DIN EN ISO 9001: 2000
zertifiziert

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

Software-Beschreibung

ADDICOUNT APCI-/CPCI-1710

Zähler/Timer

4. 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 Wind River Systems Inc.

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 finden Sie die notwendigen Belegungs- und Programmierinformationen in den einzelnen 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 Interface	ttl_io.cfg
Digitale E/A	dig_IO_d.pdf	dig_IO_e.pdf	Digital I/O	dig_IO.cfg
Impulszähler	Impulszähler_d.pdf	pulse_counter_e.pdf	Pulse counter	imp_cpt.cfg
ETM	ETM_d.pdf	ETM_e.dpf	Edge time measurement	etm.cfg

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 ZÄHLER/TIMER

2.1 Funktionsbeschreibung

Die Funktion "Zähler/Timer" arbeitet ähnlich wie die des Intel-Bausteins 82C54.

Diese Funktion eignet sich besonders für Industrie-Anwendungen, für die Zuverlässigkeit und Robustheit verlangt werden.

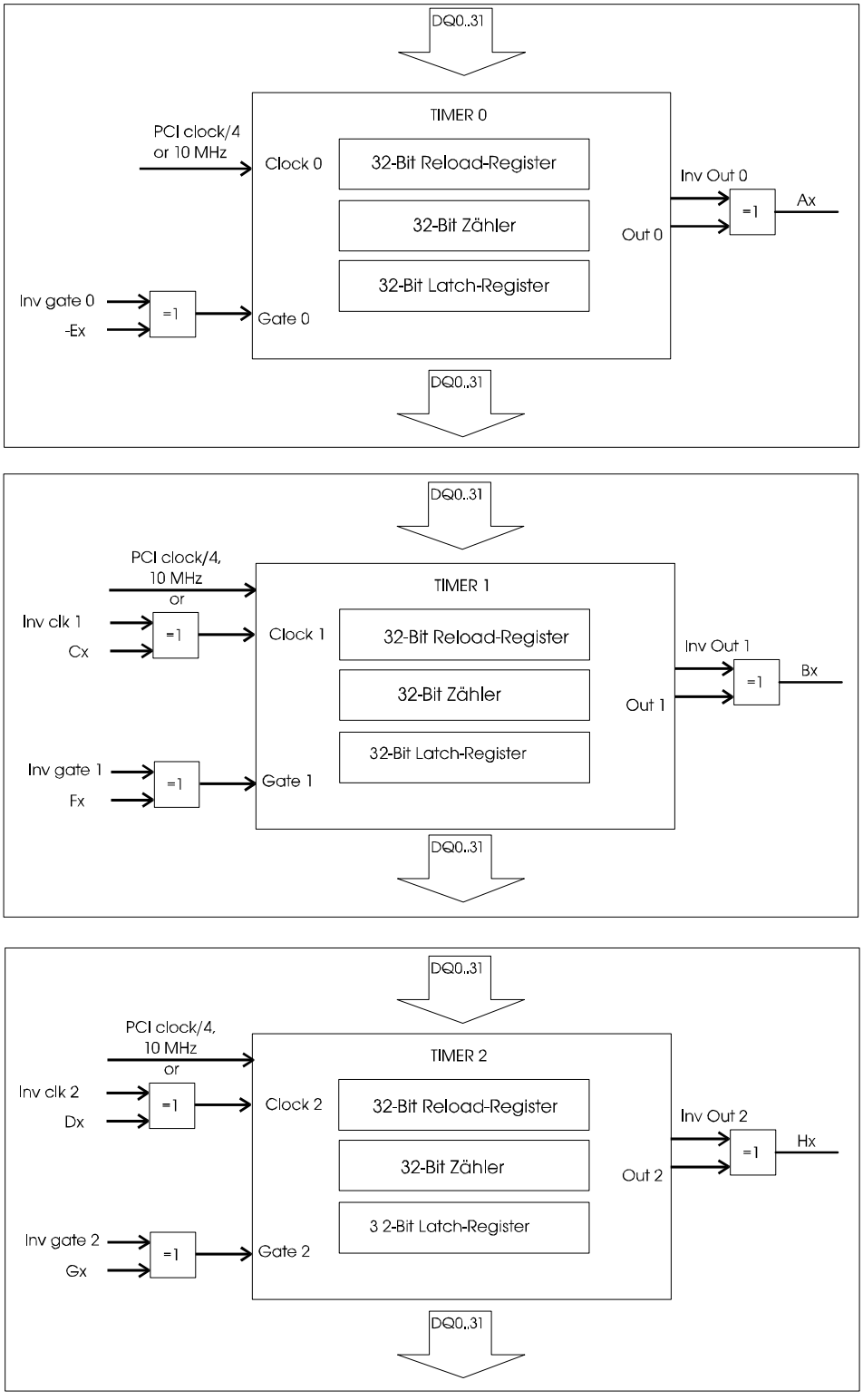
Eigenschaften:

- Zur Vermeidung von Erdschleifen wird eine komplette galvanische Trennung durch Optokoppler für die Ein-/Ausgänge herangezogen
- 3 x 32-Bit-Zähler/Timer stehen zur Verfügung (nur Binärzahlen)
- Signale bis zu 5 MHz können verarbeitet werden
- 6 programmierbare Modes
- Status-Readback und Latch-Kommando
- Eingänge und Ausgänge können per Software invertiert werden
- Hardware- und Software-GATE möglich, rücklesbar
- Einfache Schnittstelle: keine Mehrfachbelegung der Adressen
- Auslösung des Interrupts mit einem individuellen Freigabebit pro Zähler/Timer und Interruptstatus-Register
- Interner Takt (PCI/4) oder 10 MHz vom Quarzoszillator auf der Karte steht als Clock zur Verfügung, wählbar über Software

2.1.1 Blockdiagramm

Die Schnittstelle unterstützt:
- 3 voneinander unabhängige 32-Bit-Zähler/Timer, die über den Datenbus ausgelesen bzw. beschrieben werden können,
- eine Funktions- und Kontrollogik.

Abb. 2-1: Blockschaltbild der "Zähler/Timer" Funktion



2.1.2 Typische Anwendungen

- Digitaler "one-shot"
- Ereigniszähler
- Frequenzgenerator
- Komplexer Signalgenerator

2.2 Benutzte Signale

Die Funktion "Zähler/Timer" belegt **5 Eingänge (Kanäle C bis G) und 3 Ausgänge (Kanäle A, B, H)** von dem entsprechenden Funktionsmodul der **APCI-/CPCI-1710**.

Tabelle 2-1: Benutzte Signale

SIGNALE	AM STECKER	POLARITÄT	FUNKTION
OUT 1	Ax +/ -	Diff. / TTL	Ausgang des 1. Zählers/Timers
OUT 2	Bx +/ -	Diff. / TTL	Ausgang des 2. Zählers/Timers
OUT 3	Hx	24 V /Opt. 5 V	Ausgang des 3. Zählers/Timers
GATE 1	Ex	24 V /Opt. 5V	Gate Eingang des 1. Zählers/Timers
GATE 2	Fx	24V /Opt.5 V	Gate Eingang des 2. Zählers/Timers
GATE 3	Gx	24V /Opt.5 V	Gate Eingang des 3. Zählers/Timers
CLK 1	-	-	durch internem Clock belegt = PCI-Clock geteilt durch 4
CLK 2	Cx +/ -	Diff./TTL/Opt. 24V	Clock/Zähler Eingang des 2. Zählers/Timers
CLK 3	Dx +/ -	Diff./TTL/Opt. 24V	Clock/Zähler Eingang des 3. Zählers/Timers

x: Nummer des Funktionsmoduls.

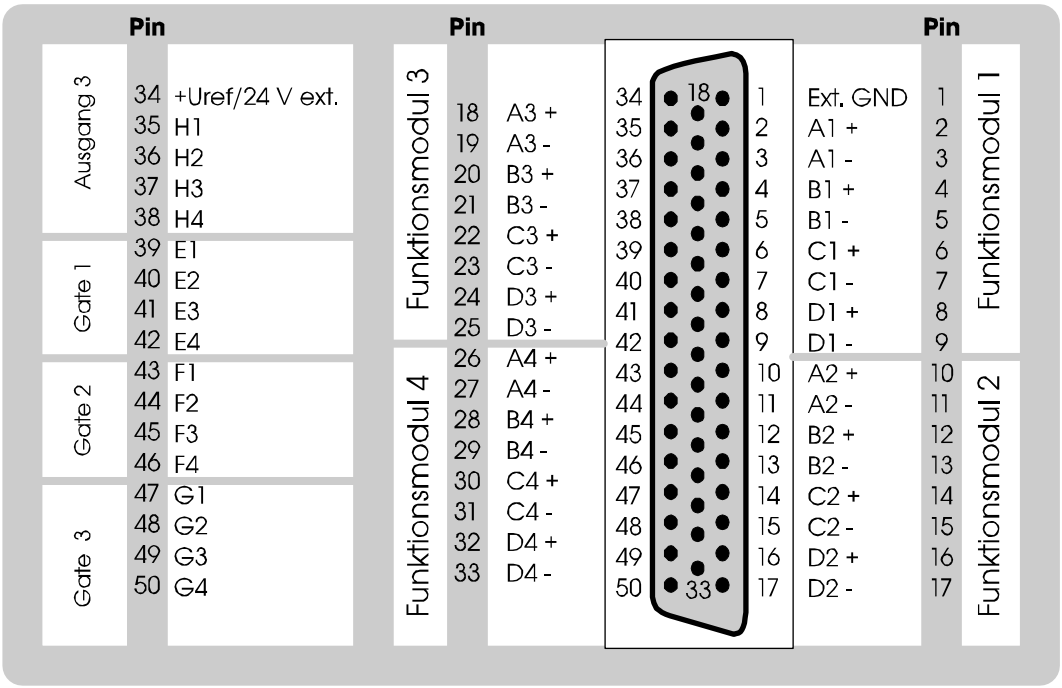
2.3 Steckerbelegung für alle Module mit Zähler/Timer-Funktion

