DIN EN ISO 9001:2000
zertifiziert

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

Funktionsbeschreibung

ADDICOUNT APCI-/CPCI-1710

Inkrementalzähler, Impulszähler

6. Ausgabe 12/2004

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 BESTIMMUNGSGEMÄSSE VERWENDUNG

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.

Die Karte **CPCI-1710** eignet sich für den Einbau in einen CompactPCI-System 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-/CPCI-1710** darf nicht als sicherheitsgerichtetes Betriebsmittel (safety related part, SRP) eingesetzt werden.

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

1.3 Technische Dokumentation

Dieses Referenzhandbuch bezieht sich sowohl auf die Karte **APCI-1710** als auch auf die Karte **CPCI-1710/1711**. Bitte vergewissern Sie sich, dass Sie außerdem folgendes bekommen haben:

- Die CD1 "Standard Software Drivers" mit dem ADDISET Parametrierprogramm und den benötigten Softwaretreibern.
- Die CD2 "Technical Manuals". Die CD enthält
  - das Handbuch **ADDICOUNT APCI-/CPCI-1710: Funktionsprogrammierbare Zählerkarte für den PCI-Bus**, das allgemeine Informationen für den Betrieb der Karte enthält,
  - ein Referenzhandbuch für jede Funktion, die Sie auf die APCI-/CPCI-1710 programmieren wollen,
- das gelbe Blatt mit den Sicherheitshinweisen.

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

Tabelle 1-1: 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	ttl_io.cfg
Digitale E/A	dig_EA_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

Bitte beachten:

Die Karte **CPCI-1710/1711** ist mit der Karte **APCI-1710** kompatibel, was die Softwareinstallation anbelangt. Die Programme ADDIREG und SET1710 machen keinen Unterschied zwischen PCI-Karten und CompactPCI-Karten.

Die API-Funktionen der Standardsoftware sind ebenfalls identisch.

1.4 Funktionsbeschreibung

Dieses Handbuch enthält neben einer globalen Beschreibung der Funktionen
- die Pinbelegung des Frontsteckers,
- eine Liste der benutzten Signale,
- den E/A-Bereich,
- ein Kapitel über die mitgelieferten API-Funktionen der Standardsoftware.

1.5 Schriftvereinbarung

Die Signale auf dem 50poligen SUB-D Stecker sind alle auf ein Funktionsmodul bezogen. Bitte beachten Sie die folgenden Schriftvereinbarungen:
- UAS: Störungssignal
- CLK: Takt
- REF: Referenzpunkt-Logik
- ENA: Enable

C**1**+ ist ein Signal für das **Funktionsmodul 1.**

2 INKREMENTALZÄHLER

2.1 Funktionsbeschreibung

Die Funktion INKREMENTALER ZÄHLER ist ein schneller Zähler für 90°-phasenverschobene Signale (Wegmesssysteme).

Er eignet sich besonders für Anwendungen, in denen Zuverlässigkeit und Robustheit in industrieller Umgebung erforderlich sind.

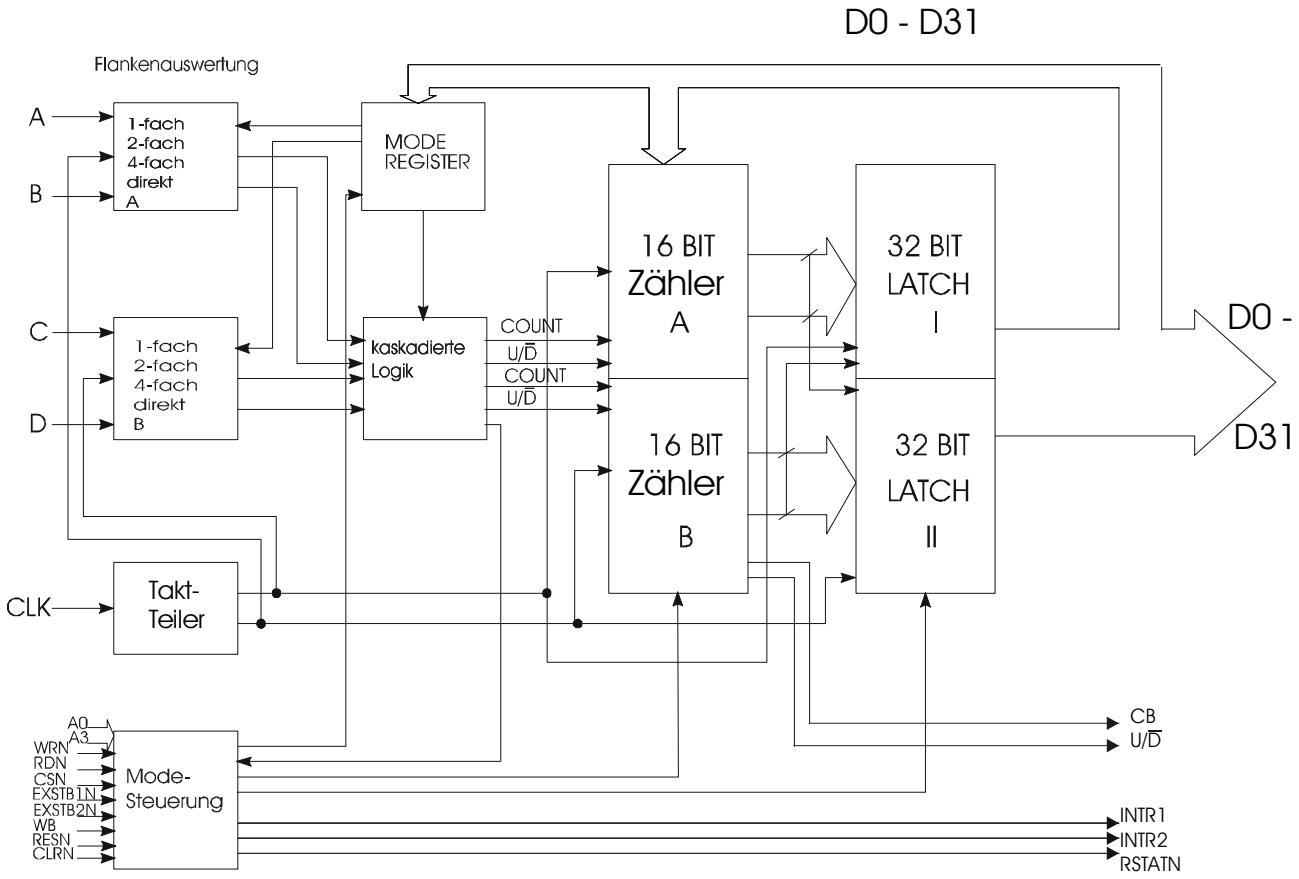
Eigenschaften:
- Anschluss von 1 oder 2 Inkrementalgeber(n), folgendermaßen konfigurierbar:
  - 1 Zähler-Kanal mit einer Zähltiefe von 32 Bit, für TTL oder differentielle Inkrementalgeber am SUB-D Frontstecker
  - 2 Zähler-Kanäle mit einer Zähltiefe von 16 Bit für TTL oder differentielle Inkrementalgeber am SUB-D Stecker (Für diese Konfiguration ist die Index-Logik nicht möglich).
- 1 "INDEX" Eingang für Referenz Punkt Logik
- 1 "UAS" Eingang, der als Störungseingang benutzt werden kann
- 1 "REF" Eingang als normaler digitaler Eingang oder für Referenz Punkt Logik
- 2 "EXTSTB" Eingänge, die das Latchen der Zählerwerte erlauben,
- Hohe Zählgeschwindigkeit
- Interruptanzeige, getriggert über die externen Strobe-Eingänge
- Integrierter Testbetrieb
- Pulsbreitenmessung über Direkteingänge, Zählrichtung programmierbar
- Impulszählung über Direkteingäne, Zählrichtung programmierbar

Funktionsumfang des Zähler Bausteins:
- 4fach- / 2fach- / 1fach-Auswertung zweier verschobener Takten. Weiterverarbeitung in 32-Bit-ladbarem Aufwärts- bzw. Abwärtszähler (oder je einem 16-Bit-Zähler)
- Richtungserkennung für Aufwärts- bzw. Abwärtszählen
- Hystereseschaltung zur Unterdrückung des ersten Pulses nach Drehrichtungsumkehr, abschaltbar
- Zwei 32-Bit-Datenlatche, getrennt programmierbar für internen/externen Strobe, Latch-Strobe synchronisiert mit internem Takt
- Arbeitsmodedefinition über internes MODE-Register, ladbar / lesbar über Datenbus
- Strobe-Eingänge, wahlweise über 2 externe Pins oder über Registerbeschreibung triggerbar

2.1.1 Blockdiagramm des Inkrementalzählers

Der Zählerbaustein enthält:
- je zwei voneinander unabhängige, intern kaskadierbare 16-Bit-Zähler,
- 4fach-/2fach-/1fach-Flankenauswerteschaltungen für zwei phasenverschobene Takte,
- Richtungsdiskriminatoren,
- Hystereseschaltungen,
- zwei 32-Bit-Latche
- sowie eine Funktions- und Kontrollogik.

Abb. 2-1: Blockschaltbild des Zählers



2.1.2 Typische Anwendungen

- Erfassung von inkrementalen Wegmesssystemen
- X-,Y-,Z-Steuerungen
- Pulsbreiten- und Frequenzmessungen
- Inkrementalgeberauswertung
- Geschwindigkeitsmessung
- Drehzahlmessung
- Toleranzmessungen
- Elektronische "Maus"

2.2 Benutzte Signale

Die Funktion "Inkrementalzähler" belegt **7 Eingänge (Kanal A bis G) und 1 Ausgang (Kanal H)** von dem entsprechenden Funktionsmodul der **APCI-/CPCI-1710**.

Auf einer Karte können Sie maximal 4 x 32-bit oder 8 x 16-Bit Inkrementalgeber anschließen.

Tabelle 2-1: Benutzte Signale

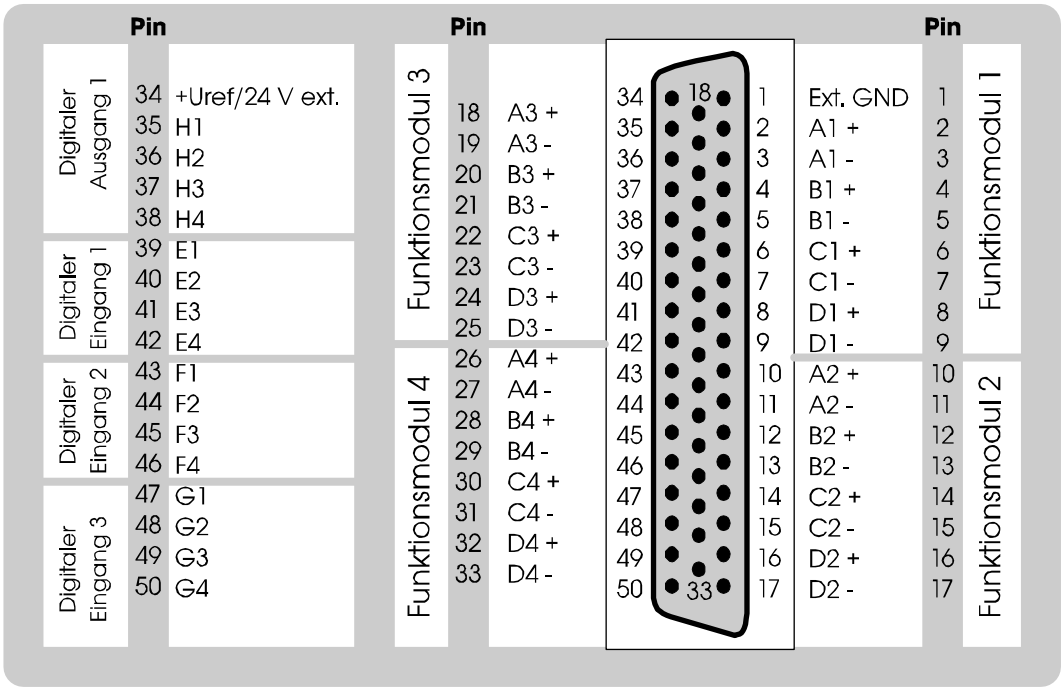
SIGNALE	AM STECKER	POLARITÄT	FUNKTION
A	Ax +/ -	Diff./TTL, optional 24V	Spur A des 1. inkrementalen Gebers
B	Bx +/ -	Diff./TTL, optional 24V	Spur B des 1. inkrementalen Gebers
INDEX	Cx + / -	Diff./TTL, optional 24V	INDEX Spur des inkrementalen Gebers wenn im 32-Bit Mode.
C			Spur A des 2. inkrementalen Gebers im 16-Bit Mode.
AUS	Dx + / -	Diff./TTL, optional 24V	Störungssignal im 32-Bit Mode.
D			Spur B des 2. inkrementalen Gebers im 16-Bit Mode
REF	Ex	24V, optional 5V	Normaler digitaler Eingang, kann über das Register eingelesen werden; kann bei der Referenzpunkt-Logik mit einwirken
ExtStrb_a	Fx	24V, optional 5V Aktiv High	Digitaler Eingang, der das Latchen des Zähler-Inhaltes in das erste Latchregister bewirkt; kann auch einen Interrupt auslösen.
ExtStrb_b	Gx	24V, optional 5V Aktiv High	Digitaler Eingang, der das Latchen des Zähler-Inhaltes in das zweite Latchregister bewirkt; kann auch einen Interrupt auslösen.
Ausgang	Hx		Normaler digitaler Ausgang

x: Nummer des Funktionsmoduls

2.3 Pinbelegung des Inkrementalzähler

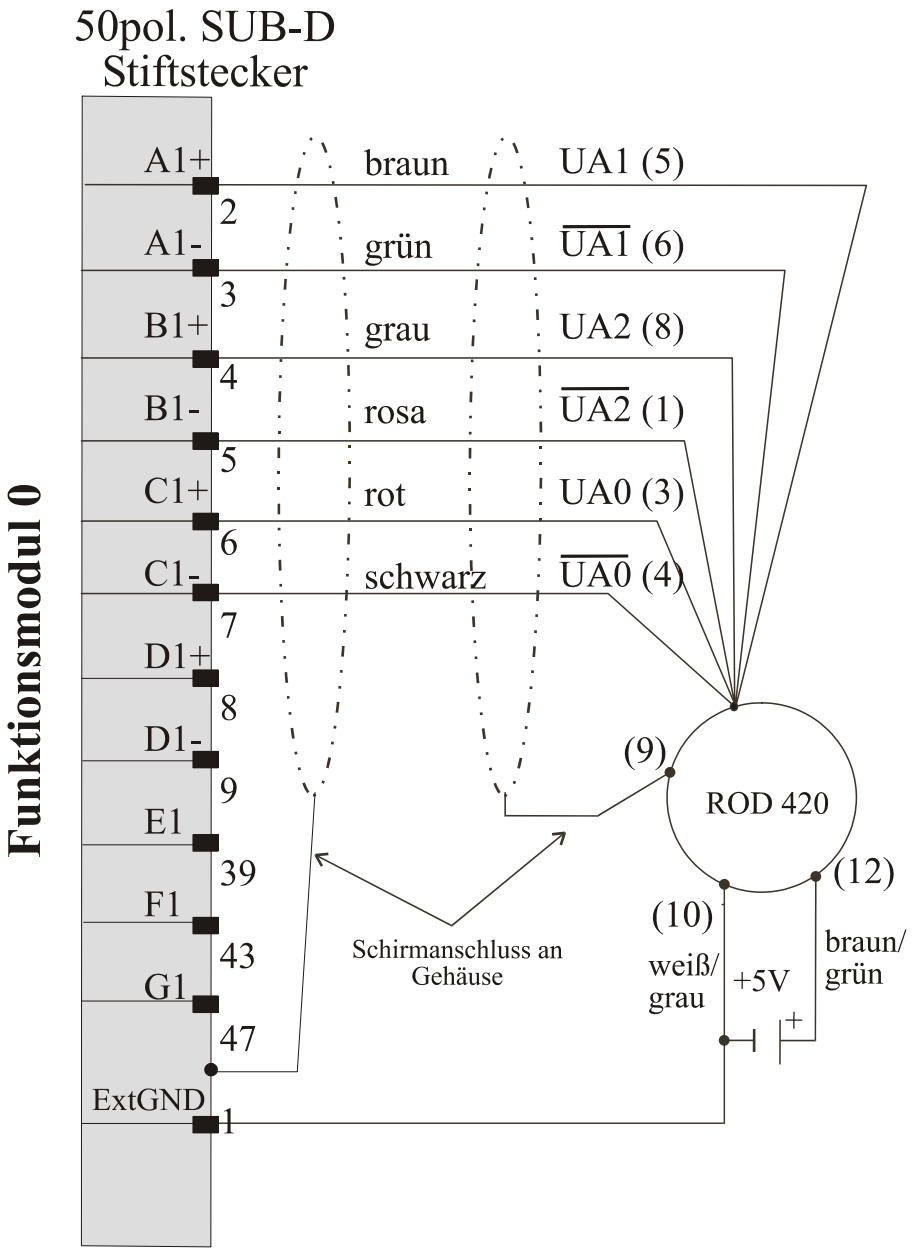
Die untere Abblidung ist ein Anschlussbeispiel: Die Funktion "Inkrementaler Zähler" ist auf allen Funktionsmodulen implementiert.

Abb. 2-2: Pinbelegung des Fronsteckers



2.4 Anschlussbeispiel: Drehgeber ROD 420

Der Drehgeber ROD 420 von HEIDENHAIN ist an Funktionsmodul 1 der **APCI-/CPCI-1710** angeschlossen.
Differenzsignale für inkrementale Information und Referenzsignale.



2.5 E/A Adressbelegung

	IORD			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE**x** + 0	y	y	Y	INT-REG
BASE**x** + 4	LAT1.3	LAT1.2	LAT1.1	LAT1.0
BASE**x** + 8	LAT2.3	LAT2.2	LAT2.1	LAT2.0
BASE**x** + 12	y	y	Y	INDEX-REG
BASE**x** + 16	y	y	Y	OVERFLOW-REG
BASE**x** + 20	y	y	MODREG2	MODREG1
BASE**x** + 24	y	y	Y	STATUS-REG
.....				
BASE**x** + 60	FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

-: keine Funktion ; y: keine relevanten Daten ,
x: Nummer des Funktionsmoduls.

	IOWR			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE**x** + 0	-	-	-	STB-REG
BASE**x** + 4	COUNT.3	COUNT.2	COUNT.1	COUNT.0
BASE**x** + 8	y	y	COUNT.1	COUNT.0
BASE**x** + 12	COUNT.3	COUNT.2	Y	y
BASE**x** + 16	-	-	-	TEST-REG
BASE**x** + 20	-	-	MODREG2	MODREG1
.....	-	-	-	-
BASE**x** + 60	-	-	-	-

-: keine Funktion ; y: keine relevanten Daten ,
x: Nummer des Funktionsmoduls.

Im 32-Bit Mode wird der Zähler (Ax und Bx) über die Adresse BASEx+4 geladen.

Im 16-Bit Mode:
- wird der erste Zähler (Ax und Bx) über die Adresse BASEx+8 geladen
- wird der zweite Zähler (Cx und Dx) über die Adresse BASEx+12 geladen

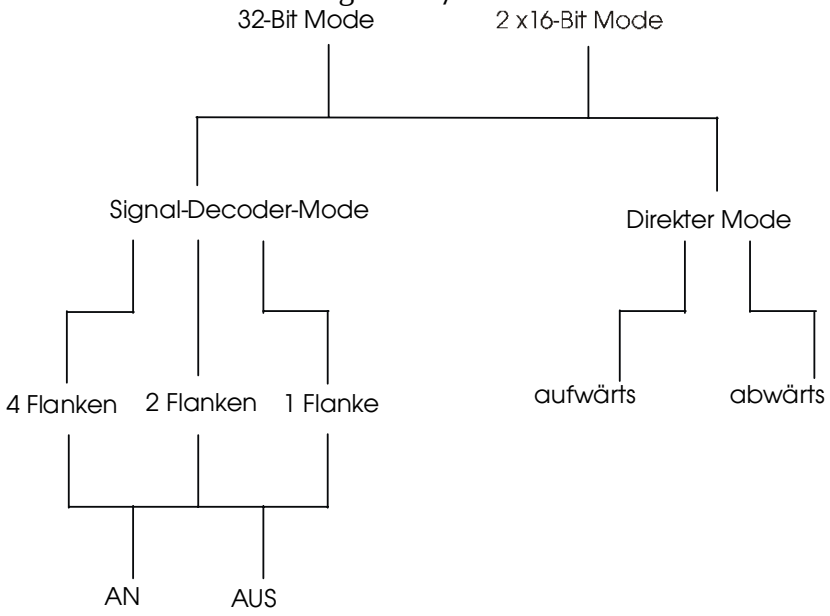
Die inkrementale Zählerfunktion belegt 7 DWORDS im E/A-Bereich des Funktionsmoduls **x**.

2.6 Beschreibung der E/A Funktionen

Allgemeines

Die Zählerfunktion wird durch das **MOD-REG1** (**Base + 20**) bestimmt. Das Register ist beschreibbar und auslesbar.

Folgende Arbeitsmodes sind möglich: Hysterese

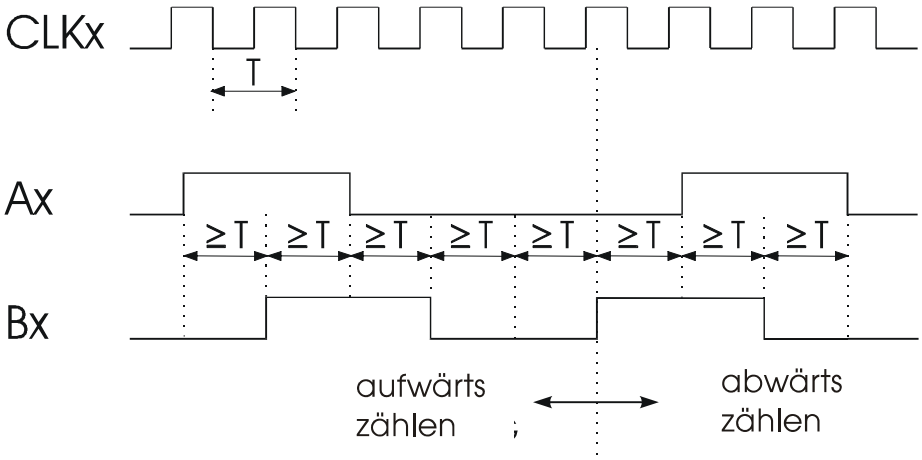


Eingänge für Flankenauswerteschaltung

A, B Eingänge für phasenverschobene Signale bei 4fach-/2fach-/1fach MODE
32-Bit-Zähler bzw. 16-Bit-Zähler A.
Signalwechsel A vor Signalwechsel B bedeutet Aufwärtszählen.
Signalwechsel B vor Signalwechsel A bedeutet Abwärtszählen.

C, D Eingänge für phasenverschobene Signale für 16-Bit-Zähler B.
Signalwechsel C vor Signalwechsel D bedeutet Aufwärtszählen.
Signalwechsel D vor Signalwechsel C bedeutet Abwärtszählen.

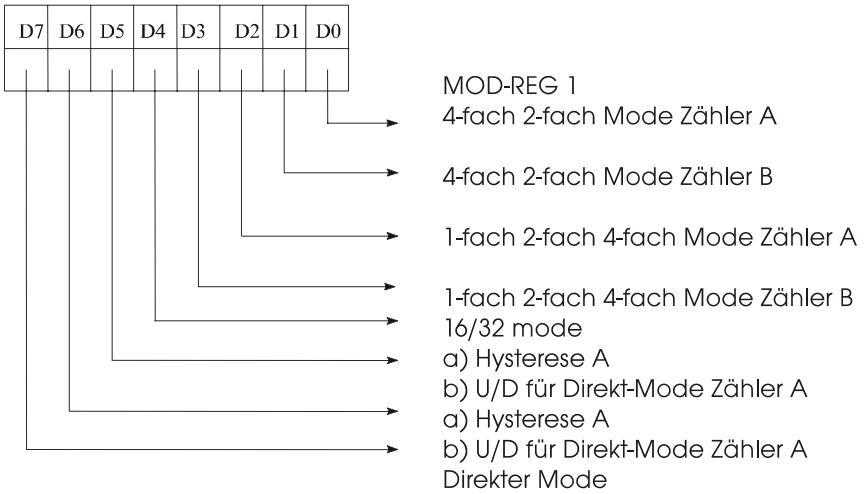
Die Flankenauswerteschaltung benötigt 3 Taktperioden zur Auswertung der Flanken bzw. der Richtungserkennung. Mit dem 4. Takt wird gezählt (gilt für alle Betriebsmodi, bezogen auf die externe negative Taktflanke). Der Takt ist gleich mit dem PCI Takt d.h. ≤ 33MHz.



Aus der Abbildung ist ersichtlich, dass keine bestimmten Phasenverschiebungen zwischen den Signalen A, B und C, D notwendig sind. Es müssen nur die zeitlich auf den Systemtakt bezogenen Mindestwerte eingehalten werden (minimaler Flankenabstand).

2.6.1 MODE-Register 1 (Base + 20)

Abb. 2-3: MODE-Register 1



Für den 1fach-MODE muss auch der 2fach-MODE programmiert werden, anderenfalls wird intern die Flankenauswertung des jeweiligen Steuerwerks (A/B) deaktiviert. Unabhängig von den Auswerteeingängen erfolgt dann keine Zählung mehr, und der letzte Zählerstand bleibt erhalten.

Der 16/32-bit-MODE wird durch das Datenbit D4 festgelegt.

MODE Register 1	Bit	7	6	5	4	3	2	1	0
32-Bit-MODE		X	X	X	0	X	X	X	X
16-Bit-MODE		X	X	X	1	X	X	X	X

Im 32-Bit-MODE wird entweder die 4fach-/2fach-/1fach-Flankenauswerteschaltung A oder der DIREKT-MODE verwendet.
CBX ist der CARRY/BORROW-Ausgang über 32 Bit

An U/D steht das in der Flankenauswerteschaltung A erkannte Aufwärts/Abwärts-Signal zur Verfügung.

Im 16-Bit-MODE wird
- entweder die 4fach-/2fach-/1fach-Flankenauswerteschaltung A für Zähler A und die 4fach-/2fach-/1fach-Flankenauswerteschaltung B für Zähler B
- oder DIREKT-MODE (Eingänge A, B für Zähler A, Eingänge C, D für Zähler B) verwendet.

CBX ist der CARRY/BORROW-Ausgang über 16 Bit, Zähler B.

An U/D steht das in der Flankenauswerteschaltung B erkannte Aufwärts/Abwärts-Signal zur Verfügung.

Der 4fach-/2fach-/1fach-MODE wird durch die Datenbits D0-D3 im MODE-Register 1 festgelegt.

L an D0	4fach-MODE für Flankenauswerteschaltung A
L an D1	4fach-MODE für Flankenauswerteschaltung B
H an D0	2fach-MODE für Flankenauswerteschaltung A
H an D1	2fach-MODE für Flankenauswerteschaltung B
L an D2, D3	4fach- oder 2fach MODE für Flankenauswerteschaltung A, B abhängig von D0, D1
H an D0 und D2, D1 und D3	1fach-MODE für Flankenauswerteschaltung A,B
L an D0 und D1, H an D2 und D3	Flankenauswerteschaltung A, B inaktiv

L: Low; **H**: High

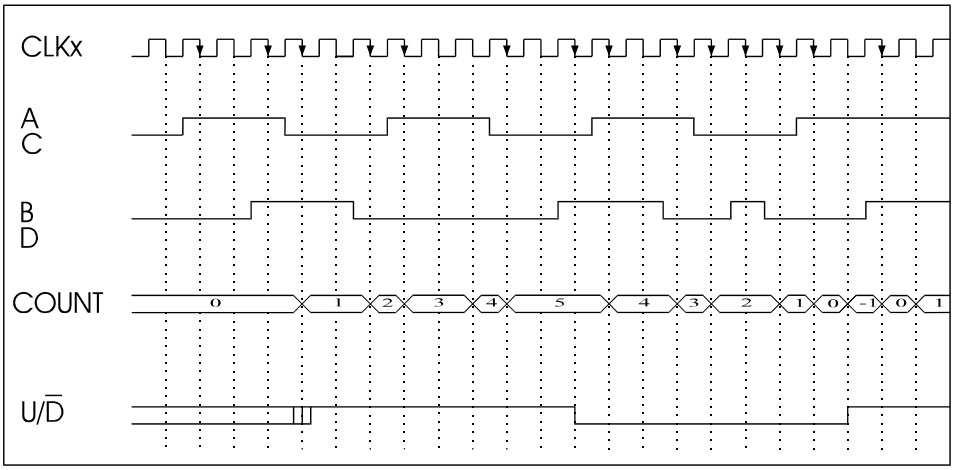
4fach-MODE

Die Flankenauswerteschaltung generiert im 4fach-MODE aus jeder Flanke zweier zueinander phasenverschobener Signale einen Zählimpuls. Diese Signale müssen im 32-Bit-MODE an die Eingänge A und B angelegt werden. Im 16-Bit-MODE stehen zwei Flankenauswerteschaltungen für die Eingänge A und B bzw. C und D zur Verfügung.

Die Signale A, B und C, D werden von dem gemeinsamen Takt CLKX abgetastet und zwischengespeichert. Dieser Takt muss mindestens die 4fache Frequenz der Signale A, B, C, D aufweisen.

Die Zählrichtung wird durch die Flankenauswerteschaltung aus der Phasenverschiebung der Signale A zu B bzw. C zu D erkannt.

Abb. 2-4: Flankenauswerteschaltung (4fach-MODE)



Abtasten (A, B, C, D) mit CLKX↓
Zählen (COUNT) mit CLKX↓

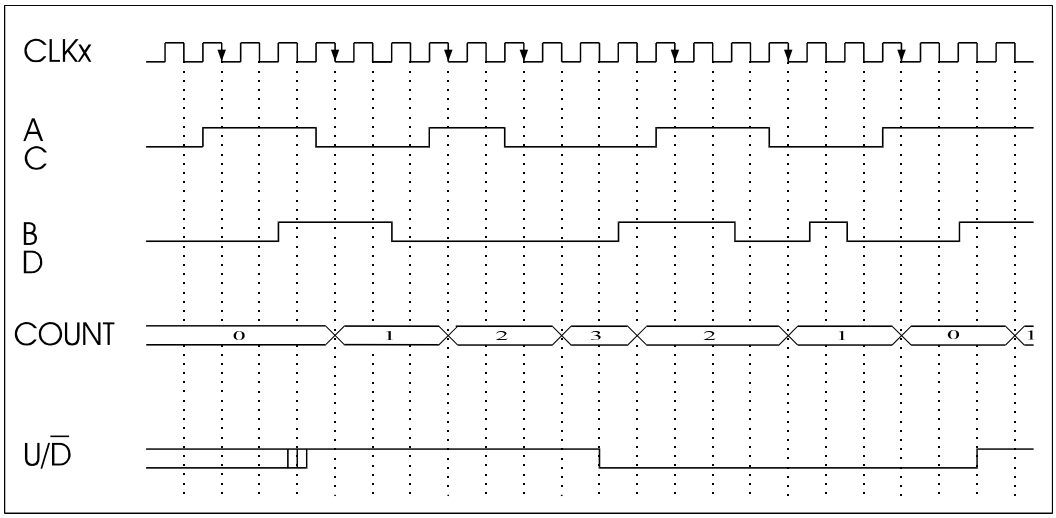
MODE Register 1	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
32-Bit-MODE		0	X	0	0	X	0	X	0
16-Bit-MODE		0	0	0	1	0	0	0	0

Hysterese jeweils ausgeschaltet

2fach-MODE

Funktion wie 4fach-MODE; Es werden jedoch nur zwei der insgesamt vier Flanken pro Periode ausgewertet.

Abb. 2-5: Flankenauswerteschaltung (2fach-MODE)



Abtasten (A, B, C, D) mit CLKX↓
Zählen (COUNT) mit CLKX↓

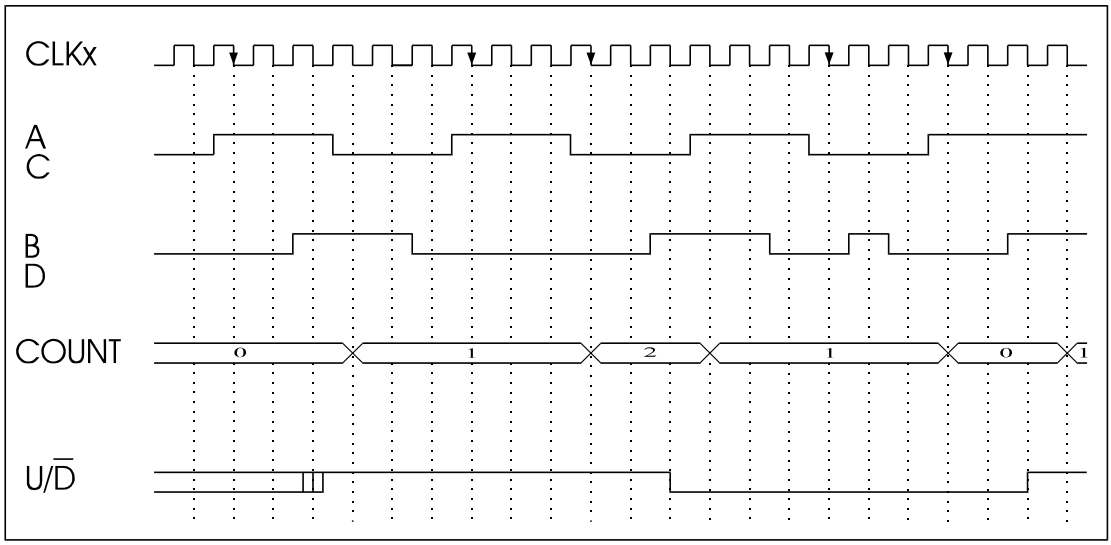
MODE Register 1	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
32-Bit-MODE		0	X	0	0	X	0	X	1
16-Bit-MODE		0	0	0	1	0	0	1	1

Hysterese jeweils ausgeschaltet

1fach-MODE

Funktion wie 4fach-MODE; Es wird jedoch nur eine der insgesamt vier Flanken pro Periode ausgewertet.

