/oder Dritten zur Nutzung überlassen werden, soweit nicht die Rechstübertragung im Rahmen der eingeräumten Produktlizenz gestattet ist. Zuwiderhandlungen können zivil- und strafrechtliche Folgen nach sich ziehen.

ADDI-DATA-Software Produktlizenz

Bitte lesen Sie diese Lizenz sorgfältig durch, bevor Sie die Standardsoftware verwenden.
Das Recht zur Benutzung dieser Software wird dem Kunden nur dann gewährt,
wenn er den Bedingungen dieser Lizenz zustimmt.
Die Software darf nur zur Einstellung der ADDI-DATA Karten verwendet werden.
Das Kopieren der Software ist verboten (außer zur Archivierung/Datensicherung und zum Austausch defekter Datenträger). Deassemblierung, Dekompilierung, Entschlüsselung und Reverse Engineering der Software ist verboten. Diese Lizenz und die Software können an eine dritte Partei übertragen werden, sofern diese Partei eine Karte käuflich erworben hat, sich mit allen Bestimmungen in diesem Lizenzvertrag einverstanden erklärt und der ursprüngliche Besitzer keine Kopien der Software zurückhält.

Warenzeichen

- ADDI-DATA ist ein eingetragenes Warenzeichen der ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ sind eingetragene Warenzeichen von Borland Insight Company.
- Microsoft C, Visual C++, Windows XP, 98, Windows 2000, Windows 95, Windows NT, EmbeddedNT und MS DOS sind eingetragene Warenzeichen von Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DasyLab, Diadem sind eingetragene Warenzeichen von National Instruments Corp.
- CompactPCI ist ein eingetragenes Warenzeichen der PCI Industrial Computer Manufacturers Group.
- VxWorks ist ein eingetragenes Warenzeichen von Windriver.

# WARNUNG

**Bei unsachgemäßen Einsatz und bestimmungswidrigem Gebrauch der Karte können:**

**Personen verletzt werden,**

**Baugruppe, PC und Peripherie beschädigt werden,**

**Umwelt verunreinigt werden.**

**Schützen Sie sich, andere und die Umwelt!**

**Sicherheitshinweise unbedingt lesen.**

Liegen Ihnen keine Sicherheitshinweise vor, so fordern Sie diese bitte an.

**Anweisungen des Handbuches beachten.**

Vergewissern Sie sich, dass Sie keinen Schritt vergessen haben. Wir übernehmen keine Verantwortung für Schäden, die aus dem falschen Einsatz der Karte hervorgehen könnten.

**Folgende Symbole beachten:**

**WICHTIG!**
kennzeichnet Anwendungstipps und andere nützliche Informationen.

**WARNUNG!**
bezeichnet eine möglicherweise gefährliche Situation.
Bei Nichtbeachten des Hinweises können Karte, PC und/oder Peripherie **zerstört** werden.

## Abbildungen

## Tabellen

# 1 DEFINITION DES VERWENDUNGSBEREICHS

## 1.1 Bestimmungsgemäßer Zweck

Die Karte **APCI-1710** eignet sich für den Einbau in einen PC mit PCI 5V/32 Bit Steckplätzen, der für elektrische Mess-, Steuer-, Regel- und Labortechnik im Sinne der EN 61010-1 (IEC 61010-1), eingesetzt wird.

## 1.2 Bestimmungswidriger Zweck

Die Karte **APCI-1710** darf nicht als sicherheitsgerichtetes Betriebsmittel (safety related part, SRP) eingesetzt werden.

Die Karte **APCI-1710** darf nicht in explosionsgefährdeten Atmosphären eingesetzt werden.

## 1.3 Allgemeine Beschreibung der Karte

Der Datenaustausch zwischen der Karte APCI-1710 und der Peripherie erfolgt über ein geschirmtes Kabel. Das Kabel ST370-16 ist an den 50pol. SUB-D Stiftstecker der Karte APCI-1710 anzuschließen.

Die Karte besitzt:

1. **bis zu 16 differentielle Eingänge** zur Verarbeitung von digitalen 5V-Signalen. Die Eingänge (RS422) können zum Anschluss von Gebern eingestellt werden. Sie können auch als TTL Eingänge benutzt werden.
- **24 V Version (APC1710-24):** Diese Eingänge können auch zur Verarbeitung von 24 V- Signalen bestückt werden

2. **bis zu 12 massenbezogene Eingänge** stehen als frei verwendbare bzw. als funktionsgebundene Eingänge zur Verfügung, soweit sie in den vorgegebenen Grenzwerten betrieben werden.
- **Option:** Diese Eingänge können auch zur Verarbeitung von 5V-Signalen bestückt werden.

3. **bis zu 12 Ausgänge** zur Ausgabe von digitalen 5V- bzw. 24 V-Signalen. (In diesem Fall stehen 20 Eingänge zur Verfügung.)
- Bis zu 8 differentielle Ausgänge (RS422) können zum Anschluss von SSI-Gebern benutzt werden.
- 4 massenbezogene 24 V Ausgänge stehen zur Verfügung als frei verwendbare bzw. als funktionsgebundene Ausgänge, soweit sie in den vorgegebenen Grenzwerten betrieben werden.
- **Option:** Die 4 massenbezogenen 24 V-Ausgänge können auch zur Ausgabe von 5 V TTL-Signalen bestückt werden.

Die **APCI-1710** Karte besteht aus 4 "Funktionsmodulen". Jedem Funktionsmodul werden digitale Ein- und Ausgänge fest zugewiesen.
Pro Modul stehen 8 Leitungen zur Verfügung: (Siehe Blockdiagramm der Ein- und Ausgänge).

Die Leitungen werden je nach Karte wie folgt aufgeteilt:
**Eingangsleitungen:**
- 2 x TTL, RS422 (Signale C, D)
- 3 x 24 V, 5 V optional (Signale E, F, G)

**Ausgangsleitungen**
- 1 x 24 V, TTL optional (Signal H)

**Frei definierbare Leitungen (Ein- oder Ausgang)**
- 2 x TTL, RS422, (Signale A, B)

**24 V Version (ACPI-1710-24 V)**
7 x 24 V Eingänge (Signale A to G)
1 x 24 V Eingänge (Signal H)

Der Einsatz der Karte **APCI-1710** in Kombination mit externen Klemmen- oder Relaisplatinen setzt eine fachgerechte Installation in einem geschlossenen Schaltschrank voraus. Prüfen Sie das Schirmdämpfungsmaß von PC-Gehäuse und Kabelschirm, bevor Sie das Gerät in Betrieb nehmen.

Der Anschluss unseres Standardkabels **ST370-16** erfüllt die Mindestforderungen:
- metallisiertes Steckergehäuse,
- geschirmtes Kabel,
- Kabelschirm über Isolierung zurückgeklappt und beidseitig fest mit dem Steckergehäuse verschraubt.

Eine andere oder darüber hinausgehende Benutzung gilt als nicht bestimmungsgemäß. Für hieraus entstehende Schäden haftet der Hersteller nicht.

Die bestimmungsgemäße Verwendung erfordert das Beachten aller Sicherheitshinweise und des Technischen Referenzhandbuches.

Beim Einsatz der Karte in den PC können sich die Störfestigkeits- und Emissionswerte des PCs verändern. Erhöhte Emissionen oder verringerte Störfestigkeit können zur Folge haben, dass die Konformität des Systems nicht mehr sichergestellt ist. Prüfen Sie daher das Schirmdämpfungsmaß von PC-Gehäuse und Kabelschirm, bevor Sie das Gerät in Betrieb nehmen.

Der Einsatz der Karte in explosionsgefährdeten Bereichen ist unzulässig. Die Karte sollte ohne zusätzliche Fixierung keinen Vibrationen ausgesetzt werden.

Die Karte muss bis zum Einsatz in ihrer antistatischen Klarsichtpackung bleiben.

Entfernen Sie nicht die Kennzeichnungsnummern der Karte, da dadurch ein Garantieverlust erfolgt.

# 2 BENUTZER

## 2.1 Qualifikation

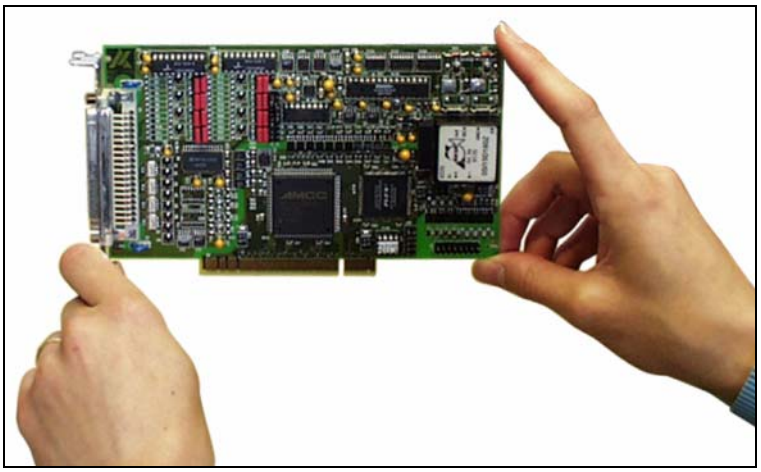
Nur eine ausgebildete Elektronikfachkraft darf folgende Tätigkeiten ausführen:
- Installation
- Inbetriebnahme
- Betrieb
- Instandhaltung.

## 2.2 Persönliche Schutzausrüstung

Beachten Sie die länderspezifischen Bestimmungen zur:
- Unfallverhütung
- Einrichtung von elektrischen und mechanischen Anlagen
- Funkentstörung.

# 3 HANDHABUNG DER KARTE

**Abb. 3-1: Richtige Handhabung**



# 4 TECHNISCHE DATEN

## 4.1 Elektromagnetische Verträglichkeit (EMV)

Der PC unterliegt der Norm EN 61326 (IEC 61326) und muss die EMV-Schutzanforderungen erfüllen.

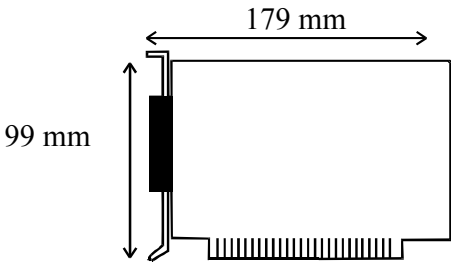
Die Karte wurde in einem akkreditierten Labor den EMV-Prüfungen nach der Norm EN61326 (IEC61326) unterzogen. Folgende Grenzwerte werden eingehalten:

| | **Istwert** | **Sollwert** |
|---|---|---|
| ESD (Kontaktentladung/Luftentladung) | 4/8 kV | 4/8 kV |
| Felder | 10 V/m | 10 V/m |
| Burst | 2 kV | 2 kV |
| Geleitete Funkstörungen | 10 V | 10 V |

## 4.2 Mechanischer Aufbau

Die Karte ist auf einer 4-Lagen Leiterplatte aufgebaut.

**Abmessungen:**




Gewicht: ..... ca. 150 g
Einbau in: ..... 32/64-Bit PCI-Steckplatz (5 V)
Anschluss zur Peripherie:..... 50-pol. SUB-D Stiftstecker
Zubehör[1]:..... Standardkabel ST370-16
Klemmenplatine: PX8000

1 Nicht im Standard-Lieferumfang enthalten.

## 4.3 Versionen

Die Karte **APCI-1710** ist in 2 Versionen erhältlich.

| Version | Format | Onboard Funktions-module | Option |
|---|---|---|---|
| APCI-1710 | PCI | 4 | 5 V Eingänge (massenbezogen) Digitale 5 V Ausgänge |
| APCI-1710-24V | PCI | 4 | - |

## 4.4 Grenzwerte

Höhenlage: ..... 2000 m über NN
Betriebstemperatur: ..... 0 bis 60°C
Luftfeuchtigkeit : ..... 30 bis 99% ohne Kondensation
Lagertemperatur: ..... -25 bis + 70°C

**PC-Mindestvoraussetzungen:**
**PCI BIOS ab Version 1.0**
Bus Geschwindigkeit: ..... < 33 MHz
Datenbuszugriff: ..... 32-Bit
Dekodierung: ..... im 64 K E/A Bereich des PCs
"Target Only"-Betrieb
Betriebssystem: ..... Windows DOS 3.11,
Windows NT 4.0, 9x, 2000

**Ressourcen:**
E/A-Bereiche
3 Bereiche: ..... 64 Bytes,
8 Bytes,
256 Bytes.
IRQs: ..... INTA vom PCI-Bus

**Energiebedarf:**
Betriebsspannung vom PC: ..... 5 V ± 5%
Stromverbrauch in mA (ohne Last): ..... Siehe Tabelle (± 10%)

| | APCI-1710 | APCI-1710-24 |
|---|---|---|
| + 5 V vom PC | 600 mA | 450 mA |
| + 24 V ext. | 10 mA | - |

## 4.4.1 Eingänge

Anzahl der Eingänge: ..... 28
Eingangstyp:
Differentielle Eingänge bzw. TTL: ..... 16
24 V masssenbezogene Eingänge ..... 12

**APCI-1710**

**Differentielle Eingänge, 5 V**
Erfüllt die EIA-Standards RS485
Nominalspannung: ..... 5 VDC
Gleichtakt Bereich: ..... +12 / -7 V
Max. Differentielle Spannung ..... ±12 V
Eingangsempfindlichkeit: ..... 200 mV
Eingangshysterese: ..... 50 mV
Eingangsimpedanz: ..... 12 kΩ
Abschlusswiderstand (typ.): ..... 150 Ω in Serie mit 10 nF
Signalverzögerung: .....120 nS (bei Nominalsp.)
Max. Eingangsfrequenz: ..... 5 MHz (bei Nominalsp.)
"Open Circuit Fail Safe Receiver Design" "1" = Eingänge offen

**Massenbezogene Eingänge, 24 V (Kanäle E, F, G)**
Nominalspannung: ..... 24 VDC
Eingangsstrom bei Nominalspannung: ..... 11 mA
Logische Eingangspegel: ..... $U_H$ max.: 30 V
$U_H$ min.: 17 V
$U_L$ max.: 15 V
$U_L$ min.: 0 V
Signalverzögerung: ..... 120 ns (bei Nominalsp.)
Maximale Eingangsfrequenz: ..... 2,5 MHz (bei Nominalsp.)

**Massenbezogene Eingänge, 5 V ( OPTION, Kanäle E,F,G)**
Nominalspannung: ..... 5 VDC
Eingangsstrom bei Nominalspannung: ..... 10 mA
Logische Eingangspegel: ..... $U_H$ max.: 7 V
$U_H$ min.: 2 V
$U_L$ max.: 0,8 V
$U_L$ min.: 0 V
Signalverzögerung: ..... 120 ns (bei Nominalsp.)
Maximale Eingangsfrequenz: ..... 5 MHz (bei Nominalsp.)

**APCI-1710-24 V**

**24 V Eingänge (Kanäle A bis G)**
Diese Kartenversion ist speziell für den Anschluss von 24 V Gebern bestimmt.
Auf den Eingängen können nur 24 V Signale angeschlossen werden.

| | | |
|---|---|---|
| Nominalspannung: ..... | 24 VDC / 11 mA | |
| Max. Eingangsfrequenz: ..... | 1 MHz (bei Nominalsp.) mit Eingangsfilter | |
| Logische Eingangspegel: ..... | $U_H$ max.: | 30 V |
| | $U_H$ min.: | 17 V |
| | $U_L$ max.: | 15 V |
| | $U_L$ min.: | 0 V |

## 4.4.2 Ausgänge

**Differentielle Ausgänge 5 V**
- erfüllen den EIA-Standard RS485

| | |
|---|---|
| Nominalspannung: ..... | 5 VDC |
| Maximale Ausgabefrequenz: ..... | 5 MHz |
| Max. Anzahl der Ausgänge: | 8 (wenn sie nicht als diff. Eingänge belegt sind) |

**Digitale Ausgänge, 24 V**

| | |
|---|---|
| Ausgangstyp: ..... | High-Side (Last an Masse) |
| Anzahl der Ausgänge: ..... | 4 |
| Nominalspannung: ..... | 24 VDC |
| Bereich der Versorgungsspannung: ..... | 10 V bis 36 VDC (über 24V ext. Pin) |
| Maximaler Ausgangsstrom für die 4 Ausgänge: | 2 A typ. (zu begrenzen an der Spannungsversorgung) |
| Maximaler Ausgangsstrom: ..... | 500 mA |
| Kurzschlussstrom / Ausgang bei 24 V, $R_{last}$ < 0,1 R: ..... | 1,5 A max. (Ausgang schaltet ab.) |
| ON-Widerstand des Ausgangs ($R_{DS}$ ON-Widerstand): ..... | 0,4 R max. |
| Übertemperatur: ..... | 170°C (alle Ausgänge schalten ab.) |

**Übertemperaturschutz (24 V Ausgänge)**

| | |
|---|---|
| Aktivierung ab ca.: ..... | 150-170°C (Chiptemperatur) |
| Deaktivierung (automatisch) ab ca.: ..... | 125-140°C (Chiptemperatur) |
| Ausgänge (bei Übertemperatur): ..... | Ausgänge schalten ab |

**Unterspannungsschutz** (wirksam bei Vext<5 V)
Ausgänge (bei Unterspannung): ..... Alle Ausgänge schalten ab.

**Schaltcharakteristik der Ausgänge**
(Vext = 24 V, T=25°C, ohmsche Last: 500 mA)
Einschaltverzögerung: ..... 200 µs
Abschaltverzögerung: ..... 15 µs

**Digitale Ausgänge, 5 V (OPTION)**
Ausgangstyp: ..... TTL
Anzahl der Ausgänge: ..... 4
Nominalspannung: ..... 5 VDC
Ausgangsstrom:.....10 mA

**Schaltcharakteristik der Ausgänge** ( T=25°C, TTL Last)
Einschaltverzögerung: ..... 0,06 µs
Abschaltverzögerung: ..... 0,02 µs

**APCI-1710, 24 V**

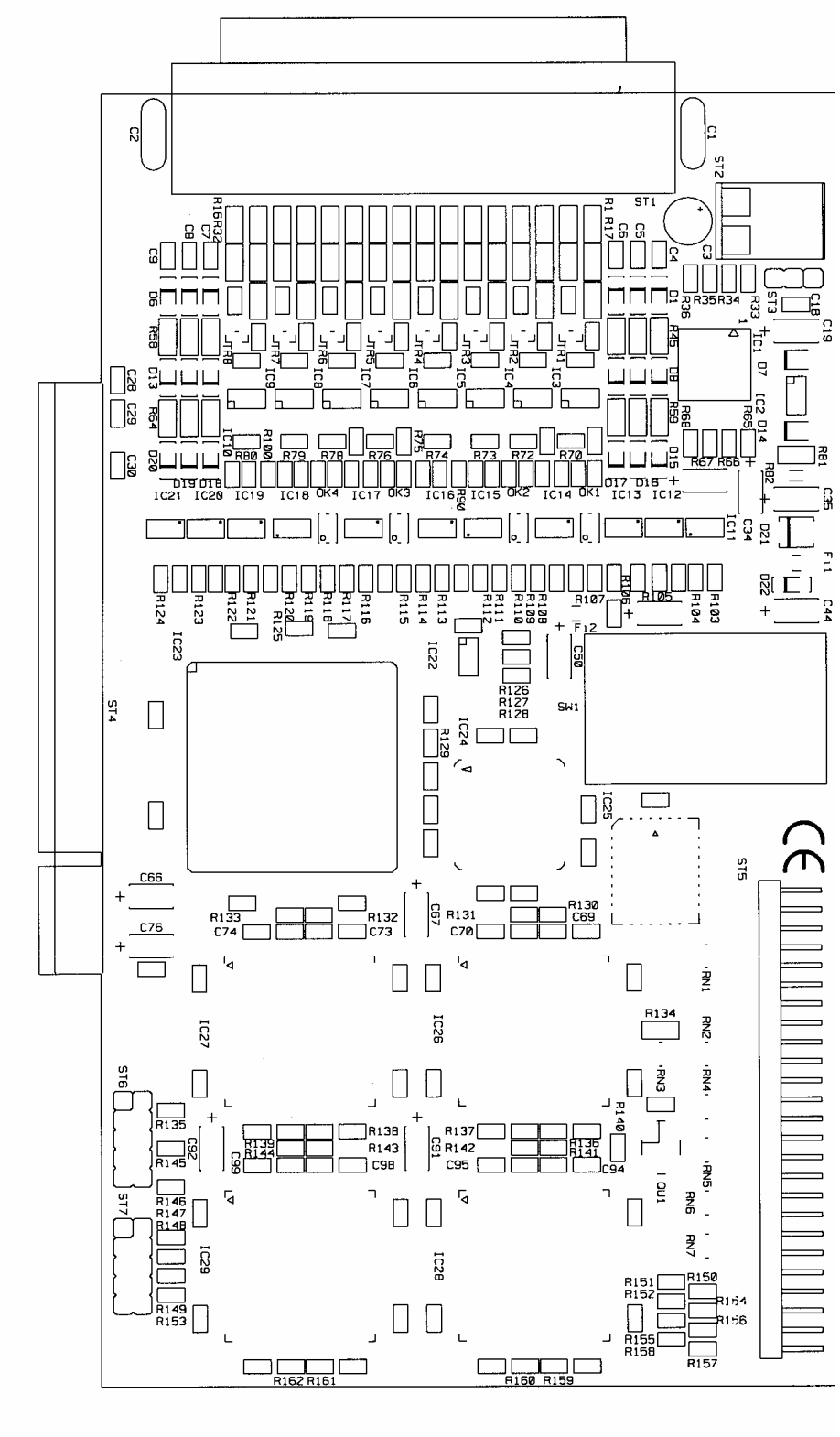
Die Ein-/Ausgänge A bis B können nur als Eingänge verwendet werden, d.h. dass die Funktion PWM und digitale E/A (nur eingeschränkt als Eingänge) nicht mit dieser Karte verwendet werden können.