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

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

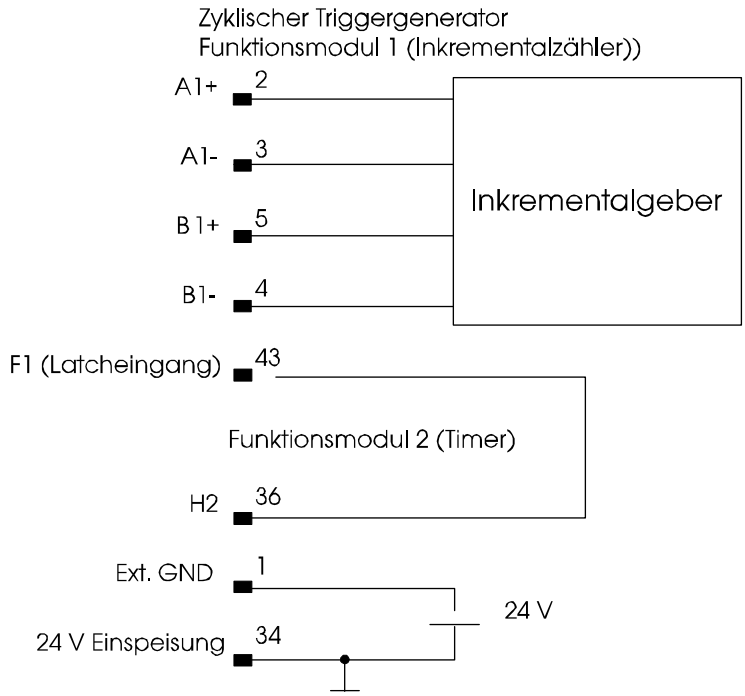
Die untere Abbildung ist ein Anschlussbeispiel: Die Funktion „Zähler/Timer“ ist auf allen Funktionsmodulen implementiert.

Abb. 2-2: Pinbelegung des 50-pol. SUB-D Steckers



2.4 Anschlussbeispiel

Abb. 2-3: Anschlussbeispiel



2.5 E/A-Adressbelegung

Tabelle 2-2: E/A-Belegung der "Zähler/Timer"-Funktion

			D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	Rd	Wr	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE**x** + 0	☑	☑	**TIMER0**	**TIMER0**	**TIMER0**	**TIMER0**
BASE**x** + 4	☑	☑	**TIMER1**	**TIMER1**	**TIMER1**	**TIMER1**
BASE**x** + 8	☑	☑	**TIMER2**	**TIMER2**	**TIMER2**	**TIMER2**
BASE**x** + 12		☑	**GLOBAL CONTROL REGISTER**			
BASE**x** + 16	☑	☑	0x00h	0x00h	**TIMER0** CONTROL REG	**TIMER0** CONTROL REG
BASE**x** + 20	☑	☑	0x00h	0x00h	**TIMER1** CONTROL REG	**TIMER1** CONTROL REG
BASE**x** + 24	☑	☑	0x00h	0x00h	**TIMER2** CONTROL REG	**TIMER2** CONTROL REG
BASE**x** + 28			-	-	-	-
BASE**x** + 32		☑	-	-	-	SET **TIMER0** REGISTER
BASE**x** + 36		☑	-	-	-	SET **TIMER1** REGISTER
BASE**x** + 40		☑	-	-	-	SET **TIMER2** REGISTER
BASE**x** + 44		☑	-	-	-	SET **TIMER0** SOFTGATE
BASE**x** + 48		☑	-	-	-	SET **TIMER1** SOFTGATE
BASE**x** + 52		☑	-	-	-	SET **TIMER2** SOFTGATE
BASE**x** + 60			FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

-: keine Funktion ; **x:** Nummer des Funktionsmoduls.

Die Funktion " Zähler/Timer " belegt 15 DWORDS im E/A Bereich des Funktionsmoduls **x**.
Die Zugriffe werden immer in 32-Bit breite gelesen oder geschrieben.

2.6 Beschreibung der E/A-Funktionen

2.6.1 Funktionsbeschreibung

Die Funktion Zähler/Timer ähnelt dem Baustein 82C54 von INTEL.

Es stehen **3 x 32-Bit Zähler/Timer** zur Verfügung. Jeder dieser Zähler/Timer kann in einem von **6 möglichen Modes** programmiert werden (Mode0 bis Mode5). Der Zähler/Timer in 32-Bit Breite kann jederzeit mit einem neuen Wert geladen sowie gelesen werden. Für das Lesen soll der Wert jedoch zuerst "gelatcht" werden.

Definition der Modi

CLK PULSE: fallende Flanke nach einer steigenden Flanke des Zähler Takt-Eingangs.

TRIGGER: eine steigende Flanke des Zähler GATE-Eingangs.

COUNTER LOADING: Zählertransfer vom Zählerregister nach Zählerbaustein

Mode0: Interrupt am Ende des Zählvorgangs

Der Mode0 eignet sich besonders für das Event-Zählen. Nach der Initialisierung ist der Ausgang "Low". Der Ausgang hält diese Position bis der Zähler 0 erreicht.

Der Ausgang wird dann auf "High" bis zum neuen Zählenzyklus oder bis zum Schreiben eines neuen Zählerwert.

Mode1: Monoflop, durch Hardware retriggerbar

Dieser Mode ist gleich wie der Mode0 außer der Funktion des GATE-Eingangs. Der GATE-Eingang wird nicht benutzt, um den Timer zu aktivieren oder de-aktivieren. Er triggert den Timer.

Mode2: Impulsgenerator

Dieser Mode funktioniert als einer durch ul_ReloadValue dividierende Zähler. Er eignet sich für die Generierung eines Echtzeit-Takt-Interrupts.

Nach der Initialisierung ist der Ausgang auf "High" gesetzt. Der Startwert wird bis auf den Wert „0“ dekrementiert; der Ausgang wird dann „Low“ während einem Taktsignal und wieder "High". Der Zähler lädt den Startwert (ul_RelaodValue) wieder und der Zyklus wird wiederholt. Ein Interrupt kann na jedem Zyklusende generiert werden.

Zeitberechnung = (ul_RelaodValue + 2) x Eingangstakt

Mode3: Rechteck-Generator

Mode3 generiert Baudrate. Er ähnelt den Mode2 außer dem Ausgangszyklus. Beim Starten dieses Zählers ist der Ausgang auf „High“ gesetzt. Der Startwert wird bis auf 0 dekrementiert, der Ausgang auf auf „Low“ gesetzt. Der Zähler wird neu mit dem Startwert geladen und wieder bis auf 0 dekrementiert. Der Ausgang wird wieder auf „High“. Der Zyklus wiederholt sich automatisch.

Zeitberechnung = (ul_RelaodValue + 2) x Eingangstakt

Mode4: Strobe, duch Software getriggert

Nach Initialisierung ist der Ausgang auf "High". Sobald der Startwert abgelaufen ist, wird der Ausgang "Low" während einem Taktimpuls und wird wieder auf "High" gesetzt. Die Zählsequenz wird getriggert, wenn einer neue Wert eingeschrieben wird.

Wenn ein neuer Wert während dem Zählzyklus eingeschrieben wird, wird dieser Wert beim nächsten Taktimpuls geladen.

Mode5: Strobe, duch Hardware getriggert (retriggerbar)

Dieser Mode ist gleich wie der Mode4 außer der Funktion des GATE-Eingangs.

Der GATE-Eingang wird nicht benutzt, um den Timer zu aktivieren oder deaktivieren. Er triggert den Timer.

2.6.2 TIMER0-Register (Base +0)

32-Bit Register, in dem der Reloadwert für den Zähler/Timer 0 geschrieben wird. Beim Lesen dieser Adresse wird der aktuelle gelatchte Wert des Zähler/Timer 0 gelesen.

2.6.3 TIMER1-Register (Base +4)

32-Bit Register, in dem der Reloadwert für den Zähler/Timer 1 geschrieben wird. Beim Lesen dieser Adresse wird der aktuelle gelatchte Wert des Zähler/Timer 1 gelesen.

2.6.4 TIMER2-Register (Base +8)

32-Bit Register, in dem der Reloadwert für den Zähler/Timer 2 geschrieben wird. Beim Lesen dieser Adresse wird der aktuelle gelachte Wert des Zähler/Timer 2 gelesen.

2.6.5 Control-Register (Base +12)

32-Bit Register, das für die 3 Zähler zuständig ist.

DQ0: = 0
DQ1: Auswahl des **Timer 0,** wenn "1" dann wird der Zählerstand und Status gelatcht

DQ2: Auswahl des **Timer 1,** wenn "1" dann wird der Zählerstand und Status gelacht
DQ3: Auswahl des **Timer 2,** wenn "1" dann wird der Zählerstand und Status gelatcht
DQ4: Schreiben einer 0 → Latchen des ausgewählten Timer-Zustands

DQ5: Schreiben einer 0 → Latchen der ausgewählten Timer-Zustands
DQ6: = 1
DQ7 : = 1
DQ8..31: keine Funktion

Mit der "LATCH" Funktion können gleichzeitig alle drei Zähler per Software gelatcht werden. Dazu müssen alle drei Zähler selektiert werden DQ1= DQ2= DQ3 = DQ6 = DQ7 =1 , & DQ4 = 0 = Latchen der selektierten Zähler.