Abb. 2-6: Flankenauswerteschaltung (1fach-MODE)



Abtasten (A, B, C, D) mit CLKX↓
Zählen (COUNT) mit CLKX↓

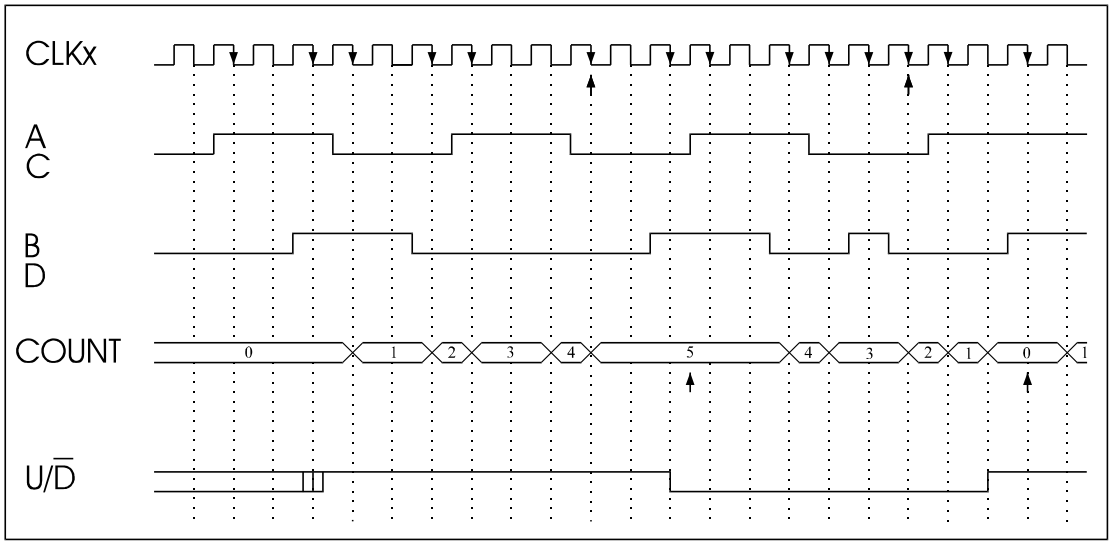
MODE Register 1	Bit	7	6	5	4	3	2	1	0
32-Bit-MODE		0	X	0	0	X	1	X	1
16-Bit-MODE		0	0	0	1	1	1	1	1

Hysterese jeweils ausgeschaltet

Hysterese

In beiden Flankenauswerteschaltungen steht eine Hysterese-Schaltung zur Verfügung. Sie unterdrückt den jeweils ersten Zählimpuls nach einer Drehrichtungsumkehr.

Abb. 2-7: Hysterese-Flankenauswerteschaltung (Beispiel des 4fach-MODEs)



Abtasten (A, B, C, D) mit CLKX↓
Zählen (COUNT) mit CLKX↓

Drehrichtungsumkehr aufwärts/abwärts/aufwärts Hysterese eingeschaltet

4-fach-MODE mit Hysterese:

MODE Register 1	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
32-Bit-MODE		0	X	1	0	X	0	X	0
16-Bit-MODE		0	1	1	1	0	0	0	0

Direkt-MODE

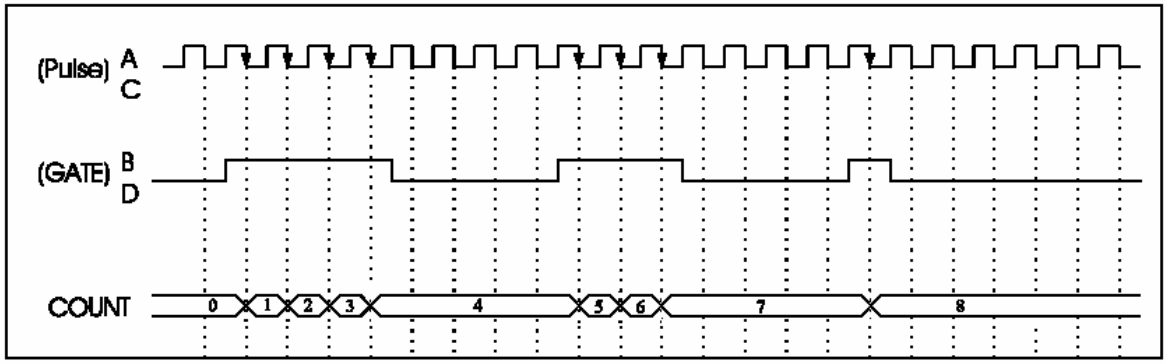
Im DIREKT-MODE werden die beiden Flankenauswerteschaltungen inaktiv.

Die Eingänge A, B im 32-Bit-MODE bzw. A, B und C, D im 16-Bit-MODE stellen je eine Takt-Torschaltung dar. Dabei können Frequenz- und Impulsdauermessungen durchgeführt werden.

Der 32-Bit-Zähler bzw. die beiden 16-Bit-Zähler können unabhängig voneinander über Bit D5 bzw. Bit D6 des MODE1-Registers auf Abwärts- oder Aufwärtszählen eingestellt werden.
Der Toreingang wird mit der fallenden Flanke von A bzw. C synchronisiert.

Wenn der TOR Eingang immer auf High Pegel gelegt wird, dann wird der Zähler die Impulse an dem Eingang A bzw. C zählen. Dies ist der Fall, wenn die differentiellen Eingänge B bzw. D offen sind.

Abb. 2-8: Impulsdiagramm der Torzeitmessung



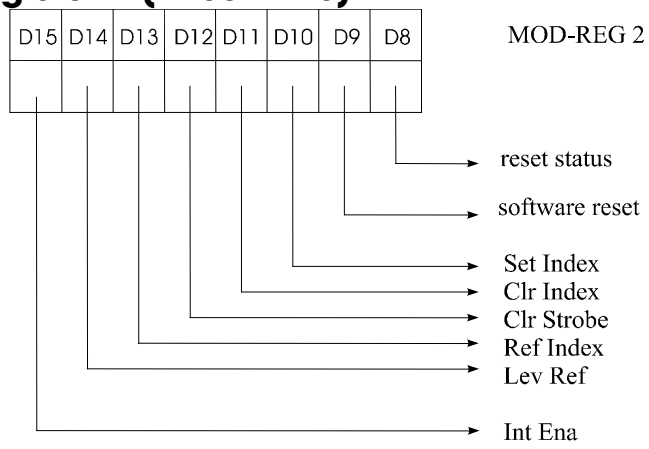
Abtasten (B, D) mit A↓, B↓ (CLK = 0)
CLK ≅ CLKX

MODE Register 1	Bit	7	6	5	4	3	2	1	0
32-Bit-MODE		1	X	1	0	X	X	X	X
16-Bit-MODE		1	0	1	1	X	X	X	X

Zähler A zählt aufwärts, Zahler B abwärts (16-Bit-MODE).

WARNUNG!
Ein gemischter Betrieb, d. h. Zähler A in 4fach-/2fach-/1fach-MODE, Zähler B im DIREKT-MODE, **ist nicht möglich!**

2.6.2 MODE-Register 2 (Base + 20)



Das MODE-Register 2 dient zur Auslösung und zur Anzeige des RESET-Zustandes des Bausteins (siehe auch Reset-Logik).

Bit 8: **RESET-STATUS**
0: wenn externer Reset erfolgt ist.
1: inaktiv (ohne Bedeutung)

Bit 9: **SOFTWARE-RESET**
1: internen Resetvorgang starten.
0: inaktiv (automatisches Rücksetzen nach Resetablauf)

Bit 10: **SET-INDX**
1: Löschen oder Strobe des Zählerinhaltes beim Referenz-Punkt.
0: inaktiv (automatisches Rücksetzen oder nicht nach Ablauf)

Bit 11: **CLR-INDX**
1: Das SET-INDX wird automatisch zurückgesetzt nach Ablauf.
0: inaktiv: SET-INDX wird nicht zurückgesetzt.

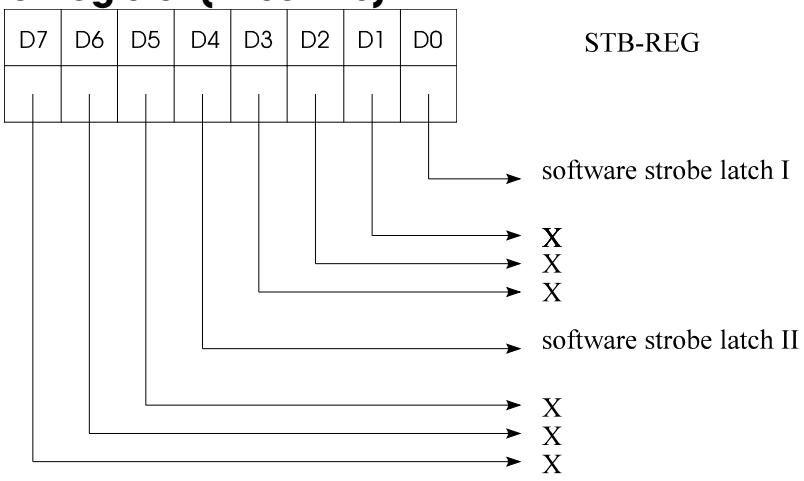
Bit 12: **CLR-STRB**
1: Zählerstand wird in das Latchregister1 geschrieben beim Erreichen des Referenz-Punkts.
0: inaktiv; Zähler wird beim Erreichen des Referenz-Punkts gelöscht.

Bit 13: **REF-INDX**
1: Der REF Eingang hat eine Auswirkung auf die Referenz-Punkt-Logik. (INDEX muss auf "1" sein **UND** der entsprechende Pegel auf REF damit die Referenz-Punkt-Logik erkannt wird.
0: inaktiv: Der REF Eingang hat keine Auswirkung auf die Referenz-Punkt-Logik.

Bit 14: **LEV-REF**
1: REF muss auf "0" sein.
0: inaktiv: REF muss auf "1" sein.

Bit 15 **INT-ENA**
1: Strobe Interrupt wird durchgeschleift.
0: (Reset) Interrupt wird gesperrt.

2.6.3 Strobe-Register (Base + 0)



Ein Software-Strobe wird durch Beschreiben der entsprechenden Datenbits des Strobe-Registers erzeugt. Der Strobe-Zyklus wird mit dem Wert 1 des betreffenden Datenbits ausgelöst. Der Wert 0 hat keine Bedeutung.

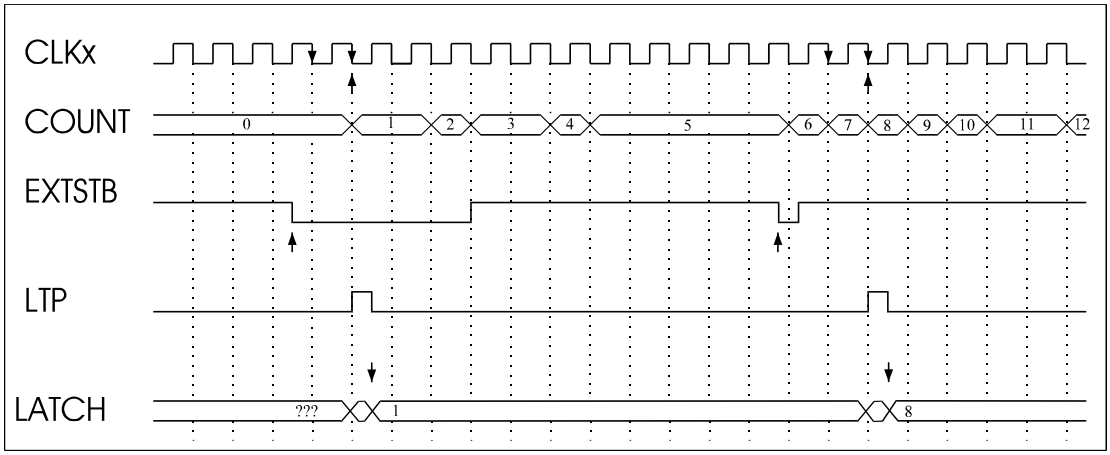
Ein Hardware-Strobe wird über die beiden Pins F (ExtStb_a, low aktiv für Latch I) und G (ExtStb_b, low aktiv für Latch II) ausgelöst.

Das Strobe-Register ist nur beschreibbar.

Latch-Logik

Intern sind zwei gleichwertige 32-Bit-Auffangregister aufgebaut, die jeweils über einen eigenen Softwarestrobe und/oder External-Strobeeingang zum Latchen des aktuellen 32-Bit-Zählerstandes veranlaßt werden. Sämtliche Strobesignale werden mit eventuell gleichzeitig intern auftretenden Zählimpulsen synchronisiert, so dass keine Zwischenzustände gespeichert werden. Durch die Einsynchronisierung wird der Zählerstand nach 2 Takten Versatz gespeichert.

Abb. 2-9: Impulsdiagramm Latch-Logik

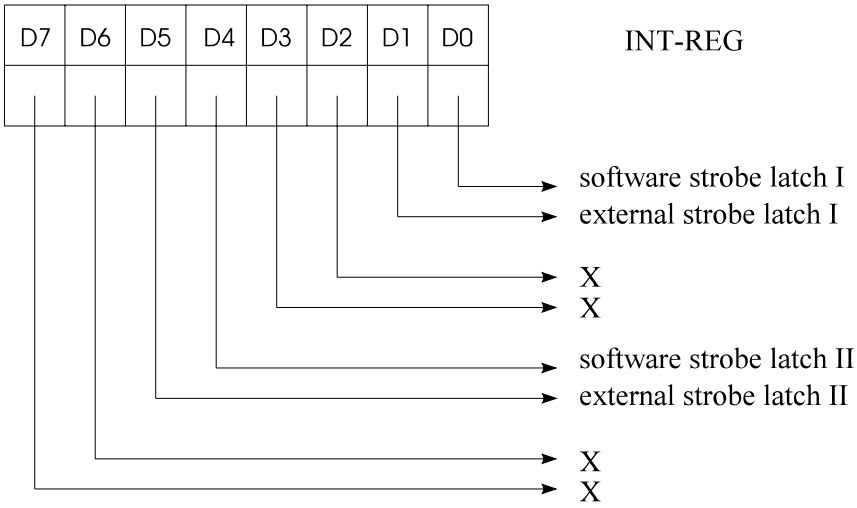


LTP: interner Latchpuls
Zählen (COUNT) mit CLKX↓

Selektierung: (Siehe auch E/A Adressbelegung 16-Bit)

LAT1.0 → BIT 0-7 LATCH I	LAT2.0 → BIT 0-7 LATCH II
LAT1.1 → BIT 8-15 LATCH I	LAT2.1 → BIT 8-15 LATCH II
LAT1.2 → BIT 16-23 LATCH I	LAT2.2 → BIT 16-23 LATCH II
LAT1.3 → BIT 24-31 LATCH I	LAT2.3 → BIT 24-31 LATCH II

2.6.4 Interrupt-Register (Base + 0)



Das Interrupt-Register enthält den Interruptstatus und ist nur auslesbar.

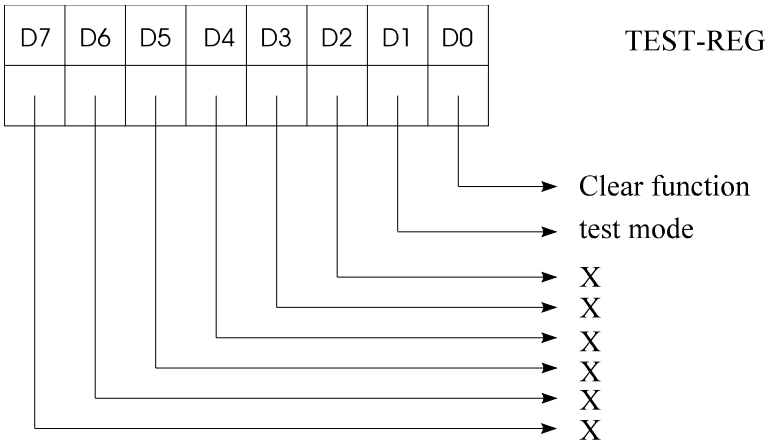
Der Wert 1 zeigt den aktiven Zustand an, wenn ein Wert gelatcht wurde. Dabei wird sowohl nach Latchgruppe als auch nach interner bzw. externer Latchtriggerung unterschieden.

Die Bitpositonen D1 und D5 sind gleichbedeutend mit der Interrupt-Anforderung. Die Bitpositionen D0 und D4 werden nach einem Software-Strobe nur aktiv, wenn intern der Latchvorgang abgeschlossen ist.

Durch einen Auslesezyklus der zugehörigen 32-Bit-Latchgruppe werden die entsprechenden Interruptanzeigen zurückgesetzt.

Hierbei ist jedoch zu beachten, dass ein während des Auslesevorganges erneut ausgelöster Strobevorgang zwar einen neuen Latchimpuls auslöst, dieser jedoch weder im Interrupt-Register noch an den Interrupt angezeigt wird.

2.6.5 Test-Register (Base + 16)



Das Testregister ist nur beschreibbar. Der Wert 1 entspricht dem aktiven Zustand. Bit 0 wird automatisch zurückgesetzt.

Für den Test des Bausteins und der angeschlossenen Peripherie ist ein Test-Mode vorgesehen, in dem intern alle 8-Bit-Zählerketten als Abwärtszähler betrieben werden. Unabhängig von externen Signalen dekrementieren alle 8-bit Zählerketten parallel bei jeder negativen Taktflanke von CLKX.

Um den Gleichlauf der Zählerketten zu erreichen, muss nach Aktivierung des Testbetriebes die CLEAR-Funktion ausgelöst werden. Nach Auslösung von Strobevorgängen über das Strobe-Register (Setzen von Bit 0) bzw. über den externen Pin F (EXSTBx1) zu beliebigen Zeitpunkten müssen alle 4 Bytes einer Latchgruppe beim Auslesen identisch ein.

Wird während des Testbetriebs das MODE-Register 1 ausgelesen, so beträgt dieser Wert unabhängig vom einprogrammierten Wert 10 HEX. Nach Beendigung des Testbetriebs ist der alte unveränderte Wert auslesbar.

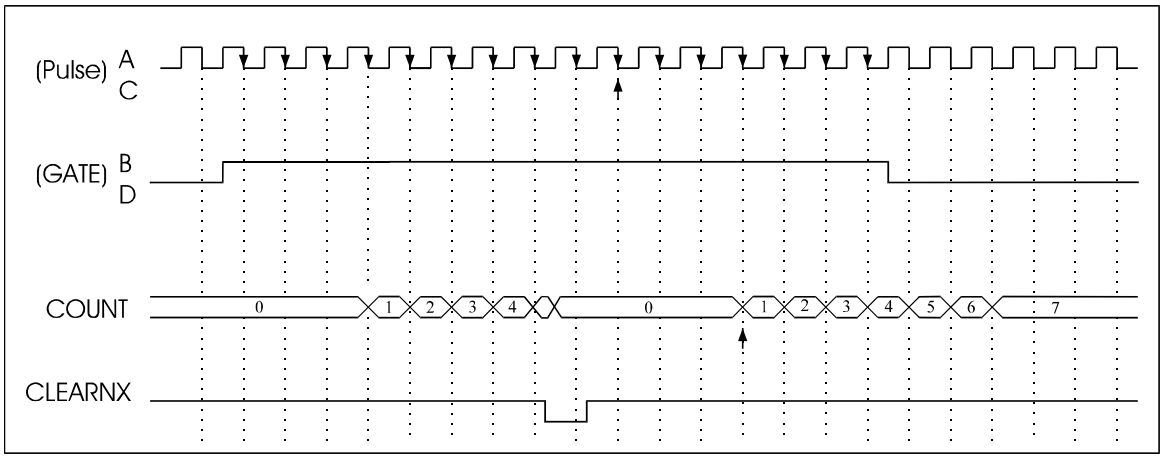
Der Testbetrieb wird verlassen, wenn entweder ein RESET-Vorgang ausgelöst wird oder in Bit 1 des Testregisters der Wert 0 eingeschrieben wird.

Die Clear-Funktion setzt nach der Ausführung das Bit 0 des Test-Registers automatisch zurück. In dieser Bitposition ist also nur ein Schreibzugriff sinnvoll.

Clear-Logik

Bei der Referenz-Punkt Logik besteht die Möglichkeit über ein externes Signal am Eingang löst ein sofortiges Rücksetzen aller Zählerketten aus. Die Wegnahme dieses Signals wird einsynchronisiert.

Abb. 2-10: Impulsdiagramm Clear-Logik



Abtasten (A, D) mit A↓, B↓ (CLK = 0)

Nach der Rückflanke des Clear-Impulses vergehen noch 3 weitere Takte, bis die Zähler wieder freigegeben werden.

Bei einem Soft-Clear (siehe Testregister) beginnt der interne Clearvorgang nach Beendigung des Schreibzyklus (WRNX = High) und dauert 5 negative Taktflanken, d. h., dass ein interner Zählvorgang frühestens mit der 6. negativen Flanke wieder erfolgen kann.

Reset-Logik

Nach erfolgtem Reset wird die Steuerlogik der Schaltung in den Basis-MODE gesetzt:

- 32-Bit-Zähler
- Flankenauswerteschaltung A im 4fach-MODE
- Hysterese aus
- Steuerwerk B inaktiv
- Bit 8/ Mode-Register 2 auf Low
- keine Beeinflussung der Zählerketten

Das Bit 8 von MODE-Register 2 bleibt jedoch solange aktiv, bis dieses Bit durch einen Schreibzyklus vom Prozessor auf 1 gesetzt wird.

Bei einem Software-Reset (siehe MODE-Register 2) beginnt der interne Resetvorgang nach Beendigung des Schreibzyklus (WRNX = High) und dauert 4 negative Taktflanken (CLKX), d. h., Daten vom Steuerwerk können mit der 5. negativen Taktflanke wieder verarbeitet werden.

Laden der Zählerketten (BASE + 4 → BASE + 12)

Der 16-/32-Bit-Zähler kann über den 32-Bit-Datenbus geladen werden.
- Über die Adresse Base + 4 wird der 32-Bit Zähler geladen.
  Über die Adresse Base + 8 wird der 16-Bit Zähler A geladen
  (Bit D0-D15).
- Über die Adresse Base + 12 wird der 16-Bit Zähler B geladen
  (Bit D16-D31).

Selektierung:
- COUNT.0→ BIT 0-7 des Zählers
- COUNT.1→ BIT 8-15 des Zählers
- COUNT.2→ BIT 16-23 des Zählers
- COUNT.3→ BIT 24-31 des Zählers

Zur Vermeidung von Zählfehlern sollte während des Ladens der Zählvorgang unterbrochen werden.

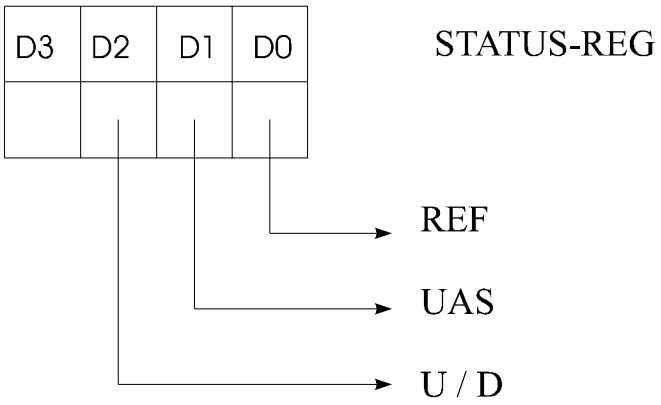
2.6.6 INDEX-Register (Base + 12)

D0 = INDX Bit =	"1":	ein Index ist erfolgt; beim Lesen dieses Wertes wird der Index zurückgesetzt.
	"0":	Kein Index ist erfolgt.

2.6.7 OVERFLOW-Register (Base + 16)

D0 = C/B Bit =	"1":	ein Überlauf bzw. Unterlauf ist erfolgt; beim Lesen dieses Wertes wird das Bit zurückgesetzt
	"0":	kein Überlauf oder Unterlauf ist erfolgt.

2.6.8 STATUS-Register (Base + 24)



STATUS-REG:

D0 = REF Bit = "1": der Eingang REF ist auf logisch "0" gesetzt.
"0: der Eingang REF ist auf logisch "1" gesetzt.

D1 = UAS Bit = "1": der Eingang UAS ist auf logisch "0" gesetzt.
"0: der Eingang UAS ist auf logisch "1" gesetzt.

D2 = U/D Bit = "1": der Zähler zählt aufwärts (Up)
"0: der Zähler zählt abwärts (Down)

2.6.9 Version Register

Auf der Basisadresse Base + 60 erfolgt die Erkennung und die Revision der Funktion in ASCI Format.

Beispiel: BASE + 60 "S" "C" "2" "3"
Bedeutet: Inkrementaler Zähler, Revision 2.3

2.7 Grenzwerte

Zähler-Tiefe: 32-Bit

Maximale Zählfrequenz: ..... 5 MHz

Minimale Phasenverschiebung: ..... 40° bei 5 MHz

Zugriffszeit auf Zähler: ..... 200 ns

3 STANDARDSOFTWARE

3.1 Define-Werte

Tabelle 3-1: Define-Tabelle

Define Name	Dezimal Wert	Hexadezimal Wert
APCI1710_DISABLE	0	0
APCI1710_ENABLE	1	1
APCI1710_16BIT_COUNTER	16	10
APCI1710_32BIT_COUNTER	0	0
APCI1710_QUADRUPLE_MODE	0	0
APCI1710_DOUBLE_MODE	3	3
APCI1710_SIMPLE_MODE	15	F
APCI1710_DIRECT_MODE	128	80
APCI1710_HYSTERESIS_ON	96	60
APCI1710_HYSTERESIS_OFF	0	0
APCI1710_INCREMENT	96	60
APCI1710_DECREMENT	0	0
APCI1710_HIGH_EDGE_CLEAR_COUNTER	0	0
APCI1710_HIGH_EDGE_LATCH_COUNTER	1	1
APCI1710_LOW_EDGE_CLEAR_COUNTER	2	2
APCI1710_LOW_EDGE_LATCH_COUNTER	3	3
APCI1710_HIGH_EDGE_LATCH_AND_CLEAR_COUNTER	4	4
APCI1710_LOW_EDGE_LATCH_AND_CLEAR_COUNTER	5	5
APCI1710_LOW	0	0
APCI1710_HIGH	1	1
APCI1710_SOURCE_0	0	0
APCI1710_SOURCE_1	1	1
APCI1710_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28

3.2 Interruptmaske

Jeder inkrementale Zähler kann einen Interrupt generieren. Um diesen Interrupt zu bekommen, sollen Sie den Interrupt aktivieren und die Interruptroutine mit der Funktion "i_APCI1710_SetBoardIntRoutineX" Funktion.

Tabelle 3-2: Interruptmaske der Funktion "Inkrementaler Zähler"

b_ModuleMask	ul_InterruptMask	Bedeutung
0000 0001	0000 0000 0000 0000 0001	Hardware Latch des 1. Registers, Modul 0 (32-Bit)
0000 0001	0000 0000 0000 0000 0010	Hardware Latch des 2. Registers, Modul 0 (32-Bit)
0000 0001	0000 0000 0000 0000 0100	Index-Interrupt aus Modul 0 (32-Bit)
0000 0001	0000 0000 0000 0000 1000	Vergleich-Interrupt aus Modul 0 (32-Bit)
0000 0001	0001 0000 0000 0000 0000	Interrupt am Ende der Frequenzmessung, Modul 0 (32-Bit)
0000 0010	0000 0000 0000 0000 0001	Hardware Latch des 1. Registers, Modul 1 (32-Bit)
0000 0010	0000 0000 0000 0000 0010	Hardware Latch des 2. Registers, Modul 1 (32-Bit)
0000 0010	0000 0000 0000 0000 0100	Index-Interrupt aus Modul 1 (32-Bit)
0000 0010	0000 0000 0000 0000 1000	vergleich-Interrupt aus Modul 1 (32-Bit)
0000 0010	0001 0000 0000 0000 0000	Interrupt am Ende der Frequenzmessung, Modul 1 (32-Bit)
0000 0100	0000 0000 0000 0000 0001	Hardware Latch des 1. Registers, Modul 2 (32-Bit)
0000 0100	0000 0000 0000 0000 0010	Hardware Latch des 2. Registers, Modul 2 (32-Bit)
0000 0100	0000 0000 0000 0000 0100	Index-Interrupt aus Modul 2 (32-Bit)
0000 0100	0000 0000 0000 0000 1000	vergleich-Interrupt aus Modul 2 (32-Bit)
0000 0100	0001 0000 0000 0000 0000	Interrupt am Ende der Frequenzmessung, Modul 2 (32-Bit)
0000 1000	0000 0000 0000 0000 0001	Hardware Latch des 1. Registers, Modul 3 (32-Bit)
0000 1000	0000 0000 0000 0000 0010	Hardware Latch des 2. Registers, Modul 3 (32-Bit)
0000 1000	0000 0000 0000 0000 0100	Index-Interrupt aus Modul 3 (32-Bit)
0000 1000	0000 0000 0000 0000 1000	vergleich-Interrupt aus Modul 3 (32-Bit)
0000 1000	0001 0000 0000 0000 0000	Interrupt am Ende der Frequenzmessung, Modul 3 (32-Bit)

Tabelle 3-3: Rückgabetabelle für den Zählwert

b_ModuleMask	ul_InterruptMask	Source	ul_CounterLatchValue Wert
b_ModuleMask = 1	ul_InterruptMask = 1	Hardware Strobe auf 1. Latch-Register des Moduls 0	Gelatchter Wert des ersten Latch-Registers (32-Bit)
b_ModuleMask = 1	ul_InterruptMask = 2	Hardware Strobe auf 2. Latch-Register des Moduls 0	Gelatchter Wert des zweiten Latch-Registers (32-Bit)
b_ModuleMask = 2	ul_InterruptMask = 1	Hardware Strobe auf 1. Latch-Register des Moduls 1	Gelatchter Wert des ersten Latch-Registers (32-Bit)
b_ModuleMask = 2	ul_InterruptMask = 2	Hardware Strobe auf 2. Latch-Register des Moduls 1	Gelatchter Wert des zweiten Latch-Registers (32-Bit)
b_ModuleMask = 4	ul_InterruptMask = 1	Hardware Strobe auf 1. Latch-Register des Moduls 2	Gelatchter Wert des ersten Latch-Registers (32-Bit)
b_ModuleMask = 4	ul_InterruptMask = 2	Hardware Strobe auf 2. Latch-Register des Moduls 2	Gelatchter Wert des zweiten Latch-Registers (32-Bit)
b_ModuleMask = 8	ul_InterruptMask = 1	Hardware Strobe auf 1. Latch-Register des Moduls 3	Gelatchter Wert des ersten Latch-Registers (32-Bit)
b_ModuleMask = 8	ul_InterruptMask = 2	Hardware Strobe auf 2. Latch-Register des Moduls 3	Gelatchter Wert des zweiten Latch-Registers (32-Bit)

3.3 Zähler-Initialisierung

1) i_APCI1710_InitCounter(...)

Syntax:
<Return Wert> = i_APCI1710_InitCounter
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_CounterRange,
BYTE b_FirstCounterMode,
BYTE b_FirstCounterOption,
BYTE b_SecondCounterMode,
BYTE b_SecondCounterOption)

Parameter
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710[1]
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_CounterRange	Auswahl des Zählerbereiches. Siehe Tabelle 3-4.
BYTE	b_FirstCounterMode	Betriebsmode des ersten Zählers. Siehe Tabelle 3-5.
BYTE	b_FirstCounterOption	Option des ersten Zählers. Siehe Tabelle 3-6, 3-7.
BYTE	b_SecondCounterMode	Betriebsmode des zweiten Zählers. Siehe Tabelle 3-5.
BYTE	b_SecondCounterOption	Option des zweiten Zählers. Siehe Tabelle 3-6, 3-7.

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Konfiguriert den Zähler-Betriebsmode des ausgewählten Moduls (*b_ModulNbr*).
Diese Funktion ist als erste aufzurufen, bevor Sie die anderen Funktionen aufrufen, die auf die Zähler zugreifen.

WICHTIG!
Wenn Sie das Modul für zwei 16-Bit Zähler konfiguriert haben, ist ein **gemischter Betrieb mit einem Zähler im 4fach-/2fach-/1fach-Mode und dem anderen Zähler im Direkt-Mode nicht möglich.**

[1] Gemeinsame Bezeichnung für APCI-1710 und CPCI-1710

Tabelle 3-4: Zählerbereich

Parameter	Einzugebender Wert	Beschreibung
b_ModulNbr	APCI1710_16BIT_COUNTER	Das Modul ist für 2 x 16-bit-Zähler konfiguriert. - b_FirstCounterMode und b_FirstCounterOption konfigurieren den ersten 16-Bit Zähler. - b_SecondCounterMode und b_SecondCounterOption konfigurieren den zweiten 16-bit Zähler.
b_ModulNbr	APCI1710_32BIT_COUNTER	Das Modul ist für einen 32-Bit Zähler konfiguriert. - b_FirstCounterMode und b_FirstCounterOption konfigurieren den 32-Bit Zähler. - b_SecondCounterMode und b_SecondCounterOption sind nicht verwendet.

Tabelle 3-5: Zähler-Betriebsmode

Parameter	Einzugebender Wert	Beschreibung
b_FirstCounterMode oder b_SecondCounterMode	APCI1710_QUADRUPLE_MODE	Im 4-fach Mode, generiert die Flankenauswerteschaltung einen Zählimpuls aus jeder Flanke zweier zueinander phasenverschobener Signale.
b_FirstCounterMode oder b_SecondCounterMode	APCI1710_DOUBLE_MODE	Funktion wie 4-fach Mode. Es werden jedoch nur 2 der 4 Flanken pro Periode ausgewertet.
b_FirstCounterMode oder b_SecondCounterMode	APCI1710_SIMPLE_MODE	Funktion wie 4-fach Mode. Es wird jedoch nur 1 der 4 Flanken pro Periode ausgewertet.
b_FirstCounterMode oder b_SecondCounterMode	APCI1710_DIRECT_MODE	Im Direkt-Mode werden beide Flankenauswerteschaltungen inaktiv. Die Eingänge A und B im 32-Bit Mode oder A, B und C, D im 16-Bit Mode stellen je eine Takt-Torschaltung dar. Dabei können Frequenz- und Impulsdauermessungen durchgeführt werden.