## 4.4.3 Sicherheit

Galvanische Trennung
(DIN VDE 0411-100): ..... 1000 V (vom PC zur externen Peripherie)
Logik: ..... positiv
Kriechstrecke: ..... 3,2 mm auf der Leiterplatte

# 4.5 Bestückungsplan

**Abb. 4-1: Bestückungsplan**



# 5 VERFÜGBARE FUNKTIONEN DER APCI-1710

## 5.1 Verfügbare Signale

### 5.1.1 Anschließbare Signalleitungen

Die Leitungen werden je nach Karte wie folgt aufgeteilt:
**Eingangsleitungen:**
- 2 x TTL, RS422 (Signale C, D)
- 3 x 24 V, 5 V optional (Signale E, F, G)

**Ausgangsleitungen**
- 1 x 24 V, TTL optional (Signal H)

**Frei definierbare Leitungen (Ein- oder Ausgang)**
- 2 x TTL, RS422, (Signale A, B)

**24 V Version (APCI-1710-24 V)**
- 7 x 24 V Eingänge (Signale A bis G)
- 1 x 24 V Ausgang, TTL optional (Signal H)

### 5.1.2 Maximale Signalbeschaltungen der APCI-1710

**Tabelle 5-1: Maximale Eingangsleitungen auf der Karte**

| Eingänge | Ausgänge verfügbar |
|---|---|
| 28 | 4 (H) |
| **davon**<br>- 16 differentielle Eingänge und<br>- 12 x 24 V Eingänge | |

**Tabelle 5-2: Maximale Eingangsleitungen auf einem Modul**

| Eingänge | Ausgänge verfügbar |
|---|---|
| 7 | 1 |
| davon<br>- 4 differentielle und<br>- 3 x 24 V Eingänge | |

**Tabelle 5-3: Maximale Ausgangsleitungen auf der Karte**

| Ausgänge | Eingänge verfügbar |
|---|---|
| 12 | 20 |
| **davon**<br>- 8 differentielle Ausgänge und<br>- 4 x 24 V Ausgänge | **davon**<br>8 differentielle<br>12 x 24 V Eingänge (E,F,G) |

**Tabelle 5-4: Maximale Ausgangsleitungen auf ein Modul**

| Ausgänge | Eingänge verfügbar |
|---|---|
| 3 Ausgangsleitungen | 1 |
| davon<br>- 2 Differentielle und<br>- 1 x 24 V Ausgang | |

# 5.2 Verfügbare Funktionen

## 5.2.1 Programmierbare Funktionen der Zählerkarte

**Tabelle 5-5: Mögliche Anwendungen der Zählerkarte**

| Funktion | Belegte Eingangskanäle | Belegte Ausgangskanäle |
|---|---|---|
| Inkrementalzähler | 7 Eingänge (A bis G) | 1 Ausgang (H) |
| SSI | 6 Eingänge (B bis G) | 2 Ausgänge (A, H) |
| Chronos | 5 Eingänge (C bis G) | 3 Ausgänge (A, B, H) |
| Zähler/Timer | 5 Eingänge (C bis G) | 3 Ausgänge (A, B, H) |
| Digitale E/A<br>oder | 7 Eingänge (A bis G)<br>5 Eingänge (C bis G) | 1 Ausgang (H)<br>3 Ausgänge (A, B, H) |
| PWM | 2 Eingänge (C, D) | 2 Ausgänge (A, B) |
| TOR | 2 Eingänge (C, D) | 2 Ausgänge (A, B) |
| TTL | 24 dig. Kanäle als Ein- oder Ausgänge konfigurierbar (PA0..7, PB0..7, PC0..7)<br>2 Ein- und Ausgänge (I,J)<br>**nur auf 1 Funktionsmodul möglich** | |
| Impulszähler | 4 Eingänge | 1 Ausgang (H) |
| ETM | 4 Eingänge (A, B, C, D) | - |

## 5.2.2 Anschlussmöglichkeiten abhängig der gewählten Funktion

**Tabelle 5-6: Maximale Anwendungsfunktionen auf der Karte**

| Anwendung | Auf der Karte | Pro Funktions-modul | Programmierte Funktion |
|---|---|---|---|
| Inkrementalgeber | 4 (32-Bit Zähltiefe)<br>8 (16-Bit Zähltiefe) | 1 (32-Bit)<br>2 (16-Bit) | Inkrementalzähler |
| Absolutdrehgeber | 12 | 3 | SSI |
| Impulszähler | 16 | 4 | Impulszähler |
| Frequenz, Duty cycle Bestimmung | 4 | 1 | Chronos |
| Frequenzausgabe, Triggersteuerung | 12 | 3 | Zähler/Timer |
| PWM | 8 | 2 | PWM |

| | | | |
|---|---|---|---|
| TTL Ein- und Ausgänge | 1 | | TTL |
| Gate-Messung | 8 | 2 | TOR |
| Edge Time Measurement | 8 | 2 | ETM |

### 5.2.3 Gelieferte Handbücher

Je nach verwendeter Funktion können Sie die notwendigen Belegungs- und Programmierinformationen in den einzelnen Handbüchern finden.

**Tabelle 5-7: Mitgelieferte Funktionshandbücher**

| Funktion | PDF Datei (CD2 technical manuals) | | Funktionsbezeichnung in SET1710 | CFG Datei |
|---|---|---|---|---|
| | deutsch | englisch | | |
| Inkrementalzähler | Inkr_zähler_d.pdf | incr_counter_e.pdf | Incremental counter | inc_cpt.cfg |
| SSI | SSI_d.pdf | SSI_e.pdf | SSI | ssi.cfg |
| Chronos | chronos_d.pdf | chronos_e.pdf | Chronos | chronos.cfg |
| Zähler/timer | Zähler_timer_d.pdf | counter_timer_e.pdf | counter/timer | 82x54.cfg |
| TOR | TOR_d.pdf | TOR_e.pdf | TOR | tor.cfg |
| PWM | PWM_d.pdf | PWM_e.pdf | Pulse width modulation | PWM.cfg |
| TTL | TTL_IO_d.pdf | TTL_IO_e.pdf | TTL I/O Interface | Ttl_io.cfg |
| Digitale E/A | dig_IO_d.pdf | dig_IO_e.pdf | Digital I/O | dig_IO.cfg |
| Impulszähler | Impulszähler_d.pdf | pulse_counter_e.pdf | Pulse counter | imp_cpt.cfg |
| ETM | ETM_d.pdf | ETM_e.dpf | Edge time measurement | etm.cfg |

# 6 EINBAU DER KARTE

**WICHTIG!**
Berücksichtigen Sie unbedingt die Sicherheitshinweise.

## 6.1 PC öffnen

- **PC und alle am PC angeschlossenen Einheiten ausschalten.**
- **Netzstecker des PCs aus der Steckdose ziehen.**
- **PC öffnen wie im Handbuch des PC Herstellers beschrieben.**

## 6.2 Auswahl eines freien Steckplatzes

Stecken Sie die Karte in einen freien PCI-5V (32-Bit) Steckplatz ein.

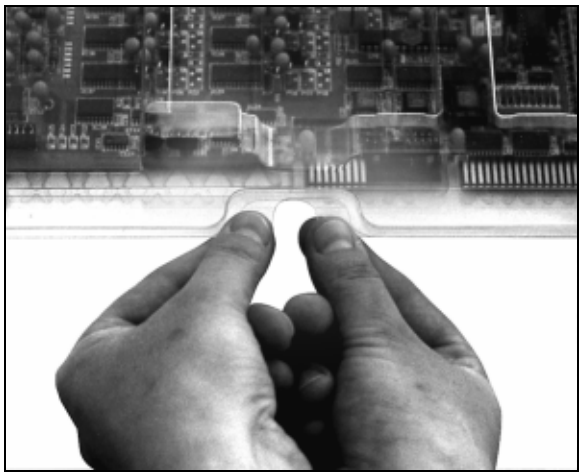
**Abb. 6-1: PCI-5V (32-Bit) Steckplatz**


**Das Blech des gewählten Steckplatzes** ausschrauben. Bitte beachten Sie hierzu die Bedienungsanleitung des PC Herstellers. Bewahren Sie das Blech auf. Sie werden es nach dem eventuellen Ausbau der Karte wieder benötigen.

**Bitte sorgen Sie für einen Potentialausgleich.**

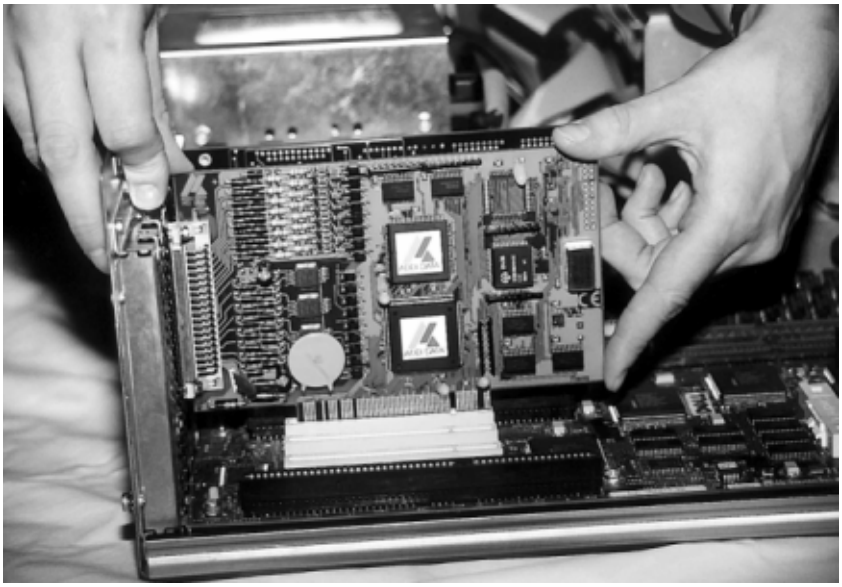
**Entnehmen Sie die Karte aus ihrer Klarsichtpackung.**

**Abb. 6-2: Klarsichtpackung öffnen**



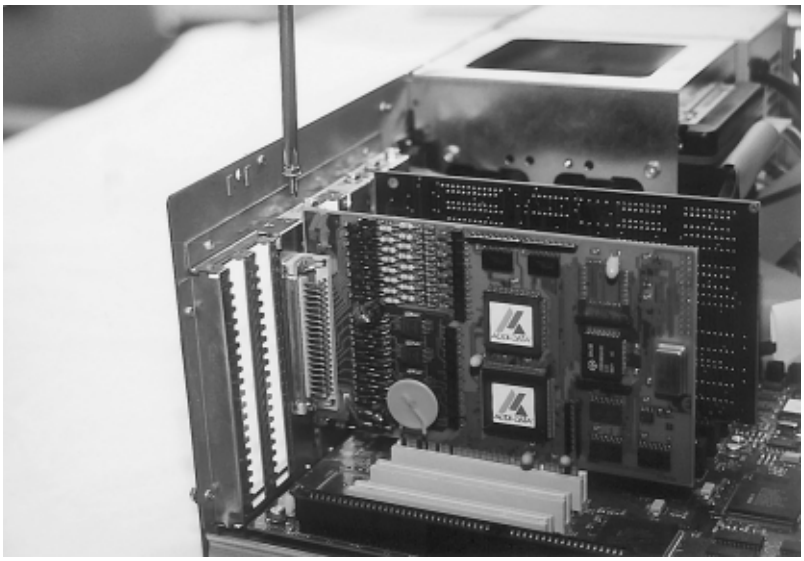
- **Karte senkrecht von oben in den gewählten Steckplatz einführen.**

**Abb. 6-3: Einbau der Karte**



- **Karte an der Gehäuserückwand mit der Schraube befestigen, mit der das Blech befestigt war.**

**Abb. 6-4: Die Karte an der Gehäuserückwand befestigen**



- **Alle gelösten Schrauben festschrauben.**

## 6.3 PC schließen

- **PC schließen wie im Handbuch des PC Herstellers beschrieben.**

# 7 SOFTWARE

## 7.1 Gelieferte Software

Die Karte wird mit einer Treiber-CD-ROM (CD 1) geliefert.
Die CD enthält:
- ADDIREG für Windows NT 4.0 und Windows 2000/95/98,
- Standardsoftware für die ADDI-DATA Karten:
  Das SET1710 Programm zur Konfiguration der Funktionsmodule
- Alle Funktionen, die für die APCI-1710 implementiert sind.

Im folgenden Kapitel werden die Software und ihre Verwendung beschrieben.

**WICHTIG!**
Die wichtigsten Informationen für das **Installieren und Deinstallieren der verschiedenen Treiber** finden Sie im mitgelieferten Handbuch **"Installationshinweise für den PCI Bus".**

Sie finden einen Link zu der entsprechenden PDF Datei im Navigationsfenster (Lesezeichen) von Acrobat Reader.

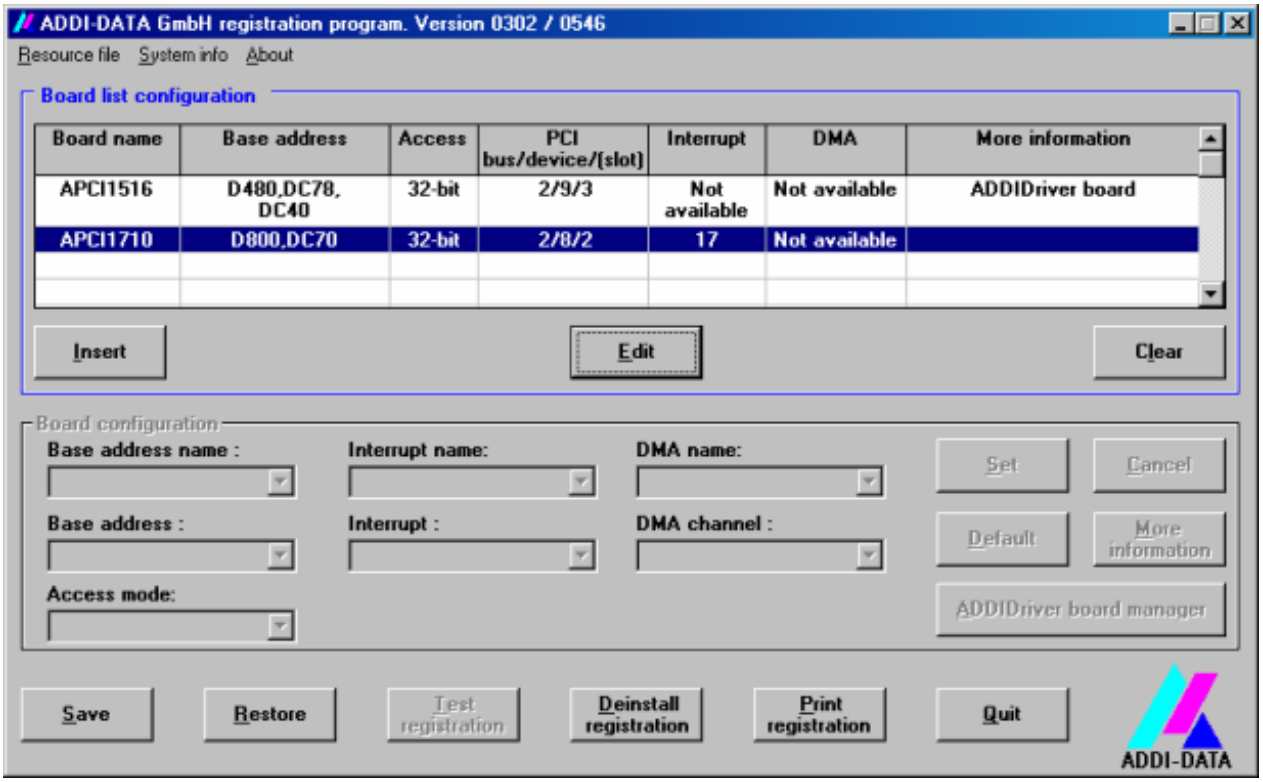
## 7.2 Konfiguration der APCI-1710 mit ADDIREG

Das ADDIREG-Registrierungsprogramm ist ein 32-Bit Programm.
Mit diesem Programm kann der Benutzer alle Hardwareinformationen registrieren, welche für den Betrieb der ADDI-DATA PC-Karten erforderlich sind.

**WICHTIG!**
**Bauen Sie zunächst die ADDI-DATA Karte ein,** die Sie registrieren wollen, bevor Sie das ADDIREG Programm aufrufen.

Wenn die Karte nicht im PC eingebaut ist, kann die Registrierung nicht geprüft werden. Beim Programmaufruf zeigt der Bildschirm folgendes Fenster an.

**Abb. 7-1: Registrierungsprogramm ADDIREG**




## Tabelle:

Die mittlere Tabelle listet die registrierten Karten und deren Parameter.

**Board name:**
Die Namen der verschiedenen registrierten Karten wird gezeigt. (z.B. APCI-7800). Wenn Sie das Programm das erste Mal benutzen, wird keine Karte unter diesem Eintrag aufgelistet.

**Base address:**
Ausgewählte Basisadresse der Karte.

**Access:**
Auswahl des Zugriff-Modes für die ADDI-DATA digitalen Karten. Zugriff in 8-Bit oder 16-Bits.

**PCI slot:**
Benutzter PCI Steckplatz. Falls die Karte keine APCI-Karte ist, erscheint die Meldung: "NO".

**Interrupt:**
Benutzter Interrupt der Karte. Falls die Karte keinen Interrupt benutzt, erscheint die folgende Meldung: "Not available".

**DMA:**
Zeigt den ausgewählten DMA-Kanal oder "Not available" an, wenn die Karte keinen benutzt.

**More information:**
Weitere Information gibt Ihnen das Dialogfenster, z.B. die Zeichenkette für den Identifier (z.B. PCI1500-50) oder die eingebauten COM Schnittstellen. Falls die Karte mit dem ADDIDRIVER programmiert ist, wird dies angezeigt.

## Textfenster:

Unter der Tabelle befinden sich 6 Text-Eintragfenster, mit deren Sie die Parameter der Karten ändern können.

**Base address name:**
Wenn die Karte mit mehreren Basisadressen betrieben wird (Eine für die erste Schnittstelle, eine für die zweite, usw.), können Sie entscheiden, welche Basisadresse geändert wird.

**Base address:**
In diesem Fenster können Sie die Basisadressen Ihrer PC-Karte auswählen. Die freien Basisadressen werden alle aufgelistet. Eine bereits benutzte Basisadresse erscheint nicht unter diesem Eintrag.

**Interrupt name:**
Wenn die Karte verschiedene Interruptleitungen (Sammel- oder Einzelinterruptleitungen) unterstützen soll, können Sie diese hier auswählen.