2.6.6 TIMER 0, 1, 2 Control-Register (Base + 16, 20, 24)

DQ0: Modebit 1 (nur Schreiben möglich), Auswahl des Zählermodes: 0,1,2,3,4,5
DQ1: Modebit 2 (nur Schreiben möglich)
DQ2: Modebit 3 (nur Schreiben möglich)
DQ3: =0
DQ4: =0
DQ5: =0
DQ6: NULL_COUNT (nur Lesen möglich), wenn 1 ist der Timer angehalten, Wenn "0" läuft der Timer ab (der Timer wurde geladen) .
DQ7: OUT (nur Lesen möglich), Bild des Ausgang-Zustands am Timer
DQ8: GATE (nur Lesen möglich), Bild des Gate-Zustands am Timer

Restliche Bits beim Rücklesen auf 0

2.6.7 SET TIMER0 Register (Base + 32)

DQ0: 1: invertiert das externe GATE0 Signal, 0: keine Invertierung (Reset)
DQ1: 1: invertiert das interne CLK0 Signal, 0: keine Invertierung (Reset)
DQ2: 1: invertiert das externe OUT0 Signal, 0: keine Invertierung (Reset)
DQ3: 1: Interruptfreigabe für den TIMER0, 0: kein Interrupt (Reset)
DQ4DQ31..4: keine Funktion
SET TIMER1 Register (Base + 36)
DQ0: 1: invertiert das externe GATE1 Signal, 0: keine Invertierung (Reset)
DQ1: 1: invertiert das externe CLK1 Signal, 0: keine Invertierung (Reset)
DQ2: 1: invertiert das externe OUT1 Signal, 0: keine Invertierung (Reset)
DQ3: 1: Interruptfreigabe für den TIMER1, 0: kein Interrupt (Reset)
DQ4: Keine Funktion
DQ5: 0: PCI-Bus Clock wird als Zeitbasis benutzt.
1: Interner Quarz /4 wird als Zeitbasis benutzt (10MHz).
DQ6 → DQ31..4: keine Funktion

2.6.8 SET TIMER1 Register (Base + 36)

DQ0: 1: invertiert das externe GATE1 Signal, 0: keine Invertierung (Reset)
DQ1: 1: invertiert das externe CLK1 Signal, 0: keine Invertierung (Reset)
DQ2: 1: invertiert das externe OUT1 Signal, 0: keine Invertierung (Reset)
DQ3: 1: Interruptfreigabe für den TIMER1, 0: kein Interrupt (Reset)
DQ4: 1: externer Clock1 wird benutzt, 0 interner Clock (pci/4)
DQ5: 0: PCI-Bus Clock wird als Zeitbasis benutzt.
1: Interner Quarz /4 wird als Zeitbasis benutzt (10MHz).
DQ6 → DQ31..4: keine Funktion

2.6.9 SET TIMER2 Register (Base + 40)

DQ0: 1: invertiert das externe GATE2 Signal, 0: keine Invertierung (Reset)
DQ1: 1: invertiert das externe CLK2 Signal, 0: keine Invertierung (Reset)
DQ2: 1: invertiert das externe OUT2 Signal, 0: keine Invertierung (Reset)
DQ3: 1: Interruptfreigabe für den TIMER2, 0: kein Interrupt (Reset)
DQ4: 1: externer Clock2 wird benutzt, 0 interner Clock(pci/4)
DQ5: 0: PCI-Bus Clock wird als Zeitbasis benutzt.
1: Interner Quarz /4 wird als Zeitbasis benutzt (10MHz).
DQ6 → DQ31..4: keine Funktion

2.6.10 SET TIMER 0,1, 2 Softgate Register(Base +44, 48, 52)

DQ0: 1 Zählen wird erlaubt in Abhängigkeit des externen GATE Signals
0 Zählen gesperrt (Reset).
DQ31..1: keine Funktion

2.6.11 Versions-REGISTER (Base +60)

Die Funktion und die Revision werden erkannt. (Lesebefehl, ASCI Format)

BASE + 60 "I" "C" "1" "3"

Bedeutet: Chronos Revision 1.3

2.7 Arbeiten mit der "Zähler/Timer"-Funktion

1. Den Zähler/Timer im gewünschten Mode programmieren.
2. Die Signale GATE bzw. OUT dem gewünschten Pegel anpassen.
3. Den Reloadwert in den Timer schreiben.
4. Zähler/Timer ist betriebsbereit.
5. Falls mit Interrupt gearbeitet wird, müssen zuerst die Freigabebits für den Interrupt auf "1" gesetzt werden.

3 STANDARDSOFTWARE

3.1 Einleitung

WICHTIG!
Merken Sie sich die folgenden Schriftweisen im Text:

Funktion: "i_APCI1710_SetBoardInformation"

Variable *ui_Address*

Tabelle 3-1: Define-Wert

Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	1	1
DLL_COMPILER_VB	2	2
DLL_COMPILER_PASCAL	3	3
DLL_LABVIEW	4	4
APCI1710_PCI_BUS_CLOCK	0	0
APCI1710_FRONT_CONNECTOR_INPUT	1	1
APCI1710_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28
APCI1710_10MHZ	10	A

3.2 Interruptmaske

Jeder Timer/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 Timer/Zähler

b_ModuleMask	ul_InterruptMask	Bedeutung
0000 000**1**	0000 0000 000**1** 0000	Interrupt auf Timer0 vom Modul 0 ausgelöst
0000 000**1**	0000 0000 00**1**0 0000	Interrupt auf Timer1 vom Modul 0 ausgelöst
0000 000**1**	0000 0000 0**1**00 0000	Interrupt auf Timer2 vom Modul 0 ausgelöst
0000 00**1**0	0000 0000 000**1** 0000	Interrupt auf Timer0 vom Modul 1 ausgelöst
0000 00**1**0	0000 0000 00**1**0 0000	Interrupt auf Timer1 vom Modul 1 ausgelöst
0000 00**1**0	0000 0000 0**1**00 0000	Interrupt auf Timer2 vom Modul 1 ausgelöst
0000 0**1**00	0000 0000 000**1** 0000	Interrupt auf Timer0 vom Modul 2 ausgelöst
0000 0**1**00	0000 0000 00**1**0 0000	Interrupt auf Timer1 vom Modul 2 ausgelöst
0000 0**1**00	0000 0000 0**1**00 0000	Interrupt auf Timer2 vom Modul 2 ausgelöst
0000 **1**000	0000 0000 000**1** 0000	Interrupt auf Timer0 vom Modul 3 ausgelöst
0000 **1**000	0000 0000 00**1**0 0000	Interrupt auf Timer1 vom Modul 3 ausgelöst
0000 **1**000	0000 0000 0**1**00 0000	Interrupt auf Timer2 vom Modul 3 ausgelöst

3.3 Initialisierung

1) i_APCI1710_InitTimer (...)

Syntax:

```
<Return-Wert> =  i_APCI1710_InitTimer
                                    (BYTE    b_BoardHandle,
                                     BYTE    b_ModulNbr,
                                     BYTE    b_TimerNbr,
                                     BYTE    b_TimerMode,
                                     ULONG   ul_ReloadValue,
                                     BYTE    b_InputClockSelection,
                                     BYTE    b_InputClockLevel,
                                     BYTE    b_OutputLevel,
                                     BYTE    b_HardwareGateLevel)
```

Parameter:

- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu konfigurieren (0 bis 2)
BYTE	b_TimerMode	Auswahl des Timer-Modes (0 bis 5) 0: Interrupt am Ende des Zählvorgangs 1: Monoflop durch Hardware retriggerbar 2: Impulsgenerator 3: Rechteck-Generator 4: Strobe durch Software getriggert 5: Strobe durch Hardware getriggert Siehe Kapitel 2 und Tabelle 3-5.
ULONG	ul_ReloadValue	Start-Zählerwert oder Teilerfaktor Siehe Tabelle 3-5.
BYTE	b_InputClockSelection	Auswahl des Eingangs-Timertakts Siehe Tabelle 3-6.
BYTE	b_InputClockLevel	Auswahl des Eingangs-Taktpegels. 0: aktiv bei "Low" 1: aktiv bei "High" (Eingang invertiert)
BYTE	b_OutputLevel	Auswahl des Ausgangs-Taktpegels. 0: aktiv bei "Low" 1: aktiv bei "High" (Ausgang invertiert)
BYTE	b_HardwareGateLevel	Auswahl des Hardware-Gate-Pegels. 0: aktiv bei "Low" (Eingang invertiert) 1: aktiv bei "High" Wird der Externgate nicht benutzt, setzen Sie den Parameter auf "0".

- Ausgabe:

Es erfolgt keine Ausgabe.

Tabelle 3-3: Timer-Mode

Mode	Bedeutung	*u_ReloadValue*	Hardware Gate
0	Interrupt am Ende des Zählvorgangs	Start-Wert	Hardware Gate
1	Monoflop, durch Hardware retriggerbar	Start-Wert	Hardware Trigger
2	Impulsgenerator	Teilerfaktor	Hardware Gate
3	Rechteck-Generator	Teilerfaktor	Hardware Gate
4	Strobe, durch Software getriggert	Start-Wert	Hardware Gate
5	Strobe, durch Hardware getriggert (retriggerbar)	Start-Wert	Hardware Trigger

Tabelle 3-4: Auswahl des Eingangstakts

b_InputClockSelection	Beschreibung
APCI1710_PCI_BUS_CLOCK	Der durch 4 dividierte PCI Bus-Takt wird für den Timer Eingangstakt benutzt. Der PCI Bus-Takt kann 30MHz oder 33MHz sein. Für den Timer 0 sind nur diese Auswahl und APCI1710_10MHZ möglich.
APCI1710_10MHZ	Der durch 4 dividierte 40MHz Takt wird für den Timer Eingangstakt benutzt. Für den Timer 0 sind nur diese Auswahl und APCI1710_PCI_BUS_CLOCK möglich.
APCI1710_ FRONT_CONNECTOR_INPUT	Auf den Frontstecker kann ein Eingangstakt für den Timer 1 und 2 eingespeist werden. Diese Taktquelle kann den Ausgangstakt des Timers 0 oder alle andere Taktquellen überschreiben.

Aufgabe:
Konfiguration des Timers (*b_TimerNbr*) und des Betriebsmodes (*b_TimerMode*) des ausgewählten Moduls (*b_ModulNbr*). Diese Funktion ist als erste aufzurufen, bevor Sie eine Funktion aufrufen, die auf den Timer zugreift.