WICHTIG!
Diese Option ist nicht verfügbar, wenn Sie den Direkt-Mode ausgewählt haben.

Tabelle 3-6: Zähler-Option für 4fach-/2fach-/1fach-Mode

Parameter	Einzugebender Wert	Beschreibung
b_FirstCounterOption oder *b_SecondCounterOption*	APCI1710_HYSTERESIS_ON	Auf beiden Flankenauswerteschaltungen steht eine Hysterese-Schaltung zur Verfügung. Sie unterdrückt den ersten Zählimpuls nach einer Drehrichtungumkehr.
b_FirstCounterOption oder *b_SecondCounterOption*	APCI1710_HYSTERESIS_OFF	Der erste Impuls wird nicht nach einer Drehrichtungumkehr unterdrückt.

WICHTIG!
Diese Option ist nicht verfügbar, wenn Sie den 4fach-/2fach-/1fach-Mode ausgewählt haben.

Tabelle 3-7: Zähler-Option für Direkt-Mode

Parameter	Einzugebender Wert	Beschreibung
b_FirstCounterOption oder *b_SecondCounterOption*	APCI1710_INCREMENT	Der Zähler inkrementiert nach jeden Zählimpuls
b_FirstCounterOption oder *b_SecondCounterOption*	APCI1710_DECREMENT	Der Zähler dekrementiert nach jedem Zählimpuls.

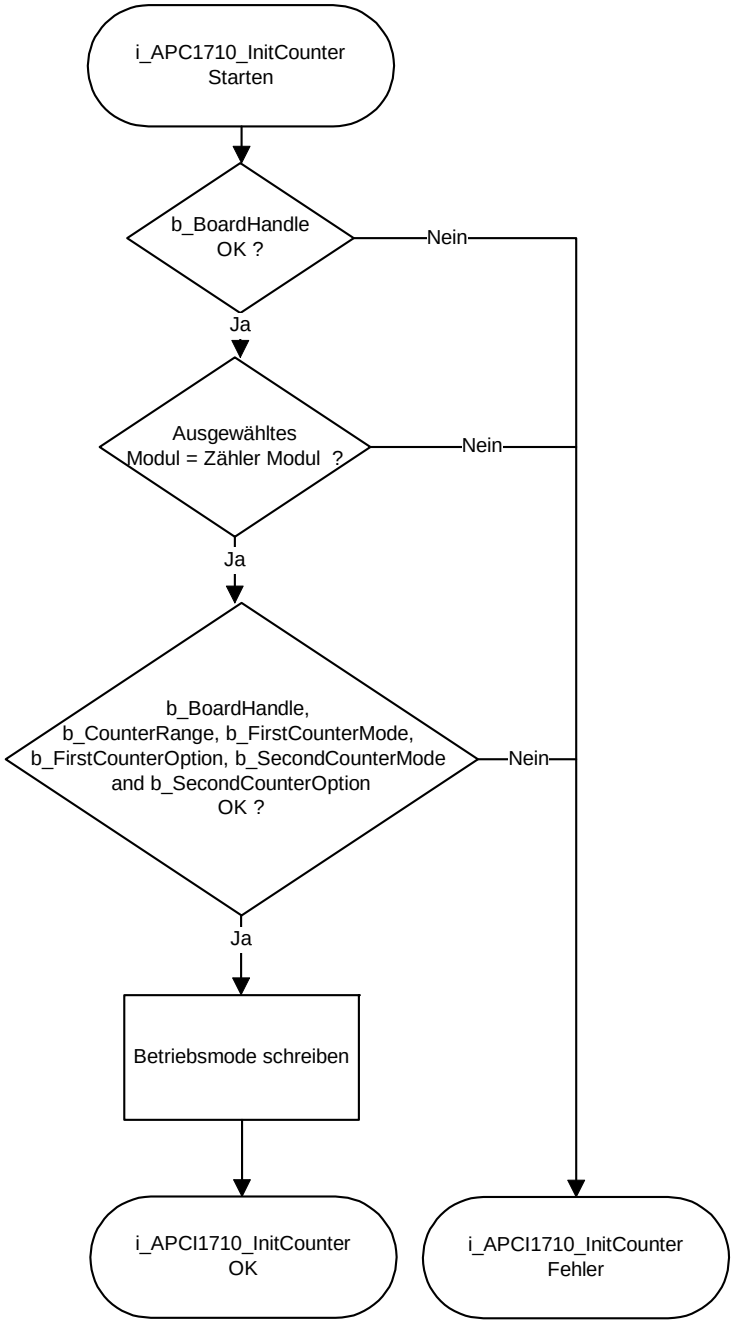
Funktionsaufruf:

```
ANSI C:
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_InitCounter
                                (b_BoardHandle,
                                 0,
                                 APCI1710_16BIT_COUNTER,
                                 APCI1710_QUADRUPLE_MODE,
                                 APCI1710_HYSTERESIS_ON,
                                 APCI1710_QUADRUPLE_MODE,
                                 APCI1710_HYSTERESIS_ON);
```

Return Wert:

0: Kein Fehler
-1:Handle-Parameter der Karte ist falsch.
-2: Das Modul ist kein Zählermodul.
-3: Der ausgewählte Zählerbereich ist falsch. Siehe Tabelle 3-4
-4: Der ausgewählte Betriebsmode des ersten Zählers ist falsch.
Siehe Tabelle 3-5
-5: Die ausgewählte Betriebsmode-Option für den ersten Zähler ist falsch.
Siehe Tabelle 3-6
-6: Der ausgewählte Betriebsmode des zweiten Zählers ist falsch.
Siehe Tabelle 3-5
-7: Die ausgewählte Betriebsmode-Option für den zweiten Zähler ist falsch.
Siehe Tabelle 3-6

Eingabe	
b_BoardHandle	b_FirstCounterOption
b_CounterRange	b_SecondCounterMode
b_FirstCounterMode	b_SecondCounterOption
Funktionsablauf	



Ausgabe
<Return Wert>

2) i_APCI1710_CounterAutoTest (...)

Syntax:
<Return Wert> = i_APCI1710_CounterAutoTest
(BYTE b_BoardHandle,
PBYTE pb_TestStatus)

Parameter:
- Eingabe:
BYTE b_BoardHandle Handle der Karte **xPCI-1710**
- Ausgabe:
PBYTE pb_TestStatus Autotest-Rückgabe. Siehe Tabelle 3-8

Aufgabe:
Ein Testmode ist für den Test des Bausteins vorgesehen. Alle 8-Bit Zählerketten werden intern als Abwärts-Zähler betrieben. Unabhängig von den externen Signalen werden alle 8-Bit Zählerketten parallel bei jeder negativen Taktflanke von CLKX dekrementiert.

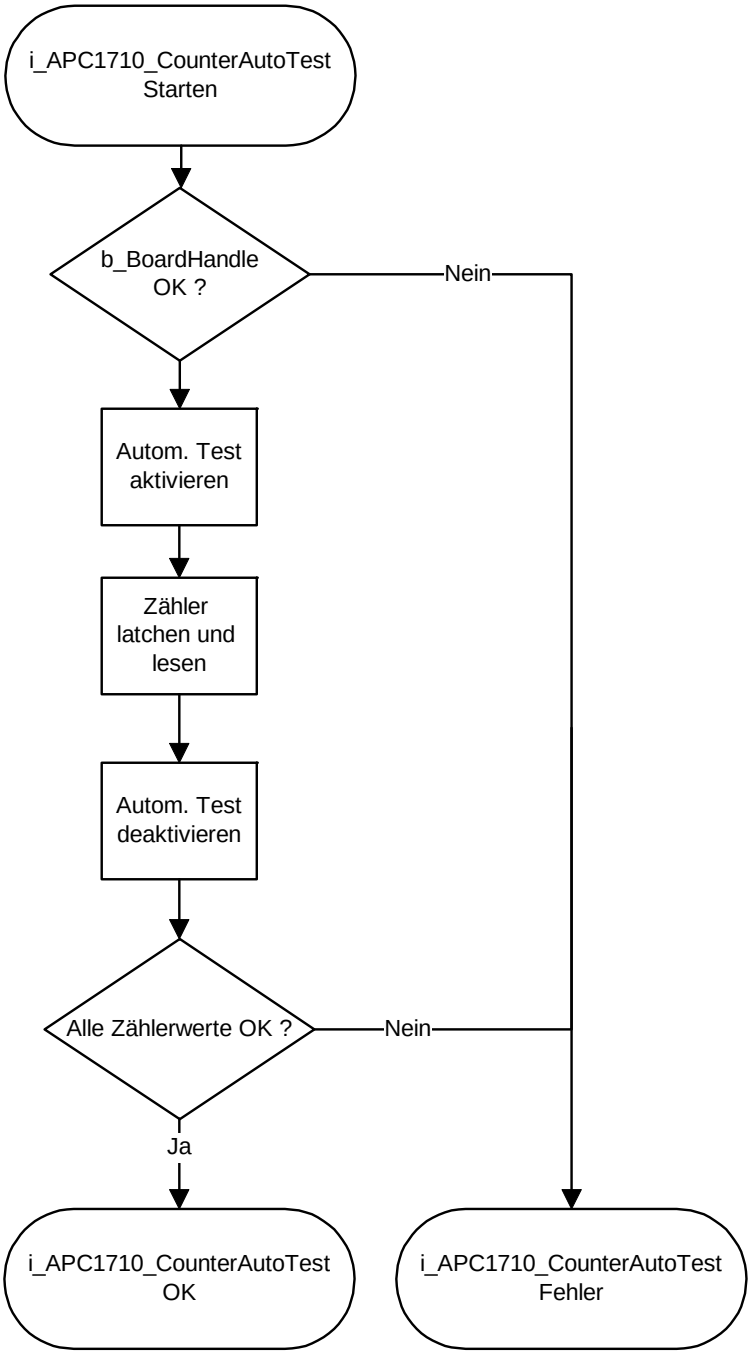
Tabelle 3-8: Autotest-Rückgabe

pb_TestStatus **Maske**	**Bedeutet**
0000	Kein Fehler
000**1**	Fehler auf Zähler 0
00**1**0	Fehler auf Zähler 1
0**1**00	Fehler auf Zähler 2
1000	Fehler auf Zähler 3

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_TestStatus;

i_ReturnValue = i_APCI1710_CounterAutoTest (b_BoardHandle,
                                            &b_TestStatus;
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Kein Zählermodul gefunden

Eingabe
b_BoardHandle
Funktionsablauf

Ausgabe
pb_TestStatus
<Return Wert>

3) i_APCI1710_ClearCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_ClearCounterValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

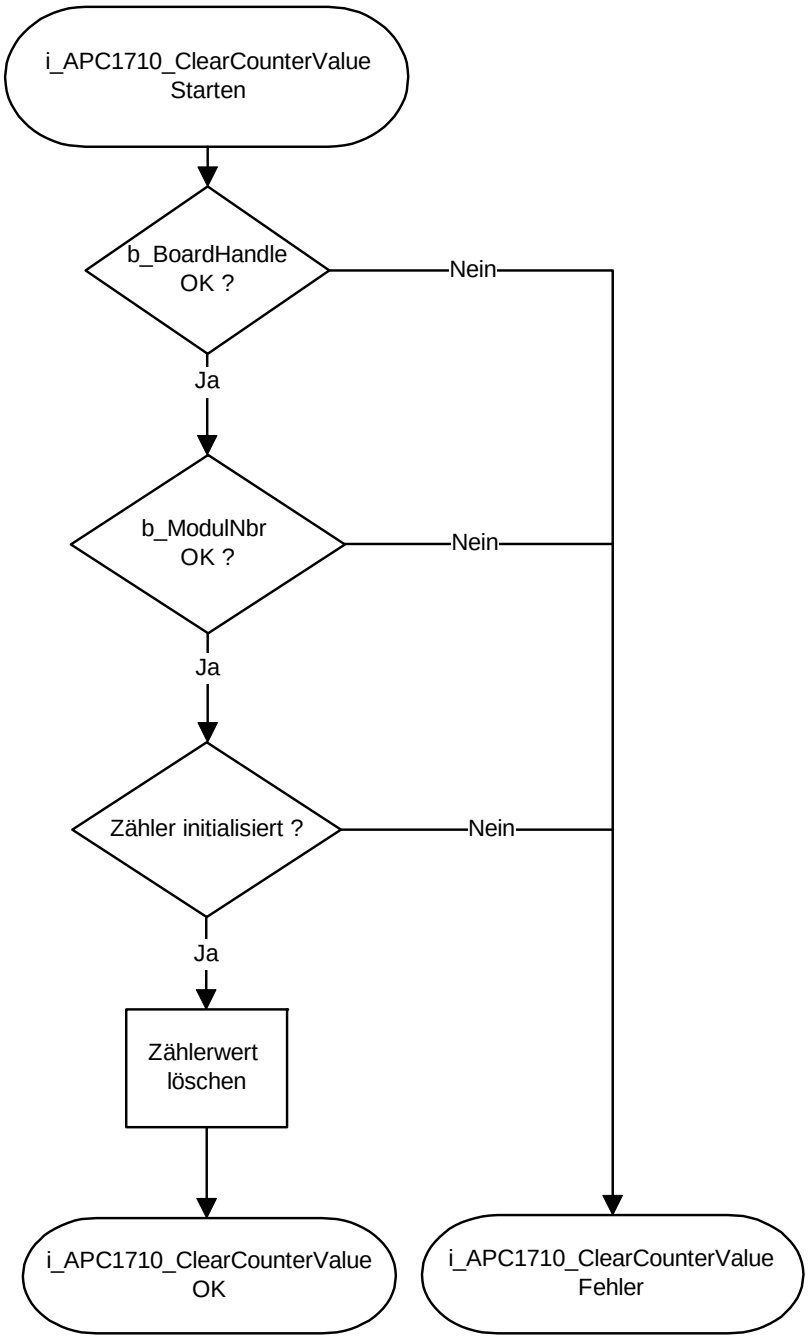
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Löscht den Zählerwert auf dem ausgewählten Modul (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;
i_ReturnValue = i_APCI1710_ClearCounterValue
                                    (b_BoardHandle,
                                     0);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
<Return Wert>

4) i_APCI1710_ClearAllCounterValue (...)

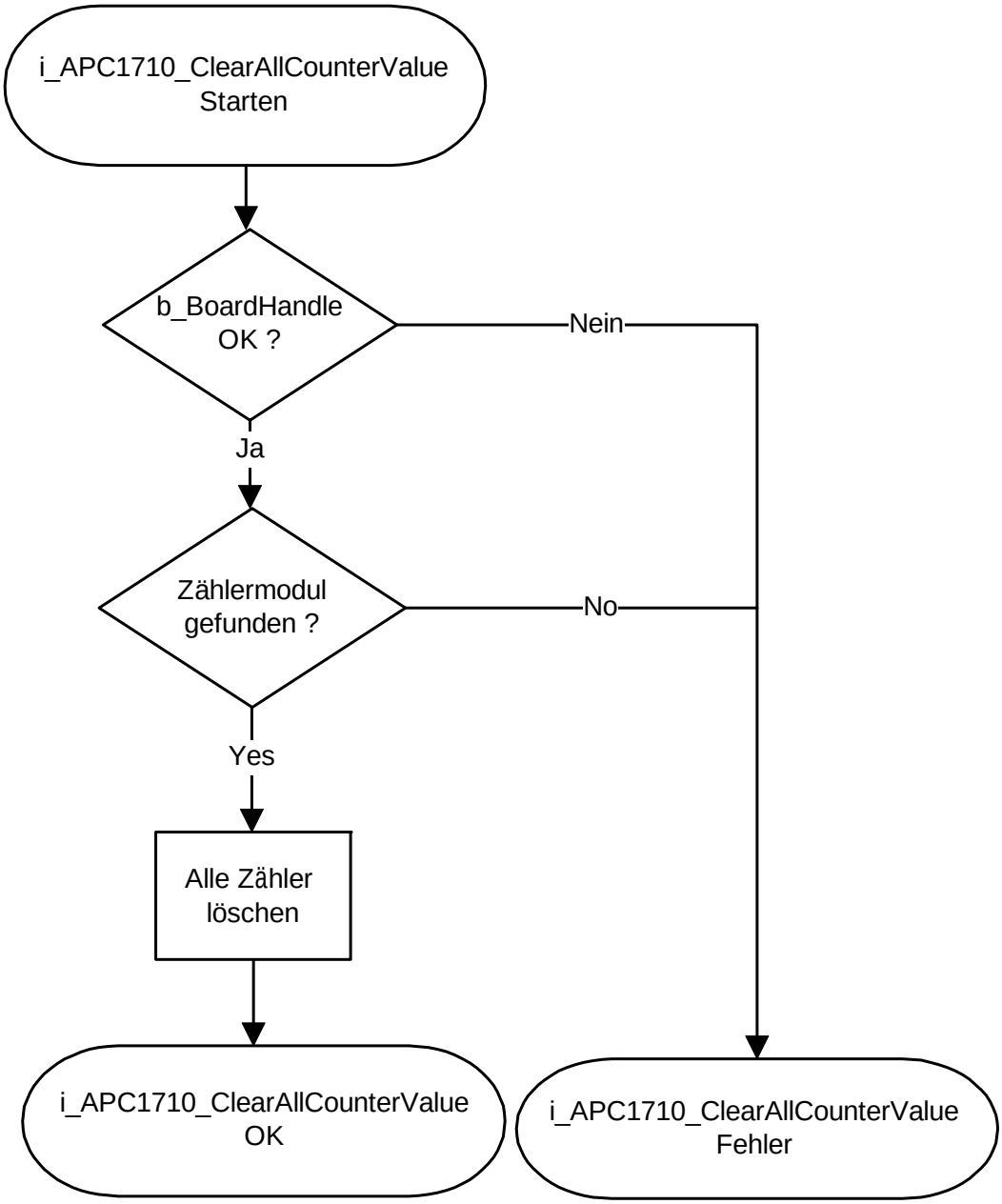
Syntax:
<Return Wert> = i_APCI1710_ClearAllCounterValue
(BYTE b_BoardHandle)

Parameter:
- Eingabe:
BYTE b_BoardHandle Handle der Karte **xPCI-1710**
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Löscht alle Zählerwerte.

Funktionsaufruf:
ANSI C :
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_ClearAllCounterValue
                                    (b_BoardHandle);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Kein Zählermodul gefunden.

Eingabe
b_BoardHandle
Funktionsablauf

Ausgabe
<Return Wert>

5) i_APCI1710_SetInputFilter (...)

Syntax:
<Return Wert> = i_APCI1710_SetInputFilter
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_PCIInputClock,
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_PCIInputClock	Auswahl des PCI Bus Takts - APCI1710_30MHZ: Der PC hat einen PCI Bustakt von 30 MHz - APCI1710_33MHZ: Der PC hat einen PCI Bustakt von 33 MHz - APCI1710_40MHz: Der PC hat einen PCI Bustakt von 40 MHz
BYTE	b_Filter	Auswahl des Filters für die diff. Eingänge A,B,C und D. Siehe Tabelle 3-9

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert oder aktiviert den Software-Filter im ausgewählten Modul (b_ModulNbr) für die Eingänge A,B,C und D.
b_Filter gibt die Filterzeit ein.

Tabelle 3-9: Filterzeit

b_PCIInputClock *b_Filter*	APCI1710_30MHZ	APCI1710_33MHZ	APCI1710_40MHZ
0	Filter nicht benutzt	Filter nicht benutzt	Filter nicht benutzt
1	266 ns (3,75 MHz)	242 ns (4,125 MHz)	200 ns (5,0000 MHz)
2	400 ns (2,5 MHz)	363 ns (2,75482 MHz)	300 ns (3,3333 MHz)
3	533 ns (1,87617 MHz)	484 ns (2,066115 MHz)	400 ns (2,5000 MHz)
4	666 ns (1,5015 MHz)	605 ns (1,652892 MHz)	500 ns (2,0000 MHz)
5	800 ns (1,25 MHz)	726 ns (1,3577410 MHz)	600 ns (1,6666 MHz)
6	933 ns (1,0718 MHz)	847 ns (1,180637 MHz)	700 ns (1,4285 MHz)
7	1066 ns (0,93808 MHz)	968 ns (1,033055 MHz)	800 ns (1,2500 MHz)
8	1200 ns (0,8333 MHz)	1089 ns (0,918273 MHz)	900 ns (1,1111 MHz)
9	1333 ns (0,75 MHz)	1210 ns (0,826446 MHz)	1000 ns (1,0000 MHz)
10	1466 ns (0,6821 MHz)	1331 ns (0,751314 MHz)	1100 ns (0,9090 MHz)
11	1600 ns (0,625 MHz)	1452 ns (0,688705 MHz)	1200 ns (0,8333 MHz)
12	1733 ns (0,577 MHz)	1573 ns (0,635727 MHz)	1300 ns (0,7692 MHz)
13	1866 ns (0,5359 MHz)	1694 ns (0,590318 MHz)	1400 ns (0,7142 MHz)
14	2000 ns (0,5000 MHz)	1815 ns (0,550964 MHz)	1500 ns (0,6666 MHz)
15	2133 ns (0,4688 MHz)	1936 ns (0,516528 MHz)	1600 ns (0,6250 MHz)

Funktionsaufruf:

ANSI C :

```
int                                 i_ReturnValue;
unsigned char                       b_BoardHandle;
i_ReturnValue = i_APCI1710_SetInputFilter
                                    (b_BoardHandle
                                     0,
                                     APCI1710_40MHz,
                                     9);
```

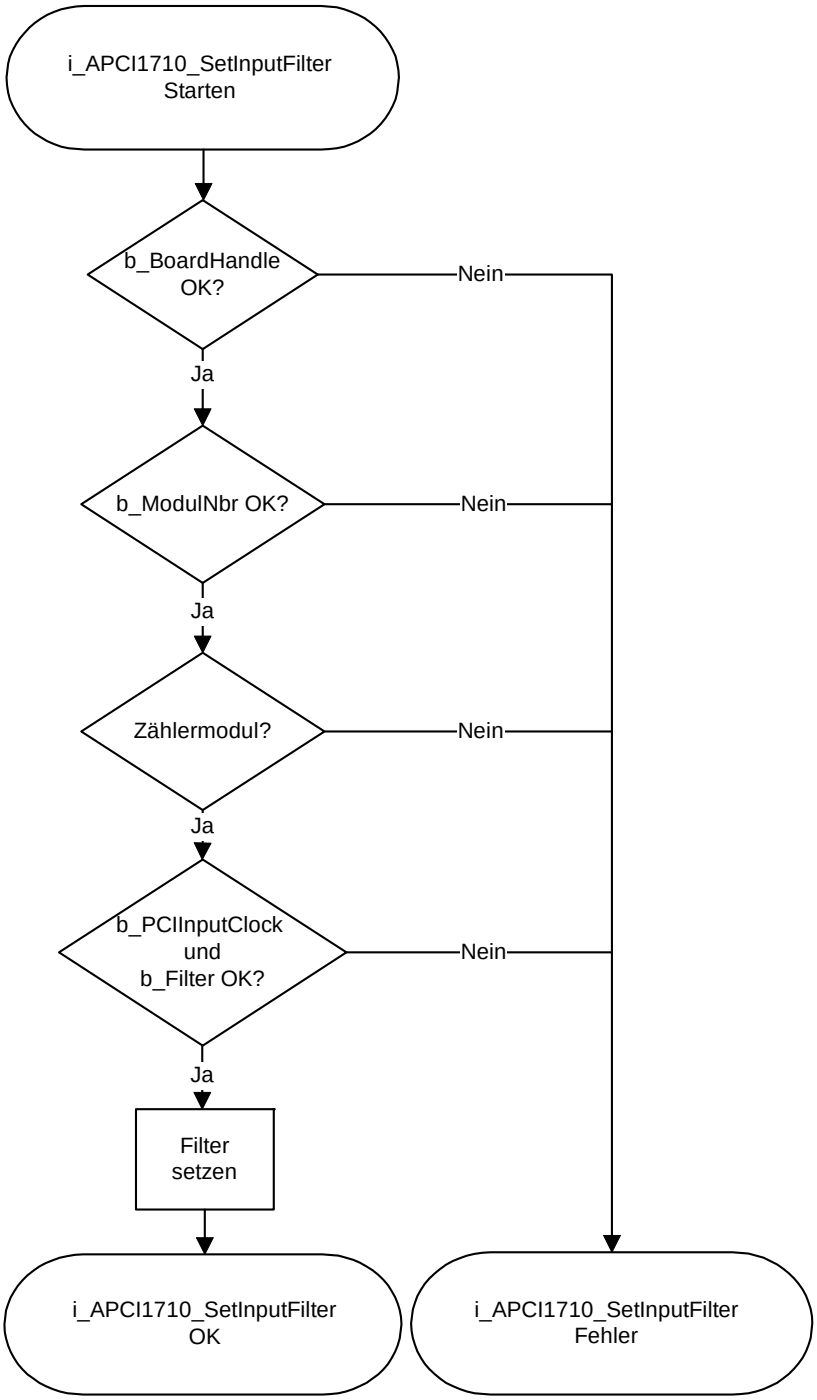
Return Wert:

0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein Zählermodul.
-4: Der ausgewählte PCI Eingangstakt ist falsch.
-5: Die ausgewählte Filterzeit ist falsch.
-6: Auf der Karte ist kein 40MHz Quarz eingebaut.

Eingabe

b_BoardHandle	b_PCIInputClock
b_ModulNbr	b_Filter

Funktionsablauf



Ausgabe

<Return Wert>

3.4 Zähler lesen

1) i_APCI1710_LatchCounter (...)

Syntax:
<Return Wert> = i_APCI1710_LatchCounter
(BYTEb_BoardHandle,
BYTE b_ModulNbr,
BYTE b_LatchReg)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_LatchReg	Auswahl des Latch-Registers 0: für das erste Latch-Register 1: für das zweite Latch-Register

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Latchen des aktuellen Wertes vom ausgewählten Modul (*b_ModulNbr*) in das ausgewählte Latch-Register (*b_LatchReg*).

Funktionsaufruf:
ANSI C :

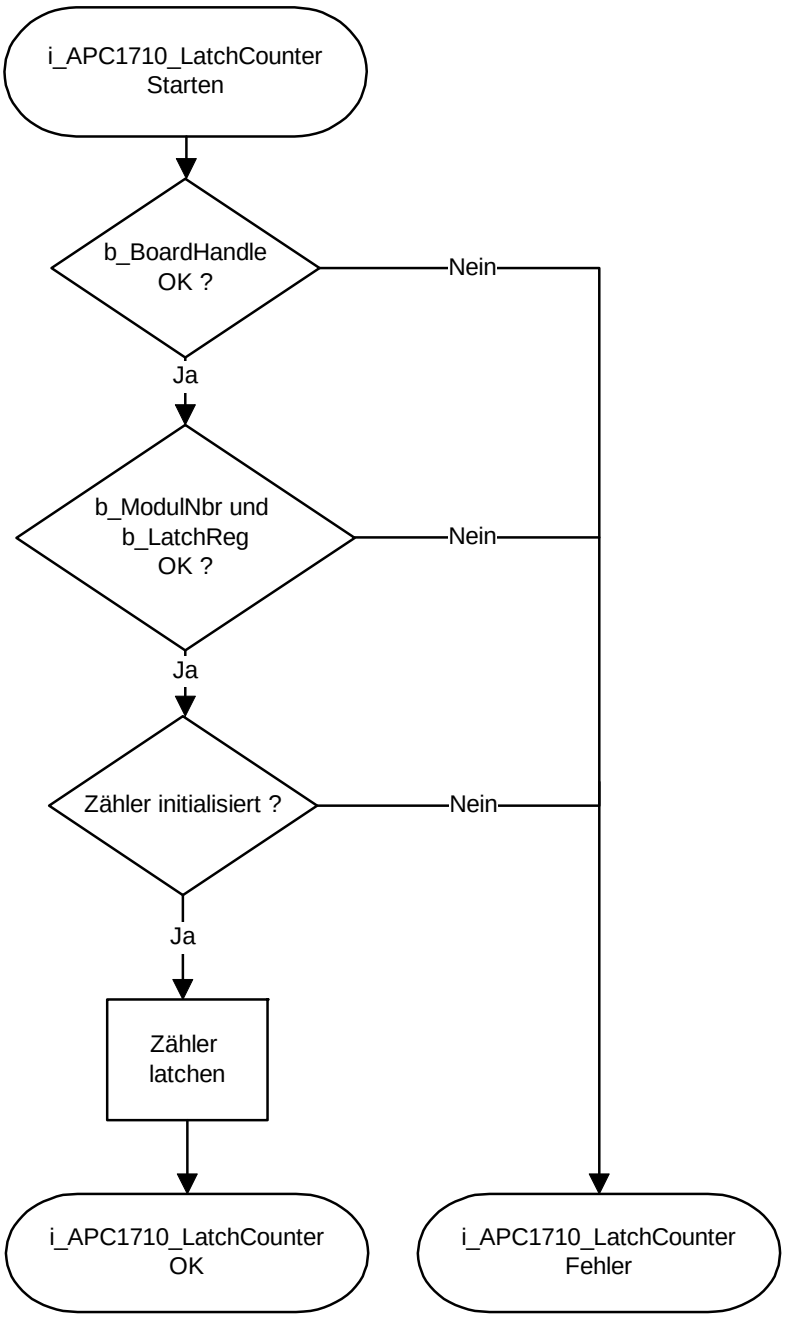
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_LatchCounter
                                (b_BoardHandle,
                                 0,
                                 0);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Das ausgewählte Latch-Register ist falsch.

Eingabe

b_BoardHandle b_LatchReg
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

2) i_APCI1710_ReadLatchRegisterStatus (...)

Syntax:
<Return Wert> = i_APCI1710_ReadLatchRegisterStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_LatchReg,
PBYTE pb_LatchStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_LatchReg	Auswahl des Latch-Registers 0: für das erste Latch-Register 1: für das zweite Latch-Register

- Ausgabe:

PBYTE	pb_LatchStatus	Latch-Register-Status. 0: Kein Latch 1: Ein Software Latch wurde ausgelöst. 2: Ein Hardware Latch wurde ausgelöst. 3: Ein Software Latch und ein Hardware Latch wurden ausgelöst.

Aufgabe:
Liest den Status des ausgewählten Latch-Registers (*b_LatchReg*) im ausgewählten Modul (*b_ModulNbr*).

Funktionsaufruf:

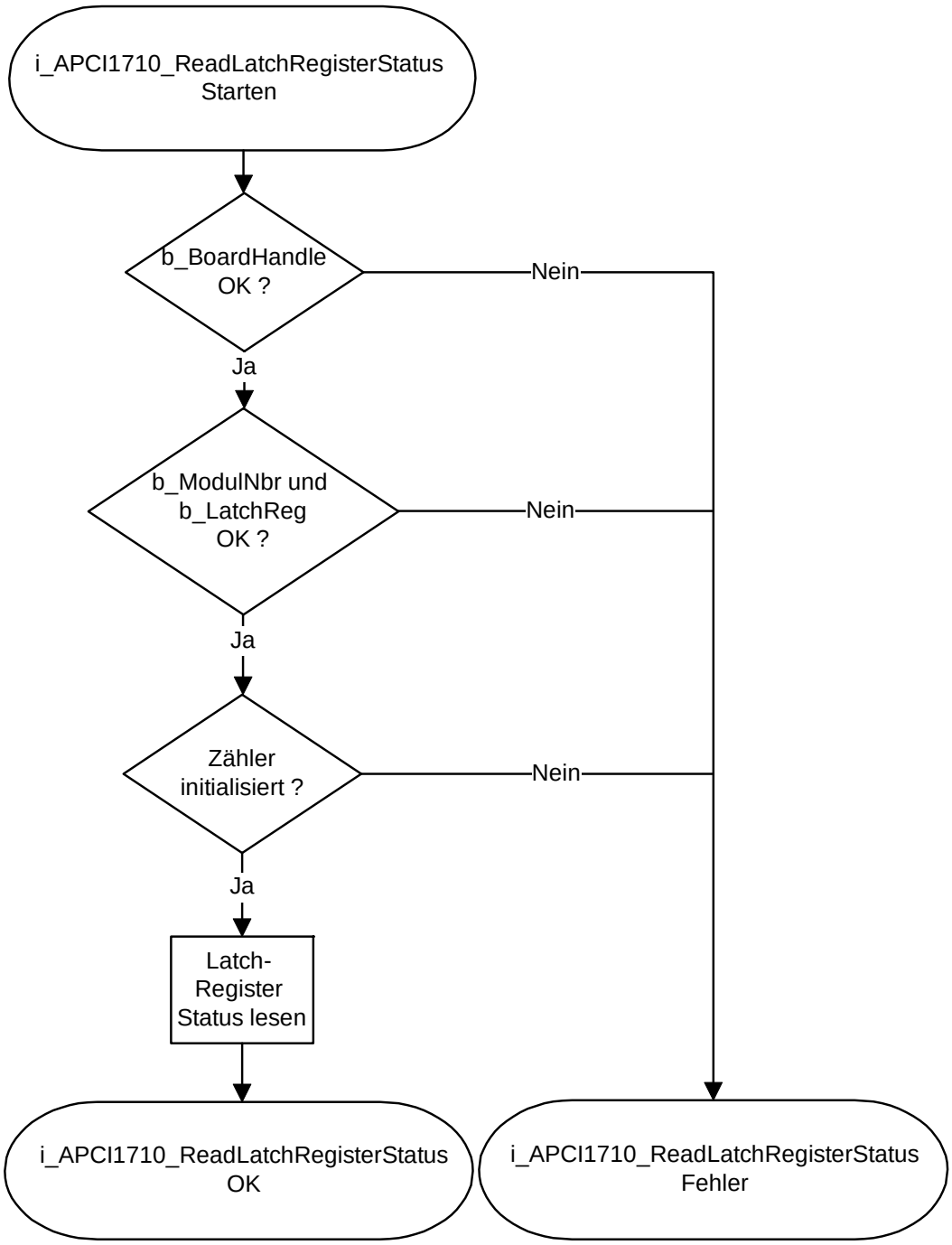
ANSI C :
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_LatchStatus;

i_ReturnValue = i_APCI1710_LatchRegisterStatus
                (b_BoardHandle, 0, 0,
                 b_LatchStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Der ausgewählte Latch-Register ist falsch.