**Interrupt:**
Auswahl der Interruptnummer, die die Karte benutzen soll.

**DMA name:**
Wenn die Karte 2 DMA Kanäle unterstützt, können Sie auswählen, welchen DMA-Kanal Sie ändern.

**DMA channel:**
Auswahl des benutzten DMA-Kanal.

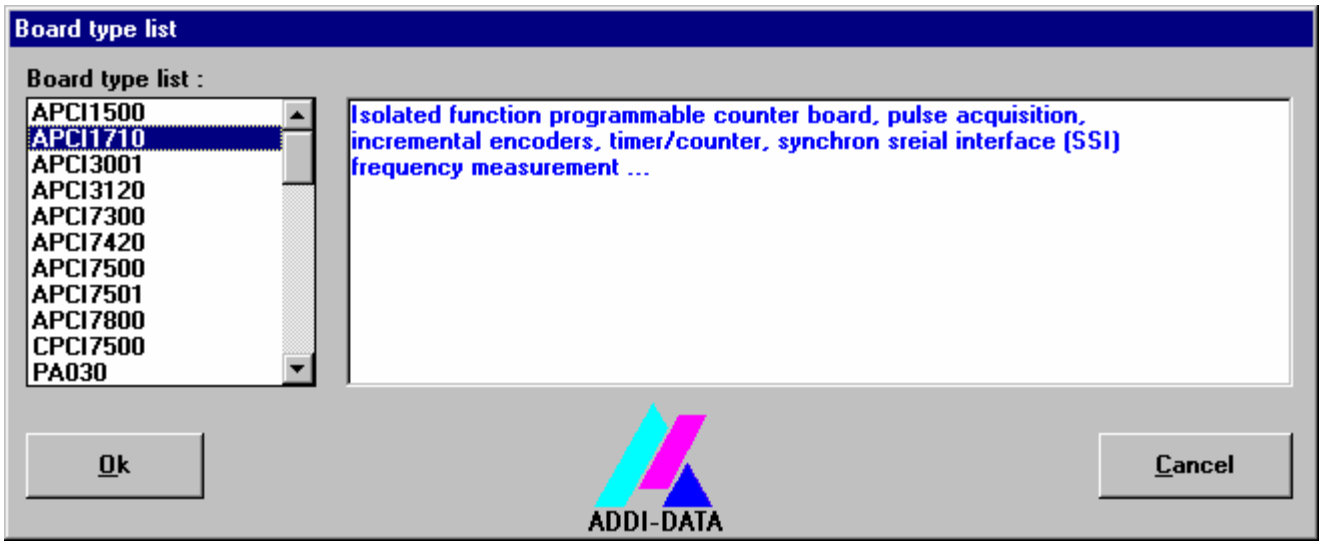
## Schaltflächen:

**Edit:**
Auswahl der markierten Karte mit den verschiedenen gesetzten Parametern der Text-Eintragsfenster.

Auf Edit klicken, um die Einträge zu bestätigen oder Doppelklick auf die ausgewählte Karte.

**Insert:**
Wenn Sie eine neue Karte einfügen wollen, klicken Sie auf „Insert“. Das folgende Fenster erscheint am Bildschirm:

**Abb. 7-2: Eine neue Karte konfigurieren**




Auf der linken Seite werden alle Karten aufgelistet, die Sie registrieren können. Die ausgewählte Karte anklicken. (Die entsprechende Zeile wird markiert.)

Auf der rechten Seite dieses Fensters stehen einige technische Informationen über die Karte zur Verfügung.

Bestätigen mit „OK“; Sie kommen zu dem ersten Bildschirm zurück.

**Clear:**
Sie können die Registrierung der Karte löschen. Markieren Sie die Karte, die Sie löschen wollen und klicken Sie auf „Clear“.

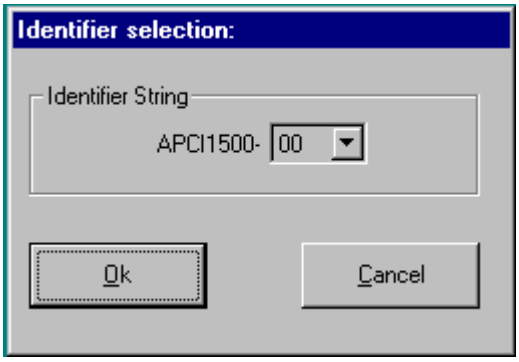
**Set:**
Setzt die parametrierte Kartenkonfiguration. Die Konfiguration soll gesetzt werden, bevor Sie sie speichern.

**Cancel:**
Setzt die geänderten Parameter auf die momentan gespeicherte Konfiguration zurück.

**Default:**
Setzt den Standardparameter der Karte.

**More information:**
Sie können damit kartenspezifische Parameter ändern, z.B. die Identifier Zeichenkette, die COM-Nummer, den Betriebsmode einer Kommunikationskarte, usw. ...

**Abb. 7-3: PCI-Karten**




Mit dieser Option können Sie die Zeichenkette für den Identifier auswählen, indem Sie die Nummer eintragen und mit „OK“ bestätigen.
Mit einem Klick auf „Cancel“ wählen Sie die alte Zeichenkette aus.

**ADDIDriver Board Manager:**
Unter Edit/ADDIDriver Board Manager können Sie die aktuellen Einstellungen jeder über den ADDEVICE Manager verwalteten Karten ansehen bzw. modifizieren.
Der ADDevice Manager wird geöffnet. Das Fenster listet alle verfügbaren Ressourcen der virtuellen Karte auf. Da die APCI-1710 nicht mit ADDIPACK programmiert wird, ist die Schaltfläche nicht aktiv.

**Save:**
Speichert die Parameter und registriert die Karte.

**Restore:**
Wiederaktivierung der zuletzt gespeicherten Parameter und Registrierung.

**Test registration:**
Überprüft, ob es einen Konflikt zwischen der Karte und den anderen Geräten gibt.
Eine Meldung zeigt den Parameter an, der den Konflikt generiert hat.
Wenn es keinen Konflikt gibt, erscheint „OK“.

**Deinstall registration:**
Deinstalliert alle Registrierungen aller Karten aus der Tabelle.

**Print registration:**
Druckt die Registrierungsparameter auf Ihren Standarddrucker aus.

**Quit:**
Verlässt das ADDIREG Programm.

### 7.2.1 Eine neue Karte registrieren

**WICHTIG!**
Um eine neue Karte zu registrieren, sind **Administrator-Rechte** erforderlich. Nur ein Administrator darf eine neue Karte registrieren oder eine bereits vorhandene Registrierung ändern.

- **Rufen Sie das ADDIREG-Programm auf.**

Die Abbildung Abb. 7-1 erscheint auf Ihrem Bildschirm. Klicken Sie auf "Insert". Wählen Sie die gewünschte Karte aus.

- **Klicken Sie auf "OK".**

Die Default-Adresse, Interrupt und die anderen Parameter werden automatisch gesetzt. Die Parameter werden in den unteren Flächen aufgelistet. Wenn die Parameter nicht automatisch durch das BIOS gesetzt werden, können Sie die Parameter ändern. Klicken Sie dafür auf die gewünschte (-n) Rollfunktion (-en) und wählen Sie einen neuen Wert aus.
Bestätigen Sie mit einem Klick.

**Wenn die gewünschte Konfiguration gesetzt ist, klicken Sie auf "Set".**

**Die Konfiguration mit "Save" speichern.**

Sie können mit einem Test prüfen, ob die Registrierung in Ordnung ist:
Wenn der Test positiv ausfällt, können Sie das ADDIREG-Programm verlassen. Die Karte wird mit den gesetzten Parametern initialisiert und kann betrieben werden.
Die Meldung "Starten Sie ihren PC neu" erscheint. Ansonsten werden die Parameter in Dateien gespeichert, die ein Neustarten des PCs nicht erfordern.

### 7.2.2 Die Registrierung einer vorhandenen Karte ändern

**WICHTIG!**
Um eine neue Karte zu registrieren, sind **Administrator-Rechte** erforderlich. Nur ein Administrator darf eine neue Karte registrieren oder eine bereits vorhandene Registrierung ändern.

**Rufen Sie das ADDIREG Programm. Markieren Sie die Karte, deren Parameter Sie ändern möchten.**

Die Parameter der Karte (Basisadresse, DMA Kanal, ..) werden in den unteren Flächen aufgelistet.

- **Klicken Sie auf die Parameter, die sie neu setzen wollen und machen Sie die Rollfunktionen auf.**
- **Wählen Sie einen neuen Wert aus. Bestätigen Sie mit einem Klick. Bitte wiederholen für jeden zu ändernden Parameter.**
- **Wenn die gewünschte Konfiguration gesetzt wird, klicken Sie auf "Set".**
- **Die Konfiguration mit "Save" speichern.**

Sie können mit einem Test prüfen, ob der Register stimmt. Wenn der Test positiv ausfällt, können Sie das ADDIREG-Programm verlassen.
Die Karte wird mit den gesetzten Parametern initialisiert und kann vertrieben werden.

## 7.3 Laden einer Funktion in ein Funktionsmodul

Das SET1710 Konfigurationsprogramm ist ein 16 bzw. 32 Bit Programm für Windows 3.11 und Windows XP/NT/2000/98. Es wird automatisch mit der Installation des Treibers in 32-Bit (Windows XP/2000/NT/98) installiert.

Im Auslieferungszustand sind alle Funktionsmodule mit der Funktion **"Inkrementalzähler"** voreingestellt.

Die Funktionen werden über die mitgelieferte **"SET1710.exe"** für jedes Funktionsmodul **einzeln** festgelegt und **einmal** programmiert. Nach dem Neustarten des Rechners ist die **APCI-1710** Karte betriebsbereit.

Der Benutzer kann auch die Konfiguration der Module durch Software setzen. Siehe die Software-Funktion i_APCI1710_ConfigureAllModule im Kapitel 9.

### 7.3.1 Modulkonfiguration mit SET1710

**WICHTIG!**
Stellen Sie unter Windows XP/2000/NT/98 zunächst die Karte mit dem Registrierungsprogramm ADDIREG ein, bevor Sie die Funktionsmodule mit dem Programm SET1710 konfigurieren.

**Abb. 7-4: SET1710: Modul-Konfigurationsprogramm**

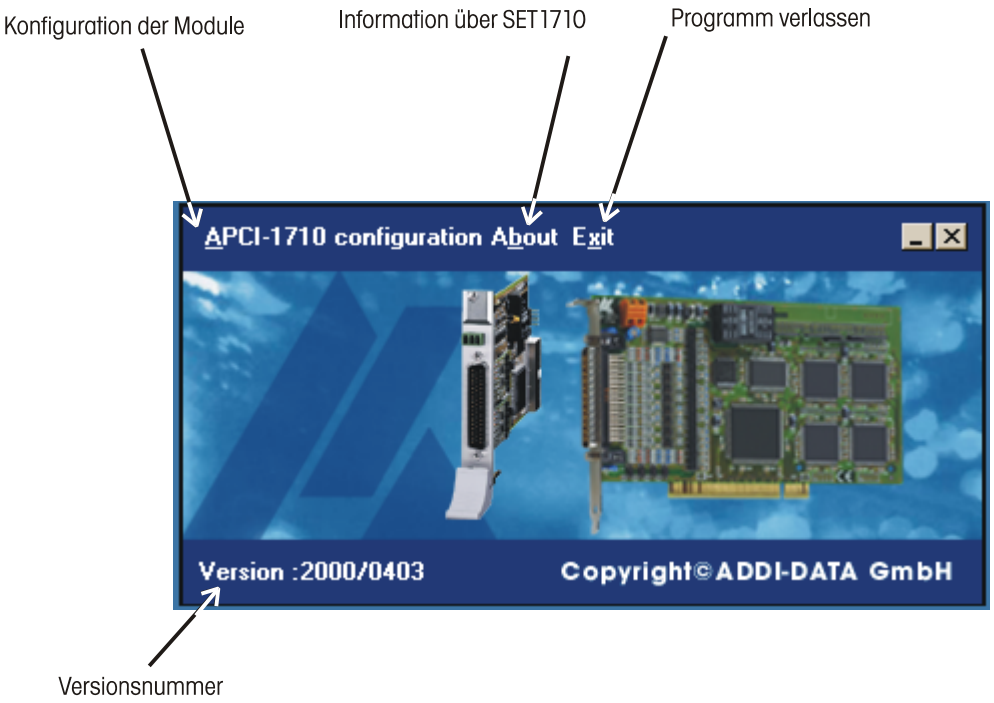


**Abb. 7-5: Auswahl einer APCI-/CPCI-1710**

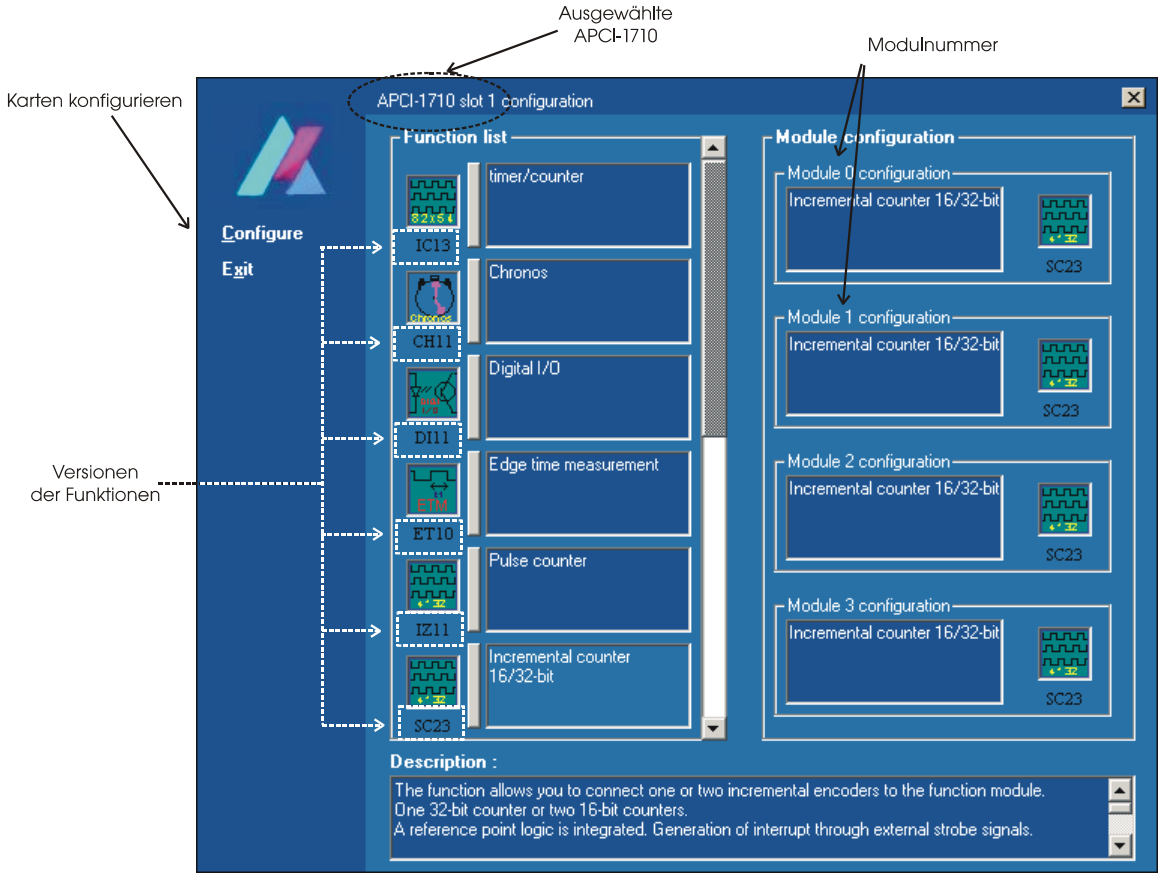

Unter "About" finden Sie die allgemeinen Informationen über die Karte und unsere Service-Abteilung.

**Abb. 7-6: Allgemeine Informationen**


Mit "Exit" verlassen Sie das SET1710 Programm.

Nachdem Sie die **APCI-1710** ausgewählt haben, erscheint das folgende Fenster:

**Abb. 7-7: Funktionsliste und Modulkonfiguration**




**Function list: (Liste der Funktionen)**
Alle verfügbaren Modulfunktionen werden aufgelistet. Jede Funktion wird in einer Konfigurationsdatei unter dem Verzeichnis CFG gesetzt.

**Module configuration: (Modulkonfiguration)**
Aktuelle Modulkonfiguration der ausgewählten Karte.

**Description: (Beschreibung)**
Beschreibung der in der Liste ausgewählten Funktion.
Die Version der Funktion, die Update-Beschreibung und alle anderen Informationen über die Modulfunktion werden in diesem Feld aufgelistet.

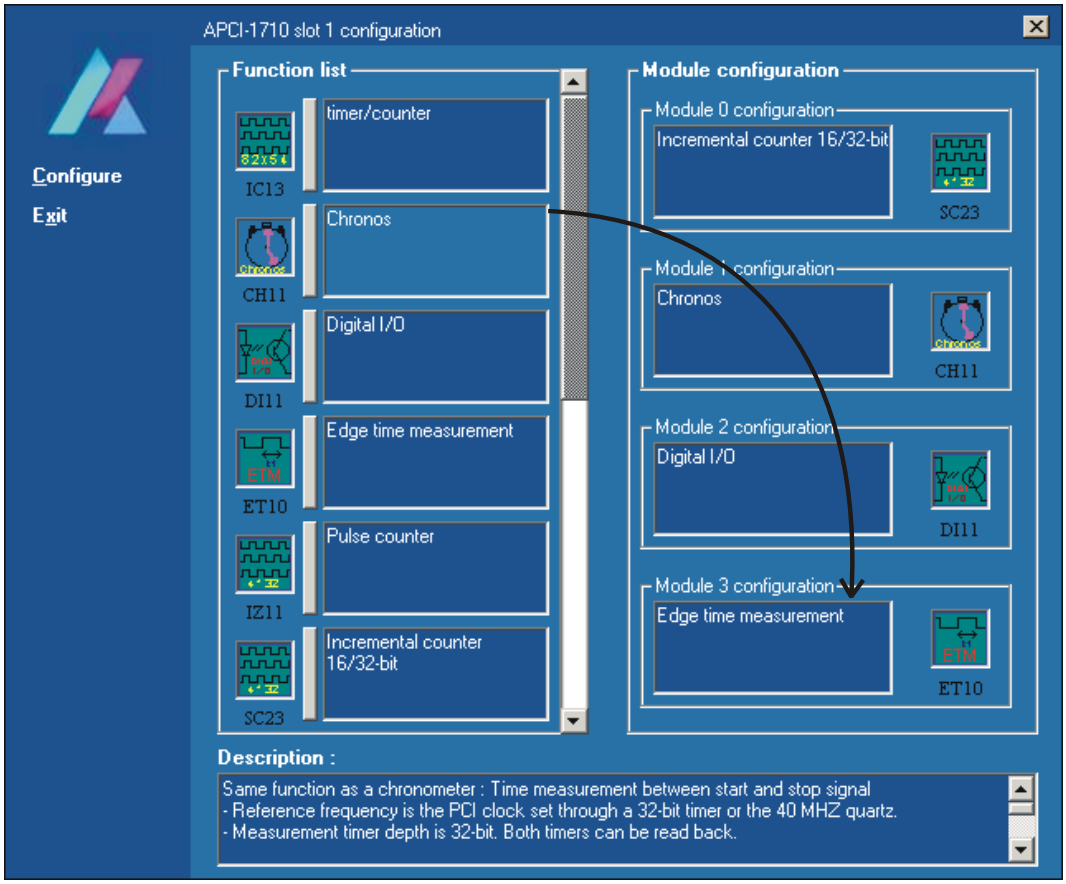
**Configure: (Konfigurieren)**
Programmiert die APCI-/CPCI-1710 Karte mit der aktuellen Modulkonfiguration.