Funktionsaufruf:
ANSI C :

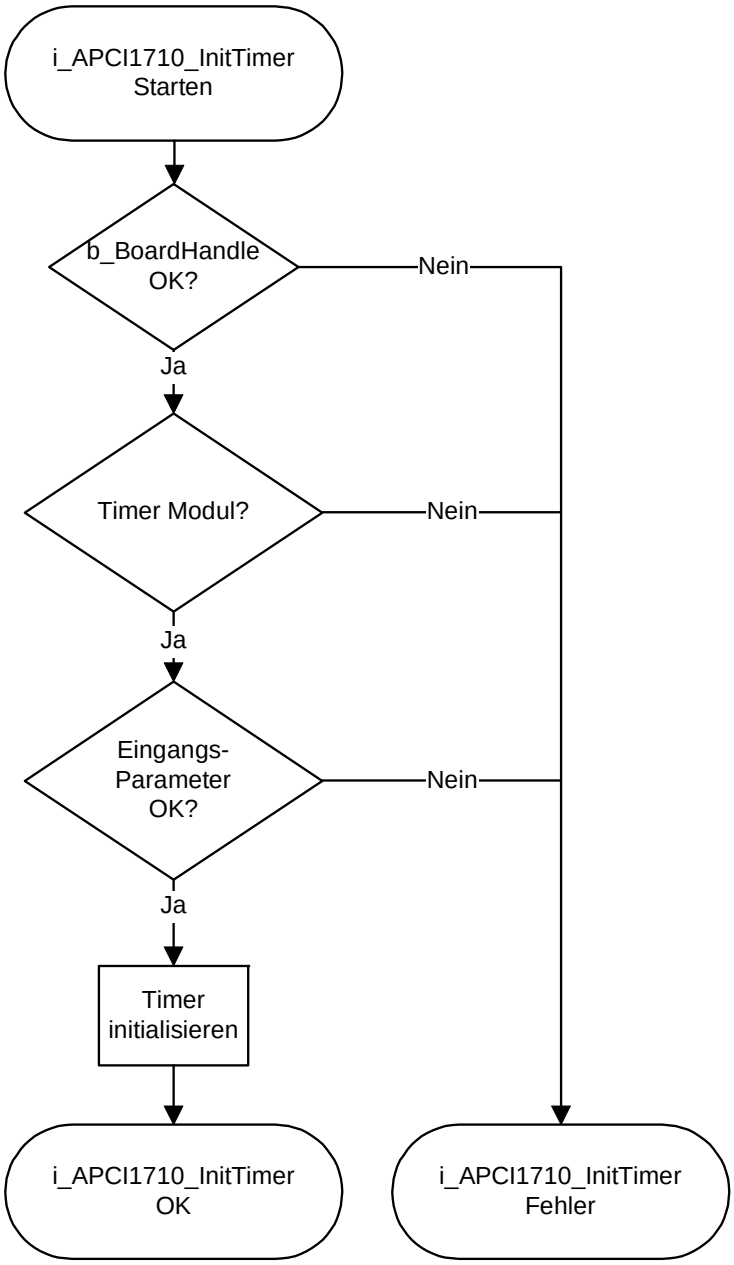
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_InitTimer
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     2,
                                     0xFF00,
                                     APCI1710_PCI_BUS_CLOCK,
                                     1,
                                     1,
                                     1);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Der ausgewählte Betriebsmode ist falsch.
-6: Der ausgewählte Timer-Eingangstakt ist falsch.
-7: Der ausgewählte Eingangs-Taktpegel ist falsch.
-8: Der ausgewählte Ausgangs-Taktpegel ist falsch.
-9: Der ausgewählte Hardware Gate-Pegel ist falsch.

Eingabe

B_BoardHandle	b_InputClockSelection
b_ModulNbr	b_InputClockLevel
b_TimerNbr	b_OutputLevel
b_TimerMode	b_HardwareGateLevel
ul_ReloadValue	

Funktionsablauf



Ausgabe

<Return-Wert>

2) i_APCI1710_EnableTimer (...)

Syntax:
<Return-Wert> = i_APCI1710_EnableTimer
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
BYTE b_InterruptEnable)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers freizugeben (0 bis 2)
BYTE	b_InterruptEnable	Aktiviert oder deaktiviert den Timer-Interrupt. APCI1710_ENABLE: Interrupt aktiviert APCI1710_DISABLE: Interrupt deaktiviert

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Aktiviert den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*).
Rufen Sie zuerst die Funktion "i_APCI1710_InitTimer", bevor Sie diese Funktion aufrufen.
Wenn der Timer-Interrupt freigegeben ist, generiert der Timer einen Interrupt, nachdem der Zählerwert "0" erreicht hat.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineX".

Funktionsaufruf:
ANSI C :