Eingabe

b_BoardHandle b_LatchReg
b_ModulNbr

Funktionsablauf



Ausgabe

pb_LatchStatus

<Return Wert>

3) i_APCI1710_ReadLatchRegisterValue (...)

Syntax:
<Return Wert> = i_APCI1710_ReadLatchRegisterValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_LatchReg,
PULONG pul_LatchValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_LatchReg	Auswahl des Latch-Registers 0: für das erste Latch-Register 1: für das zweite Latch-Register

- Ausgabe:

PULONG	pul_LatchValue	Latch-Register Wert

Aufgabe:
Liest den Wert des ausgewählten Latch-Registers (*b_LatchReg*) im ausgewählten Modul *(b_ModulNbr)*

Funktionsaufruf:
ANSI C :

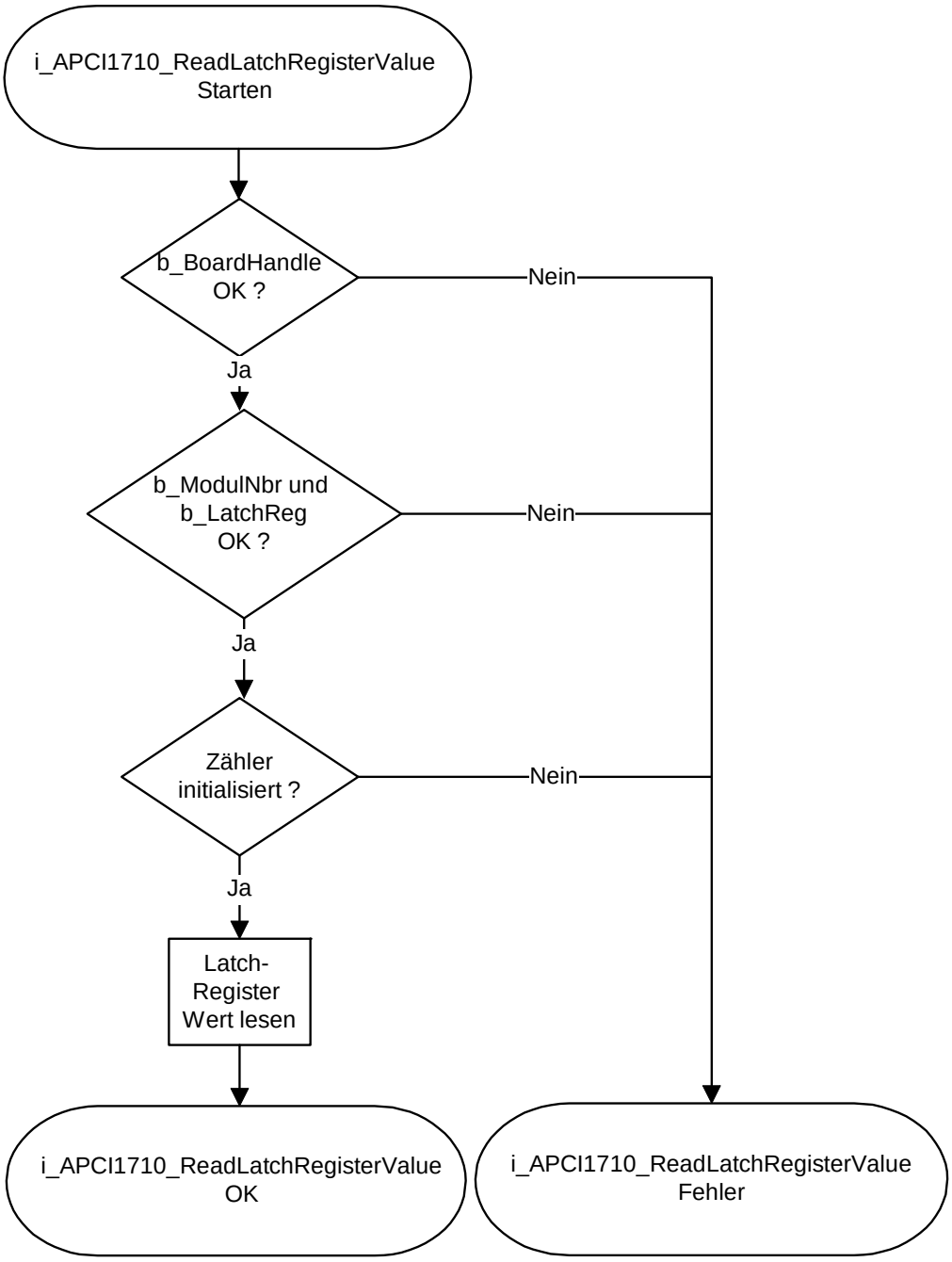
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned long   ul_LatchValue;

i_ReturnValue = i_APCI1710_ ReadLatchRegisterValue
                                (b_BoardHandle,
                                 0,
                                 0,
                                 &ul_LatchValue);
```

Return Wert:
0: Kein Fehler
-1:Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Das ausgewählte Latch-Register ist falsch.

Eingabe

b_BoardHandle b_LatchReg
b_ModulNbr

Funktionsablauf



Ausgabe

pul_LatchValue

<Return Wert>

4) i_APCI1710_EnableLatchInterrupt (...)

Syntax:
<Return Wert> = i_APCI1710_EnableLatchInterrupt
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Aktiviert den Latch-Interrupt vom ausgewählten Modul (*b_ModulNbr*).
Jedes Hardware-Latch generiert einen Interrupt.

Funktionsaufruf:
ANSI C :

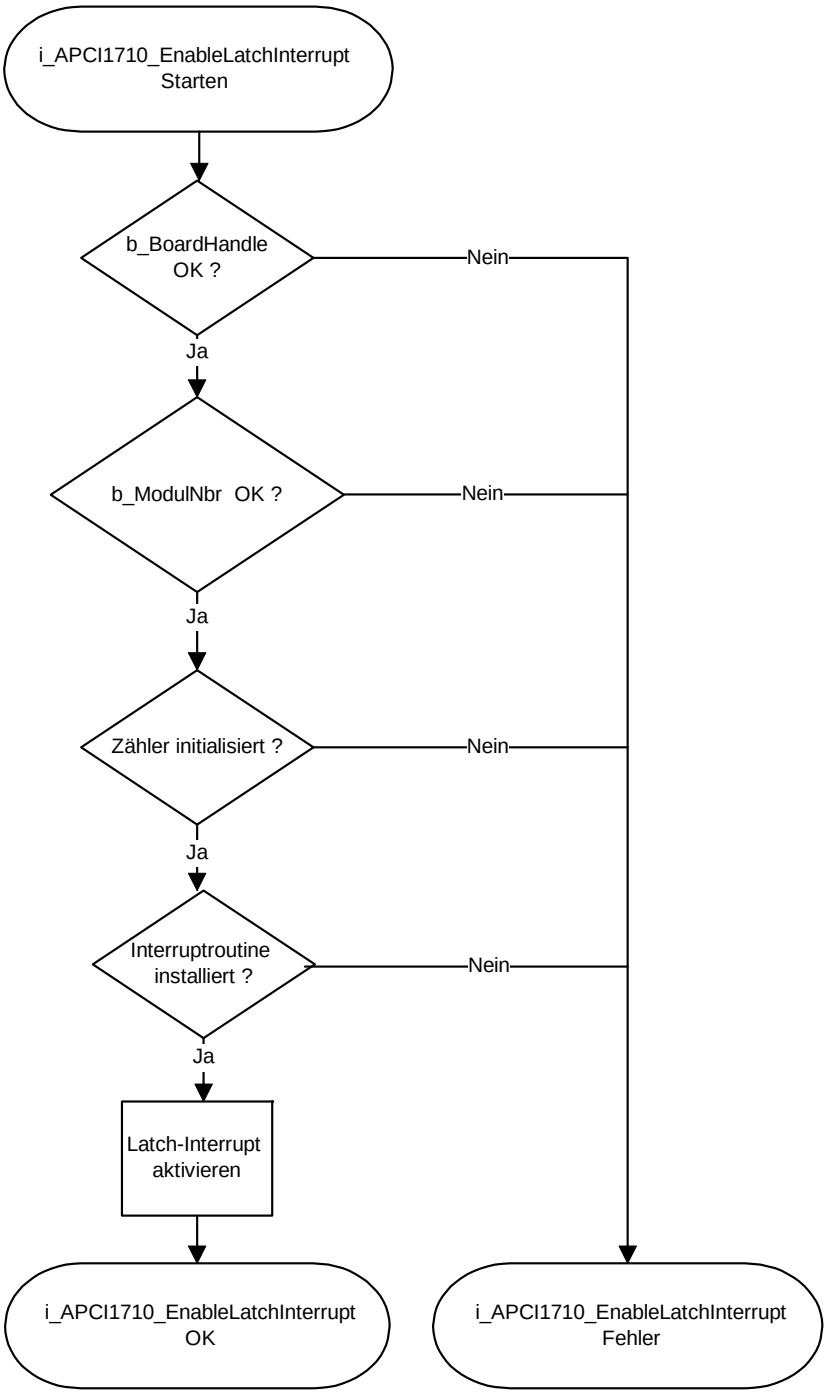
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_EnableLatchInterrupt
                                    (b_BoardHandle, 0);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Interruptroutine nicht installiert. Siehe Funktion "i_APCI1710_SetBoardIntRoutine"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

5) i_APCI1710_DisableLatchInterrupt (...)

Syntax:
<Return Wert> = i_APCI1710_DisableLatchInterrupt
(BYTEb_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert den Latch-Interrupt vom ausgewählten Modul (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

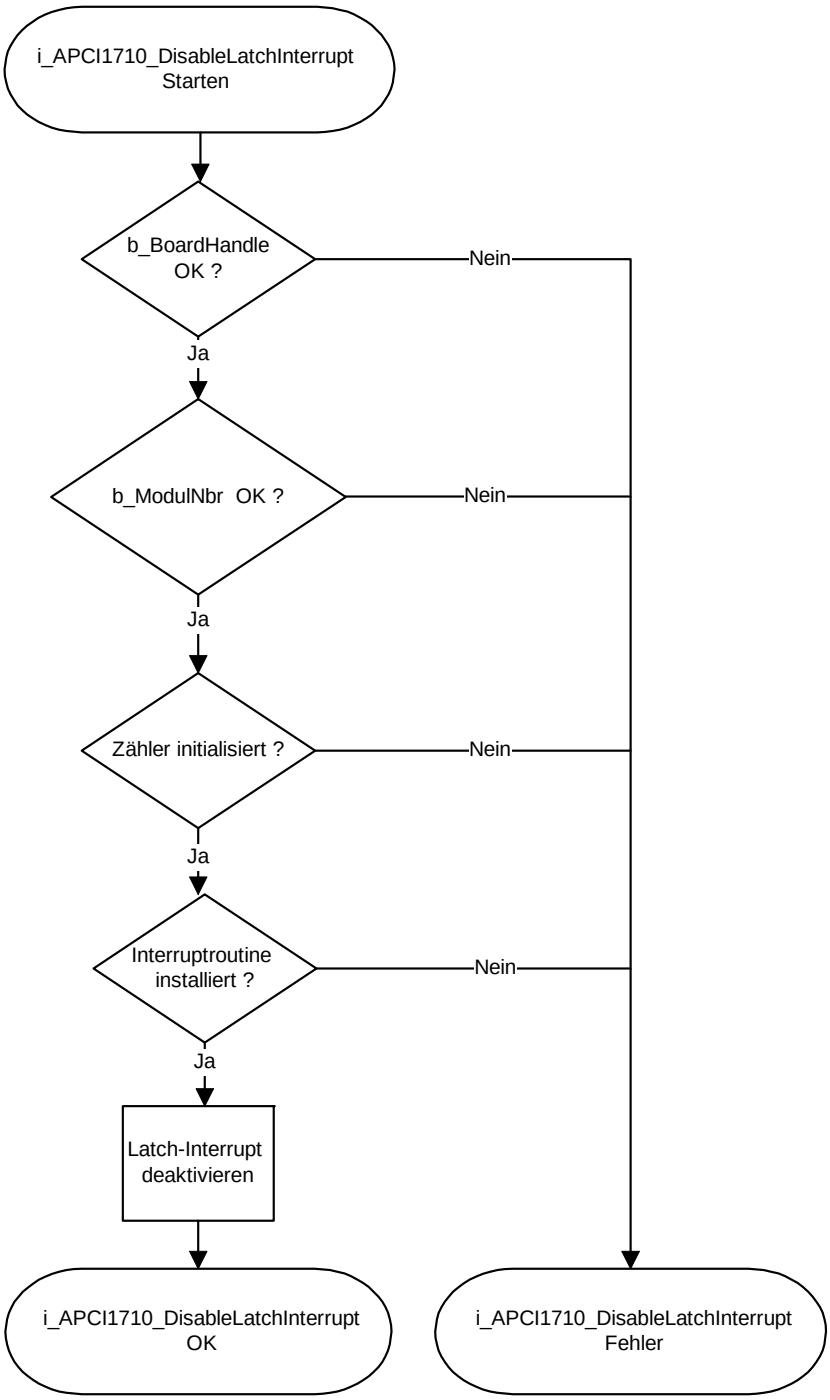
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableLatchInterrupt
                (b_BoardHandle, 0);
```

Return Wert:
0: Kein Fehler
-1:Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Interruptroutine nicht installiert. Siehe Funktion "i_APCI1710_SetBoardIntRoutine"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

6) i_APCI1710_Read16BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_Read16BitCounterValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_SelectedCounter,
PUINT pui_CounterValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_SelectedCounter	Auswahl des 16-bit Zählers (0 oder 1)

- Ausgabe:

PUINT	pui_CounterValue	16-Bit Zählerwert

Aufgabe:
Latchen des 16-Bit Zählers (*b_SelectedCounter*) vom ausgewählten Modul (*b_ModulNbr*) im ersten Latch-Register und Rückgabe des gelatchten Wertes.

Funktionsaufruf:
ANSI C :

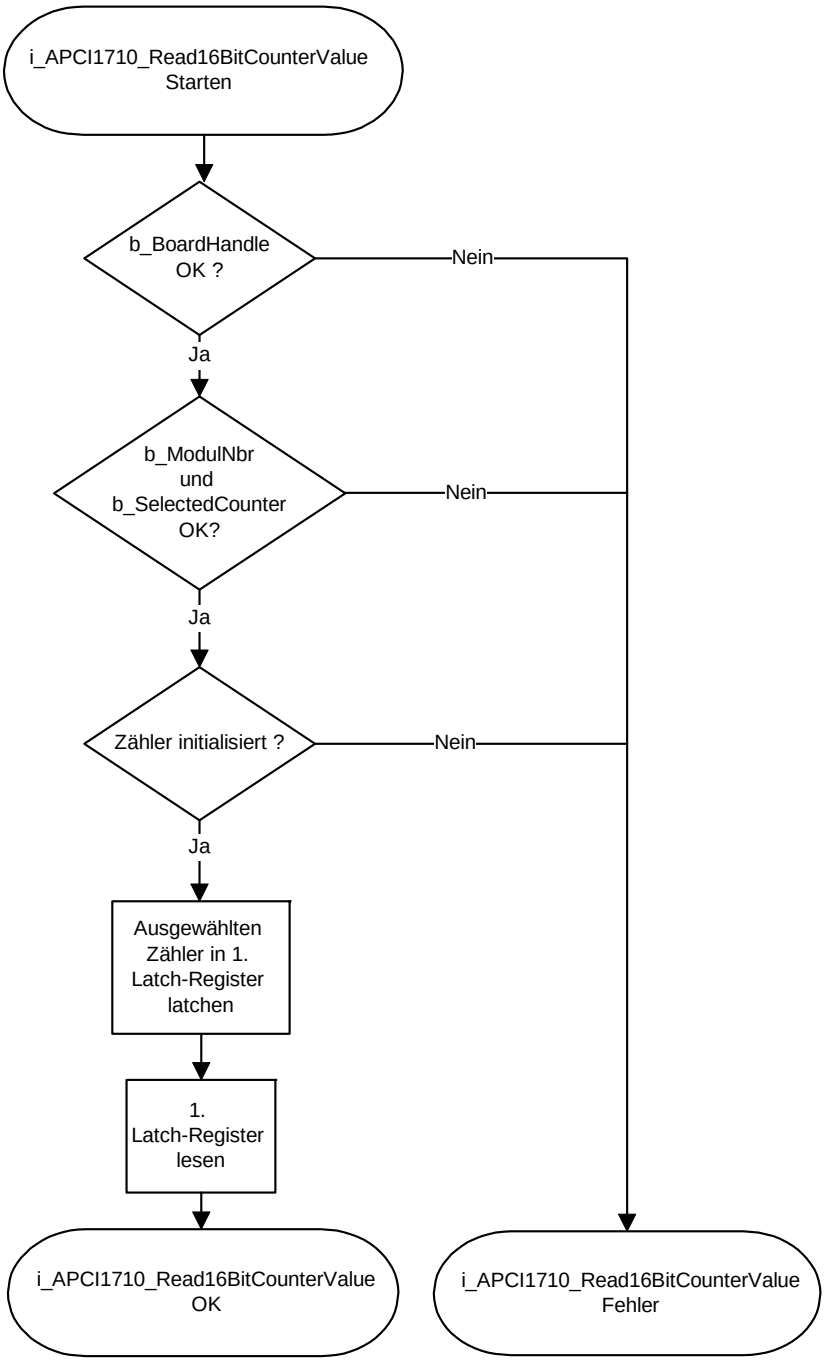
```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;
unsigned int        ui_CounterValue;

i_ReturnValue = i_APCI1710_Read16BitCounterValue
                    (b_BoardHandle,
                     0,
                     0,
                     &ui_CounterValue);
```

Return Wert:
0: Kein Fehler
-1:Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter".
-4: Der ausgewählte 16-Bit Zähler ist falsch.

Eingabe

b_BoardHandle b_SelectedCounter
b_ModulNbr

Funktionsablauf



Ausgabe

pui_CounterValue

<Return Wert>

7) i_APCI1710_Read32BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_Read32BitCounterValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr
PULONG pul_CounterValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

PULONG	pul_CounterValue	32-Bit Zählerwert

Aufgabe:
Latchen des 32-Bit Zählers vom ausgewählten Modul (*b_ModulNbr*) im ersten Latch-Register und Rückgabe des gelatchten Wertes.

Funktionsaufruf:
ANSI C :

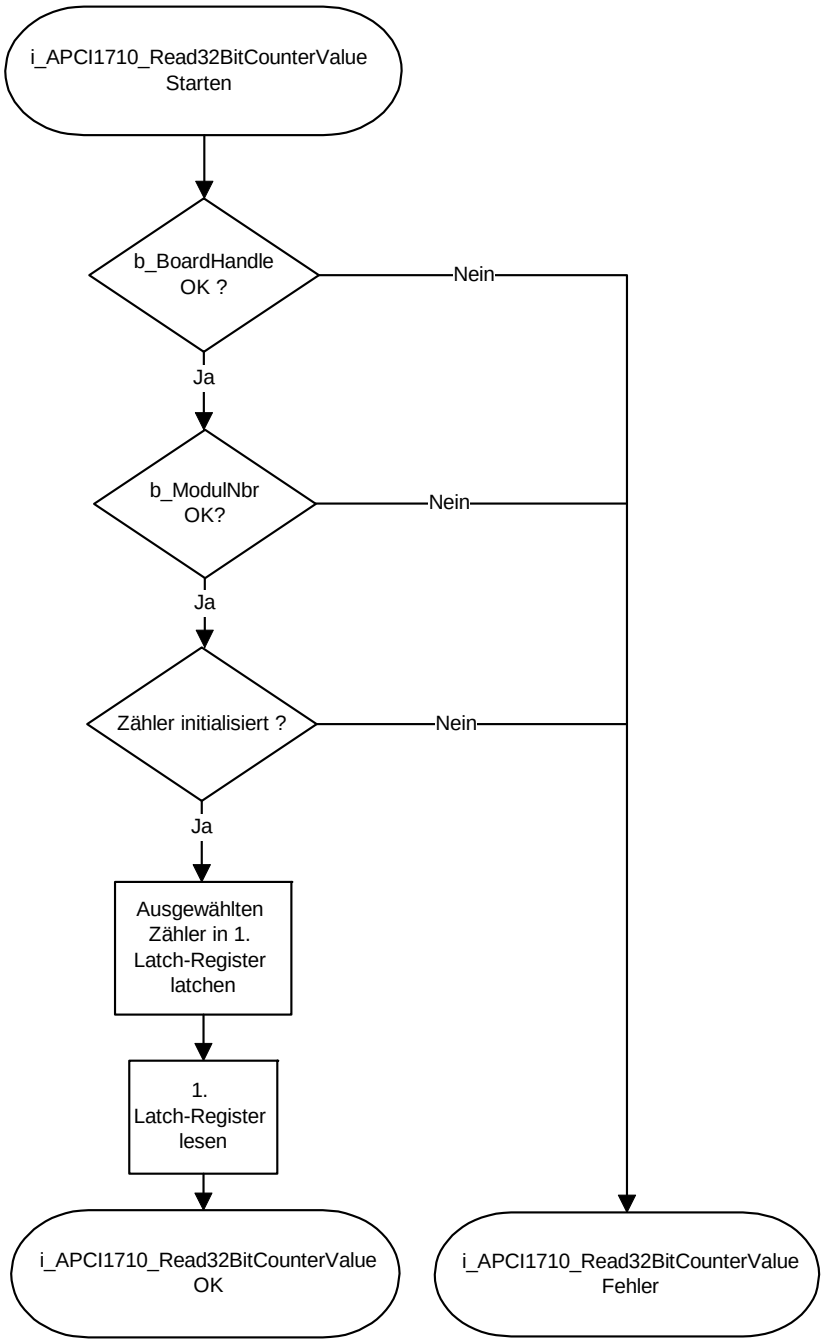
```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;
unsigned long       ul_CounterValue;

i_ReturnValue = i_APCI1710_Read32BitCounterValue
                    (b_BoardHandle,
                     0,
                     &ul_CounterValue);
```

Return Wert:
0: Kein Fehler
-1: Handle-Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

pul_CounterValue

<Return Wert>

3.5 In den Zähler schreiben

1) i_APCI1710_Write16BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_Write16BitCounterValue
(BYTE b_BoardHandle
BYTE b_ModulNbr,
BYTE b_SelectedCounter,
UINT ui_WriteValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_SelectedCounter	Auswahl des 16-Bit Zähler (0 oder 1)
UINT	ui_WriteValue	16-Bit Schreibewert

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Schreibt einen 16-Bit Wert *(ui_WriteValue)* in den 16-Bit Zähler *(b_SelectedCounter)* des ausgewählten Moduls *(b_ModulNbr).*

Funktionsaufruf:
ANSI C :