**Exit:**
Schließt das Fenster.

## 7.3.2 Modulkonfiguration setzen

## 7.3.3 Modulkonfiguration per Mausklick

**Abb. 7-8: Modulkonfiguration per Mausklick setzen**

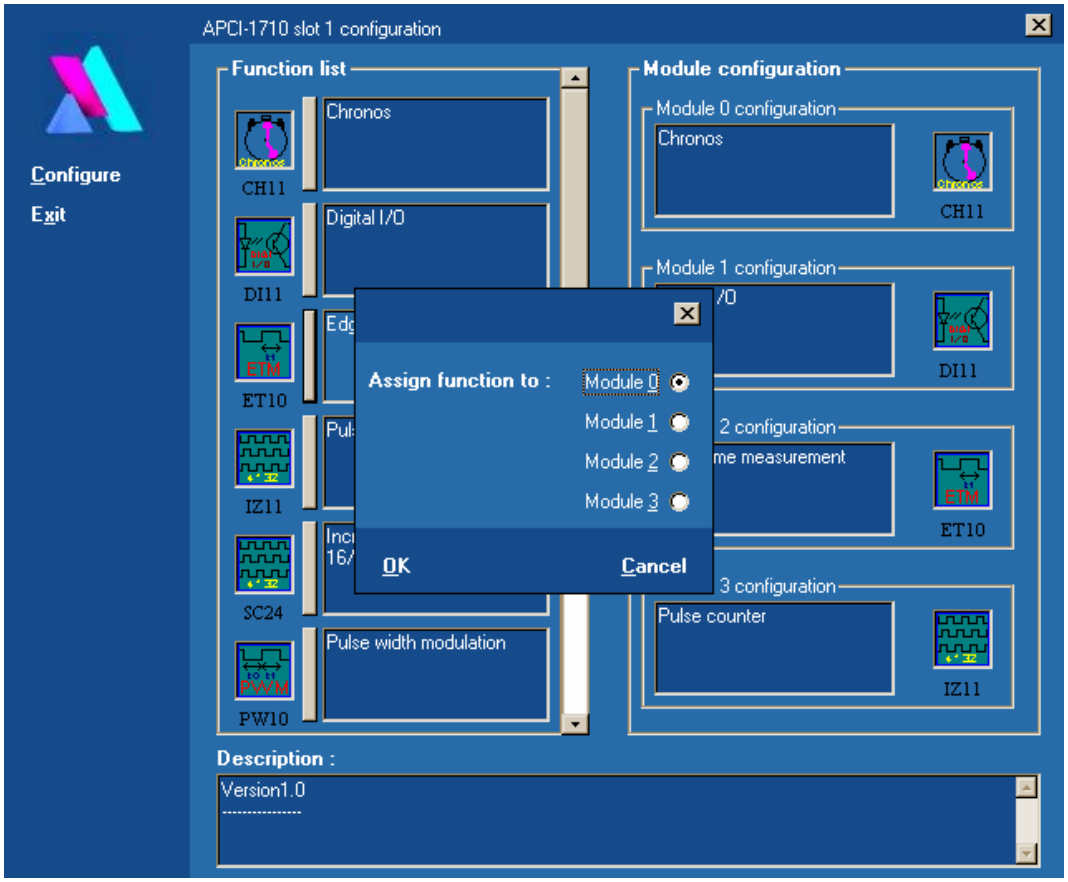



Mit der rechten Maustaste die Funktion in der Liste anklicken und in das gewünschte Modul verschieben.

### 7.3.4 Modulkonfiguration per Tastatur

Die Funktion in der Liste auswählen (Tabulator). Auf die "Return"-Taste drücken. Das folgende Dialogfenster erscheint:

**Abb. 7-9: Modulkonfiguration per Tastatur setzen**




Das gewünschte Modul auswählen und mit "OK" bestätigen.

## 7.4 ADDI-DATA im Internet

Sie können uns Ihre Fragen per E-Mail zusenden:

- per E-Mail: info@addi-data.de oder
  hotline@addi-data.de

- per Internet: http://www.addi-data.com

**Kostenloser Treiber-Download**

Die neueste Version des Gerätetreibers für Ihre ADDI-DATA Karte können Sie kostenlos vom Internet downloaden.

# 8 ANSCHLUSS AN DIE PERIPHERIE

**WICHTIG!**
Über das Anschlusskabel werden Störungen ausgestrahlt und eingekoppelt. Ein falsches Kabel könnte dann die Betriebs- und Funktionssicherheit Ihres Systems gefährden.

- **Verwenden Sie unser Standard-Anschlusskabel.**
- **Achten Sie bei der Verlegung des Anschlusskabels darauf, dass:**

- es in ausreichendem Abstand zu empfindlichen analogen Signalen verlegt wird,
- der Abstand zu potentiellen Störern, wie z.B. Frequenzumrichtern, Netzteilen so groß wie möglich ist.

Falls Sie die Ausgänge in Maximallast betreiben, sollten Sie das Anschlusskabel frei verlegen oder für eine gute Belüftung sorgen.

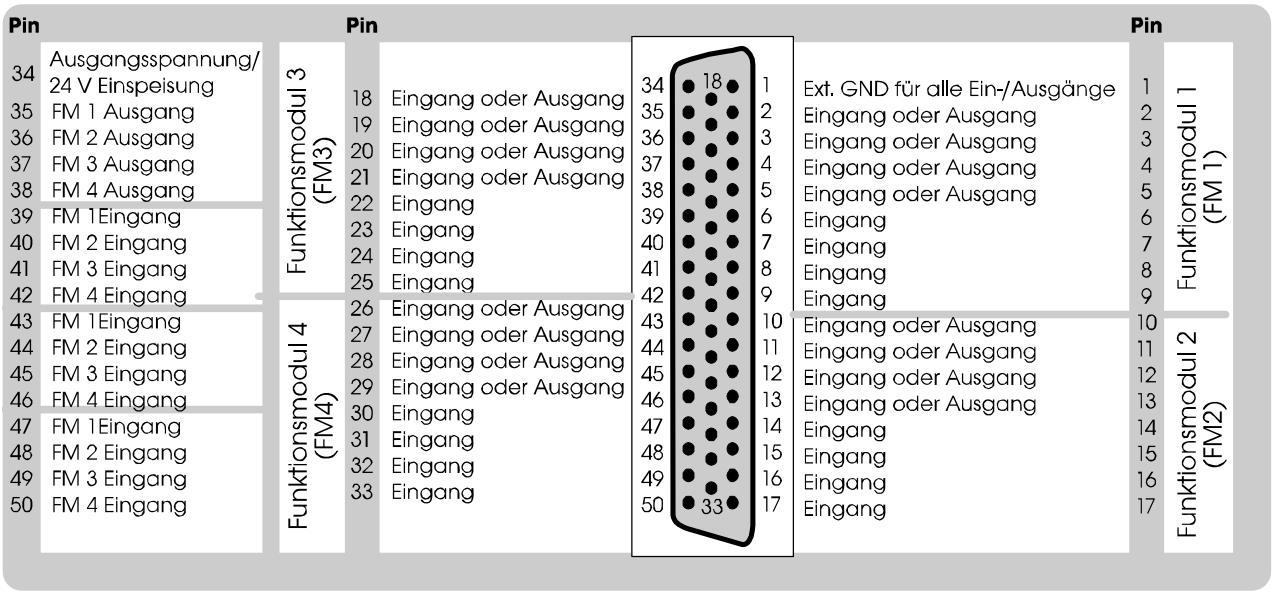
## 8.1 Steckerbelegung

**WICHTIG!**
Die Funktionsmodule weisen unterschiedliche Bezeichnungen in der Hardware- bzw. Software-Beschreibungen auf.

Für die Steckerbelegung (Hardware) werden die Module von 1 bis 4 nummeriert. Für das SET1710 Programm oder die Softwarefunktionen (Software) **BEGINNT** die Modulnummerierung mit 0.

### 8.1.1 50-pol. SUB-D Frontstecker ST1

**Abb. 8-1: 50-poliger SUB-D Stiftstecker (ST1)**



**WARNUNG!**
Der Pin 34 des Frontsteckers hat eine Doppelbelegungs-Funktion. Wenn Sie den Pin 34 nicht richtig anschließen, **kann Ihre Karte zerstört werden!**

In den folgenden Tabellen finden Sie die Pinbelegung der digitalen Ein- und Ausgänge und der jeweiligen Signale und Funktionen. Die Ein- und Ausgänge sollen nach der Programmierung der Funktion auf das Modul geschaltet werden (Richtungsumschaltung abhängig von der ausgewählten Funktion).

**Tabelle 8-1: Pinbelegung für das Funktionsmodul Nr. 1**

| Pin | Bezeichnung | Input/Output | APCI-1710 | APCI-1710-24 V |
|---|---|---|---|---|
| 2 | A1+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 3 | A1- | Input/Output [1] | Diff. / TTL | - [2] |
| 4 | B1+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 5 | B1- | Input/Output [1] | Diff. / TTL | - |
| 6 | C1+ | Input | Diff. / TTL | 24 V |
| 7 | C1- | Input | Diff. / TTL | - |
| 8 | D1+ | Input | Diff. / TTL | 24 V |
| 9 | D1- | Input | Diff. / TTL | - |
| 35 | H1 | Digital Output | 24 V | 24 V |
| 39 | E1 | Digital Input | 24 V | 24 V |
| 43 | F1 | Digital Input | 24 V | 24 V |
| 47 | G1 | Digital Input | 24 V | 24 V |

**Tabelle 8-2: Pinbelegung für das Funktionsmodul Nr. 2**

| Pin | Bezeichnung | Input/Output | APCI-1710 | APCI-1710-24 V |
|---|---|---|---|---|
| 10 | A2+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 11 | A2- | Input/Output [1] | Diff. / TTL | - |
| 12 | B2+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 13 | B2- | Input/Output [1] | Diff. / TTL | - |
| 14 | C2+ | Input | Diff. / TTL | 24 V |
| 15 | C2- | Input | Diff. / TTL | - |
| 16 | D2+ | Input | Diff. / TTL | 24 V |
| 17 | D2- | Input | Diff. / TTL | - |
| 36 | H2 | Digital Output | 24 V | 24 V |
| 40 | E2 | Digital Input | 24 V | 24 V |
| 44 | F2 | Digital Input | 24 V | 24 V |
| 48 | G2 | Digital Input | 24 V | 24 V |

[1] Für die 24 V Signale (APCI-1710-24 V) können nur Eingänge angeschlossen werden.
[2] Die Ax-, Bx-, Cx-, Dx- Pins haben bei der 24 V Karte keine Funktion. Siehe Abb. 9-5

### Tabelle 8-3: Pinbelegung für das Funktionsmodul Nr. 3

| Pin | Bezeichnung | Input/Output | APCI-1710 | APCI-1710-24 V |
|---|---|---|---|---|
| 18 | A3+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 19 | A3- | Input/Output [1] | Diff. / TTL | - |
| 20 | B3+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 21 | B3- | Input/Output [1] | Diff. / TTL | - |
| 22 | C3+ | Input | Diff. / TTL | 24 V |
| 23 | C3- | Input | Diff. / TTL | - |
| 24 | D3+ | Input | Diff. / TTL | 24 V |
| 25 | D3- | Input | Diff. / TTL | - |
| 37 | H3 | Digital Output | 24 V | 24 V |
| 41 | E3 | Digital Input | 24 V | 24 V |
| 45 | F3 | Digital Input | 24 V | 24 V |
| 49 | G3 | Digital Input | 24 V | 24 V |

### Tabelle 8-4: Pinbelegung für das Funktionsmodul Nr. 4

| Pin | Bezeichnung | Input/Output | ACPI-1710 | APCI-1710-24 V |
|---|---|---|---|---|
| 26 | A4+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 27 | A4- | Input/Output [1] | Diff. / TTL | - |
| 28 | B4+ | Input/Output [1] | Diff. / TTL | 24 V input |
| 29 | B4- | Input/Output [1] | Diff. / TTL | - |
| 30 | C4+ | Input | Diff. / TTL | 24 V |
| 31 | C4- | Input | Diff. / TTL | - |
| 32 | D4+ | Input | Diff. / TTL | 24 V |
| 33 | D4- | Input | Diff. / TTL | - |
| 38 | H4 | Digital Output | 24 V | 24 V |
| 42 | E4 | Digital Input | 24 V | 24 V |
| 46 | F4 | Digital Input | 24 V | 24 V |
| 50 | G4 | Digital Input | 24 V | 24 V |

### Tabelle 8-5: Besondere Pinbelegung

Pins mit fester Zuordnung.

| Pin | Bezeichnung | Input/Output | Funktion | Bemerkung |
|---|---|---|---|---|
| 1 | EXTGND | - | Bezugspotential | Für alle Ein- und Ausgänge |
| 34 | + UREF<br>Oder 24 V ext. | Output/Input | - Spannungsausgang für TTL Signale oder<br>- 24 V Versorgung für 24 V Ausgängen | Konfigurierbar durch Jumper ST3 |

## 8.1.2 24 V-Einspeisung für 24 V digitale Ausgänge (Kanal H)

**WARNUNG!**
Der Pin 34 des Frontsteckers hat eine Doppelbelegungs-Funktion. Wenn Sie den Pin 34 nicht richtig anschließen, **kann Ihre Karte zerstört werden!**

**Je nachdem, welcher Eingangstyp angeschlossen wird, soll der Benutzer auf den Anschluss des 24 V Ausgangskanals H achten.**

- **5 V differentielle Eingänge werden verwendet:**
  **Kanäle A±, B±, C±, D± Auslieferungszustand**

➢ Die 24 V Einspeisung der Ausgänge H1 bis H4 erfolgt über den Pin 34 des Frontsteckers ST1. Der Jumper ST3 ist auf Stellung A gesteckt.

- **TTL Eingänge werden verwendet:**

➢ Die 24 V Einspeisung der Ausgänge H1 bis H4 erfolgt **über die Klemme ST2.** Der Jumper ST3 ist auf Position B zu stellen. Der Pin 34 steht für die Hilfsspannung +UREF zur Anpassung der 5 V differentiellen Eingänge an TTL Signalen zur Verfügung.

**Abb. 8-2: Klemme ST2**

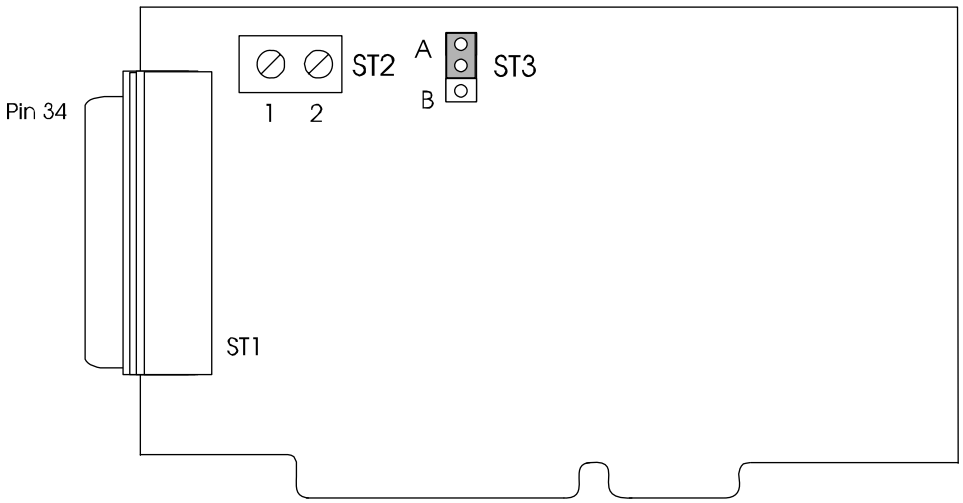


**Tabelle 8-6: Externe Einspeisung über Klemme ST2**

| Pin | Bezeichnung | Input/Output | Funktion | Bemerkung |
|---|---|---|---|---|
| ST2-1 | 24 V Einspeisung | 24 V Einspeisung | | |
| ST2-2 | EXTGND | EXTGND | Bezugspotential | mit Pin 1 des SUB-D Steckers verbunden |

## 8.1.3 50-pol. Flachbandstecker ST5

**Abb. 8-3: Lage des Flachbandstecker ST5 auf der Karte**

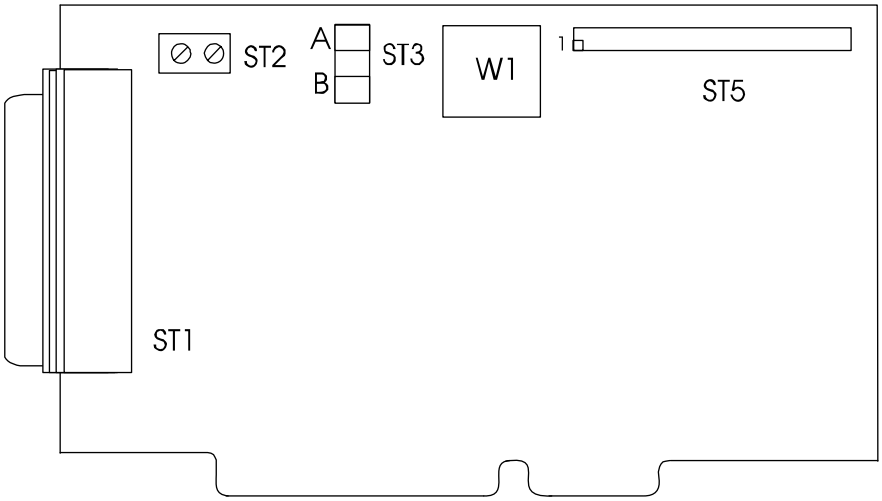
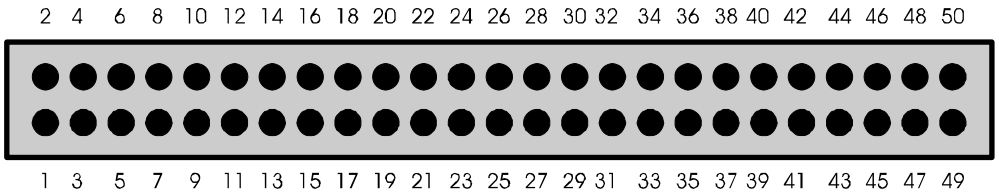



**Abb. 8-4: Flachbandsteckerbelegung**



## Tabelle 8-7: Beschreibung der Pinbelegung

| Pinnummer am Stecker | Name | Beschreibung | Pinnummer FB8000 |
|---|---|---|---|
| 1 | PC7[1] | TTL, Ein- oder Ausgang; nach Reiset: Eingang | 1 |
| 2 | GND | PC GND, nicht isoliert | 34 |
| 3 | PC6 | TTL, Ein- oder Ausgang; nach Reiset: Eingang | 18 |
| 4 | GND | PC GND, nicht isoliert | 2 |
| 5 | PC5 | TTL, Ein- oder Ausgang; nach Reiset: Eingang | 35 |
| 6 | GND | PC GND, nicht isoliert | 19 |
| 7 | PC4 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 3 |
| 8 | GND | PC GND; nicht isoliert | 36 |
| 9 | K1 | TTL, Ausgang; gleiches Signal wie H1 am Fronstecker, FM1 | 20 |
| 10 | GND | PC GND, nicht isoliert | 4 |
| 11 | K2 | TTL, Ausgang; gleiches Signal wie H2 am Fronstecker, FM2 | 37 |
| 12 | GND | PC GND; nicht isoliert | 21 |
| 13 | K3 | TTL, Ausgang; gleiches Signal wie H3 am Fronstecker, FM3 | 5 |
| 14 | GND | PC GND; nicht isoliert | 38 |
| 15 | K4 | TTL, Ausgang; gleiches Signal wie H4 am Fronstecker, FM4 | 22 |
| 16 | GND | PC GND; nicht isoliert | 6 |
| 17 | PA0 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 39 |
| 18 | PA1 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 23 |
| 19 | PA2 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 7 |
| 20 | PA3 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 40 |
| 21 | GND | PC GND, nicht isoliert | 24 |
| 22 | PA4 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 8 |
| 23 | PA5 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 41 |
| 24 | PA6 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 25 |
| 25 | PA7 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 9 |
| 26 | V. ext | | 42 |
| 27 | PB0 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 26 |
| 28 | PB1 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 10 |
| 29 | PB2 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 43 |
| 30 | PB3 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 27 |
| 31 | GND | | 11 |
| 32 | PB4 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 44 |
| 33 | PB5 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 28 |
| 34 | PB6 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 12 |
| 35 | PB7 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 45 |
| 36 | V ext | | 29 |
| 37 | PC0 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 13 |
| 38 | PC1 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 46 |
| 39 | PC2 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 30 |