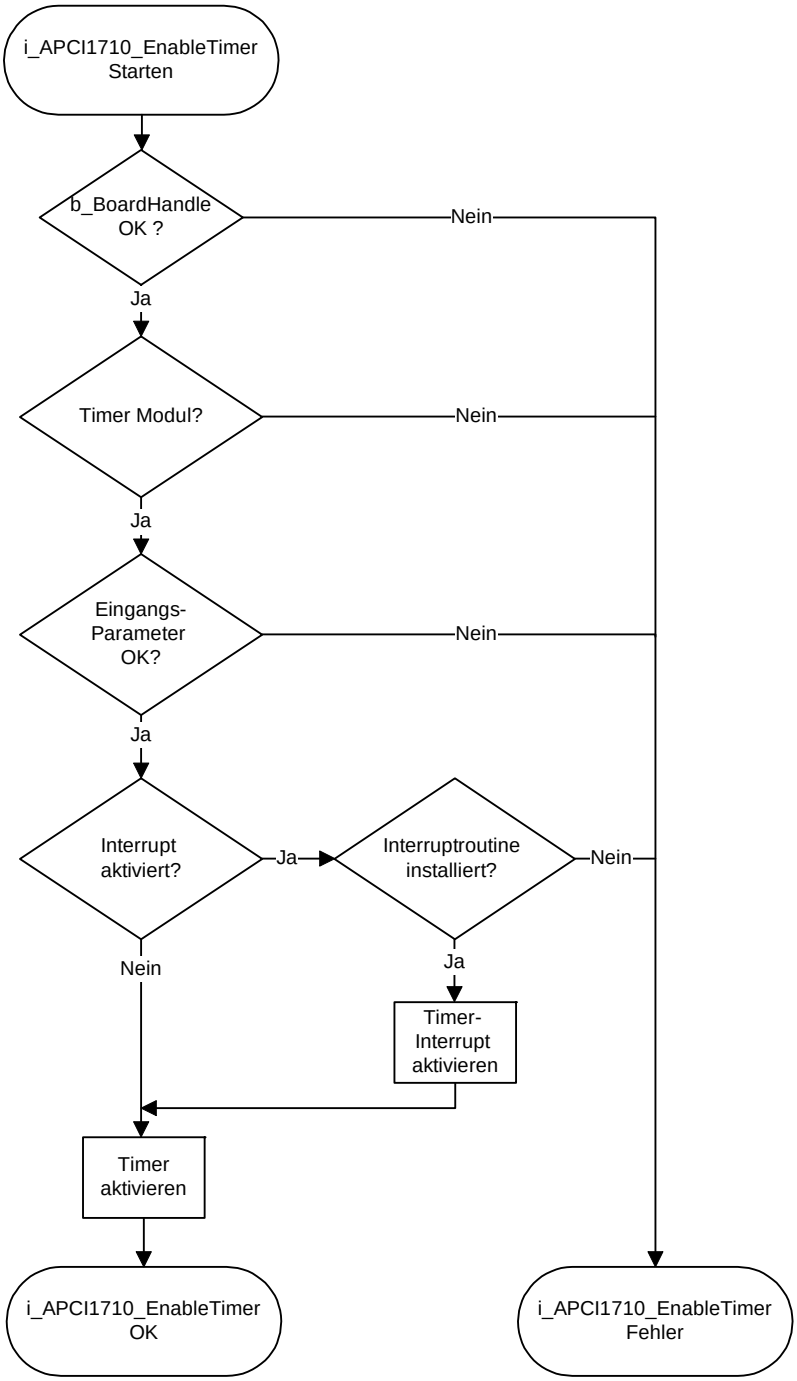
```
int            i_ReturnValue;
unsigned char  b_BoardHandle;

i_ReturnValue = i_APCI1710_EnableTimer
                (b_BoardHandle,
                 0,
                 0,
                 APCI1710_DISABLE);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"
-6: Interrupt-Parameter ist falsch.
-7: Interrupt-Funktion nicht initialisiert.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineX"

Eingabe

B_BoardHandle	b_TimerNbr
b_ModulNbr	b_InterruptEnable

Funktionsablauf



Ausgabe

<Return-Wert>

3) i_APCI1710_DisableTimer (...)

Syntax:
<Return-Wert> = i_APCI1710_DisableTimer
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_TimerNbr)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu sperren (0 bis 2)

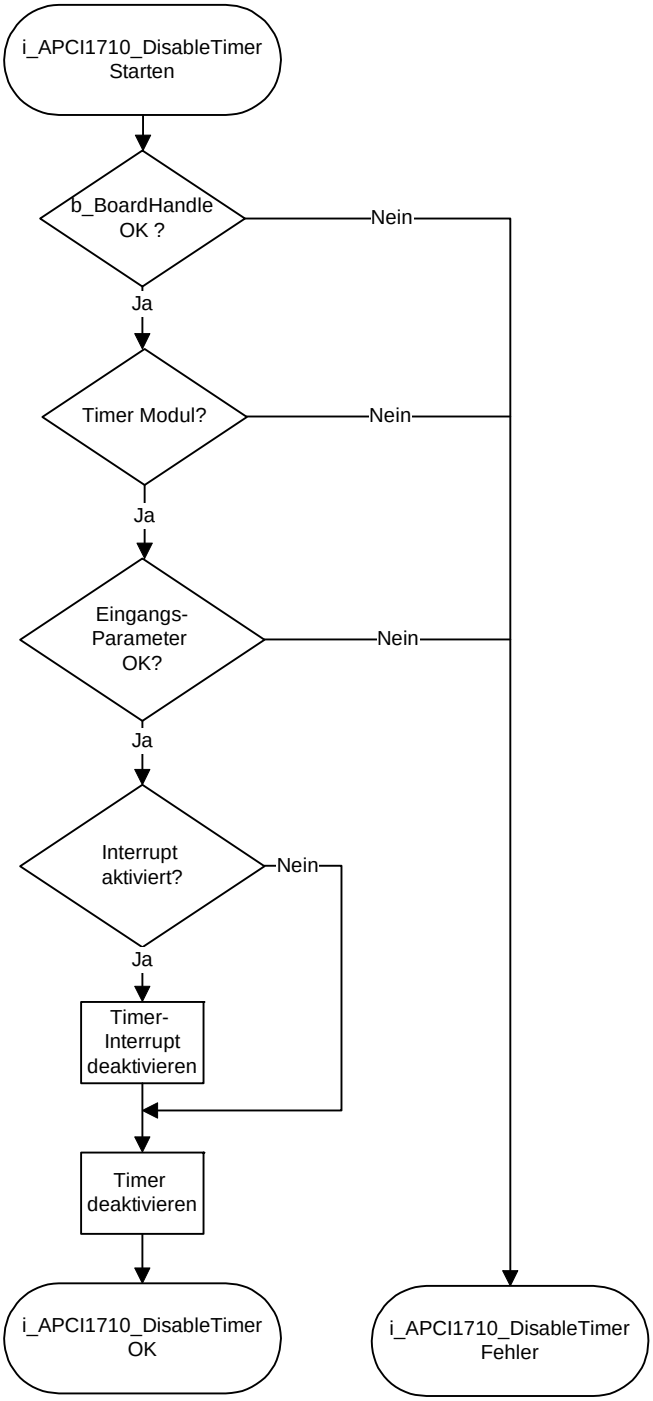
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableTimer
                (b_BoardHandle,
                 0,
                 0);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle b_ModulNbr	b_TimerNbr
Funktionsablauf	
	
Ausgabe	
<Return-Wert>	

3.3.2 Den Timer lesen

1) i_APCI1710_ReadTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_ReadTimerValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
PULONG pul_TimerValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu lesen (0 bis 2)

- Ausgabe:

PULONG	pul_TimerValue	Timer-Wert

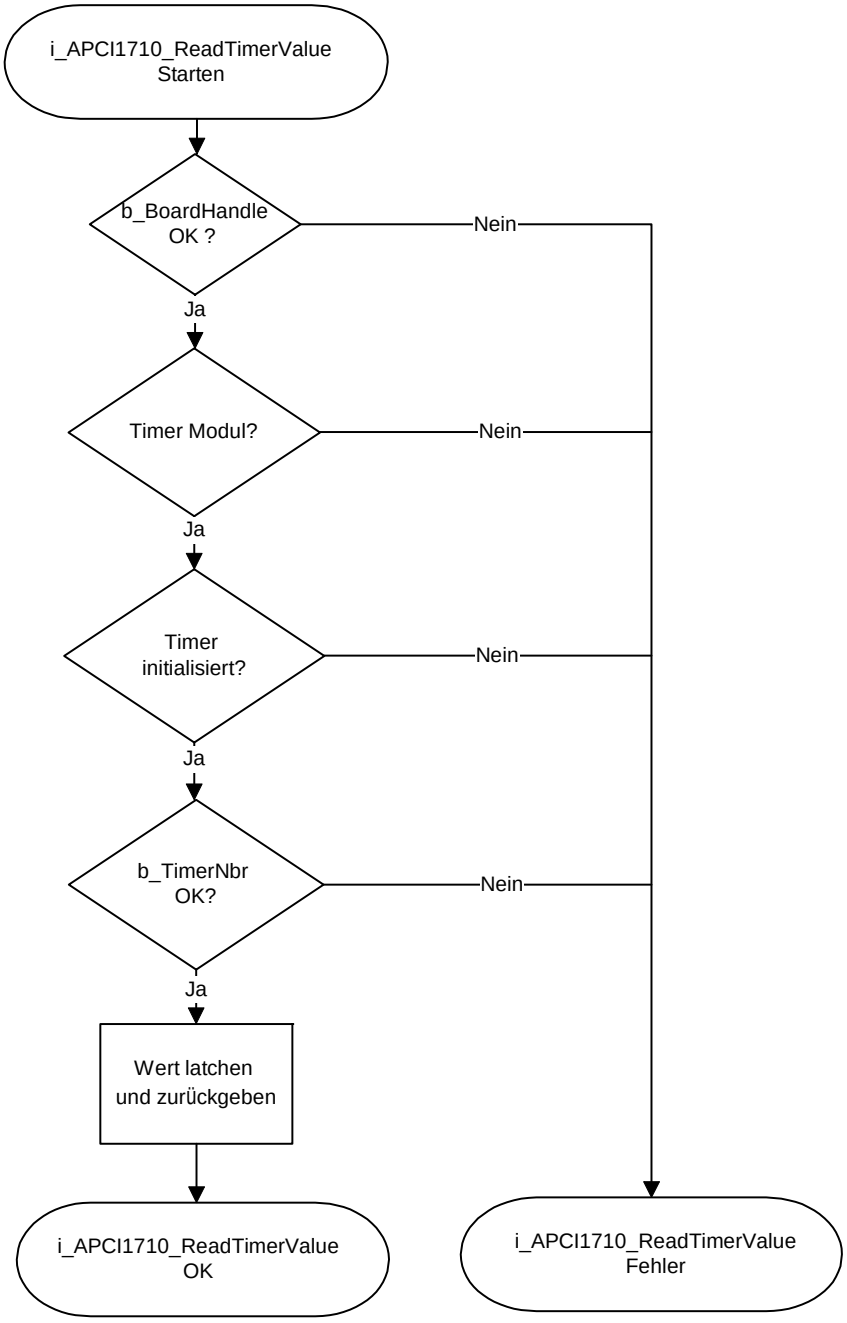
Aufgabe:
Gibt den Wert für den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

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

i_ReturnValue = i_APCI1710_ReadTimerValue
                                (b_BoardHandle,
                                 0,
                                 0,
                                 & ul_TimerValue);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle b_ModulNbr	b_TimerNbr
Funktionsablauf	



Ausgabe
Pul_TimerValue
<Return-Wert>

2) i_APCI1710_ReadAllTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_ReadAllTimerValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PULONG pul_TimerValueArray)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren

- Ausgabe:

PULONG	pul_TimerValueArray	Wertsequenz des Timers. Element 0 enthält den Wert des Timers 0 Element 1 enthält den Wert des Timers 1 Element 2 enthält den Wert des Timers 2

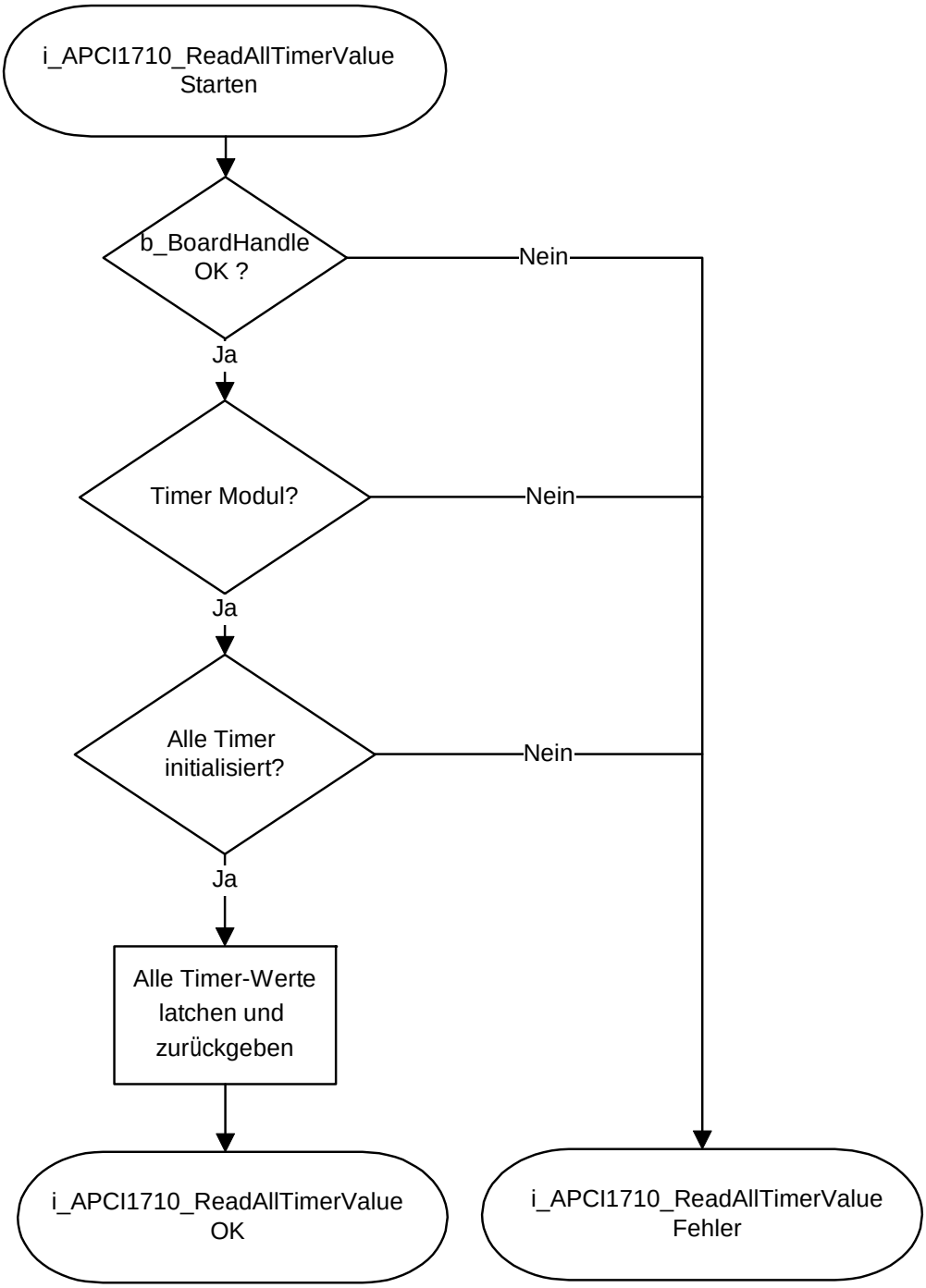
Aufgabe:
Gibt alle Timer-Werte des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned long   ul_TimerValueArray [3];

i_ReturnValue = i_APCI1710_ReadAllTimerValue
                                (b_BoardHandle,
                                 0,
                                 ul_TimerValueArray);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer 0 nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"
-6: Timer 1 nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"
-7: Timer 2 nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle	b_ModulNbr
Funktionsablauf	



Ausgabe
Pul_TimerValueArray
<Return-Wert>

3) i_APCI1710_GetTimerOutputLevel (...)

Syntax:
<Return-Wert> = i_APCI1710_GetTimerOutputLevel
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
PBYTE pb_OutputLevel)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu testen (0 bis 2)

- Ausgabe:

PBYTE	pb_OutputLevel	Pegel des Ausgangssignals "0": der Ausgang ist auf "Low" gesetzt "1": der Ausgang ist auf "High" gesetzt

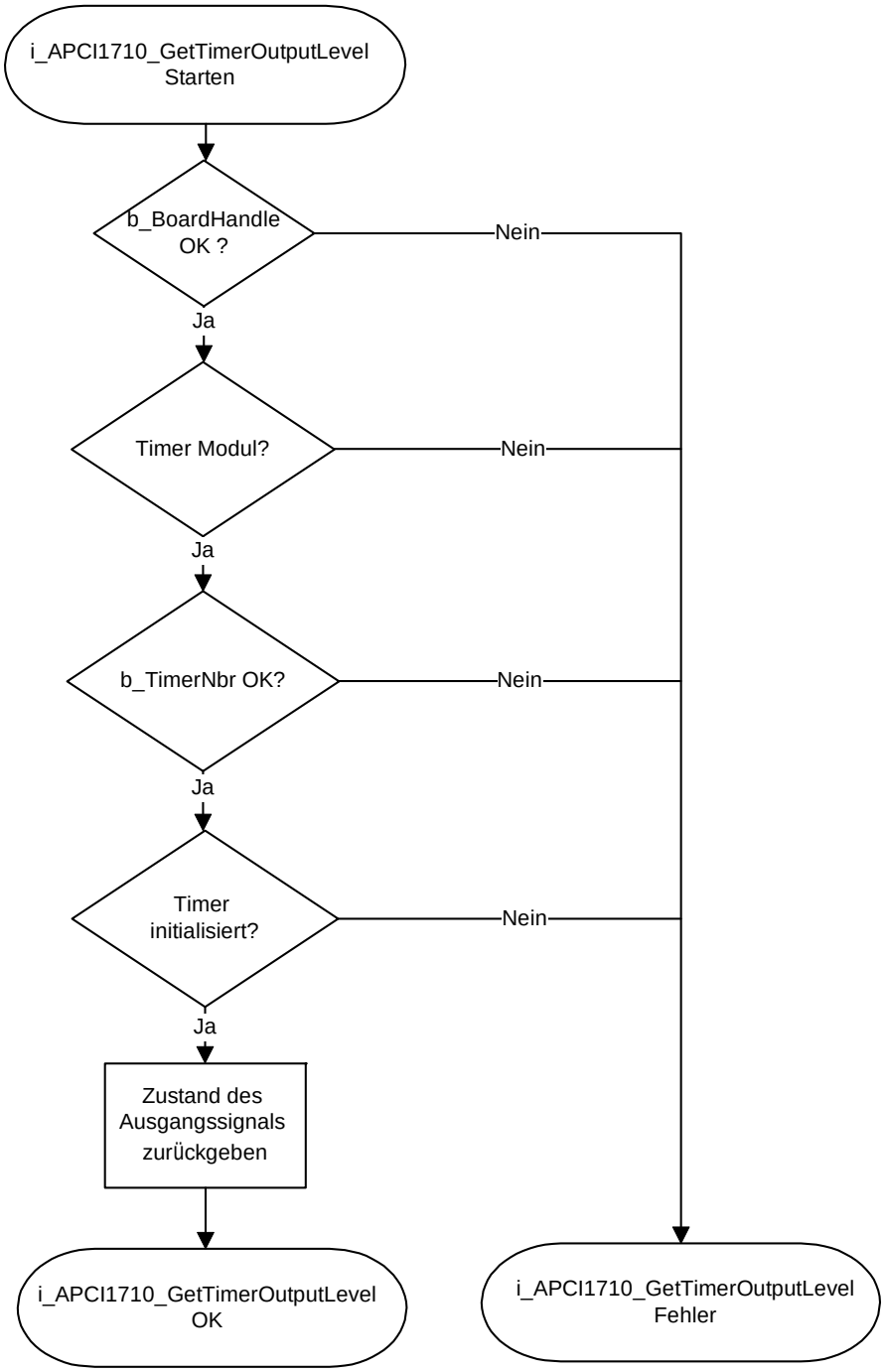
Aufgabe:
Gibt den Pegel des Ausgangssignals (*pb_OutputLevel*) für den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_OutputLevel;

i_ReturnValue = i_APCI1710_GetTimerOutputLevel
                                    (b_BoardHandle,
                                     0,
                                     0
                                     &b_OutputLevel);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle b_ModlNbr	b_TimerNbr
Funktionsablauf	



Ausgabe
Pb_OutputLevel
<Return-Wert>

4) i_APCI1710_GetTimerProgressStatus (...)

Syntax:
<Return-Wert> = i_APCI1710_GetTimerProgressStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
PBYTE pb_TimerStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu testen (0 bis 2)

- Ausgabe:

PBYTE	pb_TimerStatus	Timer-Zustand "0": Timer ist gestoppt "1": Timer läuft

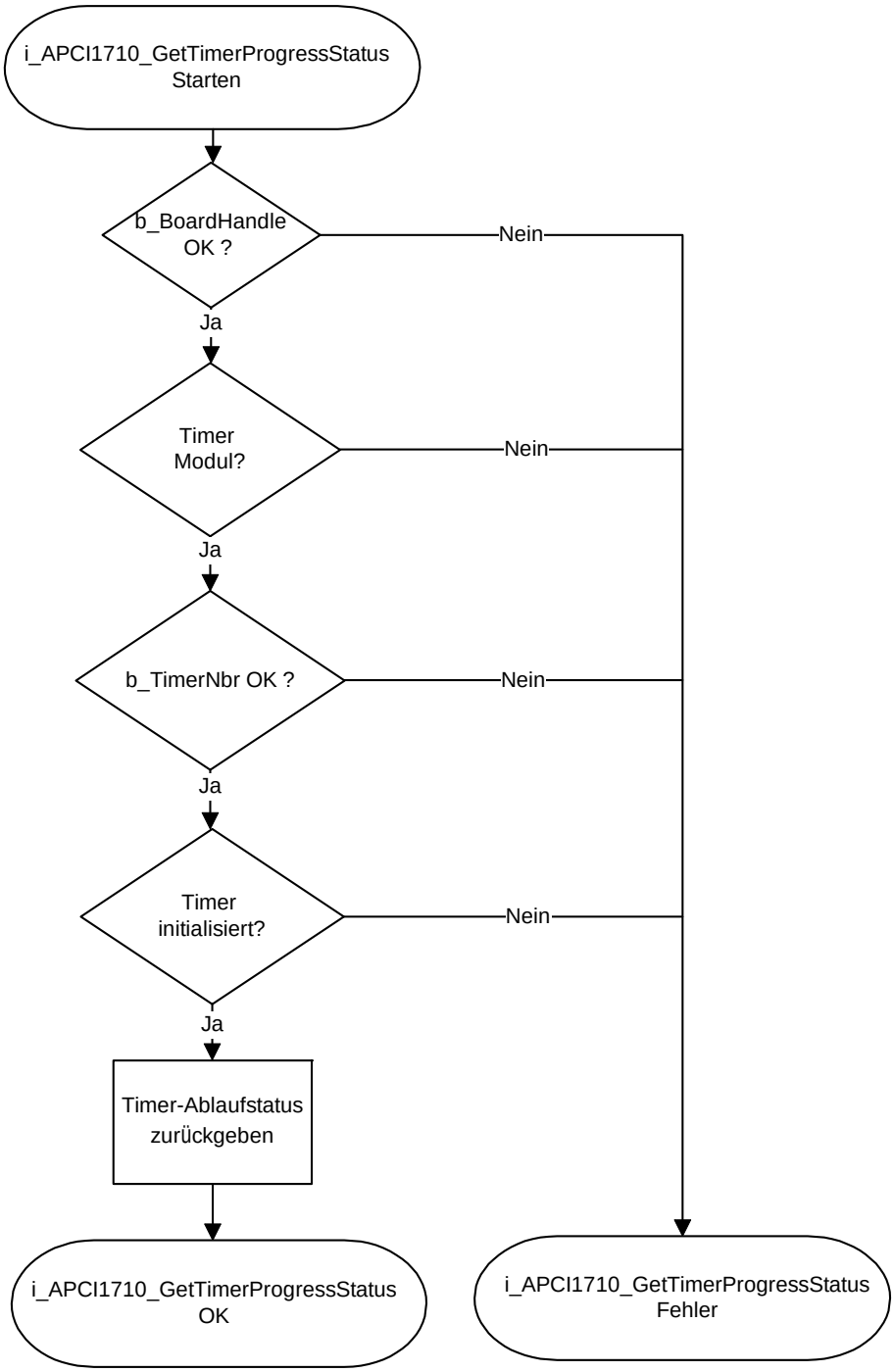
Aufgabe:
Gibt den Zustand (*pb_TimerStatus*) für den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_TimerStatus;

i_ReturnValue = i_APCI1710_GetTimerProgress