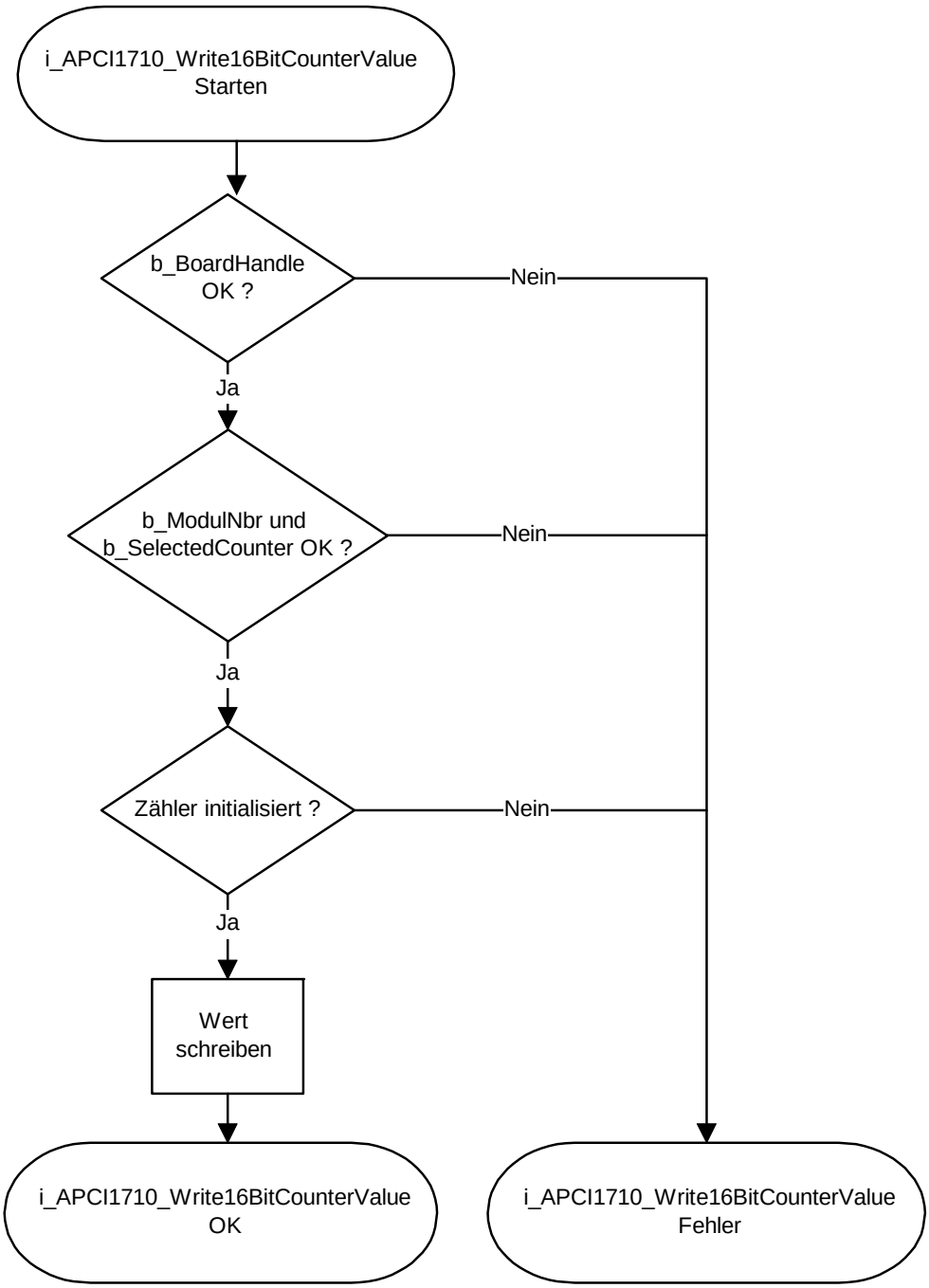
```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;

i_ReturnValue = i_APCI1710_Write16BitCounterValue (b_BoardHandle,
                                                   0,
                                                   0,
                                                   2000);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Der 16-Bit Zähler Parameter ist falsch.

Eingabe

b_BoardHandle	b_SelectedCounter
b_ModulNbr	ui_WriteValue

Funktionsablauf



Ausgabe

<Return Wert>

2) i_APCI1710_Write32BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_Write32BitCounterValue
(BYTE b_BoardHandle
BYTE b_ModulNbr,
ULONG ul_WriteValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
ULONG	ul_WriteValue	32-Bit Schreibewert

- Ausgabe:
Es erfolgt keine Ausgabe.

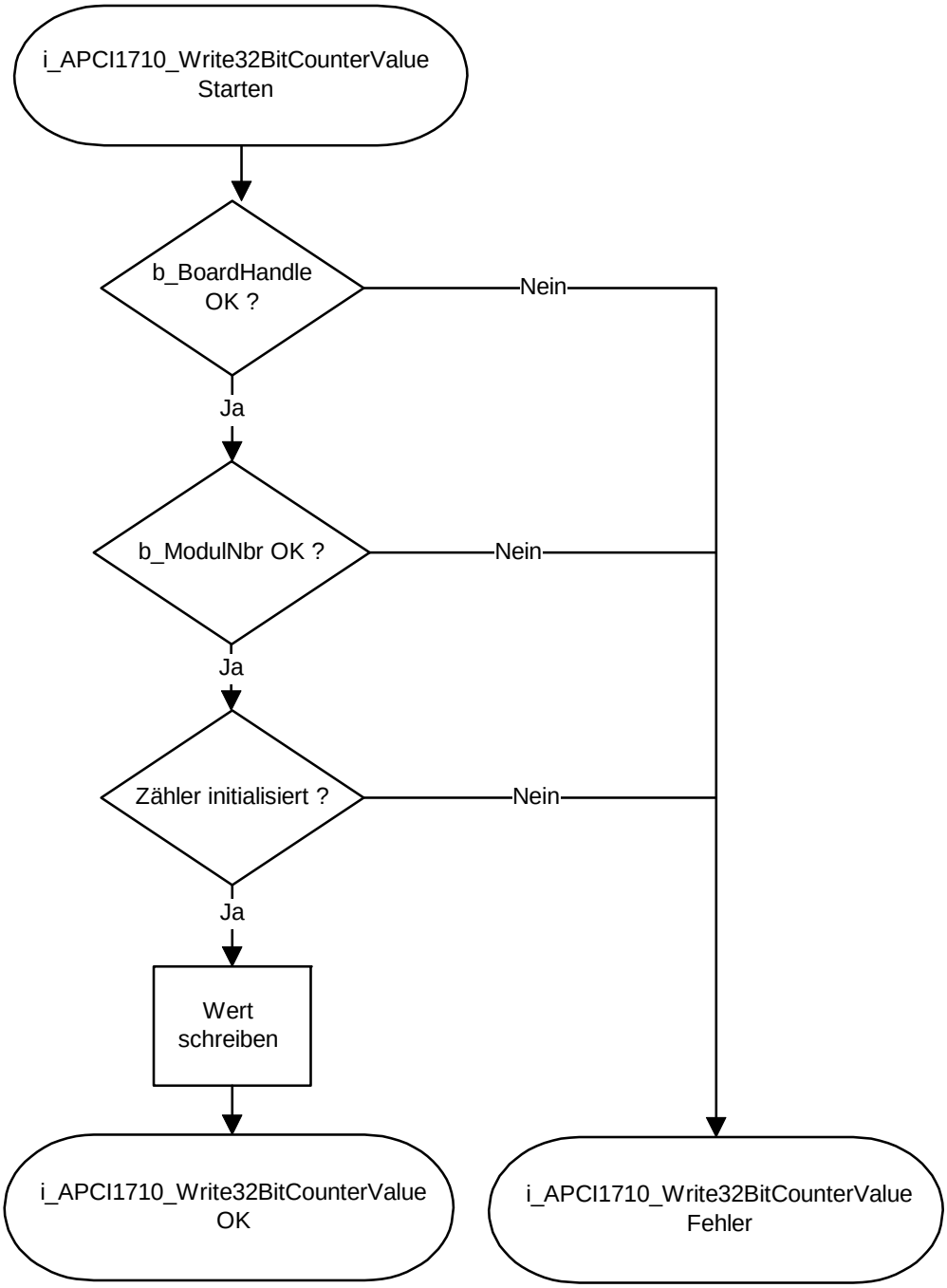
Aufgabe:
Schreibt einen 32-Bit Wert (*ui_WriteValue*) in den Zähler des ausgewählten Moduls (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;

i_ReturnValue = i_APCI1710_Write32BitCounterValue (b_BoardHandle,
                                                   0,
                                                   200000);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe	
b_BoardHandle b_ModulNbr	ul_WriteValue
Funktionsablauf	



Ausgabe
<Return Wert>

3.6 Index

1) i_APCI1710_InitIndex (...)

Syntax:
<Return Wert> = i_APCI1710_InitIndex
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_ReferenceAction,
BYTE b_IndexOperation,
BYTE b_AutoMode,
BYTE b_InterruptEnable)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **xPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_ReferenceAction	Bestimmt, ob die Referenz für die Index-Übernahme gesetzt werden soll oder nicht. - APCI1710_ENABLE: Referenz soll für die Index-Übernahme gesetzt werden, - APCI1710_DISABLE: Referenz ist keiner Bedeutung
BYTE	b_IndexOperation	Index-Betriebsmode. Siehe Tabelle 3-10
BYTE	b_AutoMode	Aktiviert oder deaktiviert den automatischen Index-Reset - APCI1710_ENABLE: Aktiviert den automatischen Mode - APCI1710_DISABLE: Deaktiviert den automatischen Mode
BYTE	b_InterruptEnable	Aktiviert oder deaktiviert den Interrupt APCI1710_ENABLE: Interrupt aktiviert APCI1710_DISABLE: Interrupt deaktiviert

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Initialisiert den Index, der dem ausgewählten Modul entspricht (*b_ModulNbr*).
Wenn sich ein INDEX Marker (Flag) ergibt, können Sie den 32-Bit Zähler löschen oder den aktuellen 32-Bit Wert in das erste Latch-Register latchen. Der *b_IndexOperation* Parameter gibt die Möglichkeit, die INDEX Aktion auszuwählen.
Wenn Sie den automatischen Mode aktiviert haben, wird jede INDEX Aktion automatisch gelöscht. Sie sollen sonst den Index Status ("i_APCI1710_ReadIndexStatus") nach jeder INDEX Aktion lesen.

Tabelle 3-10: Index-Aktion

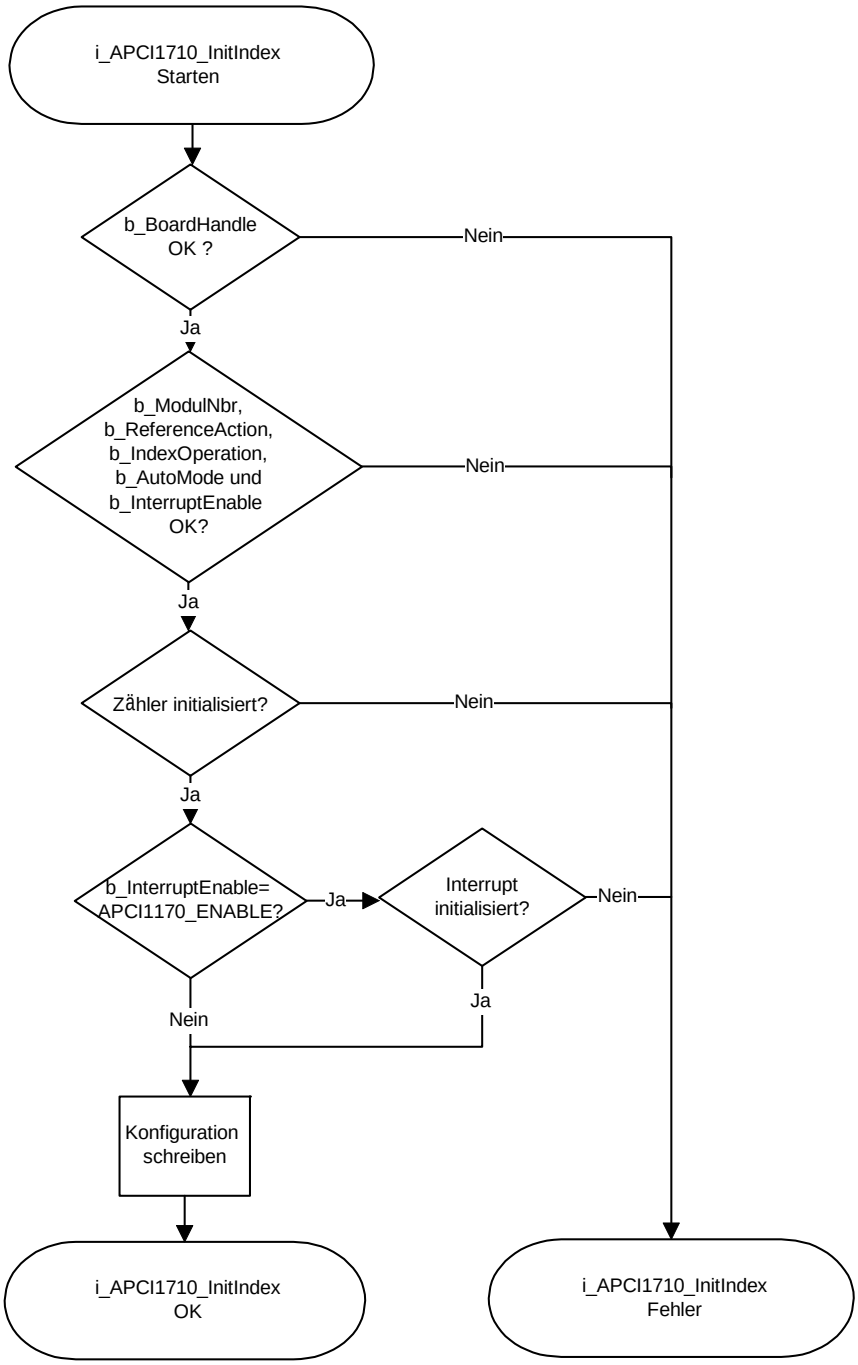
b_IndexOperation	Beschreibung
APCI1710_HIGH_EDGE_LATCH_COUNTER	Nach einem Index-Signal (High Pegel), wird der Zählerwert (32-Bit) in das erste Latch-Register gelatcht.
APCI1710_LOW_EDGE_LATCH_COUNTER	Nach einem Index-Signal (Low Pegel), wird der Zählerwert (32-Bit) in das erste Latch-Register gelatcht.
APCI1710_HIGH_EDGE_CLEAR_COUNTER	Nach einem Index-Signal (High Pegel), wird der Zählerwert gelöscht (32-Bit).
APCI1710_LOW_EDGE_CLEAR_COUNTER	Nach einem Index-Signal (Low Pegel), wird der Zählerwert gelöscht (32-Bit).
APCI1710_HIGH_EDGE_LATCH_AND_CLEAR_ COUNTER	Nach einem Index-Signal (High Pegel), wird der Zählerwert (32-Bit) in das erste Latch-Register gelatcht und dann gelöscht (32-Bit)
APCI1710_LOW_EDGE_LATCH_AND_CLEAR_ COUNTER	Nach einem Index-Signal (Low Pegel), wird der Zählerwert (32-Bit) in das erste Latch-Register gelatcht und dann gelöscht (32-Bit)

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_InitIndex
                    (b_BoardHandle,
                     0,
                     APCI1710_DISABLE,
                    APCI1710_HIGH_EDGE_LATCH_COUNTER,
                    APCI1710_ENABLE,
                    APCI1710_DISABLE);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Die ausgewählte Referenz-Aktion ist falsch.
-5: Der Index-Betriebsmode ist falsch.
-6: Der automatische Mode-Parameter ist falsch.
-7: Der Interrupt-Parameter ist falsch.
-8: Interrupt nicht initialisiert.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineXX"

Eingabe	
b_BoardHandle	b_IndexOperation
b_ModulNbr	b_AutoMode
b_ReferenceAction	b_InterruptEnable
Funktionsablauf	



Ausgabe
<Return Wert>

2) i_APCI1710_EnableIndex (...)

Syntax:
<Return Wert> = i_APCI1710_EnableIndex
(BYTEb_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Aktiviert die INDEX-Aktionen.

Funktionsaufruf:
ANSI C :

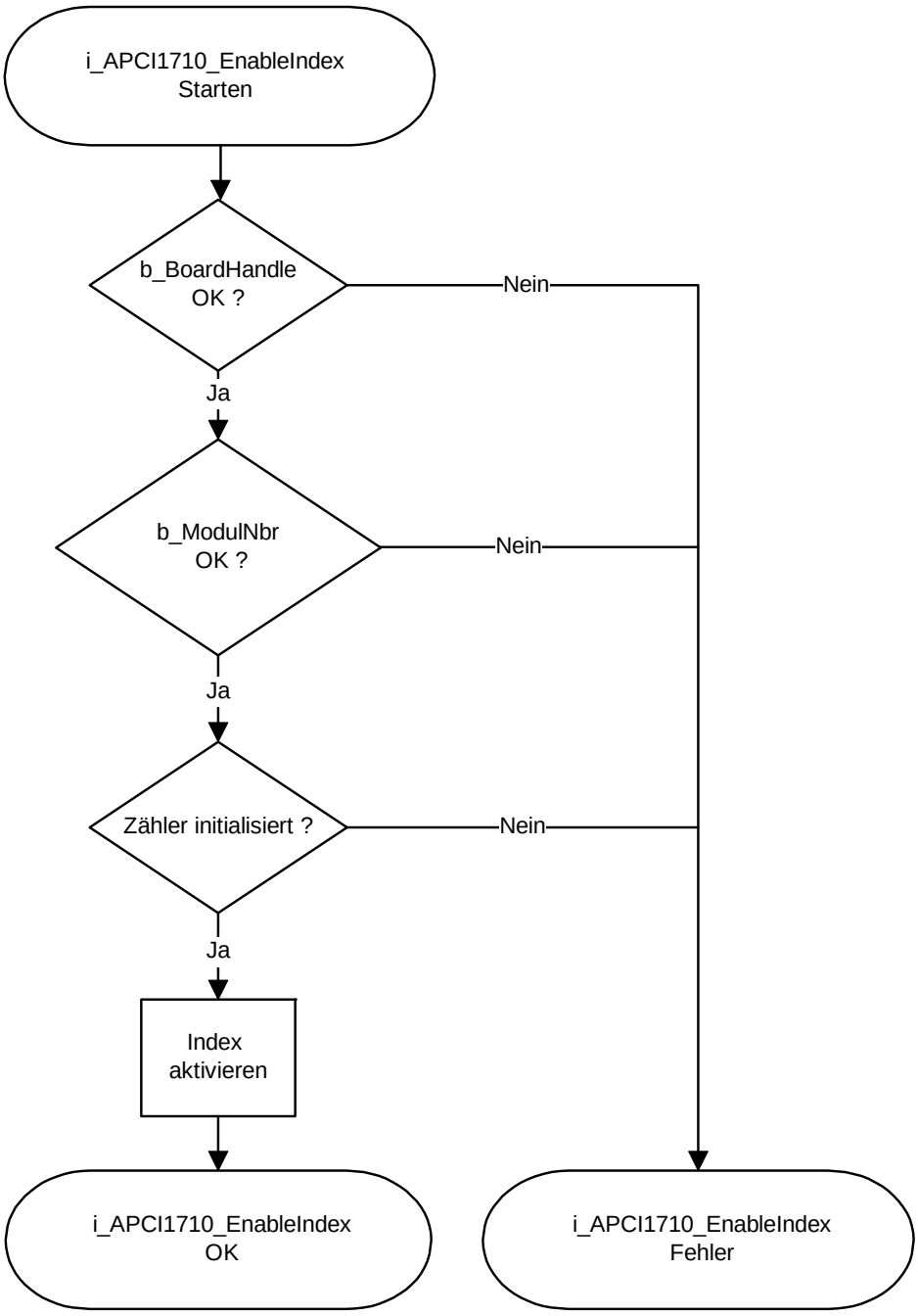
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_EnableIndex      (b_BoardHandle, 0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Index nicht initialisiert. Siehe Funktion "i_APCI1710_InitIndex"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

3) i_APCI1710_DisableIndex (...)

Syntax:
<Return Wert> = i_APCI1710_DisableIndex
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

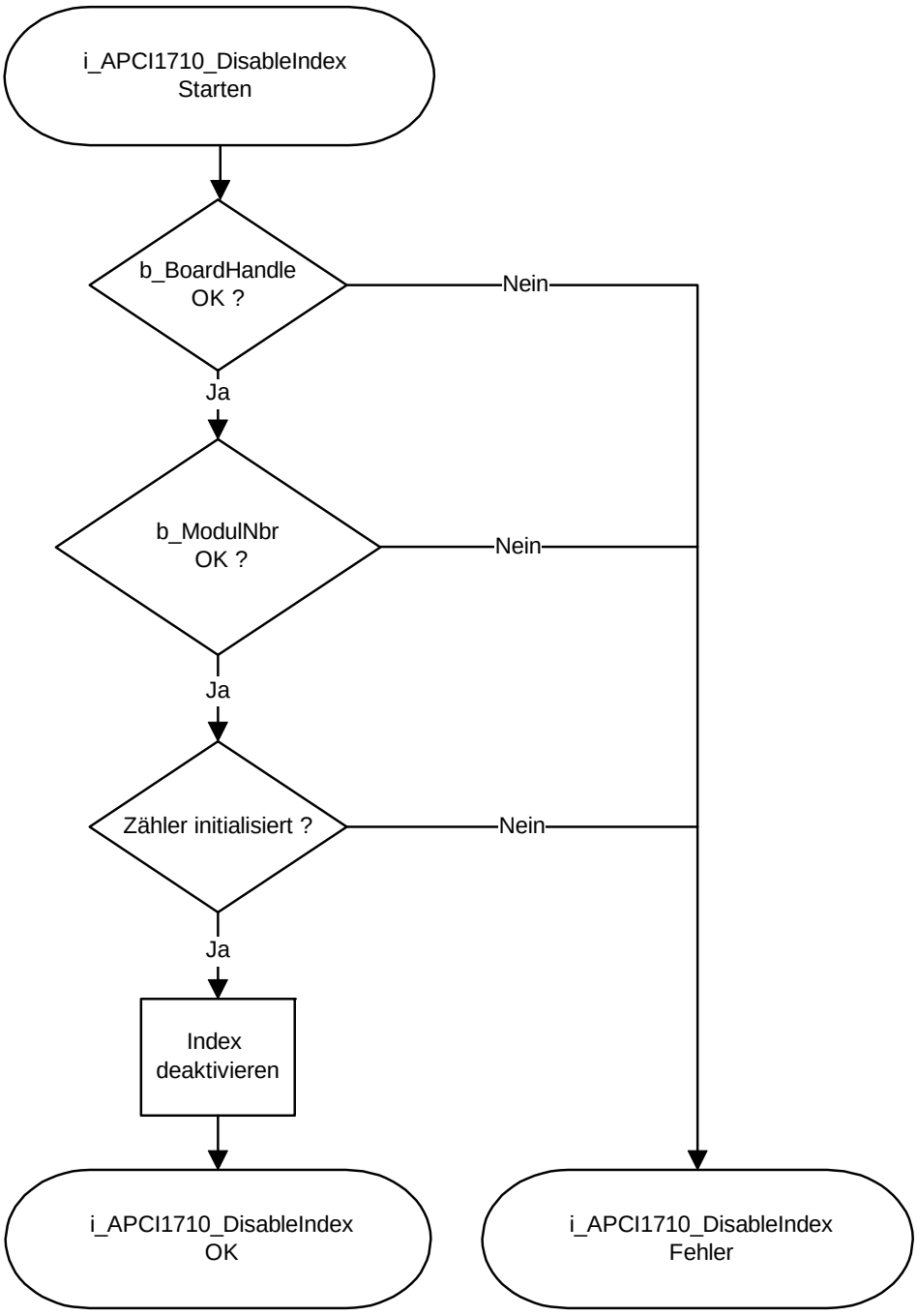
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert die INDEX-Aktionen.

Funktionsaufruf:
ANSI C :

```
int                     i_ReturnValue;
unsigned char           b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableIndex     (b_BoardHandle, 0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Index nicht initialisiert. Siehe Funktion "i_APCI1710_InitIndex"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
<Return Wert>

4) i_APCI1710_GetIndexStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetIndexStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_IndexStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

PBYTE	pb_IndexStatus	0: Kein Index 1: Ein Index wurde ermittelt

Aufgabe:
Gibt den Index-Status zurück.

Funktionsaufruf:
ANSI C :

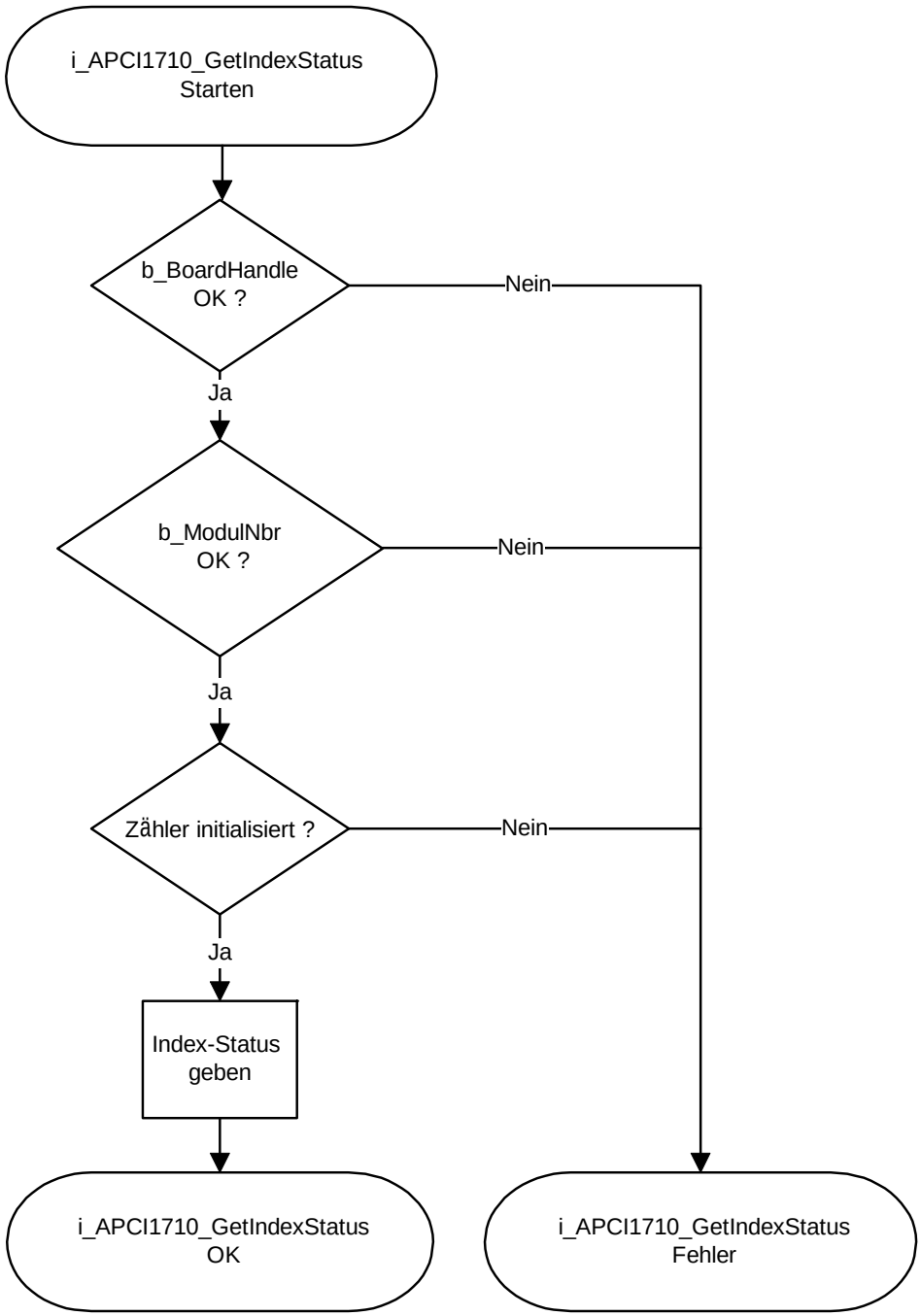
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_IndexStatus;

i_ReturnValue = i_APCI1710_ GetIndexStatus
                (b_BoardHandle,
                 0,
                 &b_IndexStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Index nicht initialisiert. Siehe Funktion "i_APCI1710_InitIndex"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

pb_IndexStatus

<Return Wert>

5) i_APCI1710_SetIndexAndReferenceSource (...)

Syntax:
<Return Wert> = i_APCI1710_SetIndexAndReferenceSource
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_SourceSelection)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_SourceSelection	Quelle des Index und der Referenzlogik Siehe Tabelle 3-11

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Ermittelt die Hardware-Quelle für den Index und die Referenzlogik. Standardmässig ist die Index-Logik an den diffentiellen Eingang C und die Referenzlogik an die 24 V des Eingangs E angeschlossen.

Tabelle 3-11: Auswahl der Index und Referenz-Quelle

b_ReferenceLevel	Anschluss
APCI1710_SOURCE_0	Die Index-Logik ist an den diff. Eingang C und die Referenz-Logik an die 24 V des Eingangs E angeschlossen. Standardkonfiguration.
APCI1710_SOURCE_1	Die Referenz-Logik ist an den diff. Eingang C und die Index-Logik an die 24 V des Eingangs E angeschlossen.

Funktionsaufruf:
ANSI C :

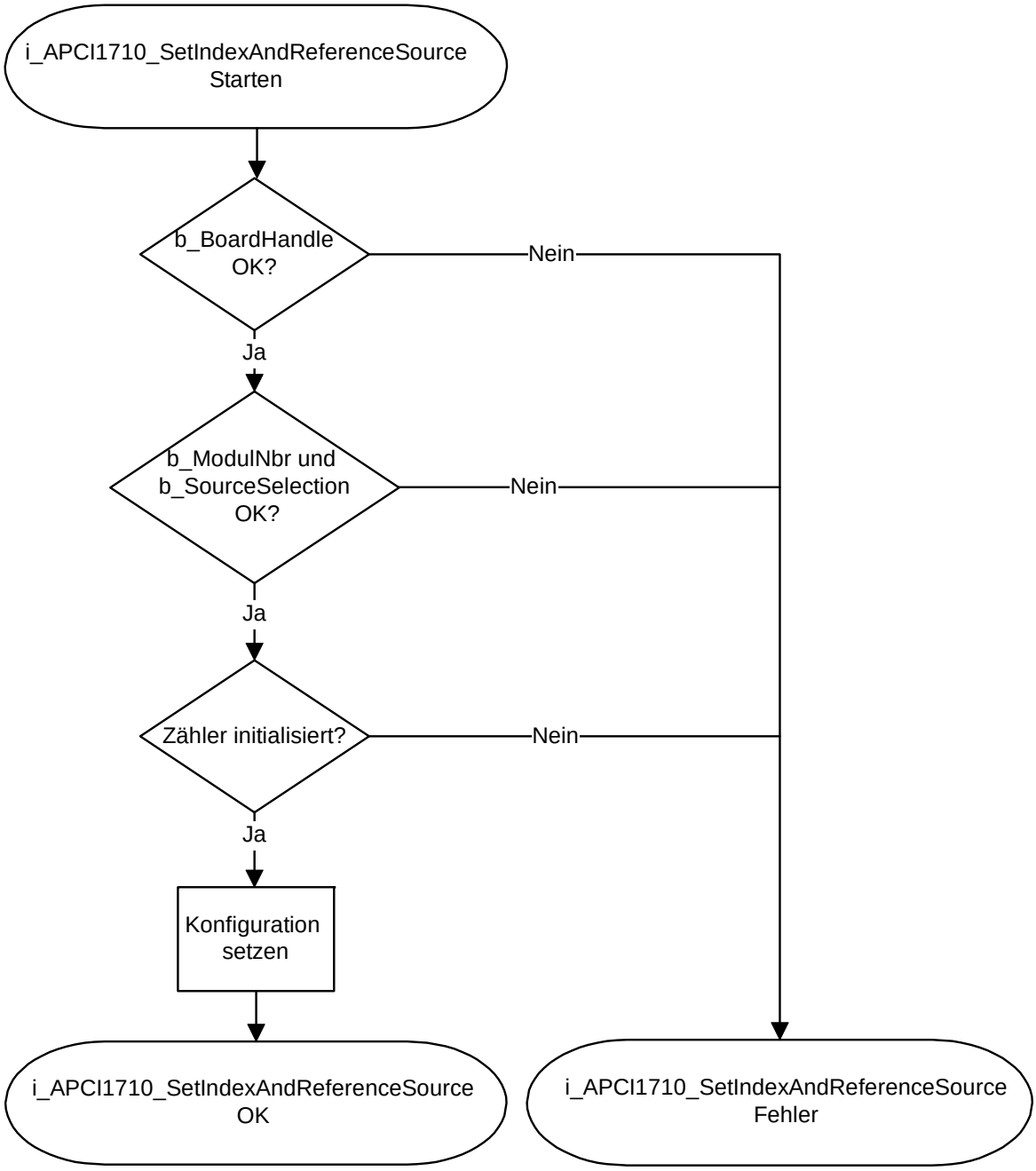
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetIndexAndReferenceSource
                                (b_BoardHandle,
                                 0,
                                 APCI1710_SOURCE_0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das Modul ist kein Zählermodul.
-4: Der Quelle-Auswahl ist falsch.

Eingabe

b_BoardHandle
b_ModulNbr
b_SourceSelection

Funktionsablauf



Ausgabe

<Return Wert>

3.7 Referenz

1) i_APCI1710_InitReference (...)

Syntax:
<Return Wert> = i_APCI1710_InitReference
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_ReferenceLevel)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_ReferenceLevel	Referenz-Pegel. Siehe Tabelle 3-12

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Initialisiert die Referenz, die dem ausgewählten Modul entspricht. (b_ModulNbr).

Tabelle 3-12: Referenz-Pegel

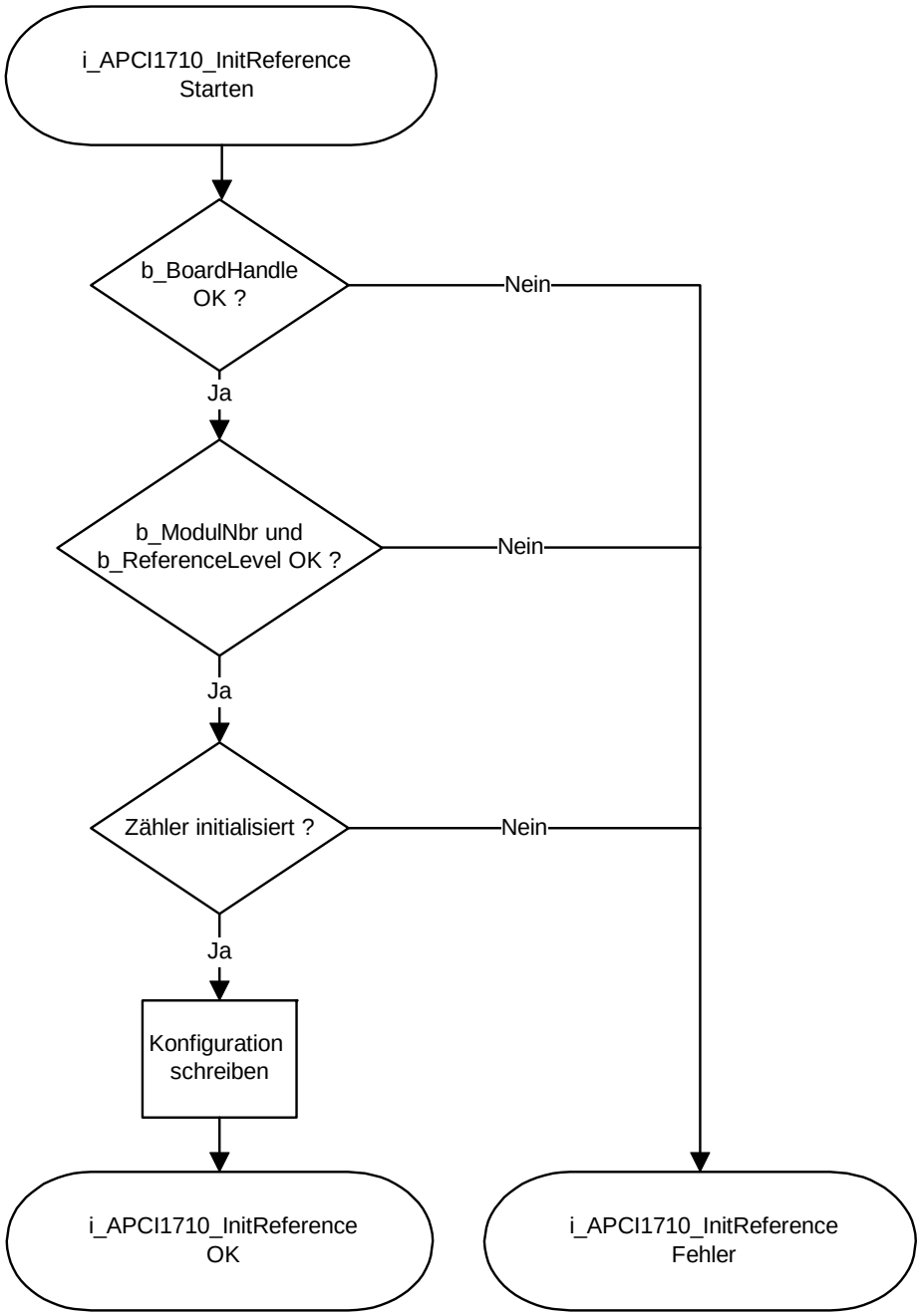
b_ReferenceLevel	Beschreibung
APCI1710_LOW	Wenn "0", Referenz initialisiert
APCI1710_HIGH	Wenn "1", Referenz initialisiert

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_InitReference
                                (b_BoardHandle,
                                 0,
                                 APCI1710_HIGH);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert Siehe Funktion "i_APCI1710_InitCounter"
-4: Referenz-Pegel ist falsch.

Eingabe	
b_BoardHandle b_ModulNbr	b_ReferenceLevel
Funktionsablauf	



Ausgabe
<Return Wert>

2) i_APCI1710_GetReferenceStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetReferenceStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_ReferenceStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

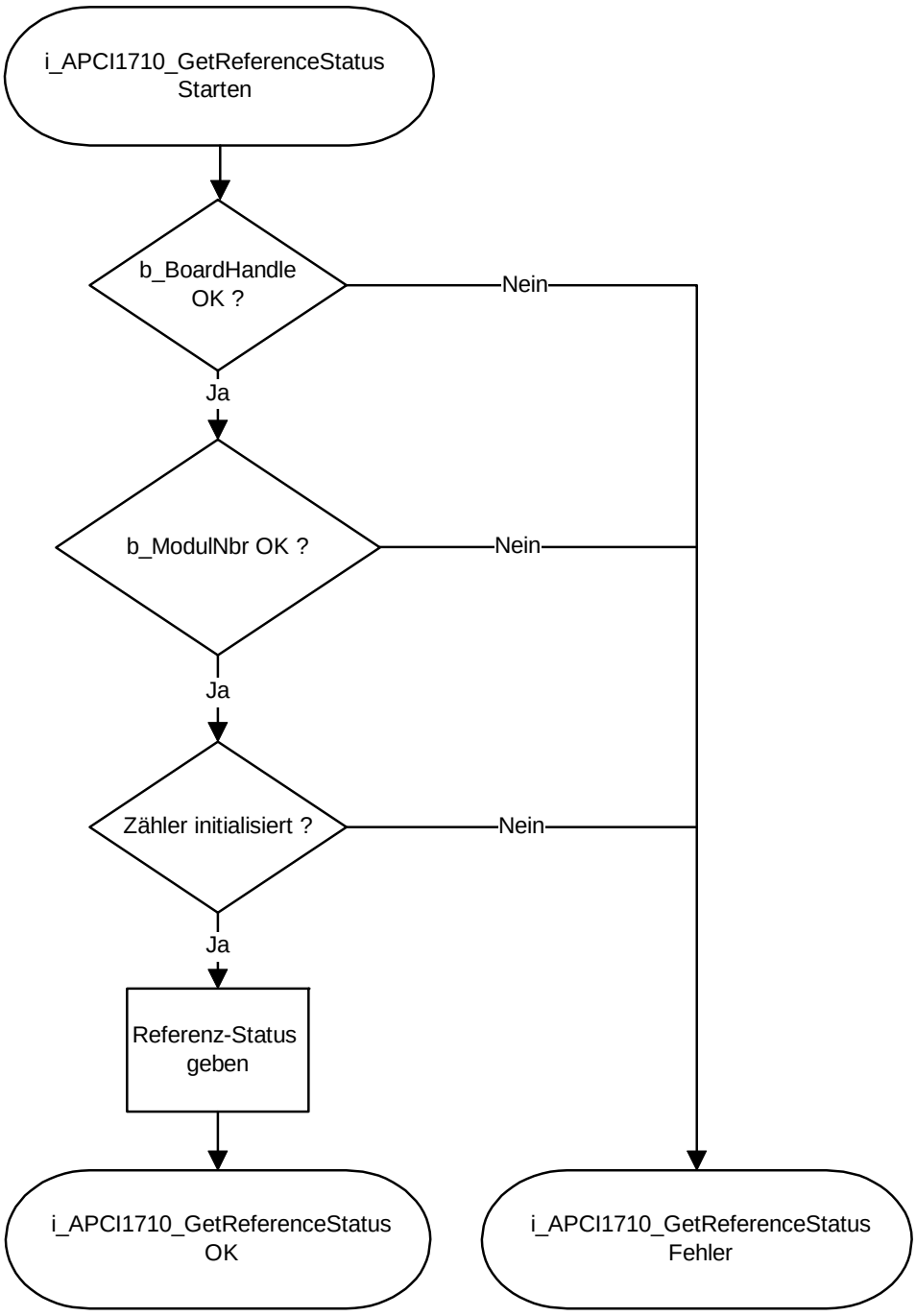
PBYTE	pb_ReferenceStatus	0: Keine Referenz 1: Eine Referenz wurde aktiviert

Aufgabe:
Gibt den Referenz-Status zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_ReferenceStatus;

i_ReturnValue = i_APCI1710_ GetReferenceStatus
                (b_BoardHandle,
                 0,
                 &b_ReferenceStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Referenz nicht initialisiert. Siehe Funktion "i_APCI1710_InitReference"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
pb_ReferenceStatus
<Return Wert>

3.8 UAS, CB, U/D#, externer Impuls (strobe)

1) i_APCI1710_GetUASStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetUASStatus
(BYTEb_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_UASStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

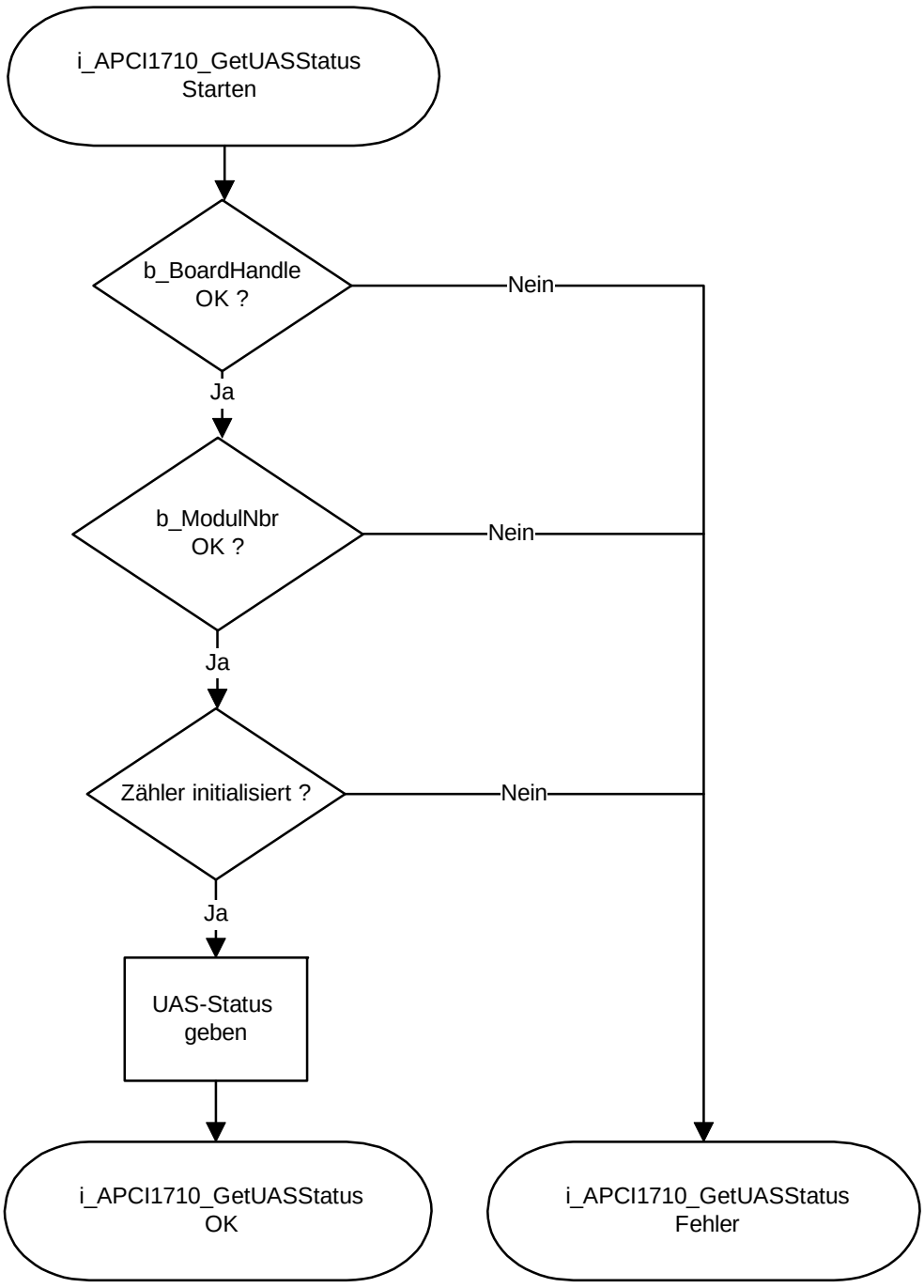
PBYTE	pb_UASStatus	0: UAS ist auf Low ("0") gesetzt 1: UAS ist auf High ("1") gesetzt