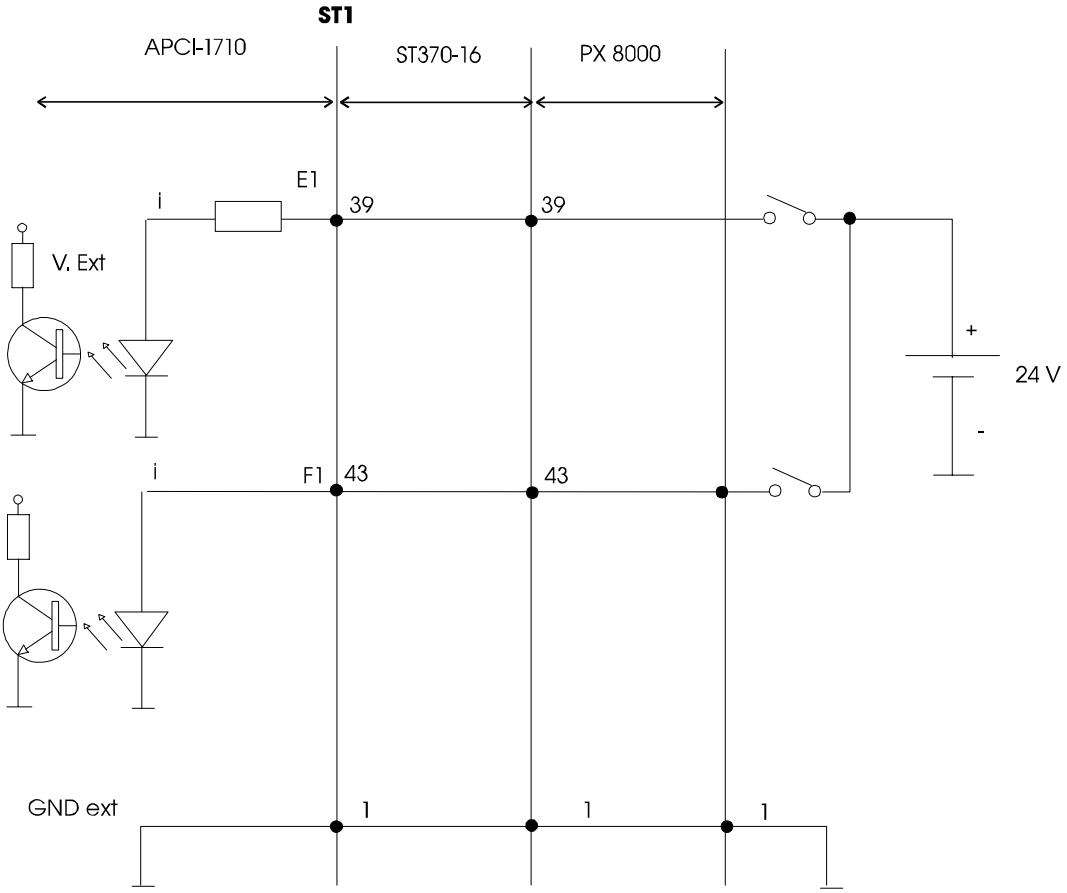
| | | | |
|---|---|---|---|
| 40 | PC3 | TTL, Ein- oder Ausgang; nach Reset: Eingang | 14 |
| 41 | GND | | 47 |
| 42 | I4[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM4 | 31 |
| 43 | J4[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM4 | 15 |
| 44 | I3 | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM3 | 48 |
| 45 | J3[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM3 | 32 |
| 46 | V ext | PC + 5 V Spannungsversorgung | 16 |
| 47 | I2[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM2 | 49 |
| 48 | J2[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM2 | 33 |
| 49 | I1[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM1 | 17 |
| 50 | J1[2] | TTL, Ein- oder Ausgang; nach Reset: Ausgang, FM4 | 50 |

1 PA, PB und PC : Pullup auf 5 V
2 Serienwiderstand 100 Ω, PD

PA, PB, PC und PD können über das Funktionsmodul “TTL I/O” genutzt werden.

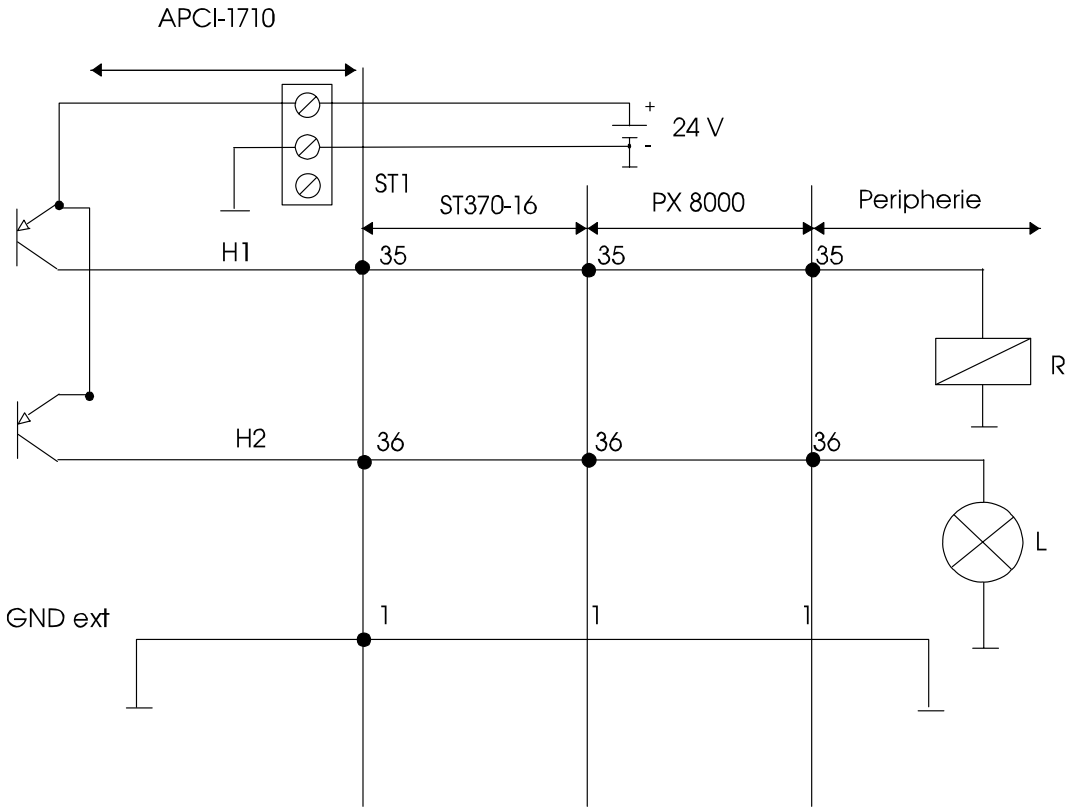
## 8.2 Anschluss massebezogener Ein- und Ausgänge

**Abb. 8-5: Anschluss eines massenbezogenen Eingangs**




ST1: Frontstecker der APCI-1710/APCI-1710-24V

**Abb. 8-6: Anschluss eines massenbezogenen Ausgangs**




ST1: Frontstecker der APCI-1710/APCI-1710-24V

## 8.3 Anschluss der differentiellen digitalen E/A

**Abb. 8-7: Anschluss eines differentiellen Eingangs**

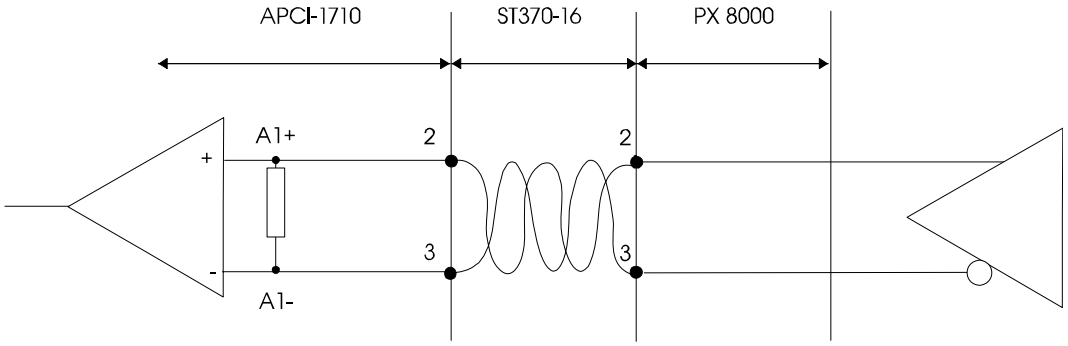



**Abb. 8-8: Beispiel: Anschluss 2 inkrementale Drehgeber an FM 1**

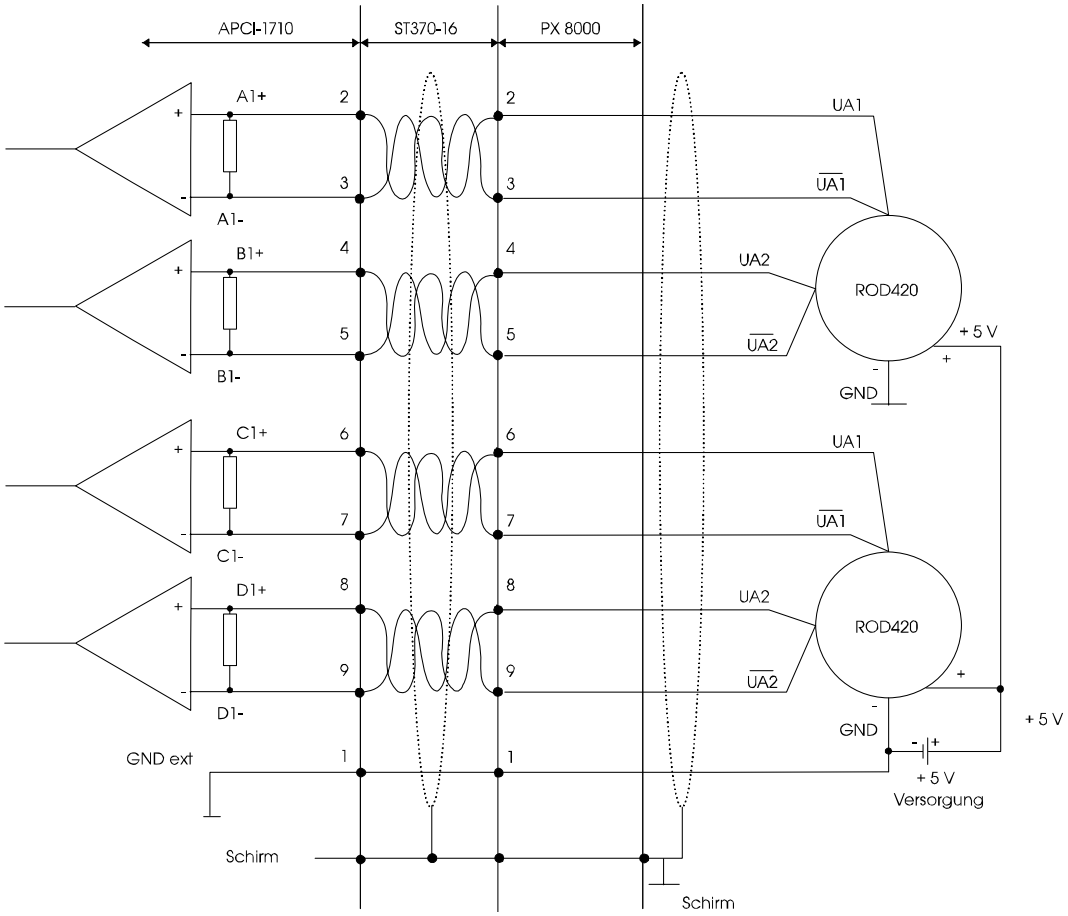
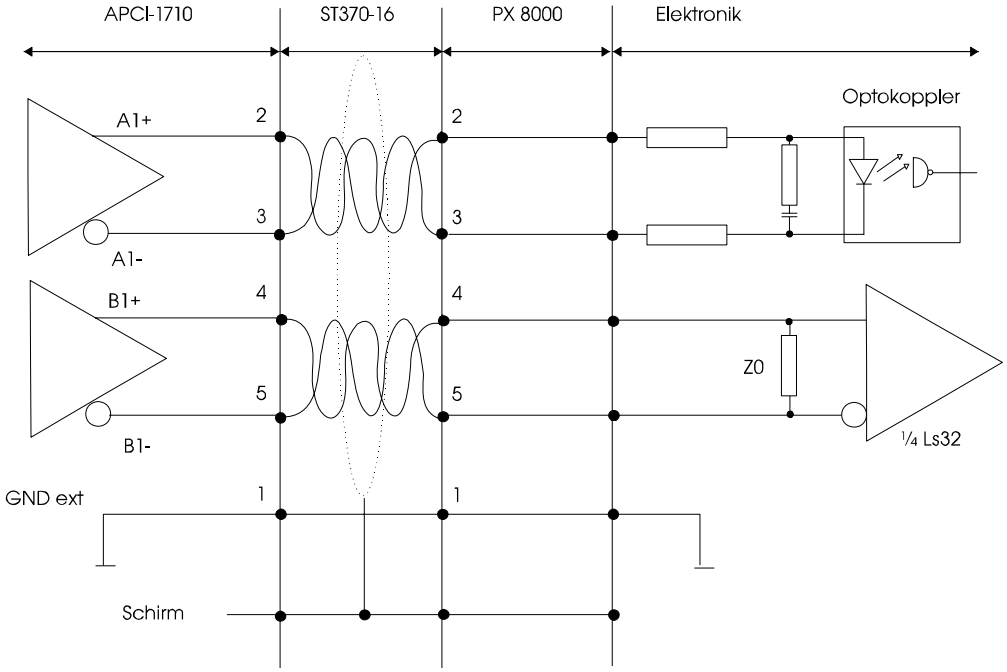
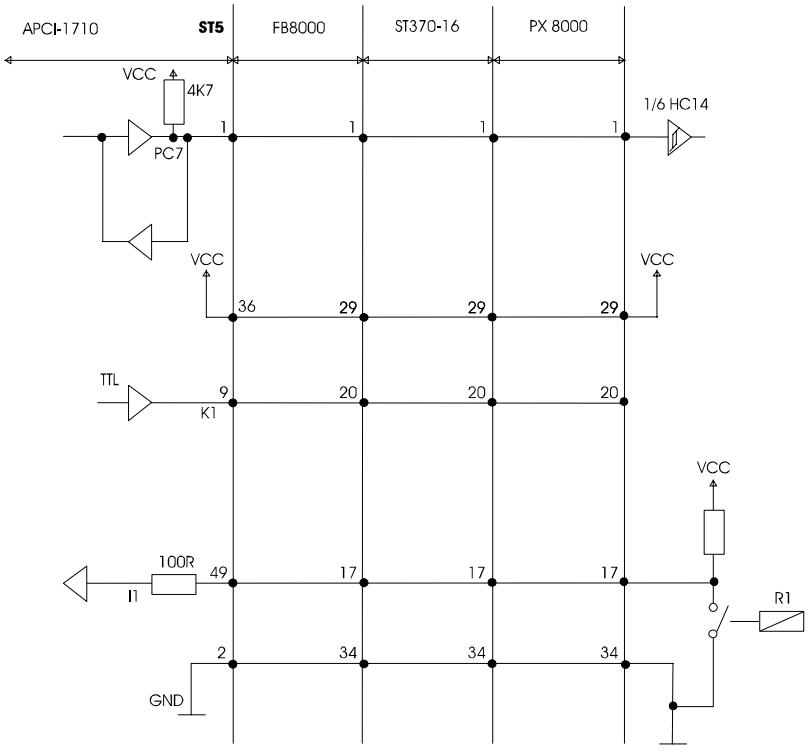


**Abb. 8-9: Anschluss eines differentiellen Ausgangs**




# 8.4 Anschluss der TTL Ein- und Ausgänge

**Abb. 8-10: Anschluss an TTL Ein- und Ausgänge**




ST5: Connector of the APCI-1710/APCI-1710-24V
for connecting ribbon cable FB8000

# 9 FUNKTIONEN DER KARTE

## 9.1 Beschreibung

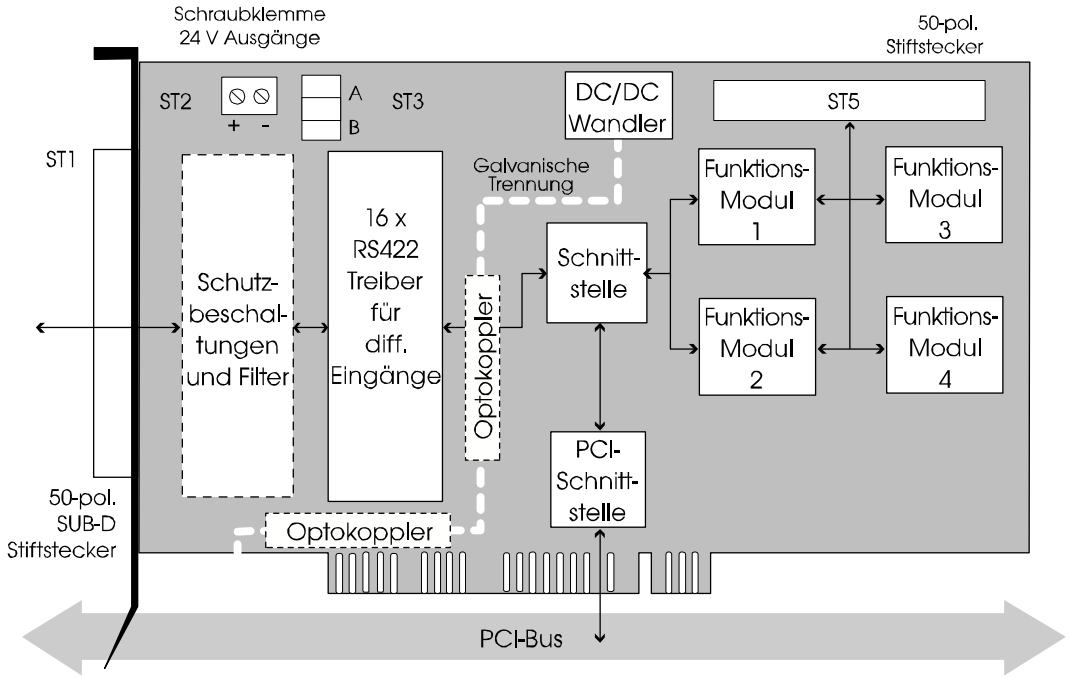
Die **APCI-1710** ist eine Erweiterungskarte für den PCI-Bus kompatibel zur PCI-Spezifikation 2.1. Diese Baugruppe dient grundsätzlich zur Verarbeitung von digitalen Signalen mit Schwerpunkt auf "Zählen und Zeitmessung".

Die digitalen Signale werden über den 50-pol. SUB-D Frontstecker ST1 an die "Funktionsmodule" der APCI-1710 Karte geführt. Sie sind galvanisch durch Optokoppler vom PC getrennt.

Ein zweiter 50-pol. Stecker ST5 ist bestückt, um ausschließlich TTL Signale an die Karte anzuschließen. Diese Signale sind nicht vom PC galvanisch getrennt.

Die APCI-1710 Karte besteht aus 4 "Funktionsmodulen", die mit 4 wieder programmierbaren CPLDs (Complex Programmable Logic Devices) bestückt sind. Jedem Funktionsmodul werden digitale Ein- und Ausgänge fest zugewiesen.

**Abb. 9-1: Blockdiagramm der APCI-1710**




Der Benutzer kann die Karte wieder programmieren, ohne sie auszubauen.
Es ergeben sich folgende Vorteile:
- einfache Erweiterung der Funktionalität der Karte
- Implementierung von Kundenwünschen
- einfaches Downloaden neu entwickelter Funktionen
- weniger Lagerplatz

Die Funktionsmodule erlauben es, digitale Ein- und Ausgabesignale zu verknüpfen und hardwaremäßig (d.h. in Echtzeit) zu verarbeiten, bevor Sie an den PC weitergeleitet werden.

Ein Funktionsmodul ist eine physikalische Einheit bestehend aus:
- digitalen Eingängen
- digitalen Ausgängen
- einem funktionsprogrammierbaren Baustein (Hardware)
- einem Interface zum PCI Bus.

Die Funktionsmodule werden auch untereinander über einen internen Bus verbunden.

## 9.2 Digitale Ein- und Ausgänge

### 9.2.1 Beschreibung

Jedem Funktionsmodul werden digitale Ein- und Ausgänge fest zugewiesen. Einige Eingänge können jedoch als Ausgänge für bestimmte Funktionen gesetzt werden, die auf dem Funktionsmodul implementiert sind.