                                (b_BoardHandle,
                                 0,
                                 0
                                 &b_TimerStatus);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle b_ModulNbr	b_TimerValue
Funktionsablauf	



Ausgabe
b_TimerStatus
<Return-Wert>

3.3.3 Auf den Timer schreiben

1) i_APCI1710_WriteTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_WriteTimerValue
(BYTE b_BoardHandle
BYTE b_ModulNbr,
BYTE b_TimerNbr,
ULONG ul_WriteValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu testen (0 bis 2)
ULONG	ul_WriteValue	Wert zu schreiben

- Ausgabe:
Es erfolgt keine Ausgabe.

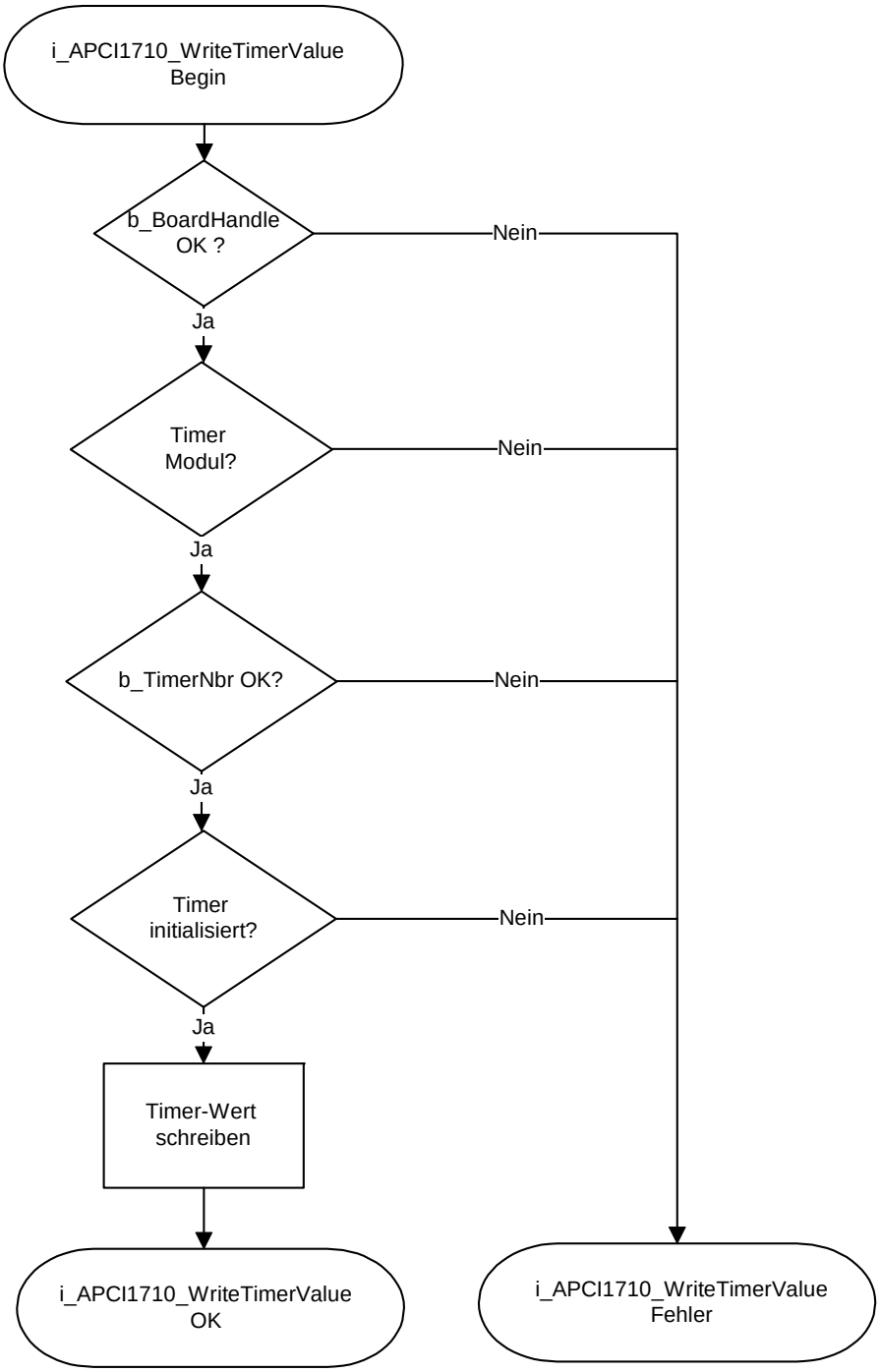
Aufgabe:
Schreibt den Wert (ul_WriteValue) in den ausgewählten Timer (*b_TimerNbr*) des ausgewählten Moduls (b_ModulNbr). Diese Funktion hängt von dem benutzten Betriebsmode ab. Siehe Tabelle 3-5.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_APCI1710_WriteTimerValue
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     0xFF00FF00);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch.
-2: Die ausgewählte Modulnummer ist falsch.
-3: Der ausgewählte Timer ist falsch.
-4: Das ausgewählte Modul ist kein "Timer"-Modul.
-5: Timer nicht initialisiert. Siehe Funktion "i_APCI1710_InitTimer"

Eingabe	
B_BoardHandle b_ModulNbr	b_TimerNbr ul_WriteValue
Funktionsablauf	



Ausgabe
<Return-Wert>

3.4 Funktionen im Kernel-Mode

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

3.4.1 Den Timer lesen

1) i_APCI1710_KRNL_ReadTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_KRNL_ReadTimerValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
PULONG pul_TimerValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu lesen (0 bis 2)

- Ausgabe:

PULONG	pul_TimerValue	Timer-Wert

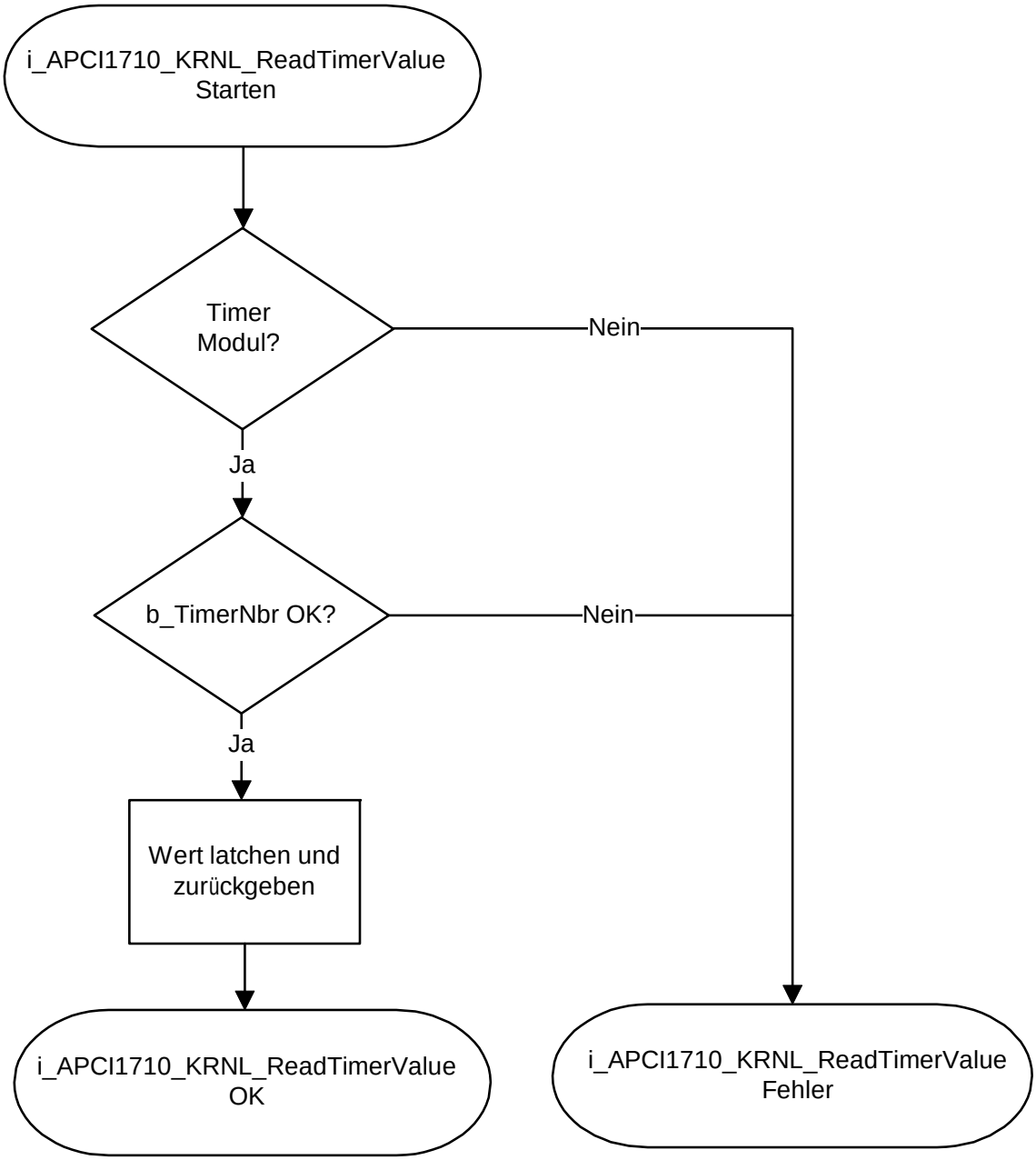
Aufgabe:
Gibt den Wert für den Timer (*b_TimerNbr*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned long ul_TimerValue;

i_ReturnValue = i_APCI1710_KRNL_ReadTimerValue
                                    (ui_BaseAddress,
                                    0,
                                    0,
                                    &ul_TimerValue);
```