Aufgabe:
Gibt den UAS-Status zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_UASStatus;

i_ReturnValue = i_APCI1710_ GetUASStatus
                                (b_BoardHandle,
                                 0,
                                 &b_UASStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
pb_UASStatus
<Return Wert>

2) i_APCI1710_GetCBStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetCBStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_CBStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

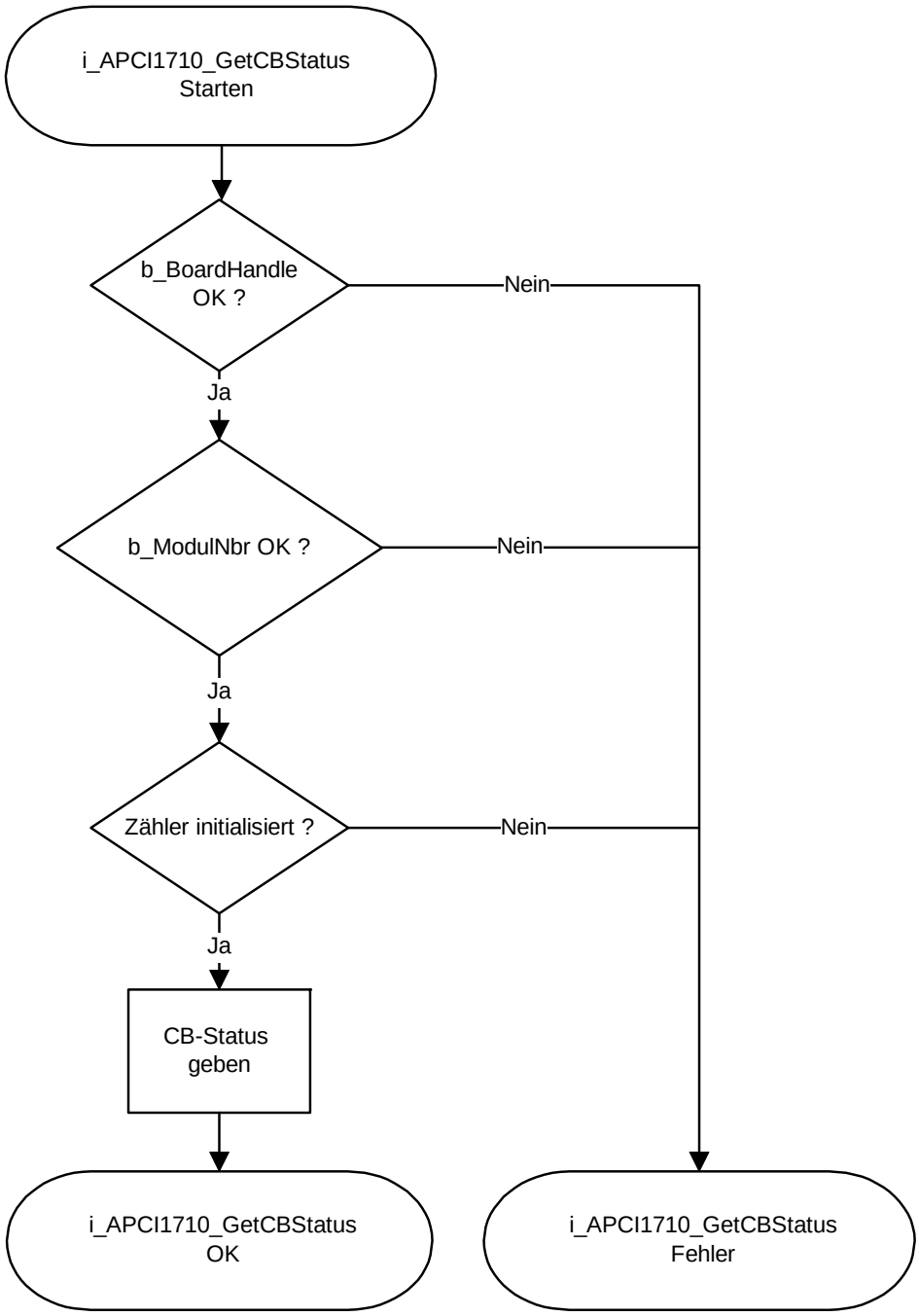
PBYTE	pb_CBStatus	0: Zähler nicht übergelaufen 1: Zähler übergelaufen

Aufgabe:
Gibt den Status des Zähler-Überlaufs zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_CBStatus;

i_ReturnValue = i_APCI1710_ GetCBStatus
                (b_BoardHandle,
                 0,
                 &b_CBStatus);
```

Return Wert:
0: Kein Fehler
1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
pb_CBStatus
<Return Wert>

3) i_APCI1710_GetUDStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetUDStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_UDStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

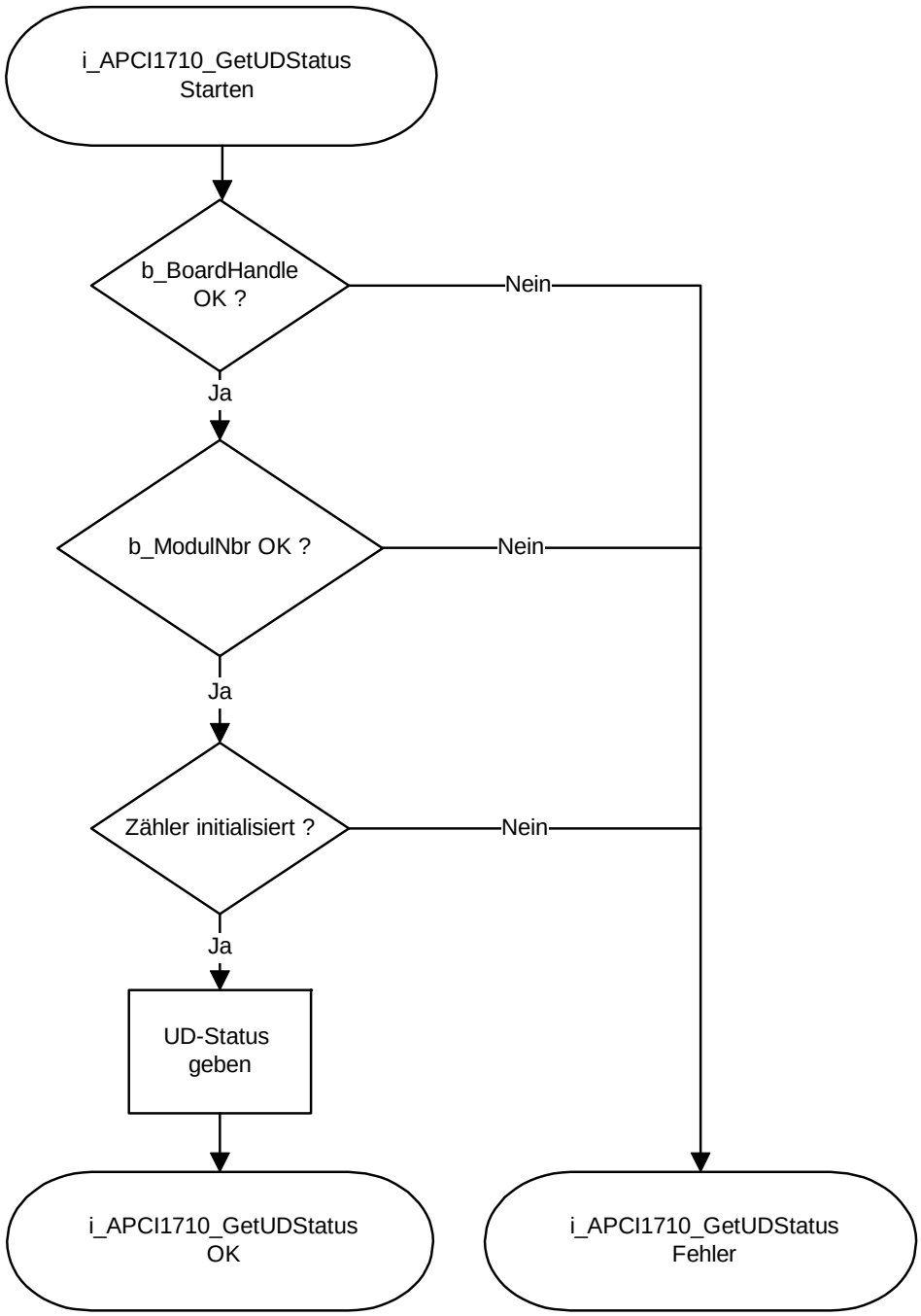
PBYTE	pb_UDStatus	0: Zähler-Ablauf im ausgewählten Mode abwärts. Siehe auch Tabelle 3-5 1: Zähler-Ablauf im ausgewählten Mode aufwärts. Siehe auch Tabelle 3-5

Aufgabe:
Gibt den Status des Zähler-Ablaufs zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_UDStatus;

i_ReturnValue = i_APCI1710_ GetCBStatus
                (b_BoardHandle,
                 0,
                 &b_UDStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
pb_UDStatus
<Return Wert>

4) i_APCI1710_GetInterruptUDLatchedStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetInterruptUDLatchedStatus
(BYTEb_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_UDStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

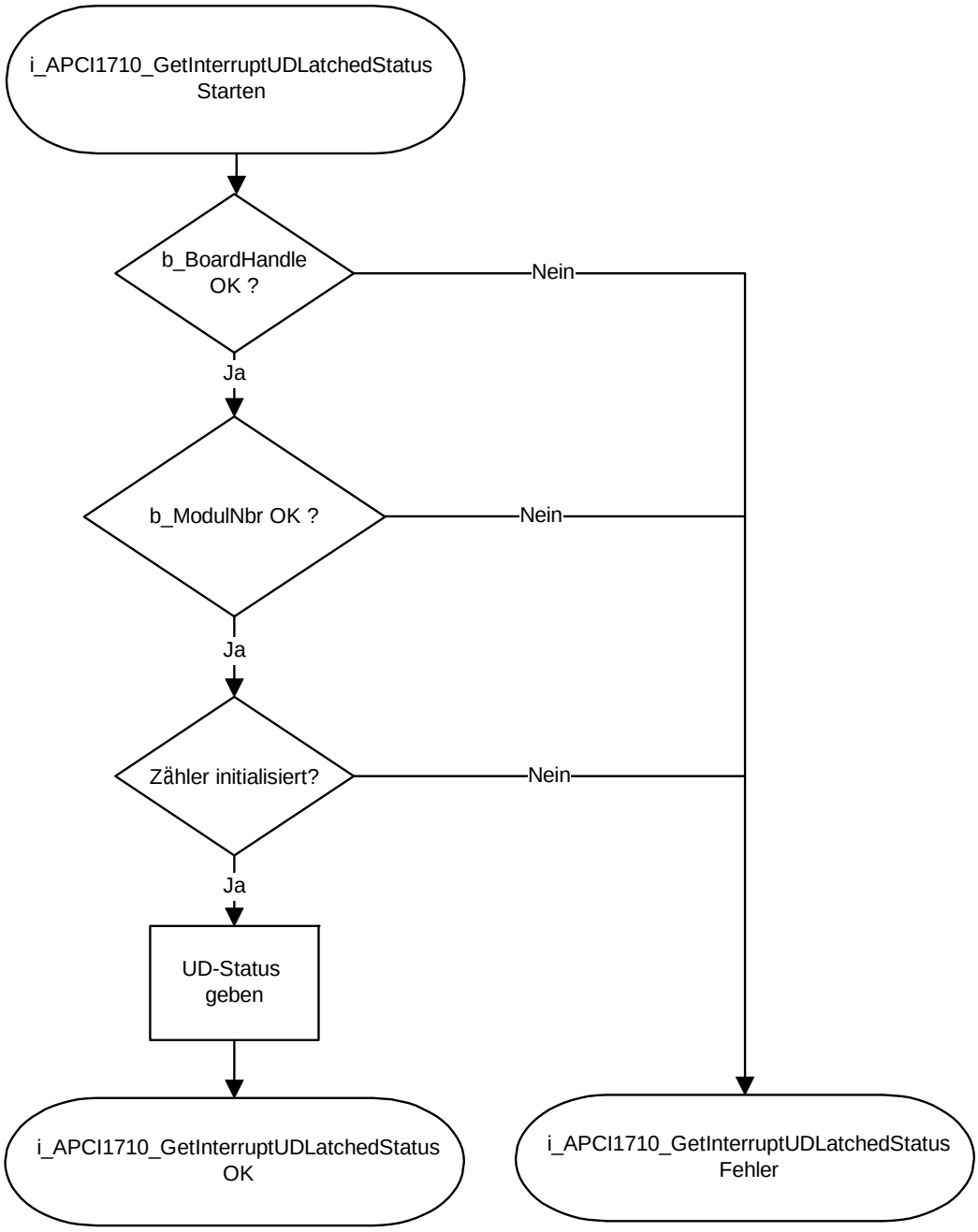
PBYTE	pb_UDStatus	0: Zähler-Ablauf im ausgewählten Mode abwärts. 1: Zähler-Ablauf im ausgewählten Mode aufwärts. 2: Es erfolgt kein Index-Interrupt Siehe Funktion "i_APCI1710_InitCounter"

Aufgabe:
Gibt den gelatchten Status des Zähler-Ablaufs zurück, nachdem ein Index-Interrupt generiert wurde.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_UDStatus;

i_ReturnValue = i_APCI1710_GetInterruptUDLatchedStatus
                                    (b_BoardHandle,
                                     0,
                                     &b_UDStatus);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Interruptfunktion nicht initialisiert.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineX"

Eingabe
b_BoardHandle b_ModulNbr
Funktionsablauf

Ausgabe
pb_UDStatus
<Return Wert>

5) i_APCI1710_InitExternalStrobe (...)

Syntax:
<Return Wert> = i_APCI1710_InitExternalStrobe
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_ExternalStrobe,
BYTE b_ExternalStrobeLevel)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_ExternalStrobe	Auswahl des externen Impuls (strobe) 0: Externer Impuls A 1: Externer Impuls B
BYTE	b_ExternalStrobeLevel	Pegel des externen Impuls. Siehe Tabelle 3-13

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Initialisiert den externen Impuls-Pegel für das ausgewählte Modul (*b_ModulNbr*).

Tabelle 3-13: Externer Impuls-Pegel

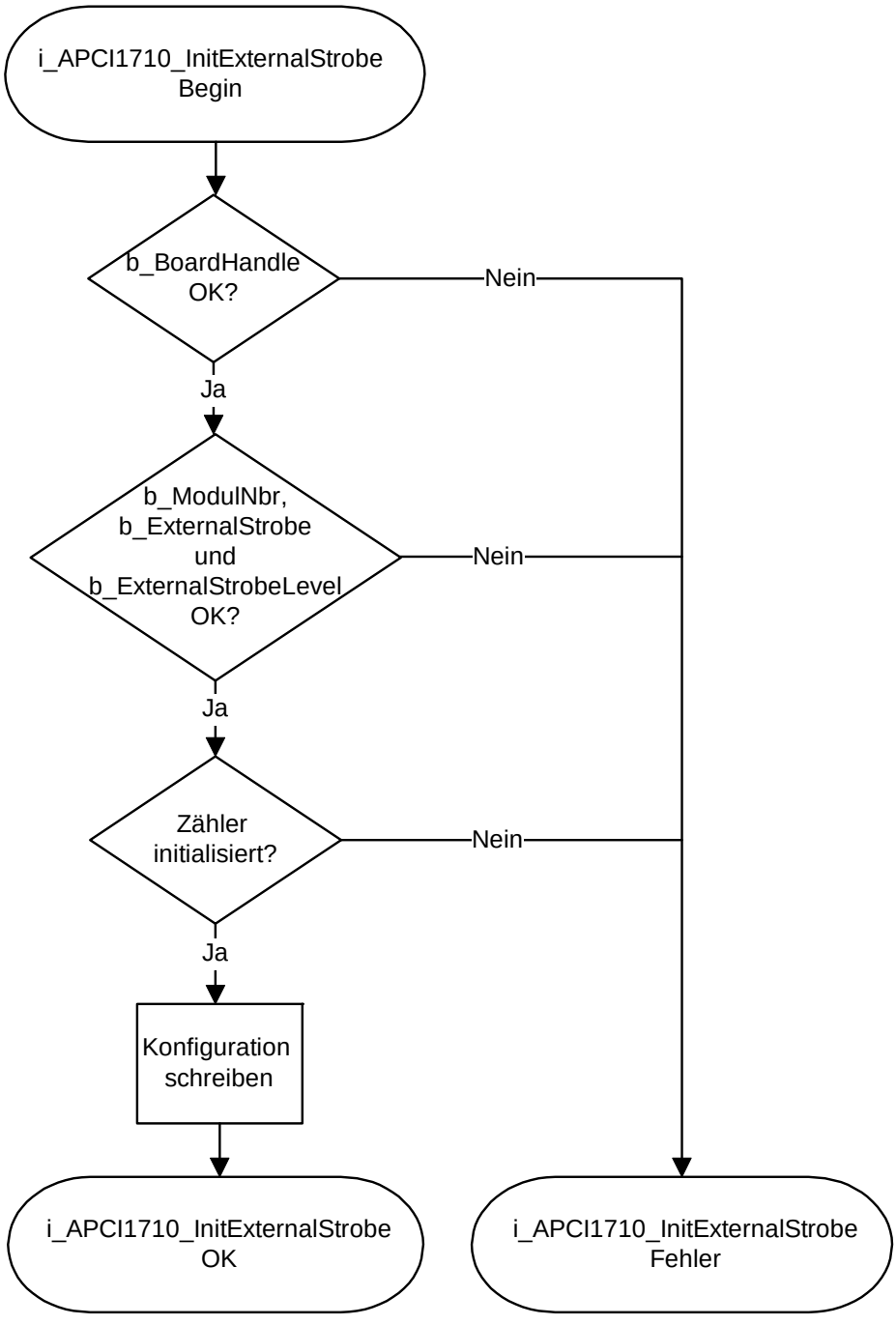
b_ReferenceLevel	Beschreibung
APCI1710_LOW	Externes Latch wird bei "0"ausgelöst
APCI1710_HIGH	Externes Latch wird bei "1"ausgelöst (Standardkonfiguration)

Funktionsaufruf:
ANSI C :

```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;
i_ReturnValue = i_APCI1710_InitExternalStrobe
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     APCI1710_HIGH);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Der ausgewählte externe Impuls ist falsch.
-5: Externimpuls-Pegel ist falsch.

Eingabe	
b_BoardHandle b_ModulNbr	b_ExternalStrobe b_ExternalStrobeLevel
Funktionsablauf	



Ausgabe
<Return Wert>

3.9 Vergleichslogik

1) i_APCI1710_InitCompareLogic (...)

Syntax:
<Return Wert> = i_APCI1710_InitCompareLogic
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
UINT ui_CompareValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
UINT	ui_CompareValue	32-Bit Vergleich-Wert

- Ausgabe:
Es erfolgt keine Ausgabe.

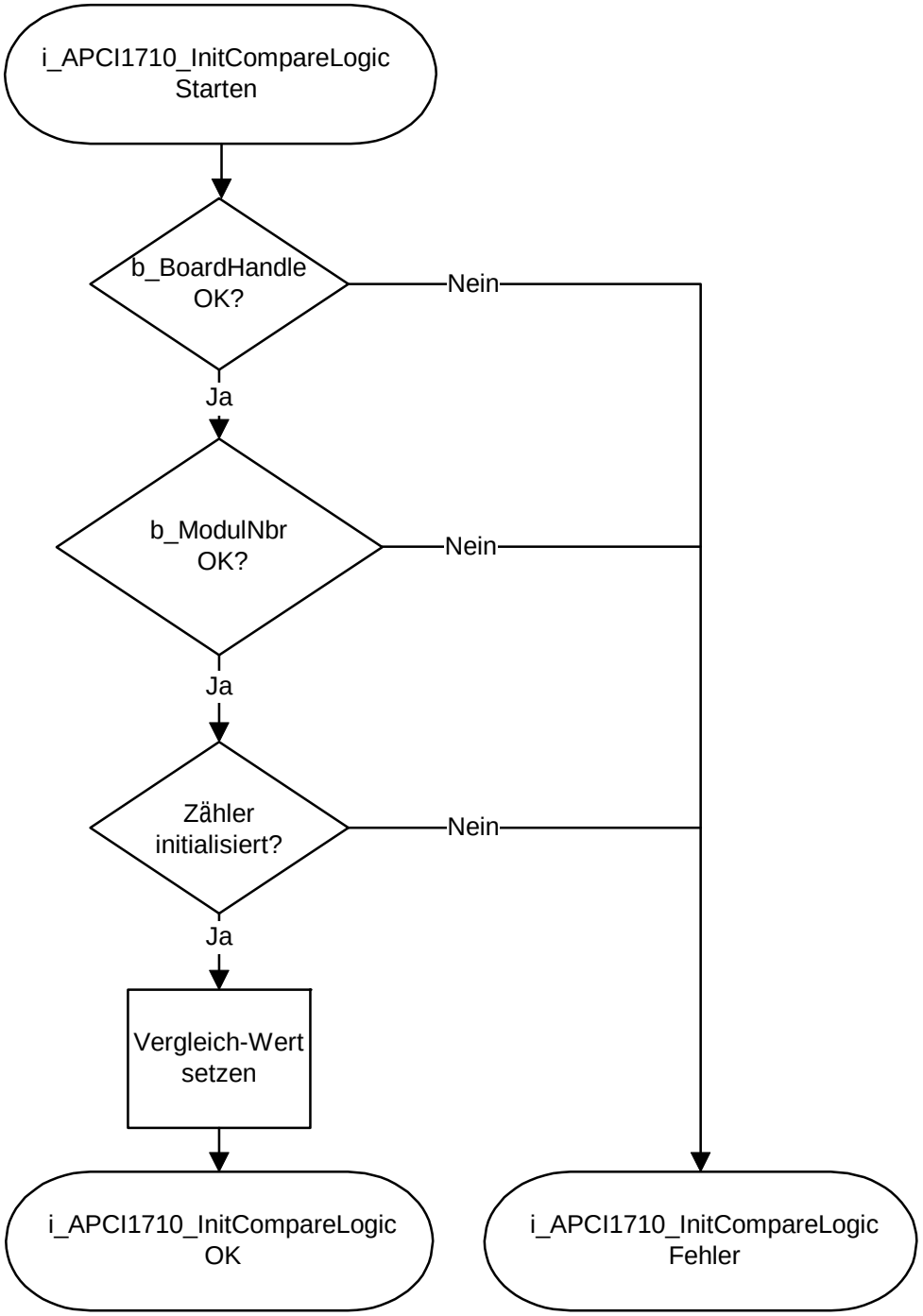
Aufgabe:
Setzt den 32-Bit Vergleich-Wert. Ein Interrupt wird generiert, sobald der Zähler den Vergleich-Wert (*ui_CompareValue*) erreicht hat.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_InitCompareLogic (b_BoardHandle,
                                             0,
                                             0xFF55);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe	
b_BoardHandle b_ModulNbr	ui_CompareValue
Funktionsablauf	



Ausgabe
<Return Wert>

2) i_APCI1710_EnableCompareLogic (...)

Syntax:
<Return Wert> = i_APCI1710_EnableCompareLogic
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

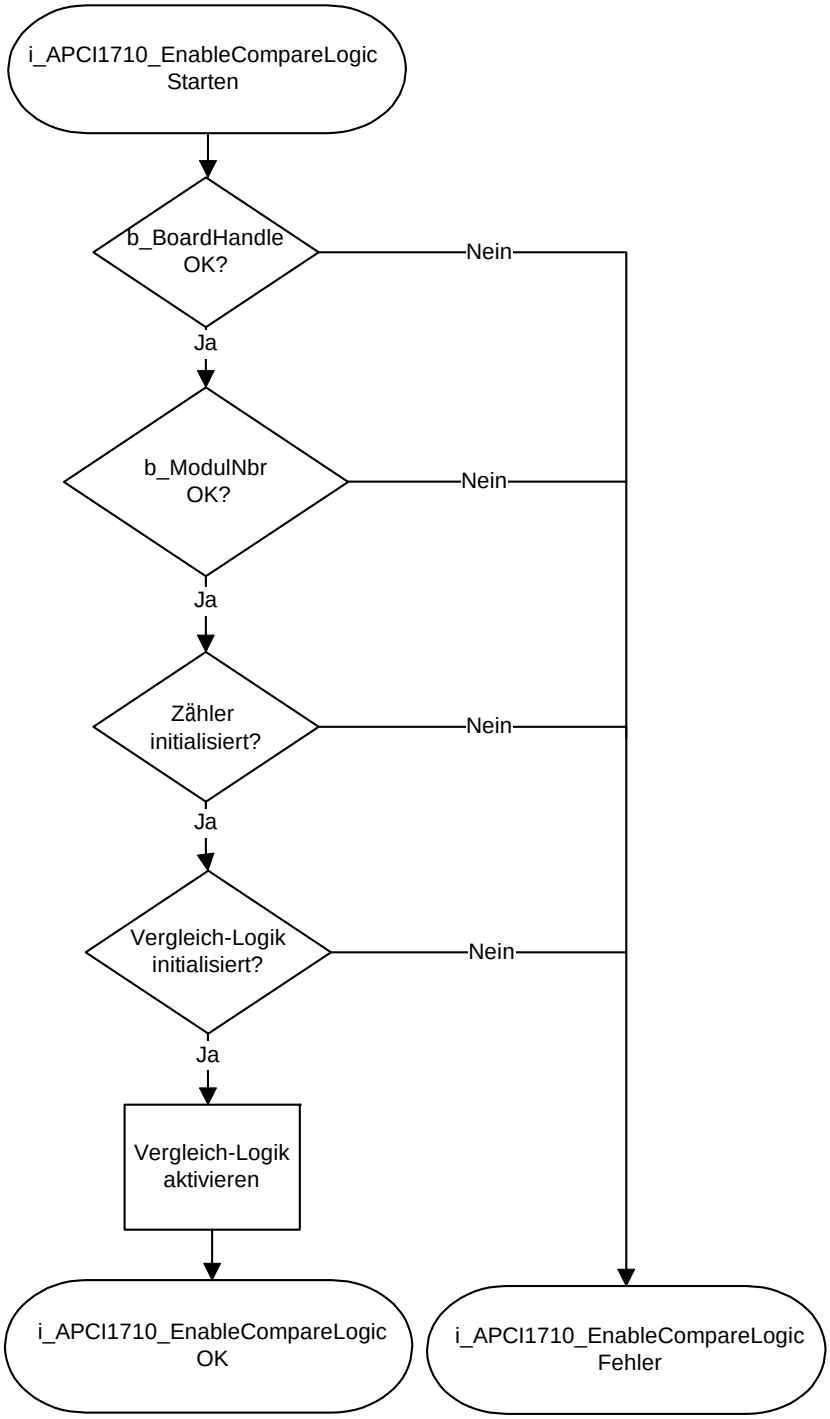
Aufgabe:
Aktiviert die 32-Bit Vergleich-Logik. Ein Interrupt wird generiert, sobald der Zähler den Vergleich-Wert (*ui_CompareValue*) erreicht hat.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_EnableCompareLogic   (b_BoardHandle,
                                                 0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Vergleich-Logik nicht initialisiert.
Siehe Funktion "i_APCI1710_InitCompareLogic"
-5: Interruptfunktion nicht initialisiert
Siehe Funktion "i_APCI1710_SetBoardIntRoutineX"

Eingabe	
b_BoardHandle	b_ModulNbr
Funktionsablauf	



Ausgabe
<Return Wert>

3) i_APCI1710_DisableCompareLogic (...)

Syntax:
<Return Wert> = i_APCI1710_DisableCompareLogic
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

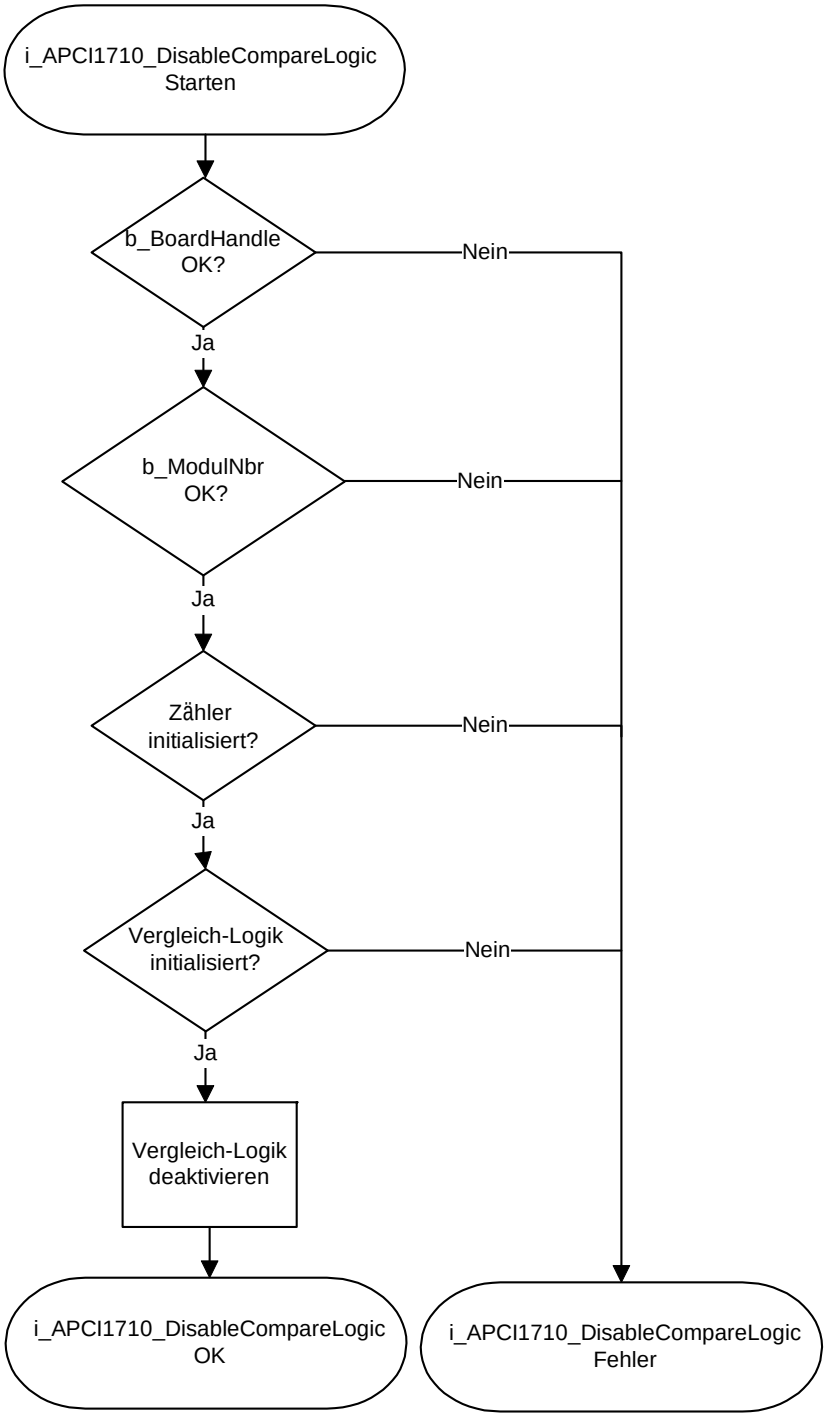
Aufgabe:
Deaktiviert die 32-Bit Vergleich-Logik.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableCompareLogic  (b_BoardHandle,
                                                 0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Vergleich-Logik nicht initialisiert. Siehe Funktion "i_APCI1710_InitCompareLogic"

Eingabe	
b_BoardHandle	b_ModulNbr
Funktionsablauf	



Ausgabe
<Return Wert>

3.10 Frequenzmessung

1) i_APCI1710_InitFrequencyMeasurement (...)

Syntax:

```
<Return Wert> =  i_APCI1710_InitFrequencyMeasurement
                                    (BYTE       b_BoardHandle,
                                     BYTE       b_ModulNbr,
                                     BYTE       b_PCIInputClock,
                                     BYTE       b_TimeUnit,
                                     ULONG      ul_TimeInterval,
                                     PULONG     pul_RealTimeInterval)
```

Parameter:

- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_PCIInputClock	Auswahl des PCI Bus Takts - APCI1710_30MHZ: Das PC hat einen PCI Bus Takt von 30 MHz - APCI1710_33MHZ: Das PC hat einen PCI Bus Takt von 33 MHz - APCI1710_40MHZ: Das PC hat einen PCI Bus Takt von 40 MHz
BYTE	b_TimeUnit	Einheit der Zeitbasis (0 bis 2) 0: ns 1: µs 2: ms
ULONG	ul_TimeInterval	Wert der Zeitbasis. Siehe Tabelle "Wert der Zeitbasis"

- Ausgabe:

PULONG	pul_RealTimeInterval	Richtiger Wert der Zeitbasis Gibt den Wert zurück, der dem in ul_TimingInterval eingegebenen Wert so genau wie möglich ähnelt.