Pro Modul stehen 8 Leitungen zur Verfügung: (Siehe Abb. 9-2: Blockdiagramm der dig. Ein- und Ausgänge)
(1 Funktionsmodul)

Die Leitungen werden je nach Karte wie folgt aufgeteilt:

**APCI-1710**
**Eingangsleitungen**
- 2 x TTL, RS422 (Signale C, D)
- 3 x 24 V, 5 V optional (Signale E, F, G)

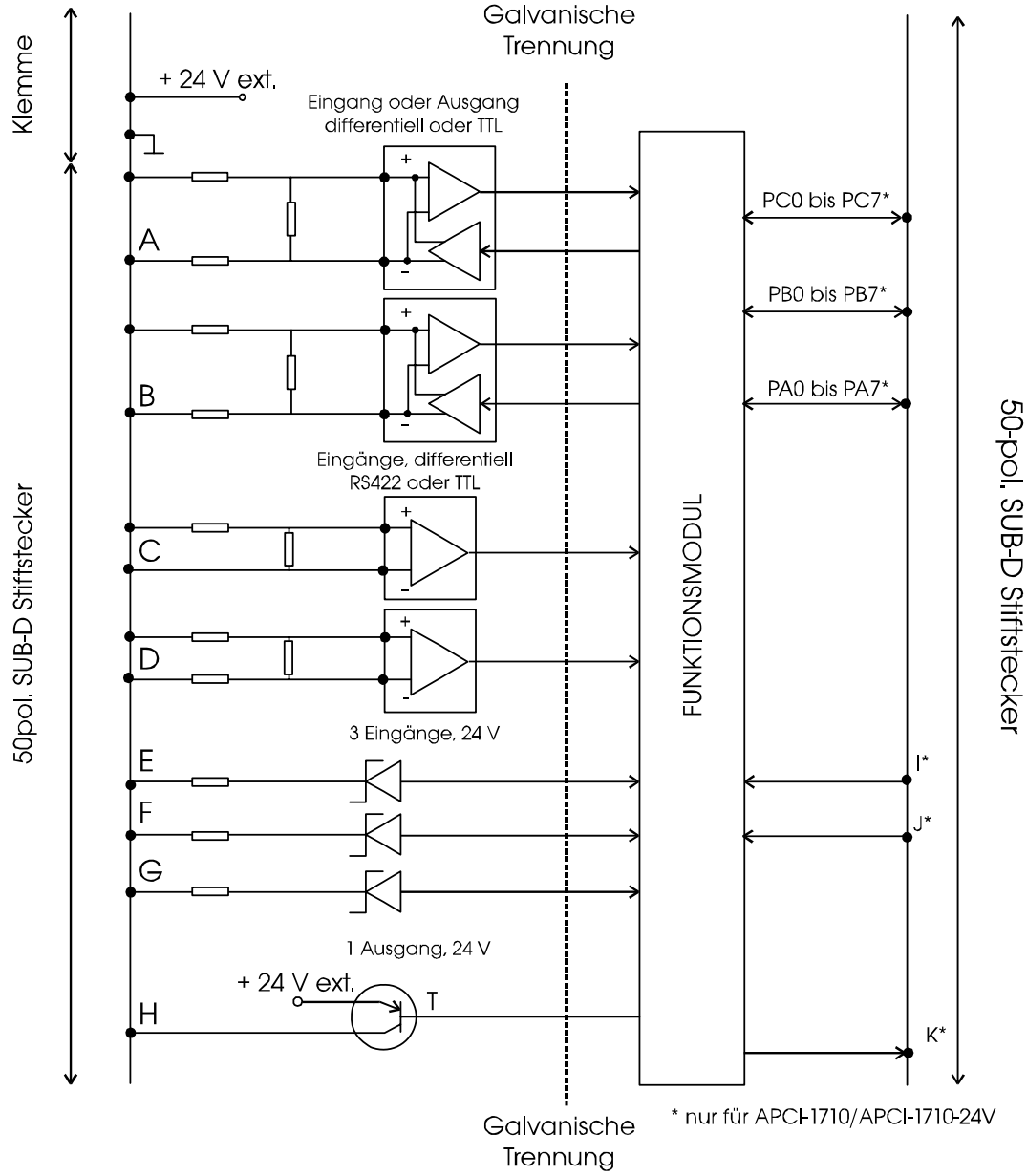
**Ausgangsleitungen**
- 1 x 24 V, TTL optional (Signal H)

**Frei definierbare Leitungen (Ein- oder Ausgang)**
- 2 x TTL, RS422, (Signale A, B)

**APCI-1710-24V**
- 7 x 24 V Eingänge (Signale A bis G)
- 1 x 24 V Ausgang, TTL optional (Signal H)

Die Funktion der digitalen Ein- und Ausgänge ist abhängig von der auf das Funktionsmodul programmierten Funktion und ist entsprechend in den jeweiligen Dokumentationen beschrieben. Nachfolgend werden nur die allgemeinen Eigenschaften der Ein- und Ausgänge beschrieben.

**Abb. 9-2 Blockdiagramm der dig. Ein- und Ausgänge (1 Funktionsmodul)**



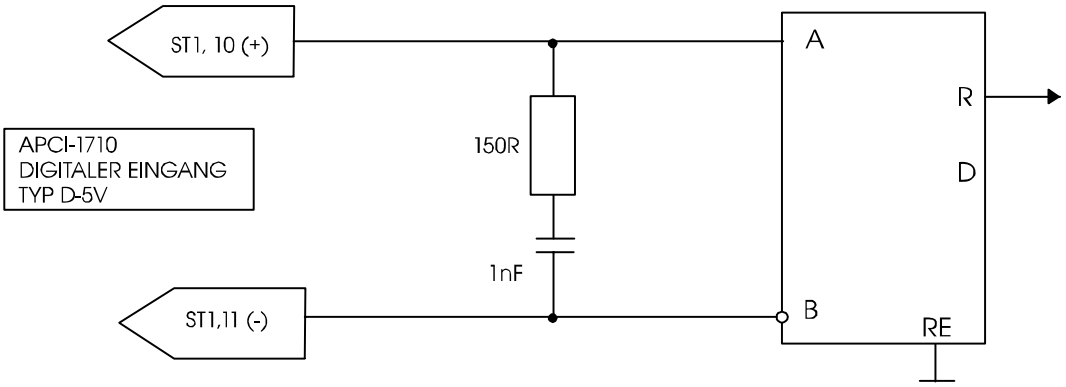
## 9.2.2 Eingänge

Die Eingänge unterscheiden sich wie folgt:
- differentielle Eingänge für sehr schnelle Signale
- massenbezogene Eingänge

### Differentielle Eingänge

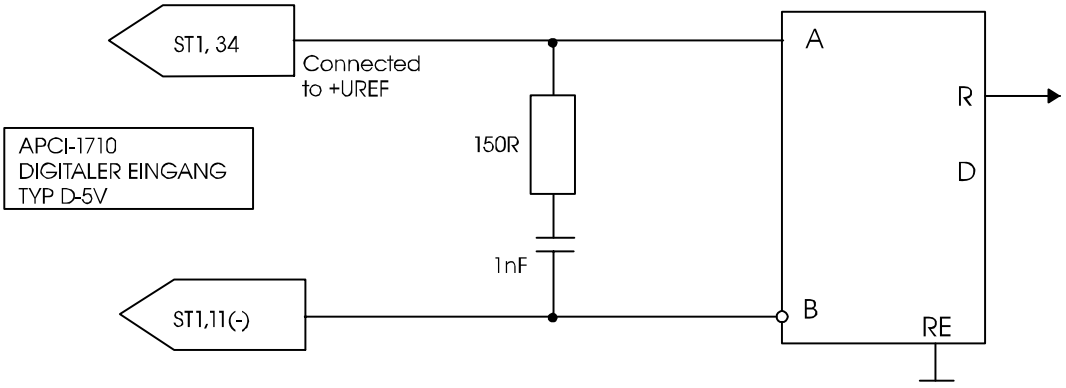
Pro "Funktionsmodul" stehen maximal 4 differentielle Eingänge (A, B, C und D) zur Verfügung. Bei Standardauslieferung entsprechen die Pegel dem RS485 (5 V) Standard.

**Abb. 9-3: Prinzipschaltbild der differentiellen Eingänge 5V**



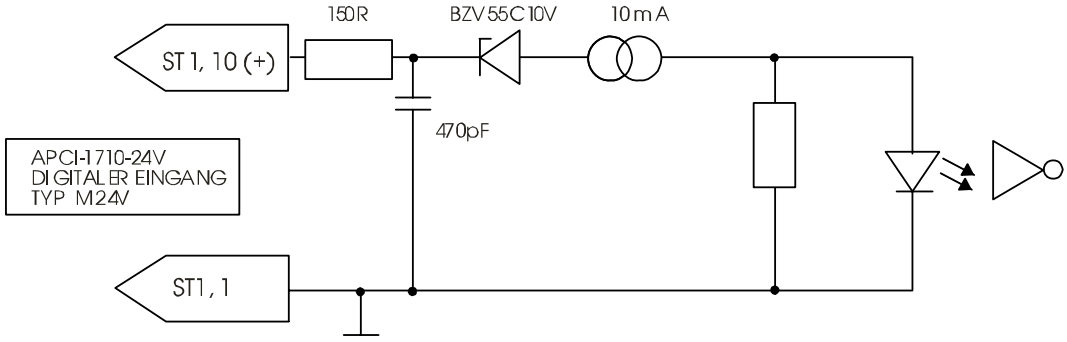

Alternativ können auch an diese Eingänge TTL Signale angeschlossen werden. Dabei ist zu beachten, dass ein Eingang des differentiellen Empfängers auf "+ UREF" verdrahtet ist, damit er auch eine Differenz bilden kann. Somit ist das TTL Signal je nach Verdrahtung an das Funktionsmodul invertiert oder nicht.

**Abb. 9-4: Prinzipschaltbild der differentiellen Eingänge 5V; als TTL Eingänge benutzt**




Optional können auch diese differentiellen Eingänge (A bis D) individuell für den Anschluss an 24 V Drehgeber / Signalgeber bestückt werden.

**Abb. 9-5: Prinzipschaltbild der differentiellen Eingänge,24 V (Option)**

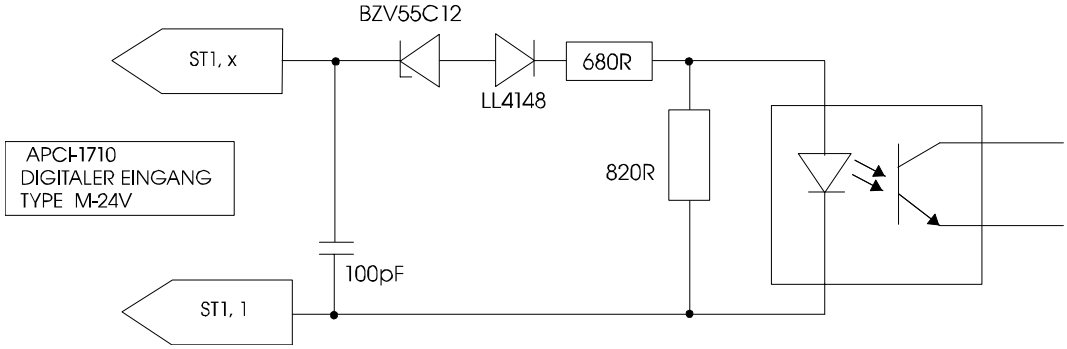



## Massenbezogene Eingänge

Pro "Funktionsmodul" stehen maximal 3 massenbezogene Eingänge (E, F und G) zur Verfügung. Bei Standardauslieferung entsprechen die Pegel dem 24 V Standard (IEC1131-2 / Typ 1).

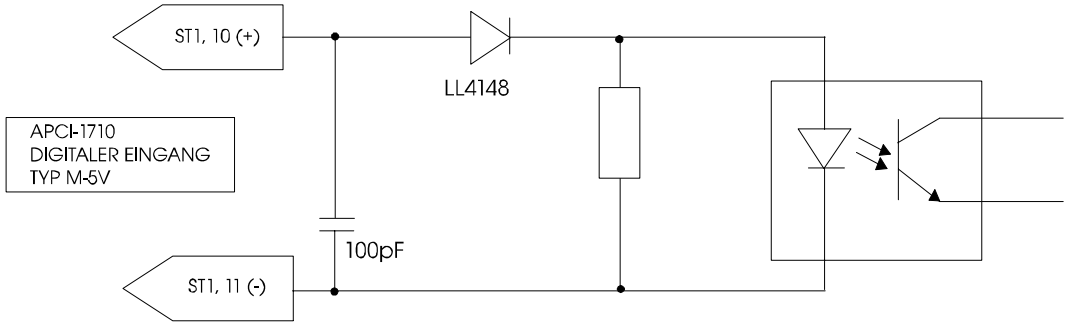
Diese Eingänge haben eine gemeinsame Masseleitung.

**Abb. 9-6: Prinzipschaltbild der digitalen Eingänge 24V**




Diese Eingänge können auf Anfrage für einen anderen Signalpegel geliefert werden (Option).

**Abb. 9-7: Prinzipschaltbild der digitalen Eingänge 5V (OPTION)**



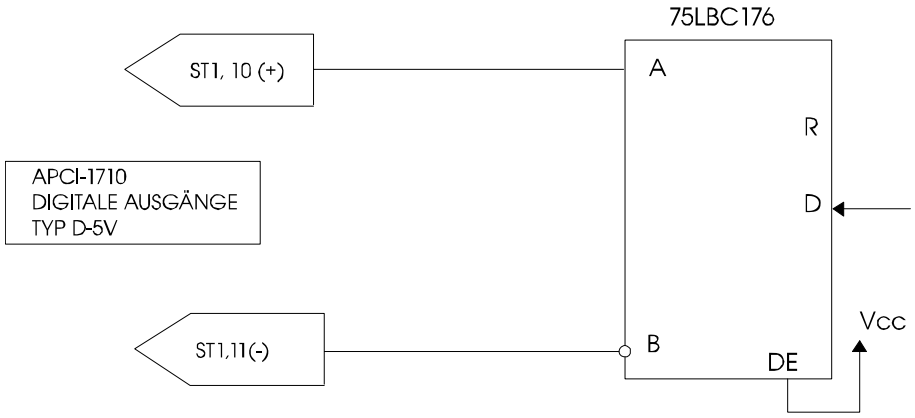
## 9.2.3 Ausgänge

Die Ausgänge unterscheiden sich wie folgt:
- differentielle Ausgänge für sehr schnelle Signale
- massenbezogene Ausgänge

### Differentielle Ausgänge

Pro "Funktionsmodul" stehen maximal 2 differentielle Ausgänge (A und B) zur Verfügung. Bei Standardauslieferung entsprechen die Pegel dem RS485 (5 V) Standard. A und B können nicht als Eingänge genutzt werden.

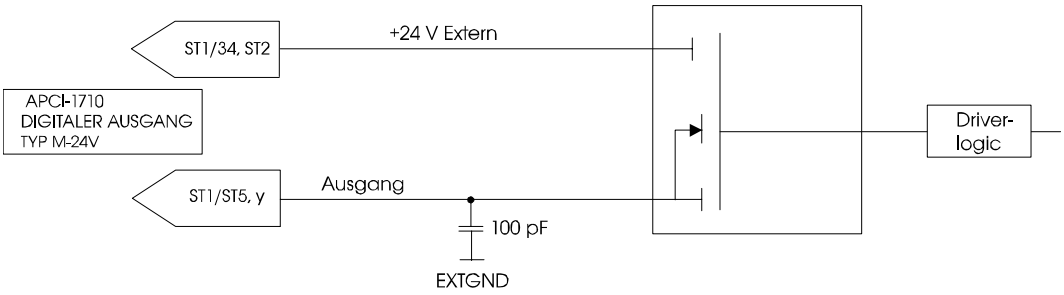
**Abb. 9-8: Prinzipschaltbild der digitalen Ausgänge 5V-Differential**




### Massenbezogene Ausgänge

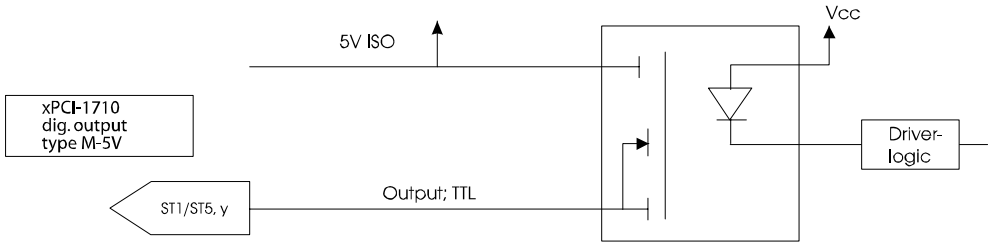
Pro Funktionsmodul steht maximal ein massenbezogener Ausgang (H) zu Verfügung. Bei Standardauslieferung entsprechen die Pegel dem 24 V Standard. (IEC1131-2 / High-Side Treiber).

**Abb. 9-9: Prinzipschaltbild des digitalen Ausgangs H - 24V**




Der Ausgang kann auf Anfrage auch als Optokoppler-Ausgang (TTL kompatibel) geliefert werden.

**Abb. 9-10: Prinzipschaltbild des digitalen Ausgangs H - 5V (OPTION**




# 9.3 TTL Ein- und Ausgänge

Die APCI-1710 stellt am Stecker ST5 weitere digitale E/A, sowie GND und Vcc des PC zur Verfügung. Diese Signale müssen dem TTL Pegel entsprechen und sorgfältig behandelt werden, um die Karte nicht zu beschädigen, falls andere Signale angeschlossen werden.

Mit dem Kabel FB8000 kann die Karte durch den Stecker ST5 an die Peripherie angeschlossen werden.

## 9.3.1 Gemeinsame Signale für alle Funktionsmodule

Die Signale PA0 bis PA7, PB0 bis PB7 und PC0 bis PC7 sind an allen "Funktionsmodulen" angeschlossen. Sie stehen nur für maximal ein "Funktionsmodul" (z. B. "TTL I/O) zur Verfügung.

**Beispiel:** Nur das Funktionsmodul Nr. 4 ist mit der Funktion TTL I/O programmiert. Die Signale können als Ein- oder Ausgänge gesetzt werden.

## 9.3.2 Einzelsignale

Pro "Funktionsmodul" stehen 2 TTL Ein- und Ausgänge (I und J) zur Verfügung. **Diese können nur in Verbindung mit dem Funktionsmodul TTL I/O genutzt werden.**

## 9.4 PCI-Bus-Schnittstelle

Die **APCI-1710** ist eine Erweiterungskarte für den PCI/CompactPCI-Bus.

Dadurch ergeben sich folgende Vorteile:
- der PCI/CompactPCI Bus ist "Plug & Play" fähig d.h. die Software stellt automatisch die Adressen, Interrupts ein.
- Die Karte wird automatisch durch das Softwareprogramm erkannt.
- Die **APCI-1710** gibt einen 32-Bit Zugriff an die Peripherie für schnellere Datenübertragungen.
- Die Karte ist nur als "Target only" auf dem PCI Bus zu betreiben. Sie hat einen Sammelinterrupt, der auf den INTA Pin des PCI-Steckers verdrahtet ist.

Nachdem der Rechner gebootet hat, und die Applikationssoftware die entsprechenden Basisadressen für die **APCI-1710** über die BIOS Funktion abgeholt hat, lässt sich auf die Karte über normale E/A Schreib- und Lesebefehle zugreifen.

Die vier Funktionsmodule belegen 256 Bytes des 64 Kbytes E/A-Bereichs des PC, jeweils 64 Bytes pro Funktionsmodul.

**WARNUNG!**
Die vom BIOS zugeordneten Adressen sollten nicht umprogrammiert werden.

Folgende Tabelle zeigt die E/A-Belegung:

**Tabelle 9-1: E/A-Belegung der Funktionsmodule**

| | IORD / IOWR | | | |
|---|---|---|---|---|
| | D31...D24 | D23...D16 | D15.....D8 | D7.....D0 |
| BYTES | HIGHBYTE | MIDHIGHBYTE | MIDLOWBYTE | LOWBYTE |
| BASE+0 | FUNKTIONSMODUL 1 | | | |
| BASE+63 | FUNKTIONSMODUL 1 | | | |
| BASE+64 | FUNKTIONSMODUL 2 | | | |
| BASE+127 | FUNKTIONSMODUL 2 | | | |
| BASE+128 | FUNKTIONSMODUL 3 | | | |
| BASE+191 | FUNKTIONSMODUL 3 | | | |
| BASE+192 | FUNKTIONSMODUL 4 | | | |
| BASE+255 | FUNKTIONSMODUL 4 | | | |

Funktionsmodul 1 kann folgende Adressen belegen: Base +0 bis Base + 63.
FM[1] 2 kann folgende Adressen belegen : Base +64 bis Base + 127.