Return-Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Der ausgewählte Timer ist falsch.
-3: Das ausgewählte Modul ist kein "Timer"-Modul.

Eingabe	
Ui_BaseAddress b_ModulNbr	b_TimerNbr
Funktionsablauf	



Ausgabe
Pul_TimerValue
<Return-Wert>

2) i_APCI1710_KRNL_ReadAllTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_KRNL_ReadAllTimerValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PULONG pul_TimerValueArray)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)

- Ausgabe:

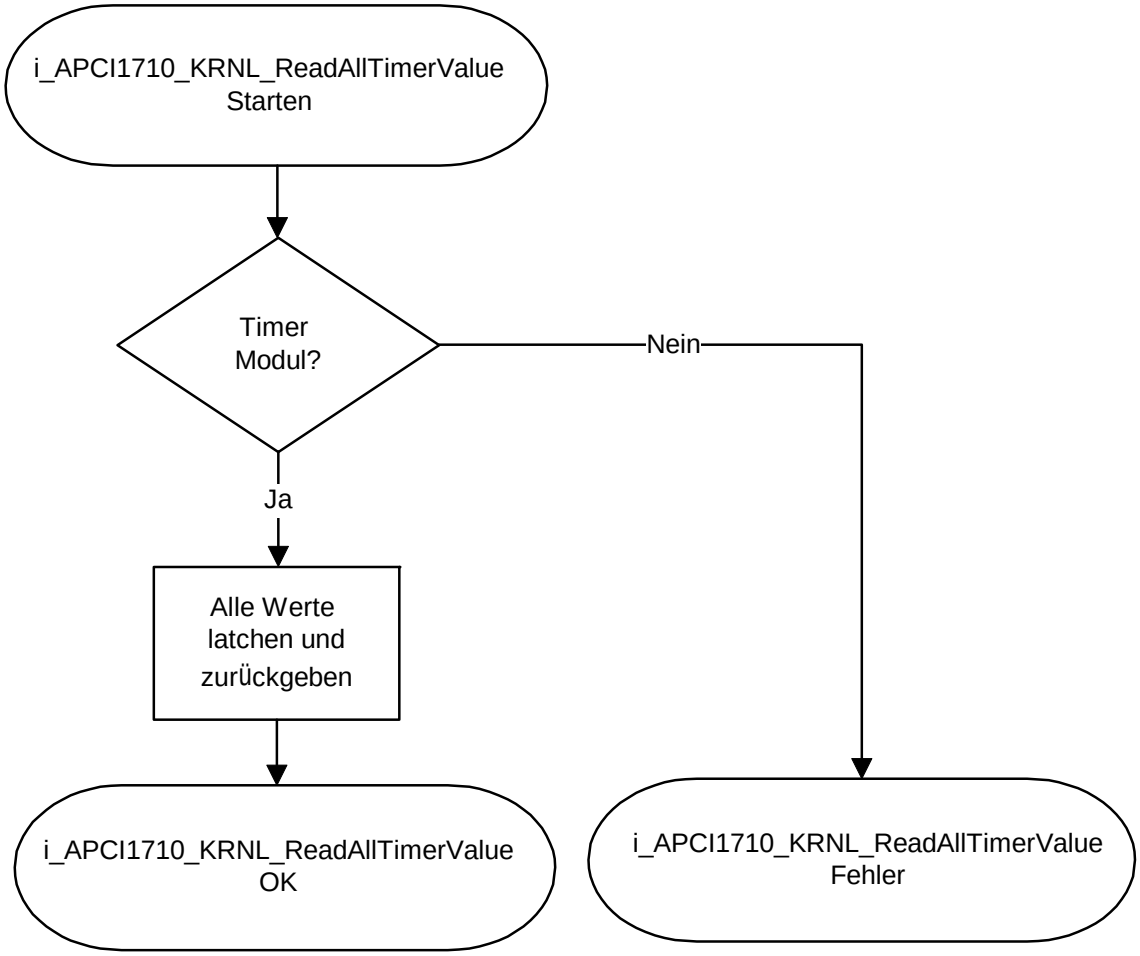
PULONG	pul_TimerValueArray	Wertsequenz des Timers. Element 0 enthält den Wert des Timers 0 Element 1 enthält den Wert des Timers 1 Element 2 enthält den Wert des Timers 2

Aufgabe:
Gibt alle Timer-Werte des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned long ul_TimerValueArray [3];

i_ReturnValue = i_APCI1710_KRNL_ReadAllTimerValue
                                    (ui_BaseAddress,
                                     0,
                                     ul_TimerValueArray);
```

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

Eingabe	
ui_BaseAddress b_ModulNbr	b_TimerNbr
Funktionsablauf	
	
Ausgabe	
pul_TimerValueArray	
<Return-Wert>	

3.4.2 Auf den Timer schreiben

1) i_APCI1710_KRNL_WriteTimerValue (...)

Syntax:
<Return-Wert> = i_APCI1710_KRNL_WriteTimerValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_TimerNbr,
ULONG ul_WriteValue)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_TimerNbr	Nummer des Timers zu testen (0 bis 2)
ULONG	ul_WriteValue	Wert zu schreiben

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Schreibt den Wert (ul_WriteValue) in den ausgewählten Timer (*b_TimerNbr*) des ausgewählten Moduls (b_ModulNbr). Diese Funktion hängt von dem benutzen Betriebsmode ab. Siehe Tabelle 3-5.

Funktionsaufruf:
ANSI C :

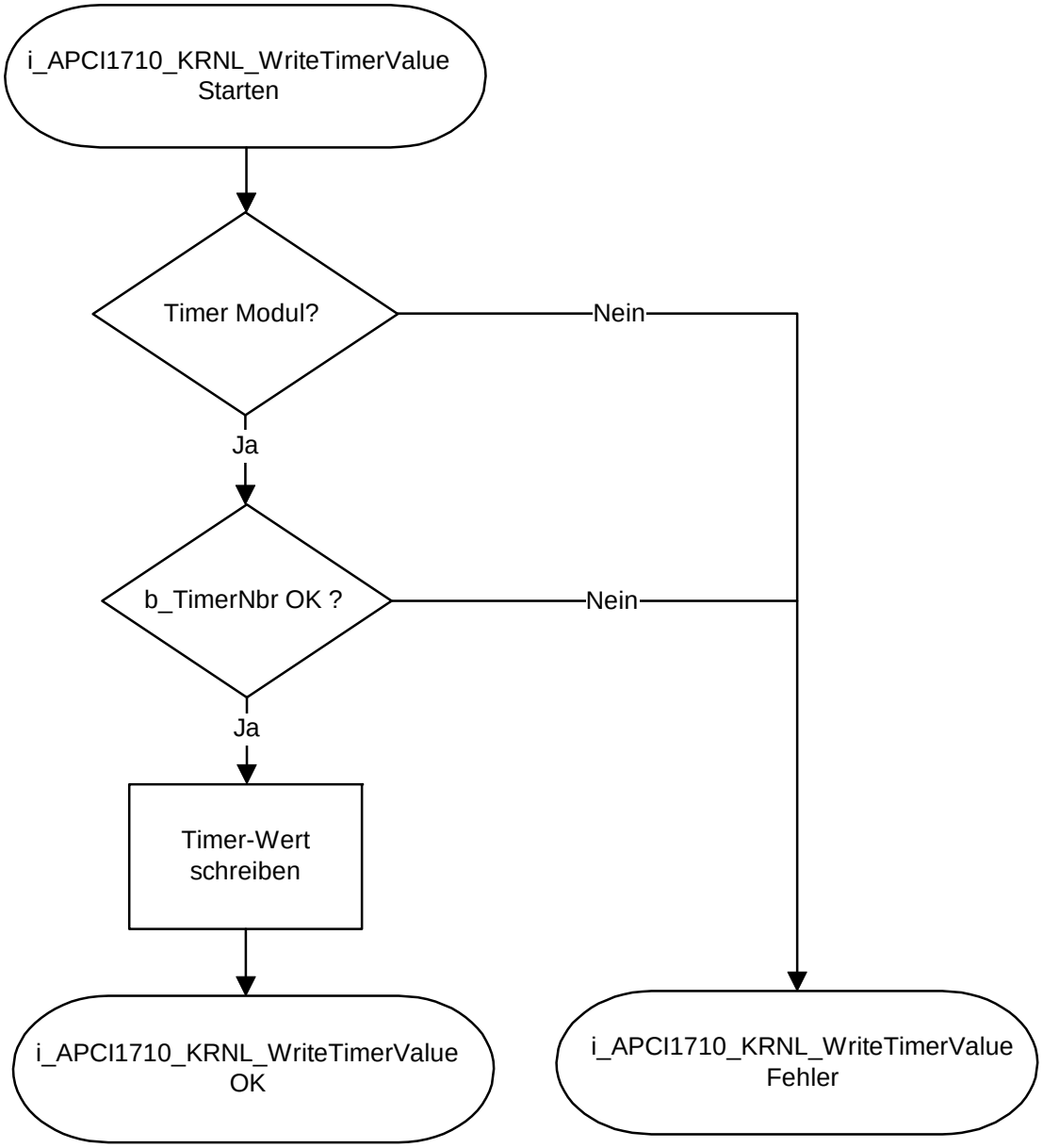
```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_WriteTimerValue
                                    (ui_BaseAddress,
                                     0,
                                     0,
                                     0xFF00FF00);
```

Return-Wert:
0: Kein Fehler
-1: Die ausgewählte Modulnummer ist falsch.
-2: Der ausgewählte Timer ist falsch.
-3: Das ausgewählte Modul ist kein "Timer"-Modul.

Eingabe

ui_Address	b_TimerNbr
b_ModulNbr	ul_WriteValue

Funktionsablauf



Ausgabe

<Return-Wert>