Tabelle 3-14: Wert der Zeitbasis

PCI-Bus- Takt	*b_TimingUnit*	*ul_TimingInterval* Minimalwert	*ul_TimingInterval* Maximalwert
30 MHz	ns (0)	266	8738133
	µs (1)	1	8738
	ms (2)	1	8
33 MHz	ns (0)	242	7943757
	µs (1)	1	7943
	ms (2)	1	7
40 MHz	ns (0)	200	6553500
	µs (1)	1	6553
	ms (2)	1	6

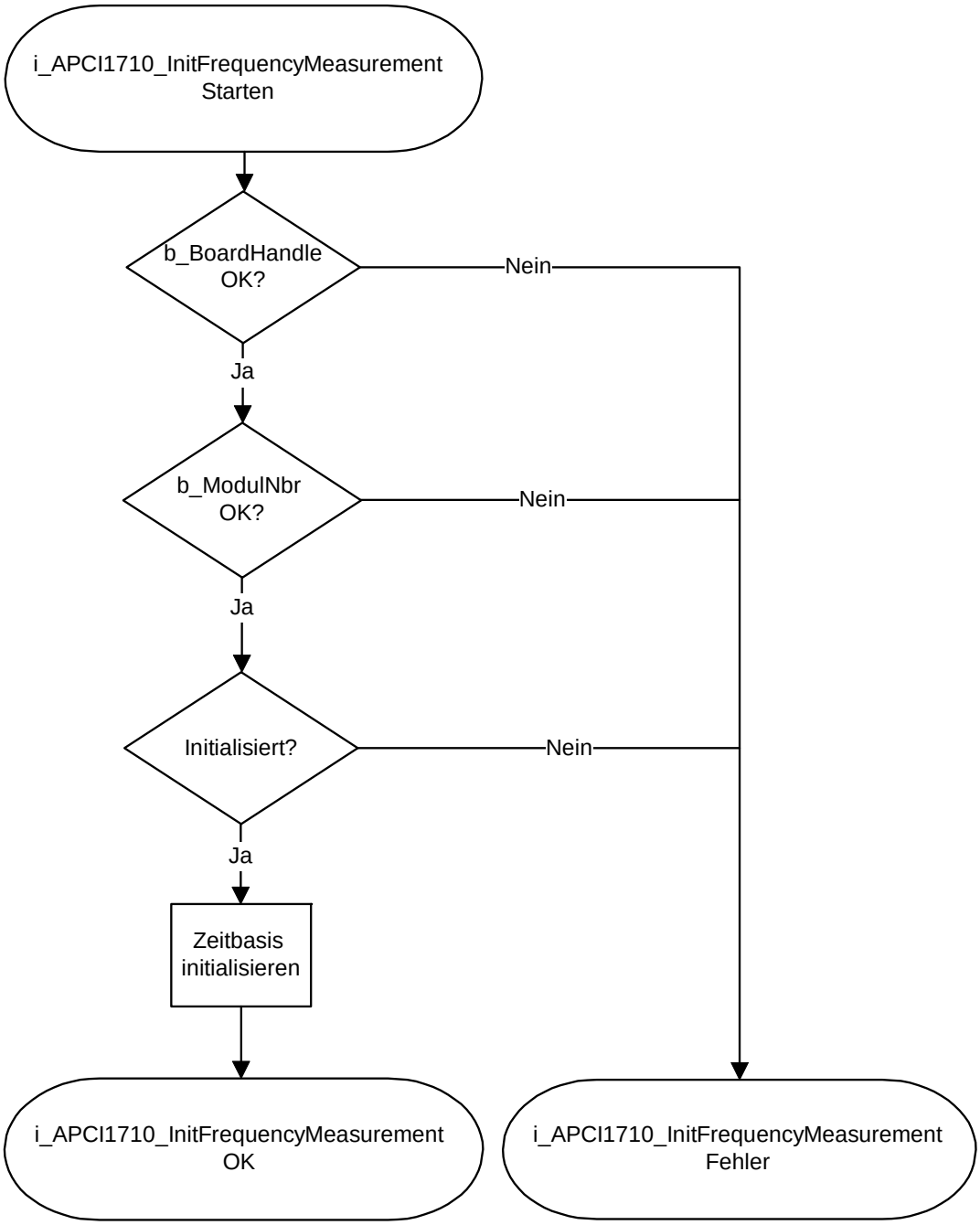
Aufgabe:
Setzt die Zeit für die Frequenzmessung.
Konfiguriert den Inkrementalzähler des ausgewählten Moduls (*b_ModulNbr*). Die Parameter *ul_TimingInterval* und *ul_TimingUnit* legen die Zeitbasis für die Messung fest. *pul_RealTimingInterval* gibt den richtigen Zeitwert zurück. Diese Funktion soll aufgerufen werden, bevor Sie eine andere Funktion aufrufen, die auf die Frequenzmessung zugreift.

Funktionsaufruf:
ANSI C :

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

i_ReturnValue = i_APCI1710_InitFrequencyMeasurement
                                (b_BoardHandle,
                                 0,
                                 APCI1710_33MHZ,
                                 2,
                                 1,
                                 &ul_RealTimingInterval);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Der ausgewählte PCI Eingangstakt ist falsch.
-5: Die ausgewählte Zeiteinheit ist falsch.
-6: Die ausgewählte Zeitbasis ist falsch.
- 7: Auf der Karte ist kein 40 MHz Quarz eingebaut.

Eingabe	
b_BoardHandle b_ModulNbr b_PCIInputClock	b_TimingUnit, ul_TimingInterval
Funktionsablauf	



Ausgabe
pul_RealTimingInterval
<Return Wert>

2) i_APCI1710_EnableFrequencyMeasurement (...)

Syntax:
<Return Wert> = i_APCI1710_EnableFrequencyMeasurement
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_InterruptEnable)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_InterruptEnable	Aktiviert oder deaktiviert den Interrupt APCI1710_ENABLE: Interrupt aktiviert APCI1710_DISABLE: Interrupt deaktiviert

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Aktiviert die Funktion für die Frequenzmessung.

Funktionsaufruf:
ANSI C :

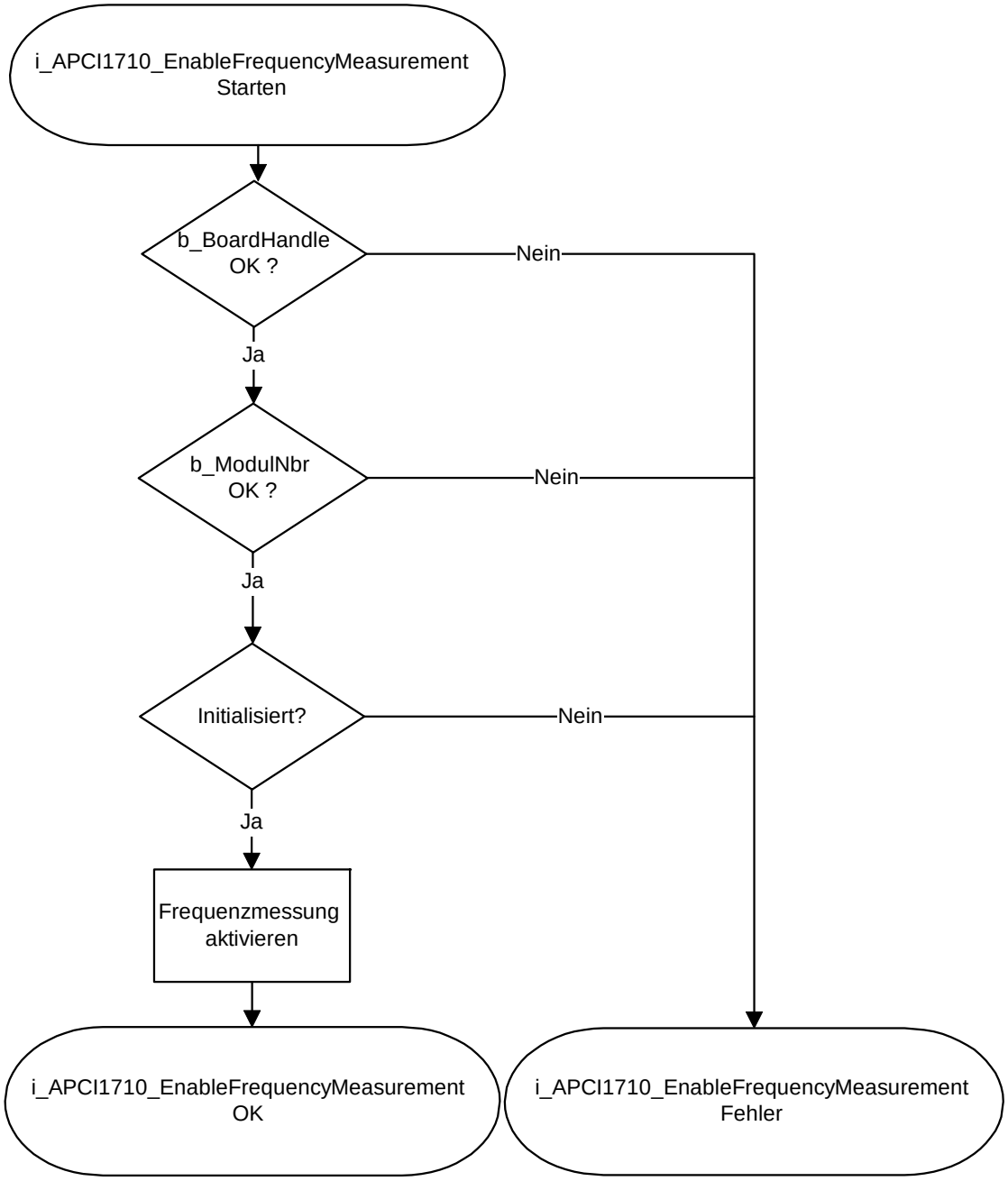
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_ EnableFrequencyMeasurement
                (b_BoardHandle,
                 0,
                 APCI1710_DISABLE);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Die Logik der Frequenzmessung ist nicht initialisiert.
Siehe Funktion "i_APCI1710_InitFrequencyMeasurement"
-5: Interruptparameter ist falsch.
-6: Interruptfunktion nicht initialisiert.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineX"

Eingabe

b_BoardHandle b_InterruptEnable
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

3) i_APCI1710_ReadFrequencyMeasurement (...)

Syntax:
<Return Wert> = i_APCI1710_ReadFrequencyMeasurement
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_Status,
PBYTE pb_UDStatus,
PULONG pul_ReadValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:

PBYTE	pb_Status	gibt den Status der Frequenzmessung zurück 0: Zählablauf nicht gestartet. 1: Zählablauf gestartet. 2: Zählablauf beendet
PBYTE	pb_UDStatus	Status des Aufwärts-/Abwärts-Zählers. Siehe Tabelle 3-15
PULONG	pul_ReadValue	Gibt die Anzahl der Inkrementen innerhalb der Zeitbasis zurück

Tabelle 3-15: Status des Zählerablaufs

	2 x 16-bit Zähler-Mode		**1 x 32-Bit Zähler-Mode**
pb_UDStatus	Zweiter 16-Bit Zähler	Erster 16-Bit Zähler	
0	Zählerablauf abwärts	Zählerablauf abwärts	Zählerablauf abwärts
1	Zählerablauf abwärts	Zählerablauf aufwärts	X
2	Zählerablauf aufwärts	Zählerablauf abwärts	X
3	Zählerablauf aufwärts	Zählerablauf aufwärts	Zählerablauf aufwärts

Aufgabe:
Gibt den Status (*pb_Status*) und die Anzahl der Inkrementen in der gesetzten Zeit zurück. Siehe Funktion "i_APCI1710_InitFrequencyMeasurement"