[1] FM für Funktionsmodul

FM 3 kann folgende Adressen belegen : Base +128 bis Base + 191.
FM 4 kann folgende Adressen belegen : Base +192 bis Base + 255.

Beispiel :

In den Funktionsmodulen 1 bis 4 werden die inkrementalen Messwertgeber mit Referenzpunktlogik programmiert. Der entsprechende E/A-Funktionsbereich für den "inkrementalen Messwertgeber" wird in die dafür eingeordneten Plätze der Funktionsmodule kopiert.

Der folgende E/A Bereich entsteht:

**Tabelle 9-2: E/A-Bereich**

| | IORD / IOWR | | | |
|---|---|---|---|---|
| | D31...D24 | D23...D16 | D15.....D8 | D7.....D0 |
| BYTES | HIGHBYTE | MIDHIGHBYTE | MIDLOWBYTE | LOWBYTE |
| BASE+0 | Inkrementaler Zähler 1 | | | |
| BASE +59 | Inkrementaler Zähler 1 | | | |
| BASE +60 bis +63 | Funktionsmodul 1 - Info **SCxx** | | | |
| BASE+64 | Inkrementaler Zähler 2 | | | |
| BASE+ 123 | Inkrementaler Zähler 2 | | | |
| BASE + 124 bis + 127 | Funktionsmodul 2 - Info SCxx | | | |
| BASE+128 | Inkrementaler Zähler 3 | | | |
| BASE+187 | Inkrementaler Zähler 3 | | | |
| BASE +188 bis +191 | Funktionsmodul 3 - Info SCxx | | | |
| BASE+192 | Inkrementaler Zähler 4 | | | |
| BASE+251 | Inkrementaler Zähler 4 | | | |
| BASE +252 bis +255 | Funktionsmodul 4 - Info SCxx | | | |

SC: Inkrementalzähler xx: Versionsnummer. z. B. SC23 Version 2.3

Alle Einstellungen auf der Baugruppe werden durch Software (API), durch die Steckerbelegung oder durch eine Wiederprogrammierung der Funktionsmodule vorgenommen (kein Jumper auf der Karte).

# 10 STANDARDSOFTWARE

## 10.1 Einleitung

**WICHTIG!**
Merken Sie sich die folgenden Schreibweisen im Text.

Funktion: "i_APCI1710_SetBoardInformation"
Variable: *ui_Address*

**Tabelle 10-1: Type-Deklaration für DOS und Windows 3.1x**

| | Borland C | Microsoft C | Borland Pascal | Microsoft Visual Basic Dos | Microsoft Visual Basic Windows |
|---|---|---|---|---|---|
| **VOID** | void | void | pointer | | any |
| **BYTE** | unsigned char | unsigned char | byte | integer | integer |
| **INT** | int | int | integer | integer | integer |
| **UINT** | unsigned int | unsigned int | word | long | long |
| **LONG** | long | long | longint | long | long |
| **PBYTE** | unsigned char * | unsigned char * | var byte | integer | integer |
| **PINT** | int * | int * | var integer | integer | integer |
| **PUINT** | unsigned int * | unsigned int * | var word | long | long |
| **PCHAR** | char * | char * | var string | string | string |

**Tabelle 10-2: Type-Deklaration für Windows 95/NT**

| | Borland C | Microsoft C | Borland Pascal | Microsoft Visual Basic Dos | Microsoft Visual Basic Windows |
|---|---|---|---|---|---|
| **VOID** | void | void | pointer | | any |
| **BYTE** | unsigned char | unsigned char | byte | integer | integer |
| **INT** | int | Int | integer | integer | integer |
| **UINT** | unsigned int | unsigned int | long | long | long |
| **LONG** | long | long | longint | long | long |
| **PBYTE** | unsigned char * | unsigned char * | var byte | integer | integer |
| **PINT** | int * | int * | var integer | integer | integer |
| **PUINT** | unsigned int * | unsigned int * | var long | long | long |
| **PCHAR** | char * | char * | var string | string | string |

**Tabelle 10-3: Define-Wert**

| Define-Name | Dezimal-Wert | Hexadezimal-Wert |
|---|---|---|
| **DLL_COMPILER_C** | 1 | 1 |
| **DLL_COMPILER_VB** | 2 | 2 |
| **DLL_COMPILER_PASCAL** | 3 | 3 |
| **DLL_LABVIEW** | 4 | 4 |
| **DLL_COMPILER_VB_5** | 5 | 5 |
| **APCI1710_DISABLE** | 0 | 0 |
| **APCI1710_ENABLE** | 1 | 1 |

## 10.2 Software-Funktionen

### 10.2.1 Initialisierung

**WICHTIG!**
In diesem Kapitel werden die gemeinsamen Funktionen für jedes Funktionsmodul aufgelistet.

Die eigenen Software Funktionen je nach Funktion der Karte APCI-1710 können Sie in den entsprechenden Handbüchern lesen.

#### 1) i_APCI1710_InitCompiler (...)

**Syntax:**
<Return-Wert>= i_APCI1710_InitCompiler
(BYTE b_CompilerDefine)

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_CompilerDefine | Der Benutzer soll die Sprache unter Windows auswählen, in der er programmieren will.<br>- DLL_COMPILER_C:<br>Der Benutzer programmiert in C.<br>- DLL_COMPILER_VB:<br>Programmierung in Visual Basic für Windows.<br>- DLL_COMPILER_VB_5:<br>Programmierung in Visual Basic 5 für Windows NT oder Windows 95/98.<br>- DLL_COMPILER_PASCAL:<br>Programmierung in Pascal oder Delphi.<br>- DLL_LABVIEW:<br>Programmierung in Labview. |

**- Ausgabe:**
Es erfolgt keine Ausgabe.

**Aufgabe:**
Wenn Sie die DLL Funktionen benutzen wollen, geben Sie ein, in welcher Sprache Sie programmieren. Rufen Sie diese Funktion als erste auf.

**WICHTIG!**
**Diese Funktion ist nur unter Windows verfügbar.**

**Funktionsaufruf:**
ANSI C :
int i_ReturnValue;

i_ReturnValue = i_APCI1710_InitCompiler (DLL_COMPILER_C);

**Return-Wert:**
0: Kein Fehler
-1: Compiler Parameter ist falsch

## 2) i_APCI1710_CheckAndGetPCISlotNumber (...)

**Syntax:**
<Return-Wert> = i_APCI1710_CheckAndGetPCISlotNumber
(PBYTE pb_SlotNumberArray)

**Parameter:**
**- Eingabe:**
Es erfolgt keine Eingabe.
**- Ausgabe**
PBYTE pb_SlotNumberArray Liste der Steckplatznummern

**Aufgabe:**
Überprüft alle **APCI-1710** und gibt die Steckplatznummer jeder Karte an.
Jeder Parameter *pb_SlotNumberArray* enthält die Steckplatznummer
(1 bis 8) einer **APCI-1710** Karte.

**Funktionsaufruf:**
ANSI C :

```
int       i_ReturnValue;

unsigned char b_SlotNumberArray [8];

i_ReturnValue = i_APCI1710_CheckAndGetPCISlotNumber
                                        (b_SlotNumberArray);
```

**Return-Wert:**
Gibt die Anzahl der **APCI-1710** zurück, die in dem PC eingebaut sind.
Wenn der Return-Wert eine "0" zurückgibt, wurde keine **APCI-1710** in dem PC gefunden.

### 3) i_APCI1710_SetBoardInformation (...)

**Syntax:**
<Return-Wert> = i_APCI1710_SetBoardInformation
(BYTE b_SlotNumber,
PBYTE pb_BoardHandle)

**Parameter:**
**- Eingabe:**
BYTE b_SlotNumber Steckplatznummer der Karte
**- Ausgabe:**
PBYTE pb_BoardHandle Handle der Karte, um die Funktionen zu benutzen.

**Aufgabe:**
Überprüft, ob die **xPCI-1710** vorhanden ist und speichert die Steckplatznummer ab.
Der Benutzer bekommt einen Handle zurück, damit die nächsten Funktionen benutzt werden können. Handles ermöglichen es, mehrere Karten zu verwalten.

**Funktionsaufruf:**
ANSI C :
```
int             i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1710_SetBoardInfomation (1, &b_BoardHandle);
```

**Return-Wert:**
-0: Kein Fehler
-1: Steckplatznummer nicht verfügbar
-2: Karte nicht vorhanden
-3: Kein Handle für die Karte verfügbar (auf 10 Handles begrenzt)
-4: Fehler beim Öffnen des Treibers in Windows NT/ Windows 95

## 4) i_APCI1710_ConfigureAllModule

**Syntax:**
<Return-Wert> = i_APCI1710_ ConfigureAllModule
(BYTE b_BoardHandle,
PCHAR pc_FileName1,
PCHAR pc_FileName2,
PCHAR pc_FileName3,
PCHAR pc_FileName4,
PULONG pul_WriteAddressError);

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der Karte |
| PCHAR | pc_FileName1 | Name der Konfigurationsdatei für FM[1] 0 |
| PCHAR | pc_FileName2 | Name der Konfigurationsdatei für FM 1 |
| PCHAR | pc_FileName3 | Name der Konfigurationsdatei für FM 2 |
| PCHAR | pc_FileName4 | Name der Konfigurationsdatei für FM 3 |

**- Ausgabe:**

| | | |
|---|---|---|
| PULONG | pul_WriteAddressError | Im Fall eines Fehlers, Adresse des Lese-/Schreibregisters |

**Aufgabe:**
Initialisiert die Funktionsmodule per Software.

**Funktionsaufruf:**
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned long   ul_WriteAddressError;

i_ReturnValue = i_APCI1710_ ConfigureAllModule
                                    (b_BoardHandle,
                                     "CFG\\INC_CPT.CFG",
                                     "CFG\\INC_CPT.CFG",
                                     "CFG\\INC_CPT.CFG",
                                     "CFG\\INC_CPT.CFG",
                                      &ul_WriteAddressError);
```

**Return-Wert:**
0: Kein Fehler
-1: Der Handle-Parameter der Karte ist falsch
-2: Fehler beim Laden der Datei
-3: Die Entschlüsselung der Altera Datei ist falsch.
-4: Fehler beim Löschen des EEPROM Bausteins (Flash)
-5: Die Programmierung des Flash ist falsch
-6: Fehler beim Auslesenvorbereitung des Flash
-7: Das Auslesen des Flash ist falsch
-8: Vergleich zwischen dem Schreiben und dem Lesen des Flash nicht korrekt
-9: Fehler beim Laden des Flex-Bausteins

[1] FM: Funktionsmodul

-10: Der Ladentest des Flex-Bausteins ist falsch

## 5) i_APCI1710_GetHarwareInformation

**Syntax:**
<Return-Wert> = i_APCI1710_GetHardwareInformation
(BYTE b_BoardHandle,
PUINT pui_BaseAddress,
PBYTE pb_InterruptNbr,
PBYTE pb_SlotNumber)

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der Karte |

**- Ausgabe:**

| | | |
|---|---|---|
| PUINT | pui_BaseAddress | Basisadresse der Karte |
| PBYTE | pb_InterruptNbr | Interruptkanal der Karte |
| PBYTE | pb_SlotNumber | Steckplatznummer der Karte |

**Aufgabe:**
Gibt die Basisadresse , den Interrupt und die Steckplatznummer der Karte zurück.

**Funktionsaufruf:**
ANSI C :

```
int               i_ReturnValue;
unsigned int      ui_BaseAddress;
unsigned char     b_InterruptNbr;
unsigned char     b_SlotNumber;
unsigned char     b_BoardHandle;

i_ReturnValue = i_APCI1710_GetHardwareInformation
                                        (b_BoardHandle,
                                         &ui_BaseAddress,
                                         &b_InterruptNbr,
                                         &b_SlotNumber);
```

**Return-Wert:**
0: Kein Fehler
-1: Der Handle Parameter der Karte ist falsch

### 6) i_APCI1710_CloseBoardHandle (...)

**WICHTIG!**
Rufen Sie diese Funktion jedesmal auf, wenn Sie das Benutzerprogramm verlassen wollen!

**Syntax:**
<Return-Wert> = i_APCI1710_CloseBoardHandle
(BYTE b_BoardHandle)

**Parameter:**
**- Eingabe:**
BYTE b_BoardHandle Handle der **xPCI-1710**
- Ausgabe:
Es erfolgt keine Ausgabe
**Aufgabe:**
Gibt den Handle der Karte frei. Sperrt den Zugriff auf die Karte.

**Funktionsaufruf:**
ANSI C :

```
int                 i_ReturnValue;
unsigned char  b_BoardHandle;

i_ReturnValue = i_APCI1710_CloseBoardHandle  (b_BoardHandle);
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch

## 10.2.2 Interrupt

### 7) i_APCI1710_SetBoardIntRoutineDos (..)

**WICHTIG!**
Diese Funktion kann nur für C/C++ und Pascal für DOS benutzt werden.

**Syntax:**
<Return-Wert> = i_APCI1710_SetBoardIntRoutineDos
(BYTE b_BoardHandle,
VOID v_FunctionName
(BYTE b_BoardHandle,
BYTE b_ModuleMask,
ULONG ul_InterruptMask,
ULONG ul_CounterLatchValue))

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der xPCI-1710 Karte |
| VOID | v_FunctionName | Name der Benutzer-Interruptroutine |

**- Ausgabe:**
Es erfolgt keine Ausgabe.

**Aufgabe:**
Diese Funktion ist für alle xPCI-1710 Karten aufzurufen, auf denen Sie einen Interrupt aktivieren wollen.
Beim ersten Aufruf der Funktion (erste Karte):
- wird die Benutzer-Interruptroutine installiert,
- werden die Interrupts ermöglicht.
Falls Sie mehrere xPCI-1710 betreiben, die auf Interrupts reagieren sollen, müssen Sie die Funktion so viel aufrufen, wie Sie xPCI-1710 Karten betreiben.
Die Variable *v_FunctionName* hat nur beim ersten Aufruf eine Bedeutung
Ab dem zweiten Aufruf der Funktion (nächste Karten) werden Interrupts ermöglicht.

**Interrupt**
Wenn ein Interrupt erzeugt wird, wird die Benutzer-Interruptroutine vom System aufgerufen. Wenn mehrere Karten betrieben werden, und mehrere auf Interrupts reagieren sollen, gibt die Variable *b_BoardHandle* die Identifikationsnummer (Handle) der Karte, die den Interrupt erzeugt hat.
Die Benutzer-Interruptroutine muss die folgende Syntax haben:
VOID_ v_FunctionName (BYTE b_BoardHandle,
BYTE b_ModuleMask
ULONG ul_InterruptMask,
ULONG ul_CounterLatchValue)

| | |
|---|---|
| *v_FunctionName* | Name der Benutzer-Interruptroutine |
| *b_BoardHandle* | Nummer des xPCI-1710-Handles, der den Interrupt generiert hat. |
| *b_ModulMask* | Maske des Moduls, das den Interrupt generiert hat. (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |
| *ul_InterruptMask* | Maske der Events, das den Interrupt generiert. (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |
| u*l_CounterLatchValue* | Die geschalteten Werte des Timers werden zurückgegeben. (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |

Der Benutzer kann einen anderen Name für v_FunctionName, b_BoardHandle, ul_InterruptMask, ul_CounterLatchValue vergeben.

**Funktionsaufruf:**
ANSI C :

```
void    v_FunctionName      (unsigned char  b_BoardHandle,
                             unsigned char  b_ModuleMask,
                             unsigned long ul_InterruptMask
                             unsigned long ul_CounterLatchValue)
                {
                .
                .
                }
int             i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_APCI1710_SetBoardIntRoutineDos
                                        (b_BoardHandle,
                                         v_FunctionName );
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch
-2: Interrupt schon installiert

### 8) i_APCI1710_SetBoardIntRoutineVBDos (..)

**WICHTIG!**
Diese Funktion kann nur für Visual Basic DOS benutzt werden.

**Syntax:**
<Return-Wert> = i_APCI1710_SetBoardIntRoutineVBDos
(BYTE b_BoardHandle)

**Parameter:**
**- Eingabe:**
BYTE b_BoardHandle Handle der Karte
**- Ausgabe:**
Es erfolgt keine Ausgabe.

**Aufgabe:**
Diese Funktion ist für alle Karten xPCI-1710 aufzurufen, auf die Sie einen Interrupt aktivieren wollen. Wenn einen Interrupt aktiviert ist, wird ein Visual Basic Event generiert.
Beim ersten Aufruf der Funktion (erste Karte):
- werden die Interrupts für die ausgewählte Karte ermöglicht.
Falls Sie mehrere xPCI-1710 betreiben, die auf Interrupts reagieren sollen, müssen Sie die Funktion so viel aufrufen, wie Sie xPCI-1710 Karten betreiben.

***Interrupt***
Wenn ein Interrupt erzeugt wird, wird die Benutzer-Interruptroutine vom System aufgerufen.

***Interruptverwaltung steuern***
Benutzen Sie die Funktionen "ON UEVENT GOSUB xxxxxxxxx" von Visual Basic DOS und "i_APCI1710_TestInterrupt"
Diese Funktion testet den Interrupt der xPCI-1710. Sie wird benutzt, um die Werte von *b_BoardHandle, b_ModuleMask, ul_InterruptMask und ul_CounterLatchValue* zu erhalten.

**Funktionsaufruf:**
Visual Basic DOS:

```
Dim Shared i_ReturnValue        As Integer
Dim Shared i_BoardHandle        As Integer
Dim Shared i_ModuleMask         As Integer
Dim Shared l_InterruptMask      As Long
Dim Shared l_CounterLatchValue  As Long
IntLabel:
```

```
i_ReturnValue = i_APCI1710_TestInterrupt      (i_BoardHandle, _
                                               i_ModuleMask,  _
                                               l_InterruptMask, _
                                               l_CounterLatchValue)
.
.
.

Return


ON UEVENT GOSUB IntLabel
UEVENT ON
```

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineVBDos (b_BoardHandle)
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch
-2: Interrupt bereits installiert

### 9) i_APCI1710_SetBoardIntRoutineWin16 (..)

**WICHTIG!**
Diese Funktion kann nur für Windows 3.1 und Windows 3.11 benutzt werden.

**Syntax:**

```
<Return-Wert> = i_APCI1710_SetBoardIntRoutineWin16
                              (BYTE   b_BoardHandle,
                               VOID   v_FunctionName
                                      (BYTE     b_BoardHandle,
                                       BYTE     b_ModuleMask,
                                       ULONG    l_InterruptMask,
                                       ULONG    ul_CounterLatchValue))
```

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der Karte |
| VOID | v_FunctionName | Name der Benutzer-Interruptroutine |

**- Ausgabe:**
Es erfolgt keine Ausgabe.

**Aufgabe:**
Diese Funktion ist für alle **xPCI-1710** Karten aufzurufen, auf denen Sie einen Interrupt aktivieren wollen.
Beim ersten Aufruf der Funktion (erste Karte):
- wird die Benutzer-Interruptroutine installiert,
- werden die Interrupts ermöglicht.

Falls Sie mehrere **xPCI-1710** betreiben, die auf Interrupts reagieren sollen, müssen Sie die Funktion so oft aufrufen, wie Sie xPCI-1710 Karten betreiben.
Die variable *v_FunctionName* hat **nur beim ersten Aufruf** eine Bedeutung
Ab dem zweiten Aufruf der Funktion (nächste Karten) werden Interrupts ermöglicht.

**Interrupt**
Wenn ein Interrupt erzeugt wird, wird die Benutzer-Interruptroutine vom System aufgerufen.
Wenn mehrere Karten betrieben werden, und mehrere auf Interrupts reagieren sollen, gibt die Variable *b_BoardHandle* die Identifikationsnummer (Handle) der Karte, die den Interrupt erzeugt hat.
Die Benutzer-Interruptroutine muss die folgende Syntax haben:

```
VOID   v_FunctionName   (BYTE     b_BoardHandle,
                         BYTE     b_ModuleMask,
                         ULONG    ul_InterruptMask,
                         ULONG    ul_CounterLatchValue)
```

| | |
|---|---|
| *v_FunctionName* | Name der Benutzer-Interruptroutine |
| *b_BoardHandle* | Handle der **xPCI-1710,** die den Interrupt generiert hat. |
| *b_ModuleMask* | Maske des Moduls, das den Interrupt generiert hat. (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |
| *ul_InterruptMask* | Maske der Events, die den Interrupt generiert haben. (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |
| *ul_CounterLatchValue* | Die latched Werte des Timers werden zurückgegeben (Siehe Tabelle der Interrupt-Maske im entsprechenden Referenzhandbuch) |

Der Benutzer kann einen anderen Name für v_FunctionName, b_BoardHandle, b_ModuleMask, ul_InterruptMask und ul_CounterLatchValue vergeben.

**WICHTIG!**
Wenn Sie Visual Basic für Windows benutzen, gibt es den folgenden Parameter nicht. Benutzen Sie die Funktion "i_APCI1710_TestInterrupt".