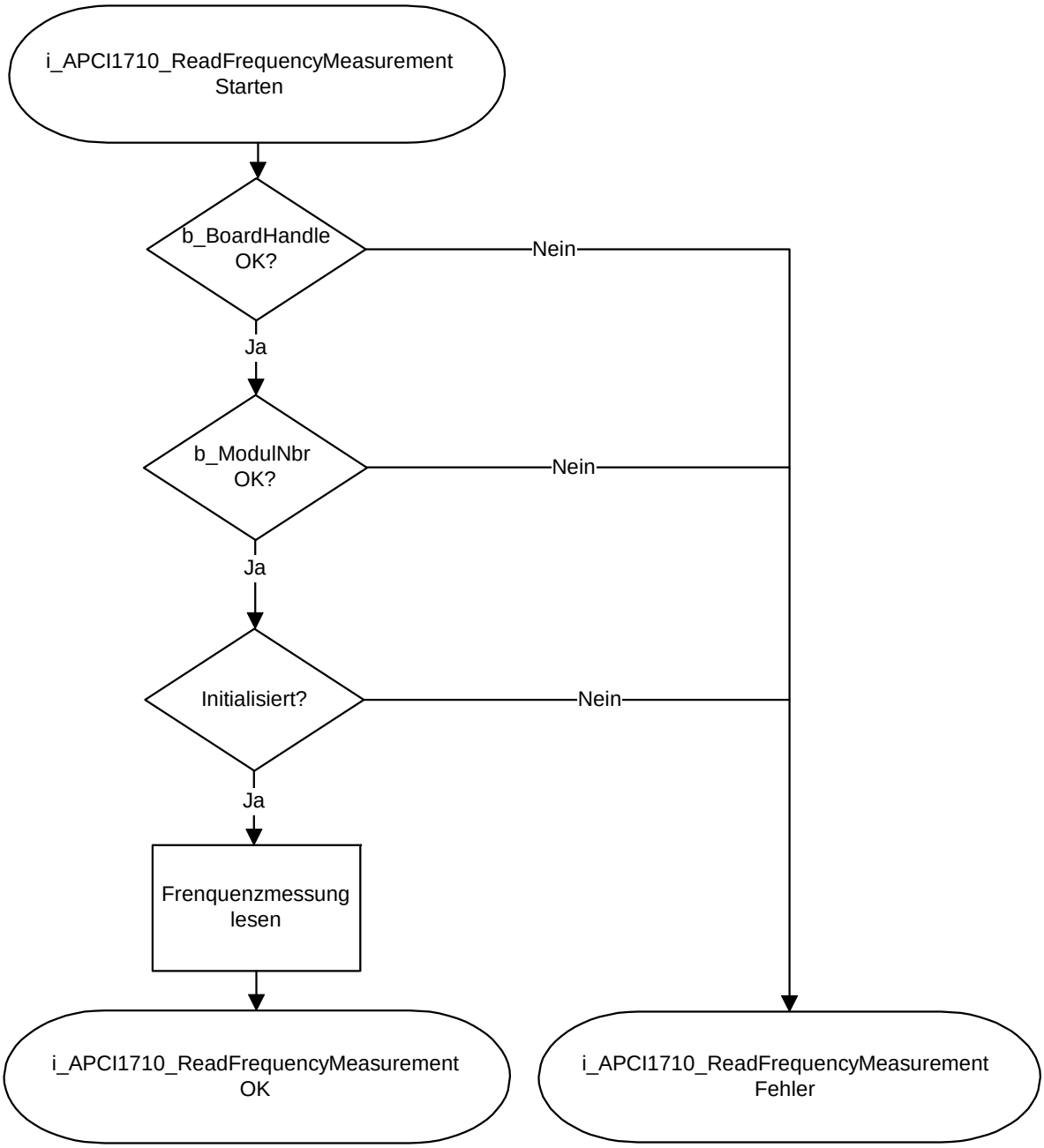
Funktionsaufruf:
ANSI C :

```
int                     i_ReturnValue;
unsigned char           b_BoardHandle;
unsigned char           b_Status;
unsigned char           b_UDStatus;
unsigned long           ul_ReadValue;

i_ReturnValue = i_APCI1710_ReadFrequencyMeasurement
                                    (b_BoardHandle,
                                     0,
                                     &b_Status,
                                     &b_UDStatus,
                                     &ul_ReadValue);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Die Logik der Frenquenzmessung ist nicht initialisiert.
Siehe Funktion "i_APCI1710_InitFrequencyMeasurement"

Eingabe	
b_BoardHandle	b_ModulNbr

Funktionsablauf



Ausgabe	
pb_Status pb_UDStatus	pul_ReadValue

<Return Wert>

4) i_APCI1710_DisableFrequencyMeasurement (...)

Syntax:
<Return Wert> = i_APCI1710_DisableFrequencyMeasurement
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert die Frequenzmessung.

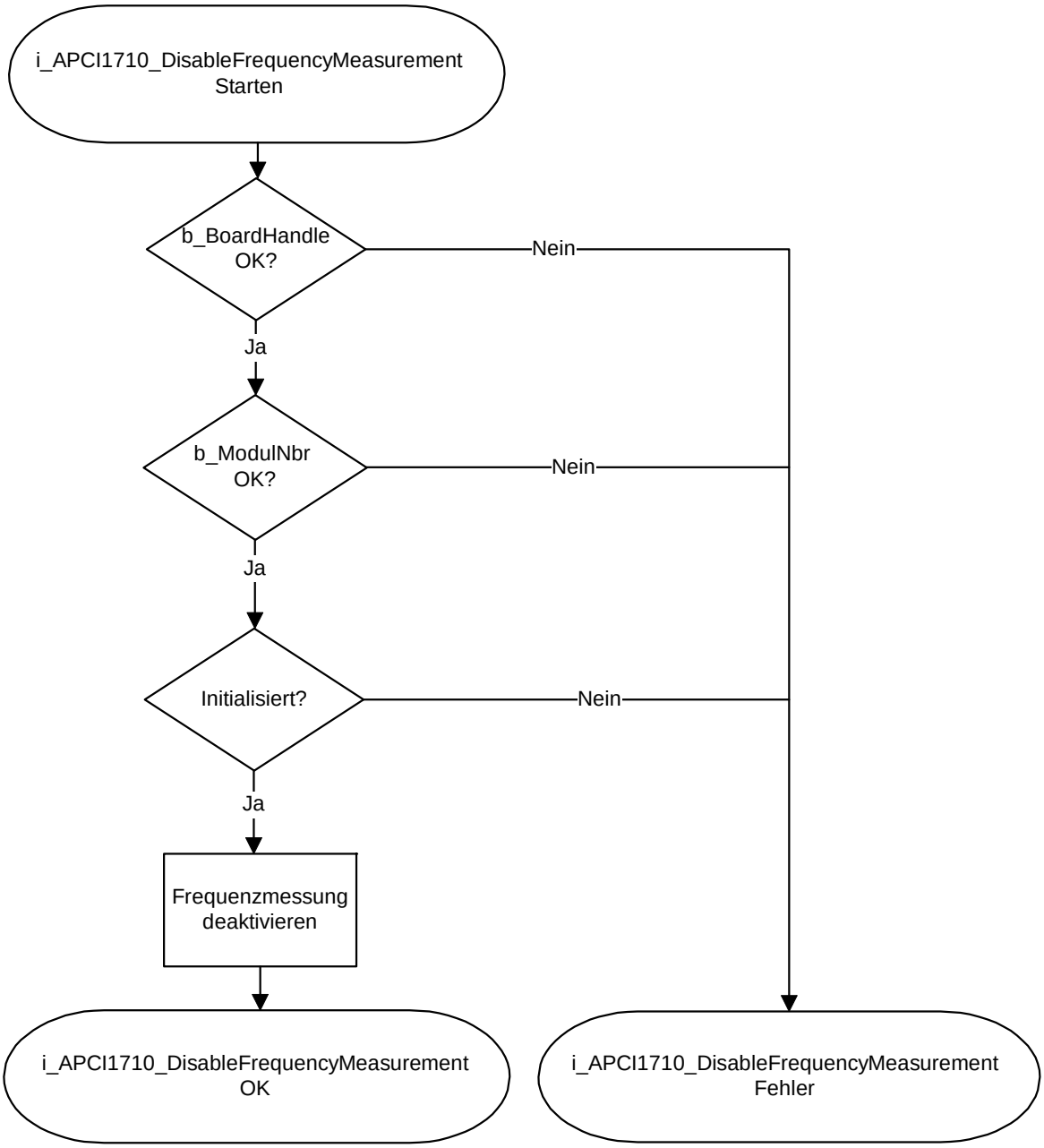
Funktionsaufruf:

ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableFrequencyMeasurement
                                        (b_BoardHandle,
                                         0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"
-4: Die Logik der Frequenzmessung ist nicht initialisiert.
Siehe Funktion "i_APCI1710_InitFrequencyMeasurement"

Eingabe	
b_BoardHandle	b_ModulNbr
Funktionsablauf	



Ausgabe
<Return Wert>

3.11 Digitaler Ausgang

1) i_APCI1710_SetDigitalChlOn (...)

Syntax:
<Return Wert> = i_APCI1710_SetDigitalChlOn
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

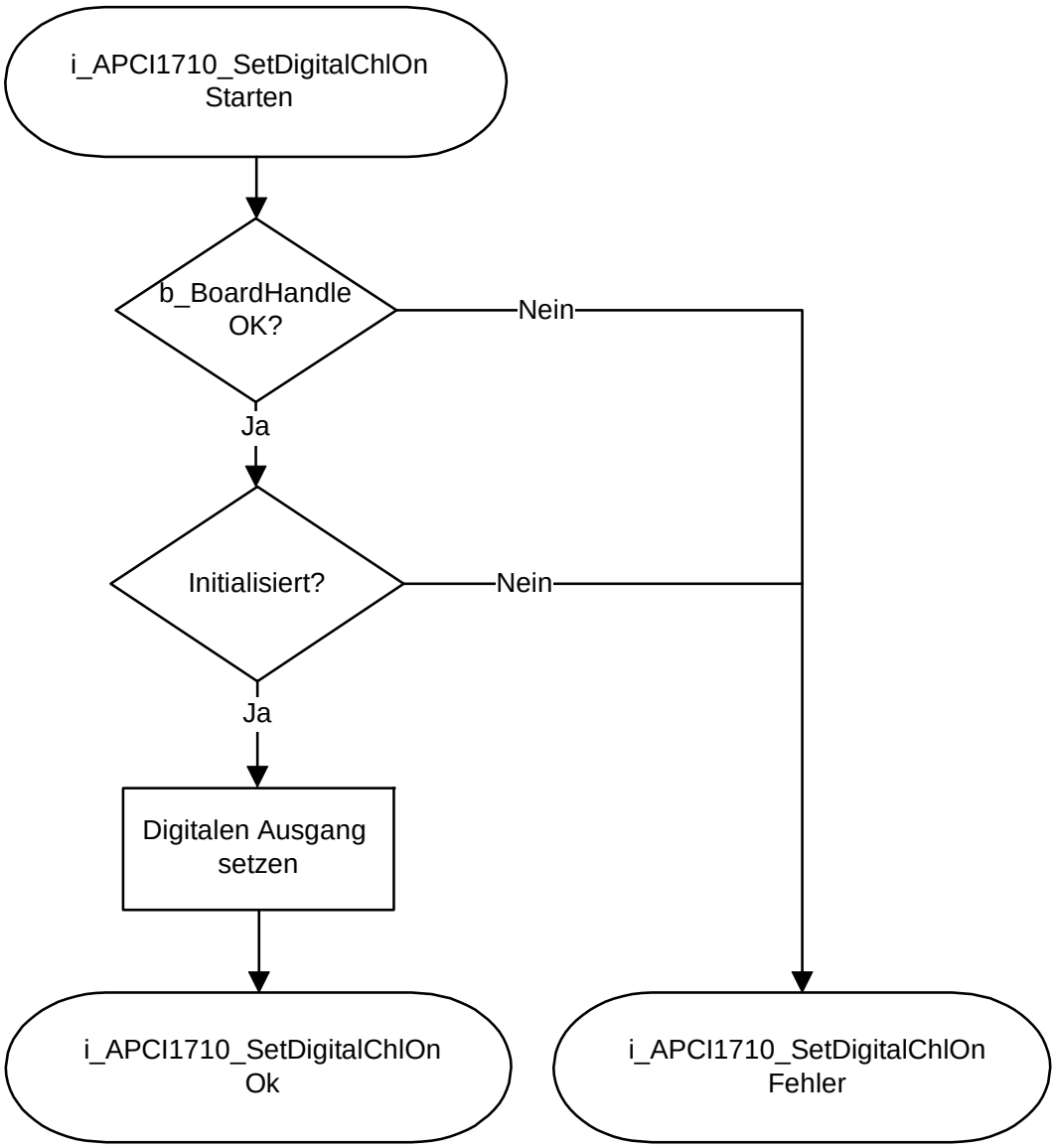
Aufgabe:
Setzt den digitalen Ausgang H. Einen Ausgang setzen bedeutet, ihn auf High setzen.

Funktionsaufruf:

ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetDigitalChlOn (b_BoardHandle,
                                            0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe	
b_BoardHandle	b_ModulNbr
Funktionsablauf	
	
Ausgabe	
<Return Wert>	

2) i_APCI1710_SetDigitalChlOff (...)

Syntax:
<Return Wert> = i_APCI1710_SetDigitalChlOff
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte xPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

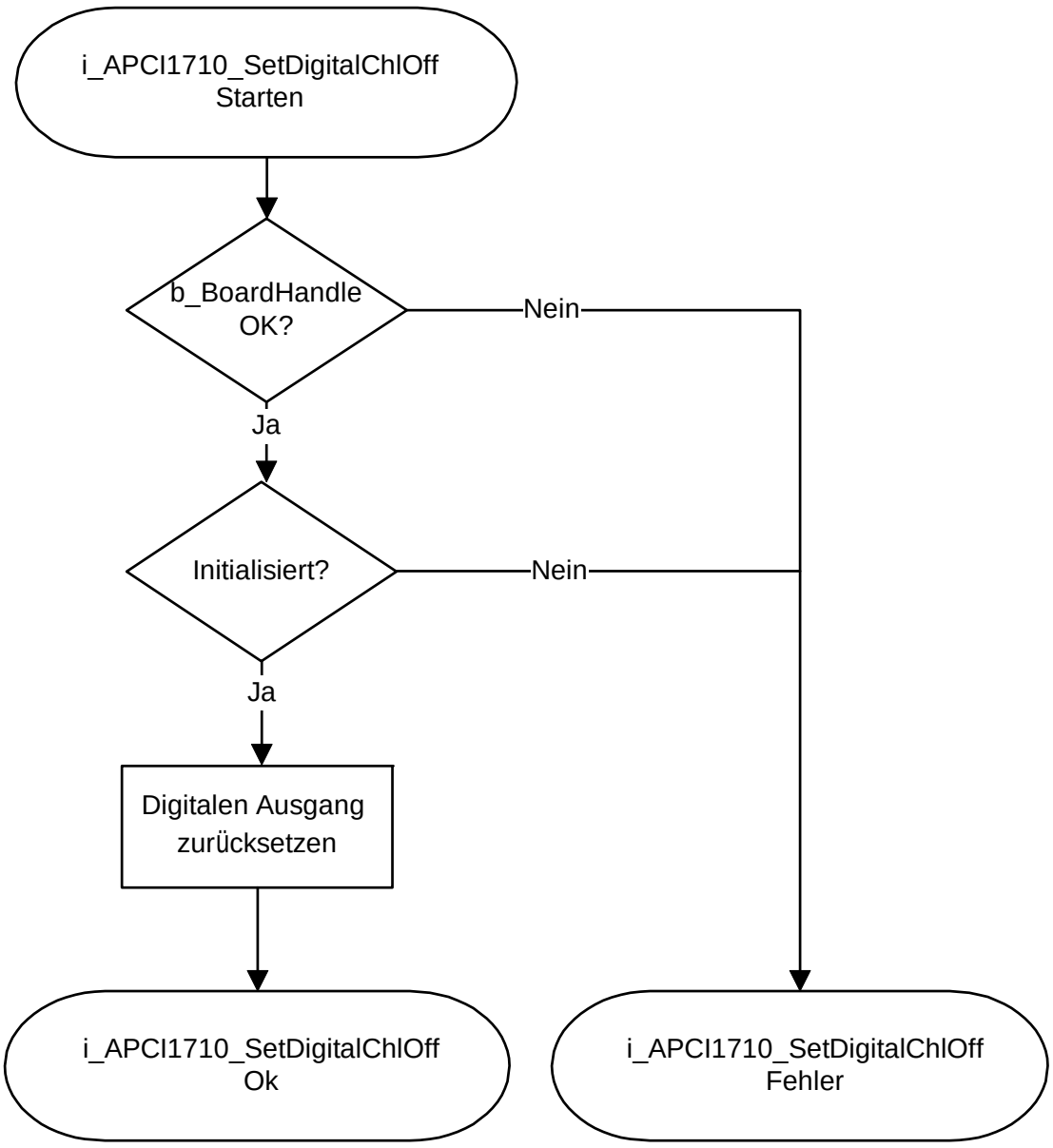
Aufgabe:
Setzt den digitalen Ausgang H zurück. Einen Ausgang rücksetzen bedeutet auf Low setzen.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetDigitalChlOff (b_BoardHandle,
                                             0);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Zähler nicht initialisiert. Siehe Funktion "i_APCI1710_InitCounter"

Eingabe	
b_BoardHandle	b_ModulNbr
Funktionsablauf	



Ausgabe
<Return Wert>

3.12 Funktionen im Kernel-Mode benutzen

WICHITIG!
Diese Funktionen stehen nur für die Benutzer Interruptroutine unter Windows NT und Windows 95/98 im synchronen Mode zur Verfügung. Siehe Funktion "i_APCI1710_SetBoardIntRoutineWin32"

1) i_APCI1710_KRNL_ClearCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ClearCounterValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

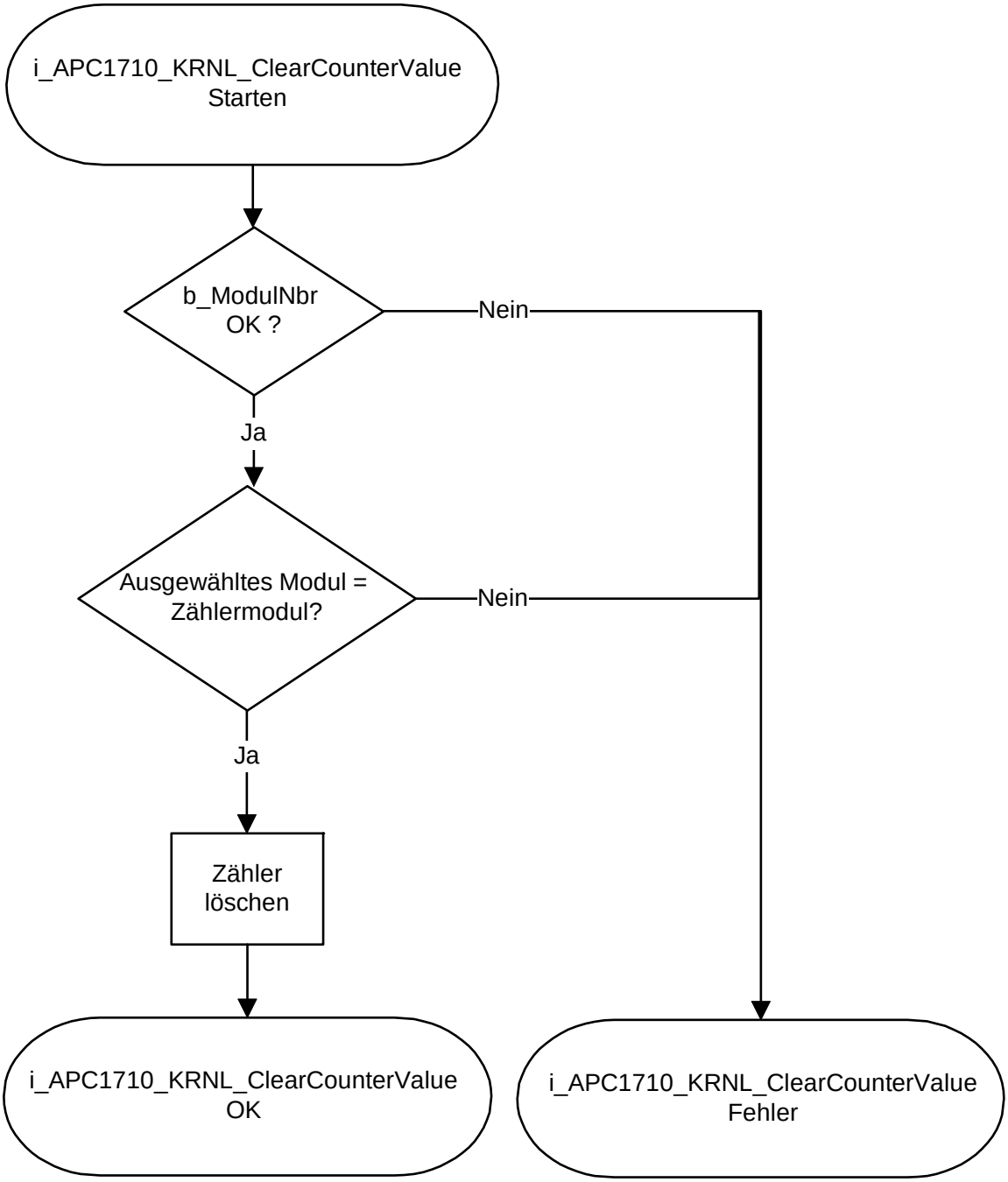
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Löscht den Zählerwert des ausgewählten Moduls (*b_ModulNbr*).

Funktionsaufruf:
ANSI C:

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_ClearCounterValue
                                    (ui_BaseAddress,
                                     0);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnmmer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe
ui_BaseAddress b_ModulNbr
Funktionsablauf

Ausgabe
<Return Wert>

2) i_APCI1710_KRNL_Read16BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_Read16BitCounterValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_SelectedCounter,
PUINT pui_CounterValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)
BYTE	b_SelectedCounter	Ausgewählter 16-Bit Zähler (0 oder 1)

- Ausgabe:

PUINT	pui_CounterValue	16-Bit Zählerwert

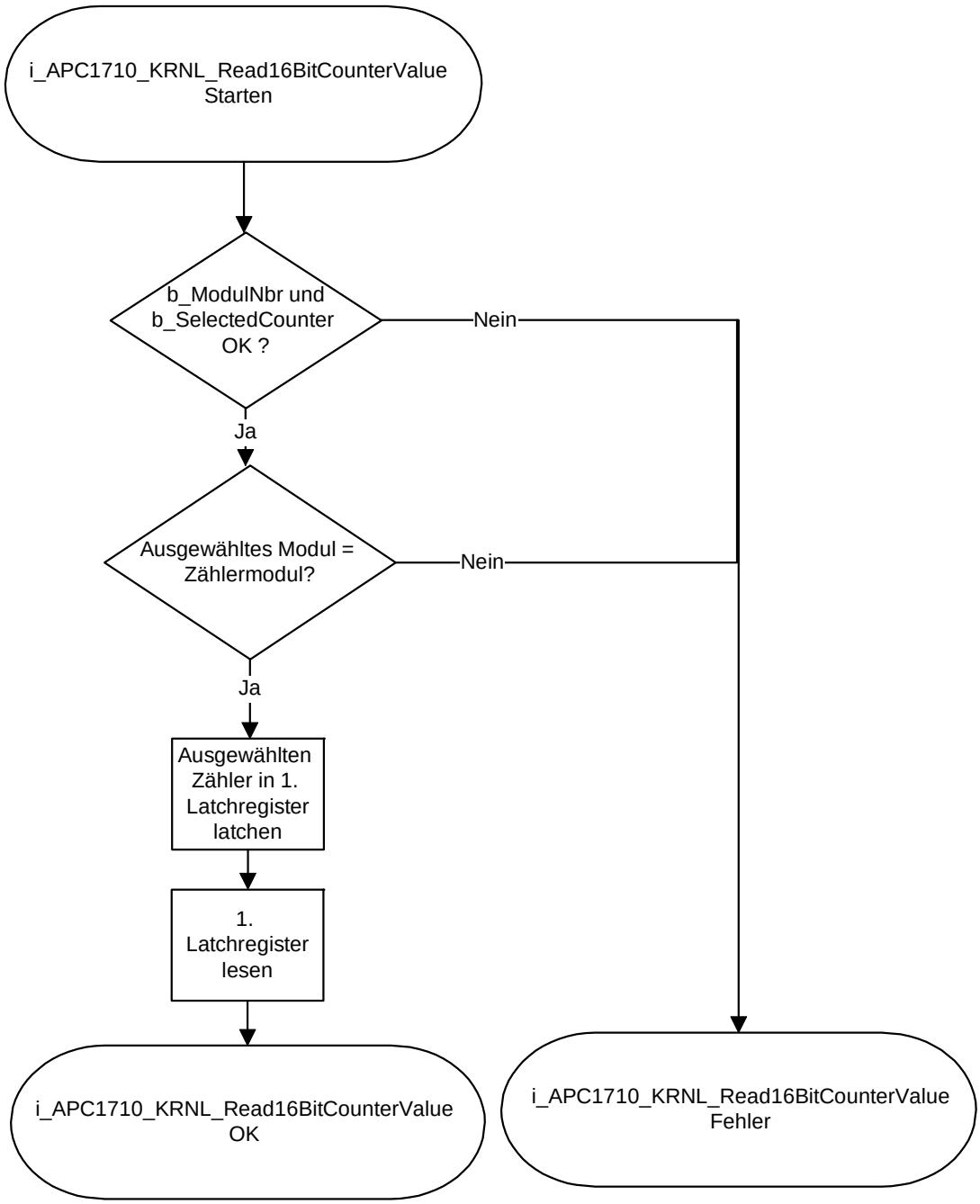
Aufgabe:
Latchen des 16-Bit Zählers (*b_SelectedCounter*) vom ausgewählten Modul (*b_ModulNbr*) in das erste Latch-Register und Rückgabe des gelatchten Wertes.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned int    ui_CounterValue;

i_ReturnValue = i_APCI1710_KRNL_Read16BitCounterValue
                (ui_BaseAddress,
                 0,
                 0,
                 &ui_CounterValue);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.
-3: Der ausgewählte 16-Bit Zähler ist falsch.

Eingabe	
ui_BaseAddress b_ModulNbr	b_SelectedCounter
Funktionsablauf	



Ausgabe
pui_CounterValue
<Return Wert>

3) i_APCI1710_KRNL_Read32BitCounterValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_Read32BitCounterValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PULONG pul_CounterValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

- Ausgabe:

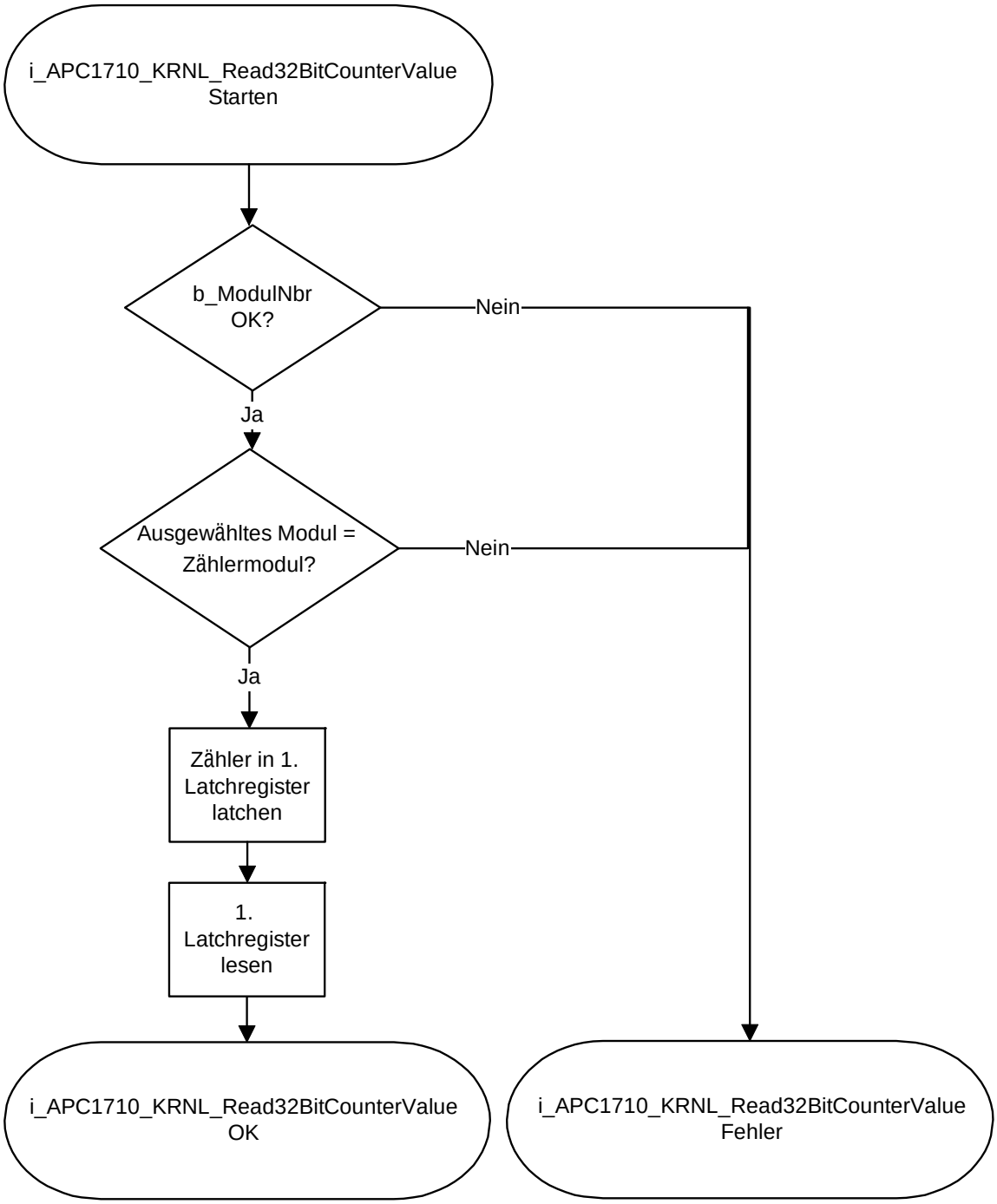
PULONG	pul_CounterValue	32-Bit Zählerwert

Aufgabe:
Latchen des 32-Bit Zählers (*b_SelectedCounter*) vom ausgewählten Modul (*b_ModulNbr*) in das erste Latch-Register und Rückgabe des gelatchten Wertes.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned long   ul_CounterValue;

i_ReturnValue = i_APCI1710_KRNL_Read32BitCounterValue
                                (ui_BaseAddress,
                                 0,
                                 &ul_CounterValue);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe
ui_BaseAddress b_ModulNbr
Funktionsablauf

Ausgabe
pul_CounterValue
<Return Wert>

4) i_APCI1710_KRNL_Write16BitCounterValue (...)

Syntax:
< Return Wert > = i_APCI1710_KRNL_Write16BitCounterValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_SelectedCounter,
UINT ui_WriteValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)
BYTE	b_SelectedCounter	Ausgewählter 16-Bit Zähler (0 oder 1)
UINT	ui_WriteValue	16-Bit Schreibewert

- Ausgabe:
Es erfolgt keine Ausgabe.

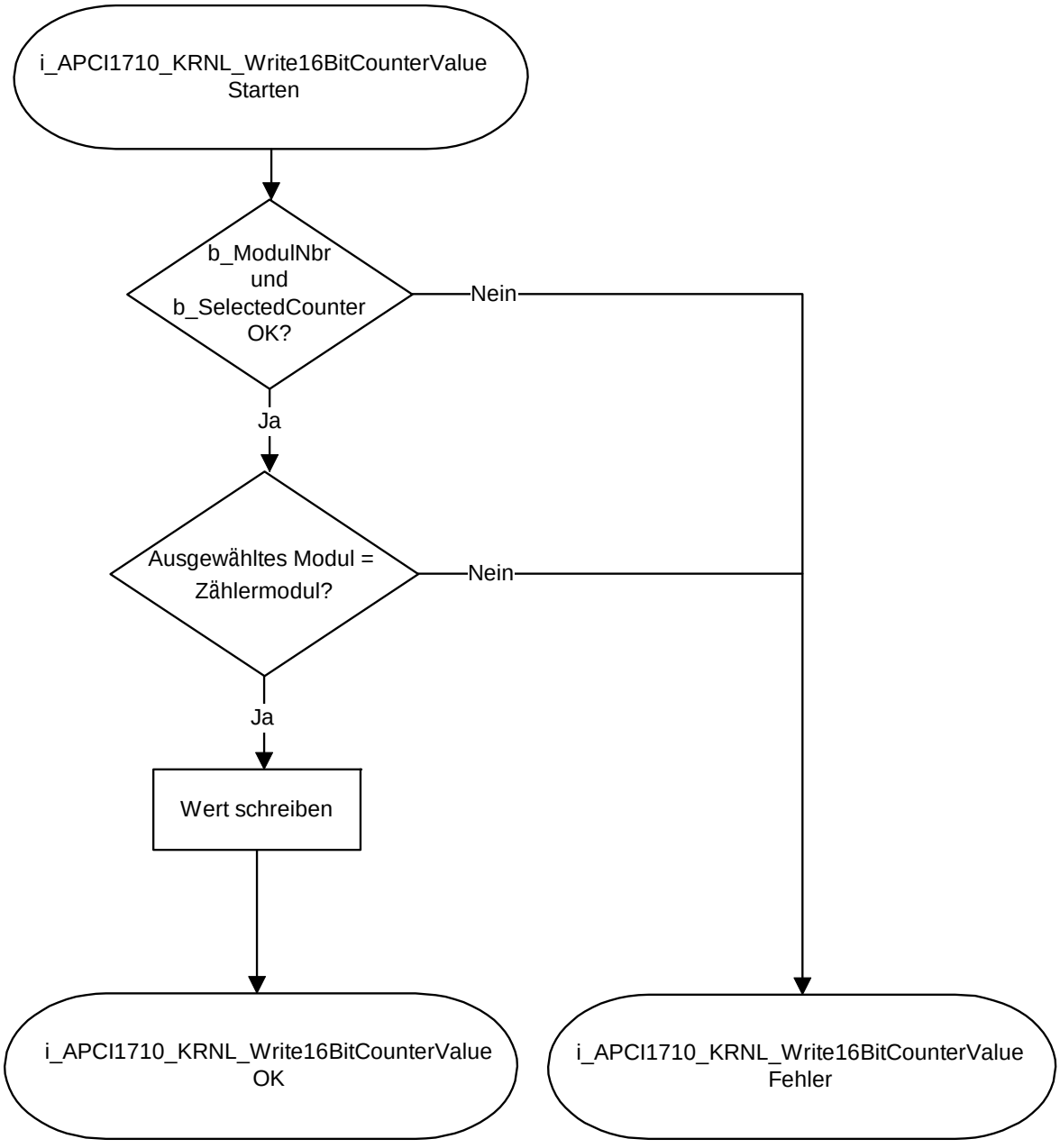
Aufgabe:
Schreibt einen 16-Bit Wert (*ui_WriteValue*) in den 16-Bit Zähler (*b_SelectedCounter*) des ausgewählten Moduls *(b_ModulNbr)*

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_Write16BitCounterValue
                                    (ui_BaseAddress,
                                     0,
                                     0,
                                     2000);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.
-3: Der ausgewählte 16-Bit Zähler ist falsch.

Eingabe	
ui_BaseAddress b_ModulNbr	b_SelectedCounter ui_WriteValue
Funktionsablauf	



Ausgabe
<Return Wert>

5) i_APCI1710_KRNL_Write32BitCounterValue (...)

Syntax:
< Return Wert > = i_APCI1710_KRNL_Write16BitCounterValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
ULONG ul_WriteValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)
ULONG	ul_WriteValue	16-Bit Schreibewert

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Schreibt einen 32-Bit Wert (*ui_WriteValue*) in den 32-Bit Zähler des ausgewählten Moduls *(b_ModulNbr)*

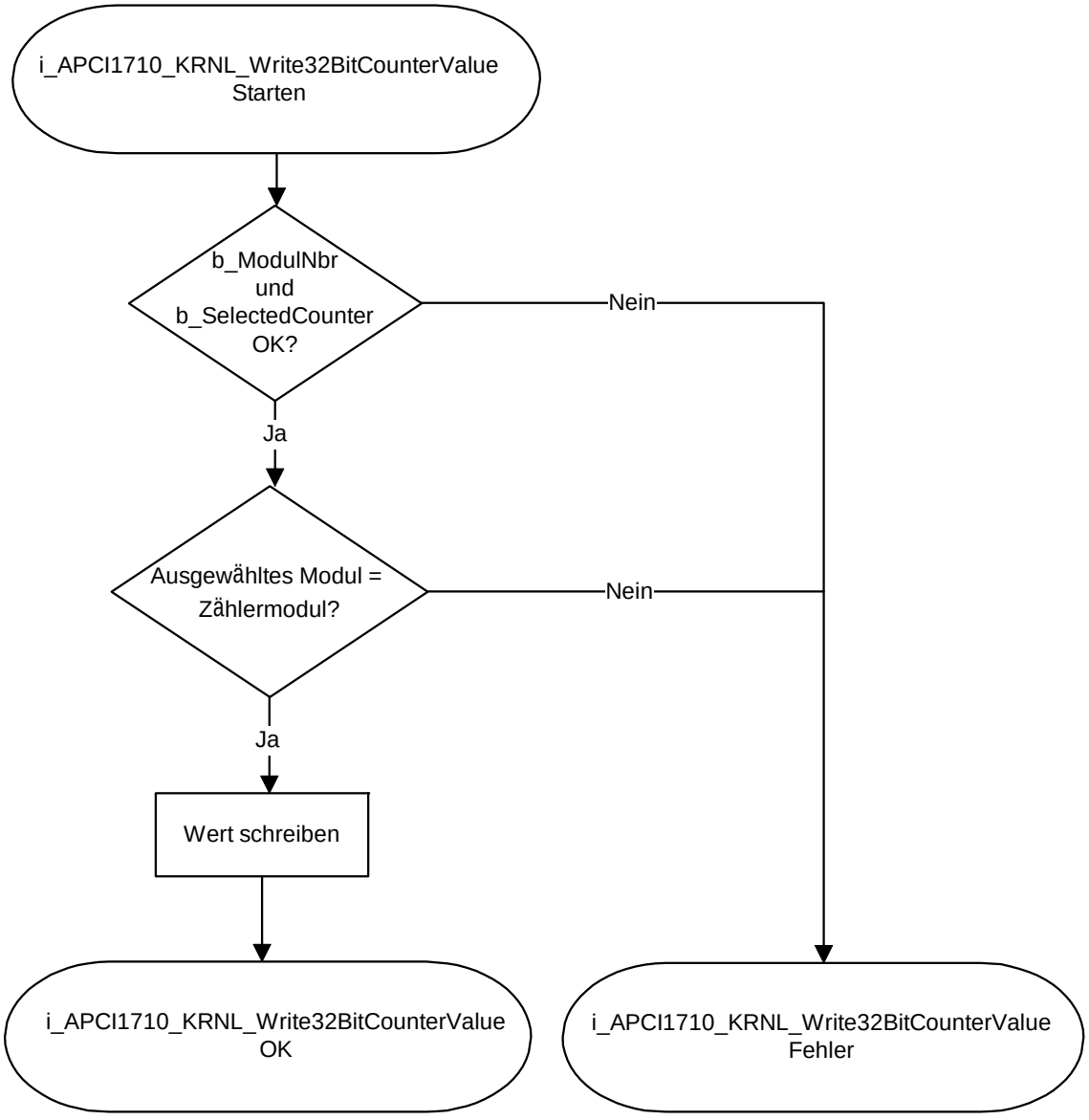
Funktionsaufruf:

ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_Write32BitCounterValue
                (ui_BaseAddress,
                 0,
                 200000);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe	
ui_BaseAddress b_ModulNbr	ul_WriteValue
Funktionsablauf	



Ausgabe
<Return Wert>

6) i_APCI1710_KRNL_GetInterruptUDLatchedStatus (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_GetInterruptUDLatchedStatus
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_UDStatus)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

- Ausgabe:

PBYTE	pb_UDStatus	0: Zähler-Ablauf im ausgewählten Mode abwärts. 1: Zähler-Ablauf im ausgewählten Mode aufwärts. 2: Es erfolgt kein Index-Interrupt Siehe Funktion "i_APCI1710_InitCounter"

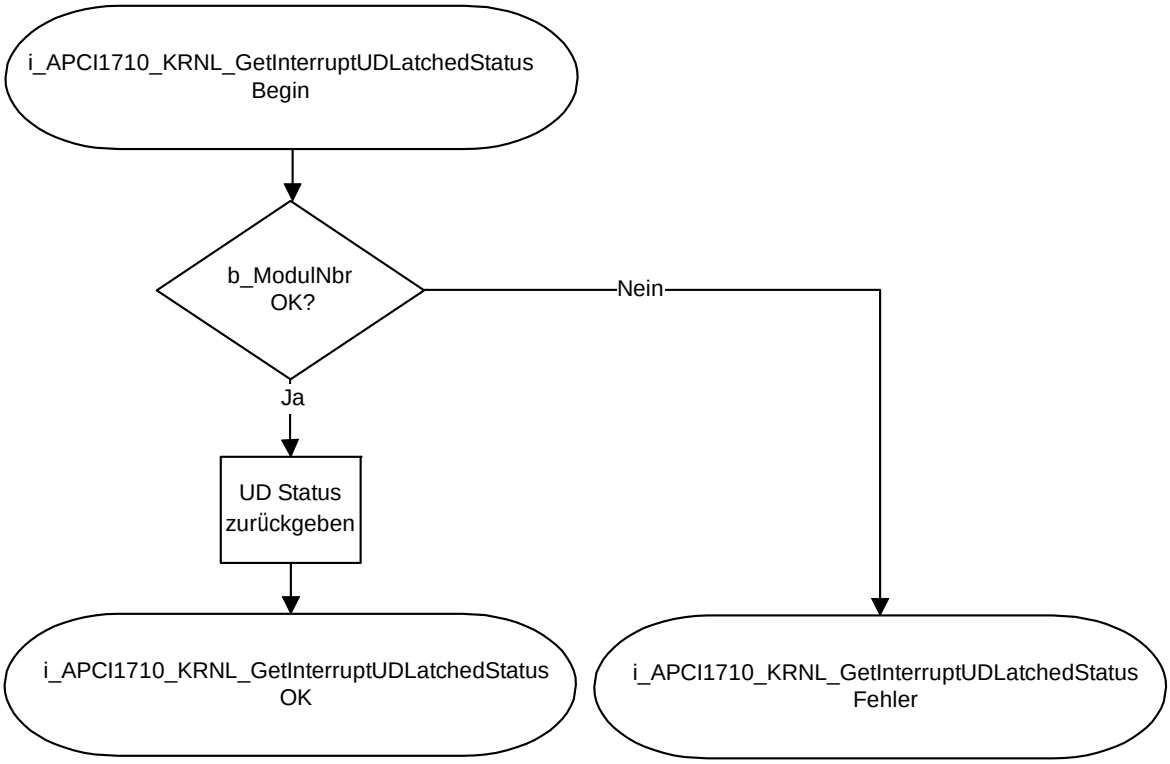
Aufgabe:
Gibt den gelatchten Status des Zählerablaufs zurück, nachdem ein Index Interrupt erfolgt ist.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_UDStatus;

i_ReturnValue = i_APCI1710_GetInterruptUDLatchedStatus
                                    (ui_BaseAddress,
                                     0,
                                     &b_UDStatus);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe	
ui_BaseAddress	b_ModulNbr
Funktionsablauf	



Ausgabe
pb_UDStatus
<Return Wert>

7) i_APCI1710_KRNL_ReadFrequencyMeasurement (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ReadFrequencyMeasurement
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_UDStatus,
PBYTE pb_Status,
PULONG pul_ReadValue)

Parameter:
- Eingabe

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

- Ausgabe

PBYTE	pb_Status	gibt den Status der Frequenzmessung zurück 0: Zählablauf nicht gestartet. 1: Zählablauf gestartet. 2: Zählablauf gestoppt
PBYTE	pb_UDStatus	Status des aufwärts/abwärts Zählers. Siehe Tabelle 3-12
PULONG	pul_ReadValue	Gibt die Anzahl der Inkrementen innerhalb der Zeitbasis zurück

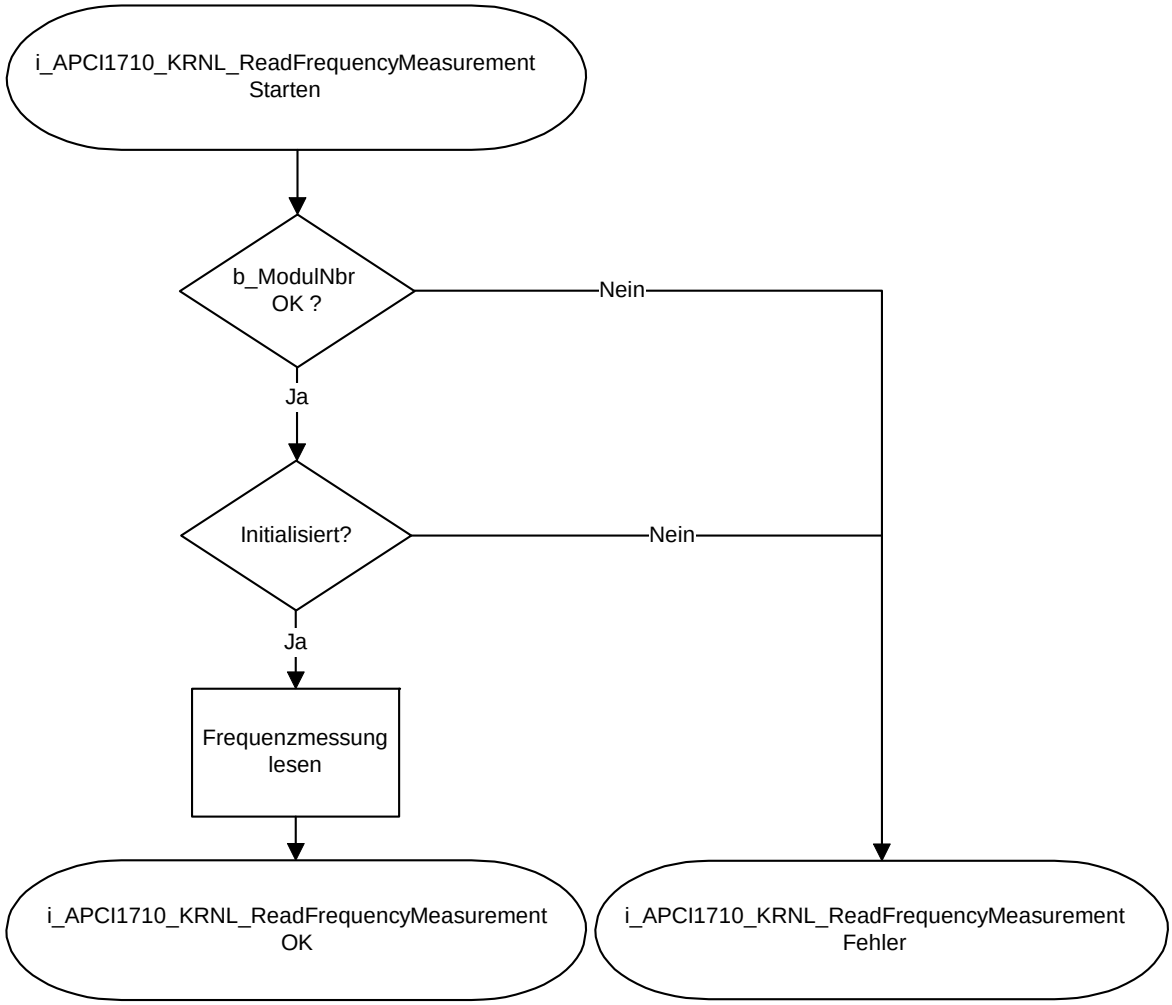
Aufgabe:
Gibt den Status (*pb_Status*) und die Anzahl der Inkrementen in der gesetzten Zeit zurück. Siehe Funktion "i_APCI1710_InitFrequencyMeasurement"

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_Status;
unsigned char   b_UDStatus;
unsigned long   ul_ReadValue;

i_ReturnValue = i_APCI1710_KRNL_ReadFrequencyMeasurement
                (ui_BaseAddress,
                 0,
                 &b_Status,
                 &b_UDStatus,
                 &ul_ReadValue);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe	
ui_BaseAddress	b_ModulNbr
Funktionsablauf	



Ausgabe	
pb_Status pb_UDStatus	pul_ReadValue
<Return Wert>	

8) i_APCI1710_KRNL_SetDigitalChlOn (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetDigitalChlOn
(UINT ui_BaseAddress,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

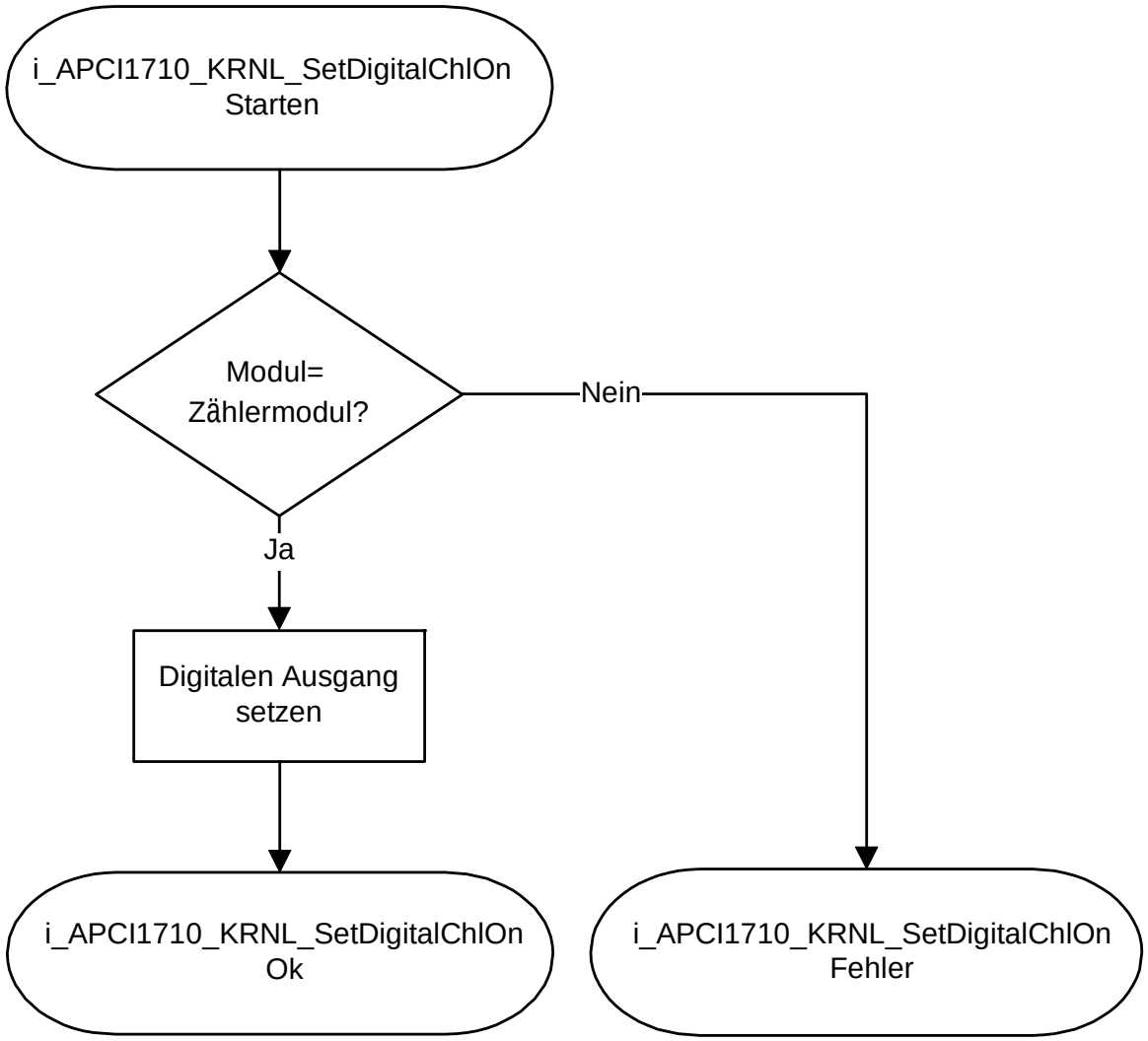
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Setzt den digitalen Ausgang H. Einen Ausgang setzen bedeutet einen Ausgang auf High setzen.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetDigitalChlOn
                (ui_BaseAddress,
                 0);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe
ui_BaseAddress b_ModulNbr
Funktionsablauf

Ausgabe
<Return Wert>

9) i_APCI1710_KRNL_SetDigitalChlOff (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetDigitalChlOff
(UINT ui_BaseAddress,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der xPCI-1710. Siehe "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 to 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Setzt den digitalen Ausgang H zurück. Einen Ausgang rücksetzen bedeutet einen Ausgang auf Low setzen.

Funktionsaufruf:
ANSI C :

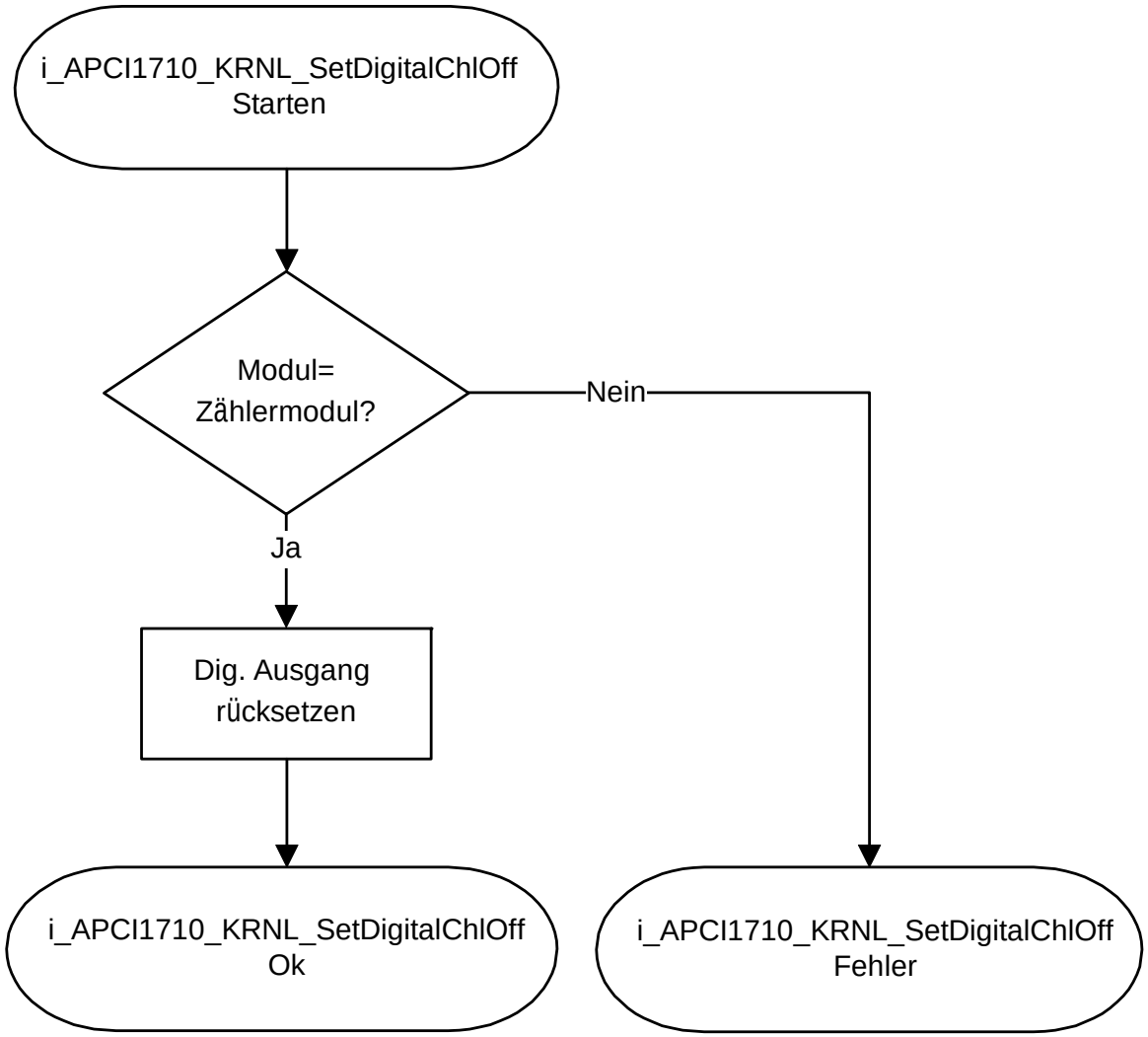
```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetDigitalChlOff
                (ui_BaseAddress,
                 0);
```

Return Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Das Modul ist kein Zählermodul.

Eingabe

ui_BaseAddress
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>