```
VOID   v_FunctionName          (BYTE     b_BoardHandle,
                                BYTE     b_ModuleMask,
                                ULONG    ul_InterruptMask,
                                ULONG    ul_CounterLatchValue)
```

**Funktionsaufruf:**
ANSI C :

```
void    v_FunctionName      (unsigned char   b_BoardHandle,
                             unsigned char   b_ModuleMask,
                             unsigned long   ul_InterruptMask,
                             unsigned long   ul_CounterLatchValue)
                {
                .
                .
                }

int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin16
                        (b_BoardHandle,
                         v_FunctionName );
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch
-2: Interrupt schon installiert

### 10) i_APCI1710_SetBoardIntRoutineWin32 (..)

**WICHTIG!**
Diese Funktion ist nur für Windows NT und Windows 9x verfügbar.

**Syntax:**

```
<Return-Wert> = i_APCI1710_SetBoardIntRoutineWin32
                        (BYTE       b_BoardHandle,
                         BYTE       b_UserCallingMode,
                         ULONG      ul_UserSharedMemorySize,
                         VOID **    ppv_UserSharedMemory,
                         VOID       v_FunctionName
                                    (BYTE     b_BoardHandle,
                                     BYTE     b_ModuleMask,
                                     ULONG    ul_InterruptMask,
                                     ULONG    ul_CounterLatchValue,
                         BYTE       b_UserCallingMode,
                                     VOID *   pv_UserSharedMemory))
```

**Parameter:**

**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der Karte **xPCI-1710** |
| BYTE | b_UserCallingMode | APCI1710_SYNCHRONOUS_MODE: Die Benutzerroutine wird direkt durch die Interruptroutine des Treibers aufgerufen APCI1710_ASYNCHRONOUS_MODE: Die Benutzerroutine wird direkt durch den Interrupt Thread des Treibers aufgerufen |
| VOID | v_FunctionName | Name der Benutzer-Interruptroutine |
| ULONG | ul_UserSharedMemorySize | Bestimmt die Größe in Bytes des Benutzer Gemeinschaftsspeichers Sie können den Parameter nur benutzen, wenn Sie den Mode APCI1710_SYNCHRONOUS_MODE ausgewählt haben. |

**WICHTIG!**
**Die Größe des User Shared Memory ist auf 63 MB begrenzt. Falls mehr Speicherplatz verwendet wird, könnte dies zu Problemen führen.**

**- Ausgabe:**

| | | |
|---|---|---|
| VOID ** | ppv_UserSharedMemory | Adresse des Benutzer-Gemeinschaftsspeichers (Shared memory) Sie können den Parameter nur benutzen, wenn Sie den Mode |

APCI1710_SYNCHRONOUS_MODE
ausgewählt haben.

**WICHTIG!**
Für Windows NT und Windows 95 stehen 4 Rings zur Verfügung (Ring 0 bis Ring 3).

- Das Benutzer-Anwendungsprogramm läuft unter Ring 3. In diesem Ring steht kein Zugriff auf Hardware zur Verfügung.
- VXD und SYS Treiber laufen unter Ring 0 und haben einen Hardwarezugriff.
- Ring 0 kann nicht auf die Variable von Ring 3 zugreifen und soll den Gemeinschaftsspeicher (Shared memory) benutzen.
- Ring 0 und Ring 3 verfügen über einen Zeiger, der diesen Gemeinschaftsspeicher ermittelt. Beide Rings haben eine verschiedene Adresse.

Diese Funktion ist für alle **xPCI-1710-**Karten aufzurufen, auf die Sie einen Interrupt aktivieren wollen.
Beim ersten Aufruf der Funktion (erste Karte):
- wird die Benutzer-Interruptroutine installiert,
- werden die Interrupts ermöglicht,
- wird der Benutzer-Gemeischaftsspeicher zugeteilt, wenn der APCI1710_SYNCHRONOUS_MODE aktiviert ist.

Falls Sie mehrere **xPCI-1710** betreiben, die auf Interrupts reagieren sollen, müssen Sie die Funktion so oft aufrufen, wie Sie xPCI-1710 Karten betreiben. Die variable *v_FunctionName* hat **nur beim ersten Aufruf** eine Bedeutung. Ab dem zweiten Aufruf der Funktion (nächste Karten) werden Interrupts ermöglicht.

***Interrupt***
Wenn ein Interrupt erzeugt wird, wird die Benutzer-Interruptroutine vom System aufgerufen.
Wenn mehrere Karten betrieben werden und auf Interrupts reagieren sollen, gibt die Variable *b_BoardHandle* die Identifikationsnummer (Handle) der Karte, die den Interrupt erzeugt hat.
Die Benutzer-Interruptroutine kann wie folgt aufgerufen werden:
- direkt von der Interruptroutine des Treibers (Synchron-Mode). Das Code der Benutzer-Interruptroutine läuft unmittelbar unter Ring 0.
- oder vom Interrupt-Thread des Treibers (Asynchron-Mode). Ein Event wird generiert und der Interrupt Thread ruft die Benutzer-Interruptroutine auf. Das Code der Benutzer-Interruptroutine läuft unter Ring 3.
  Der Interrupt Thread des Treibers hat die höchste Priorität (31) im System.

**Abb. 10-1: Synchroner und asynchroner Mode**

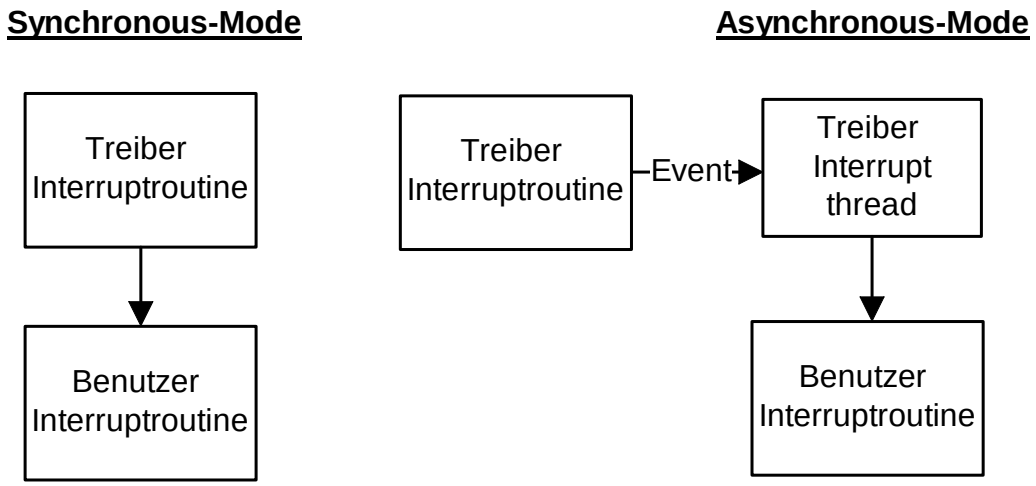



**Abb. 10-2: Synchroner und asynchroner Mode**

| | **SYNCHRONER MODE** |
|---|---|
| **VORTEILE** | Die Benutzer-Interruptroutine wird direkt von der Interruptroutine des Treibers aufgerufen (Ring 0). Die Zeit zwischen Interrupt und der Benutzer-Interruptroutine ist reduziert. |
| **EINSCHRÄNKUNGEN** | Ein Debug der Benutzer-Interruptroutine ist nicht möglich. |
| | Die Benutzer-Interruptroutine kann nicht die Windows API Funktionen aufrufen. |
| | Die Benutzer-Interruptroutine kann nicht die Funktionen aufrufen, die Zugriff auf die gemeinsame Variable zur Verfügung haben. Der Benutzer kann jedoch einen Gemeinschaftsspeicher anwenden. |
| | Die Benutzer-Interruptroutine kann nur die Funktionen des xPCI-1710 Gerätetreibers aufrufen, die die folgende Extension haben: "i_APCI1710_KRNL_XXX" |
| | Dieser Mode kann nicht unter Visual Basic benutzt werden. |

| | **ASYNCHRONER MODE** |
|---|---|
| **VORTEILE** | Ein Debug der Benutzer-Interruptroutine ist möglich. |
| | Die Benutzer-Interruptroutine kann die API Funktionen aufrufen. |
| | Die Benutzer-Interruptroutine kann die Funktionen aufrufen, die Zugriff auf die gemeinsamen Variable zur Verfügung haben. |
| | Die Benutzer-Interruptroutine kann alle Funktionen des xPCI-1710 Gerätetreibers aufrufen. Syntax: "i_APCI1710_XXX" |
| **EINSCHRÄNKUNGEN** | Das Code der Benutzer-Interruptroutine wird vom Interrupt Thread des Treibers aufgerufen (Ring 3). Die Zeit zwischen Interrupt und der Benutzer-Interruptroutine wird erhöht. |

**Gemeinschaftsspeicher (Shared memory):**
Wenn Sie den APCI1710_Synchronous_Mode ausgewählt haben, können Sie keinen Zugriff auf die allgemeinen Funktionen haben. Sie haben aber die Möglichkeit, einen Gemeinschaftsspeicher zu erstellen (ppv_UserSharedMemory), in dem alle vorgegebene Compiler oder Benutzer Define gespeichert werden.
Die Variable ul_UserSharedMemorySize ermittelt die Größe in Byte des ausgewählten Benutzer Typs.
Ein Zeiger der Variable pv_UserSharedMemory wird der Interruptroutine mit der Variable pv_ USerSharedMemory zurückgegeben. Diese Funktion ist nicht möglich in Visual Basic.
Die Benutzer-Interruptroutine soll folgende Syntax haben:

```
VOID    v_FunctionName          (BYTE     b_BoardHandle,
                                 BYTE     b_ModuleMask,
                                 ULONG    ul_InterruptMask,
                                 ULONG    ul_CounterLatchValue,
                                 BYTE     b_UserCallingMode,
                                 VOID *   pv_UserSharedMemory)
```

| | |
|---|---|
| v_FunctionName | Name der Benutzer-Interruptroutine |
| b_BoardHandle | Handle der **xPCI-1710,** die den Interrupt generiert hat. |
| b_ModuleMask | Maske des Moduls, das den Interrupt generiert hat.<br>Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |
| ul_InterruptMask | Maske der Events, die den Interrupt generiert haben.<br>Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |
| ul_CounterLatchValue | Die latched Werte des Zählers werden zurückgegeben.<br>Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |

| | |
|---|---|
| b_UserCallingMode | APCI1710_SYNCHRONOUS_MODE:<br>Die Benutzerroutine wird direkt vom Treiber-Interruptroutine aufgerufen.<br>APCI1710_ASYNCHRONOUS_MODE:<br>Die Benutzerroutine wird direkt vom Treiber-Interrupt-Thread aufgerufen. |
| pv_UserSharedMemory | Zeiger des Benutzer-Gemeinschaftsspeichers |

Der Benutzer kann andere Name für v_FunctionName, b_BoardHandle, b_ModuleMask, ul_InterruptMask, ul_CounterLatchValue, b_UserCallingMode und pv_UserSharedMemory vergeben.

**WICHTIG!**
Wenn Sie Visual Basic 4 benutzen, haben die folgenden Parameter sind keine Bedeutung. Benutzen Sie die Funktion "i_APCI1710_TestInterrupt".

```
BYTE       b_UserCallingMode,
ULONG      ul_UserSharedMemorySize,
VOID **    ppv_UserSharedMemory,
VOID       v_FunctionName      (BYTE    b_BoardHandle,
                                BYTE    b_ModuleMask,
                                ULONG   ul_InterruptMask,
                                ULONG   ul_CounterLatchValue,
                                BYTE    b_UserCallingMode,
                                VOID *  pv_UserSharedMemory)
```

**Funktionsaufruf:**

ANSI C :

```
typedef struct
        {
        .
        .
        .
        }str_UserStruct;
   str_UserStruct * ps_UserSharedMemory;
void v_FunctionName     (unsigned char     b_BoardHandle,
                         unsigned char     b_ModuleMask,
                         unsigned long     ul_InterruptMask,
                         unsigned long     ul_CounterLatchValue,
                         unsigned char     b_UserCallingMode,
                         void *            pv_UserSharedMemory)
                {
str_UserStruct * ps_InterruptSharedMemory;
ps_InterruptSharedMemory = (str_UserStruct *) pv_UserSharedMemory;
                .
                .
                }
int             i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin32
                        (b_BoardHandle,
                        APCI1710_SYNCHRONOUS_MODE,
                         sizeof (str_UserStruct),
                         (void **) &ps_UserSharedMemory,
                         v_FunctionName);
```

**Visual Basic 5:**

```
Sub v_FunctionName      (ByVal i_BoardHandle          As Integer,
                         ByVal i_ModuleMask           As Integer,
                         ByVal l_InterruptMask        As Long,
                         ByVal l_CounterLatchValue    As Long,
                         ByVal b_UserCallingMode      As Integer,
                         ByVal l_UserSharedMemory     As Long)
        End Sub

Dim i_ReturnValue   As Integer
Dim i_BoardHandle As Integer

i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin32
                                (i_BoardHandle,
                                 APCI1710_ASYNCHRONOUS_MODE,
                                 0,
                                 0,
                                 AddressOf v_FunctionName)
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch
-2: Interrupt schon installiert
-3: Der ausgewählte Aufrufmode der Benutzer-Interruptroutine ist falsch
-4: Kein Speicherplatz für den Benutzer-Gemeinschaftsspeicher verfügbar

## 11) i_APCI1710_TestInterrupt

**Syntax:**
<Return-Wert> = i_APCI1710_TestInterrupt
(BYTE b_BoardHandle,
BYTE b_ModuleMask
PULONG pul_InterruptMask,
PULONG pul_CounterLatchValue)

**Parameter:**
**- Eingabe:**
Es erfolgt keine Eingabe.
**- Ausgabe:**

| | | |
|---|---|---|
| PBYTE | pb_BoardHandle | Handle der xPCI-1710, die den Interrupt ausgelöst hat. Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |
| PBYTE | pb_ModuleMask | Maske des Moduls, das den Interrupt ausgelöst hat. Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |
| PULONG | pul_InterruptMask | Maske der Events, die den Interrupt ausgelöst haben. Siehe die Tabelle der Interrupt-Maske in dem entsprechenden Referenzhandbuch |
| PULONG | pul_CounterLatchValue | Gibt die latched Werte des Zählers zurück. |

**Aufgabe:**
Überprüft, ob eine xPCI-1710 einen Interrupt ausgelöst hat. Wenn ja, gibt den Handle der Karte und die Quelle des Interrupts zurück.

**WICHTIG!**
Diese Funktion kann nur in Visual Basic für DOS und Windows benutzt werden.

**Funktionsaufruf:**
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_ModuleMask;
unsigned char   ul_InterruptMask;
unsigned int    ul_CounterLatchValue;

i_ReturnValue = i_APCI1710_TestInterrupt    (&b_BoardHandle,
                                             &b_ModuleMask
                                             &ul_InterruptMask,
                                             &ul_CounterLatchValue;
```

**Return-Wert:**
-1: Kein Interrupt
> 0: IRQ Nummer

## 12) i_APCI1710_ResetBoardIntRoutine (..)

**Syntax:**
<Return-Wert> = i_APCI1710_ResetBoardIntRoutine
(BYTE b_BoardHandle)

**Parameter:**
- Eingabe:
  BYTE b_BoardHandle Handle der Karte
- Ausgabe:
  Es erfolgt keine Ausgabe

**Aufgabe:**
Stoppt die Interruptverwaltung der xPCI-1710.
Deinstalliert die Interruptroutine, falls die Interruptverwaltung aller xPCI-1710 gestoppt ist.

**Funktionsaufruf:**
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
i_ReturnValue = i_APCI1710_ResetBoardIntRoutine      (b_BoardHandle);
```

**Return-Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch
-2: Keine Interrupt installiert mit der Funktion
"i_APCI1710_SetBoardIntRoutineXXX"

## 10.2.3 Initialisierung Eingangsfilter

### 13) i_APCI1710_InitInputFilter (..)

**Syntax:**
<Return Wert> = i_APCI1710_InitInputFilter
(BYTE b_BoardHandle,
BYTE b_Modul,
BYTE b_TimeBase,
BYTE b_Filter)

**Parameter:**
**- Eingabe:**

| | | |
|---|---|---|
| BYTE | b_BoardHandle | Handle der Karte x**PCI-1710** |
| BYTE | b_ModulNbr | Nummer des zu konfigurierenden Moduls (0 bis 3) |
| BYTE | b_TimeBase | Auswahl des Filter-Clock<br>- APCI1710_30 MHZ: Der PC hat einen PCI Bustakt von 30 MHz<br>- APCI1710_33MHZ: Der PC hat einen PCI Bustakt von 33 MHz<br>- APCI1710_40 MHz: Die Karte hat einen 40 MHz Quarz |
| BYTE | b_Filter | Auswahl des Filters. Siehe |

Tabelle 10-4.

**- Ausgabe:**
Es erfolgt keine Ausgabe.

**Aufgabe:**
Deaktiviert oder aktiviert den Filter im ausgewählten Modul (b_Modul).
b_Filter gibt die Filterzeit an.

**Tabelle 10-4: Filterzeit**

| *b_TimeBase*<br>*b_Filter* | APCI1710_30MHZ | APCI1710_33MHZ | APCI1710_40MHZ |
|---|---|---|---|
| 0 | Filter nicht benutzt | Filter nicht benutzt | Filter nicht benutzt |
| 1 | 133 ns | 121 ns | 100 ns |
| 2 | 200 ns | 182 ns | 150 ns |
| 3 | 267 ns | 242 ns | 200 ns |
| 4 | 333 ns | 303 ns | 250 ns |
| 5 | 400 ns | 364 ns | 300 ns |
| 6 | 467 ns | 424 ns | 350 ns |
| 7 | 533 ns | 485 ns | 400 ns |
| 8 | 600 ns | 545 ns | 450 ns |
| 9 | 667 ns | 606 ns | 500 ns |
| 10 | 733 ns | 667 ns | 550 ns |
| 11 | 800 ns | 727 ns | 600 ns |
| 12 | 867 ns | 788 ns | 650 ns |
| 13 | 933 ns | 848 ns | 700 ns |
| 14 | 1000 ns | 909 ns | 750 ns |
| 15 | 1067 ns | 970 ns | 800 ns |

**WICHTIG!**
Diese Funktion ist nur für folgende Module vorhanden:
Chronos, Impulszähler, Inkrementalzähler

**Funktionsaufruf:**
ANSI C :

```
int                              i_ReturnValue;
unsigned char                    b_BoardHandle;
i_ReturnValue = i_APCI1710_InitInputFilter
                                 (b_BoardHandle
                                  0,
                                  APCI1710_40MHz,
                                  9);
```

**Return Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul hat kein Input Filter.
-4: Der ausgewählte Filter-Clock ist falsch.
-5: Die ausgewählte Filterzeit ist falsch.
-6: Auf der Karte ist kein 40MHz Quarz vorhanden.

### 14) i_APCI1710_CheckInputFilter40MHzStatus (..)

**Syntax:**
<Return Wert> = i_APCI1710_CheckInputFilter40MHzStatus
(BYTE b_BoardHandle,
PBYTE pb_40MHz_Status)

**Parameter:**
**- Eingabe:**
BYTE b_BoardHandle Handle der Karte x**PCI-1710**

**- Ausgabe:**
PBYTE pb_40MHz_Status Verfügbarkeit 40MHz auf der Karte
0: 40MHz nicht vorhanden
1: 40MHz vorhanden

**Aufgabe:**
Prüft die Verfügbarkeit des 40 MHz Taktes auf der Karte.

**WICHTIG!**
Diese Funktion ist nur für die folgenden Module vorhanden:
Chronos, Impulszähler, Inkrementalzähler

**Funktionsaufruf:**
ANSI C :

```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;
unsigned char       b_40MHz_Status;

i_ReturnValue = i_APCI1710_CheckInputFilter40MHzStatus
                    (b_BoardHandle
                     &b_40MHz_Status);
```

**Return Wert:**
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die Funktion ist in keinem der vier Funktionsmodule verfügbar