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

Chronos

3. 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_EA_d.pdf	TTL_IO_e.pdf	TTL I/O	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 CHRONOS

2.1 Funktionsbeschreibung

Die Funktion "CHRONOS" ist eine Timer-Schnittstelle, die es erlaubt, die Zeit zwischen zwei "Events" wie ein Chronometer zu messen.
3 Funktionen sind implementiert:

- ein 32-Bit Timer, um eine Referenzzeit zu bilden,
- ein 32-Bit Messtimer, der die Zeit zwischen Start- und Stopimpulse bestimmt und misst.
- 3 digitale Eingänge und 3 digitale Ausgänge

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

Eigenschaften:

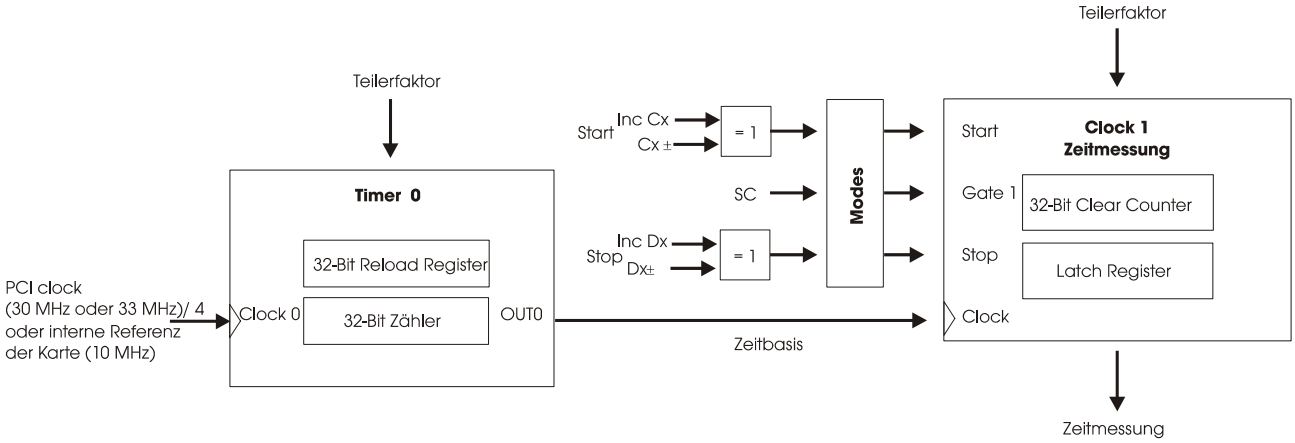
- Zur Vermeidung von Erdschleifen wird eine komplette galvanische Trennung durch Optokoppler für die Ein-/Ausgänge herangezogen.
- Interruptmöglichkeit beim Messende
- Signale bis zu 5 MHz können verarbeitet werden.
- Timer rücklesbar
- Eingänge und Ausgänge können per Software invertiert werden.
- Software GATE möglich

2.1.1 Blockdiagramm der Chronos Funktion

Die Schnittstelle enthält:

- 2 voneinander unabhängige 32-Bit Timer, die über den Datenbus ausgelesen bzw. beschrieben werden können
- digitale Ein- und Ausgänge

Abb. 2-1: CHRONOS Blockschaltbild



2.1.2 Typische Anwendungen

- Zeitmessung zwischen 2 Ereignissen,
- Echtzeituhr,
- Timer

2.2 Benutzte Signale

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

Auf einer Karte können Sie maximal 4 Chronometer (1 pro Modul) nutzen.

Tabelle 2-1: Benutzte Signale

AM STECKER	POLARITÄT	FUNKTION
A **x** +/ -	Diff. / TTL	**Digitaler Ausgang 1**, nach Reset auf Logisch 0
B **x** +/ -	Diff. / TTL	**Digitaler Ausgang 2,** nach Reset auf Logisch 0
C **x** +/ -	Diff. / TTL / Opt. 24V	**Start-Impuls** für die Messung, z.B. 0→1 Flanke = Start Eingang gefiltert. Impulsbreite > 100 ns
D **x** +/ -	Diff. / TTL / Opt. 24V	**Stop-Impuls** für die Messung, z.B. 0→1 Flanke = Stop Interruptfähig, wenn das Freigabe-Bit gesetzt ist. Eingang gefiltert Pulsbreite > 100 ns
E **x**	24V / Opt. 5V	**Digitaler Eingang 0** Wenn Spannung < als 15 V = Logisch 1 Wenn Spannung > 17 V = Logisch 0 (invertierend)
F **x**	24V / Opt. 5V	**Digitaler Eingang 1,** invertierend
G **x**	24V / Opt. 5V	**Digitaler Eingang 2,** invertierend
H **x**	24 V / Opt. TTL	**Digitaler Ausgang 0,** nach Reset auf Logisch 0

x: Nummer des Funktionsmoduls.

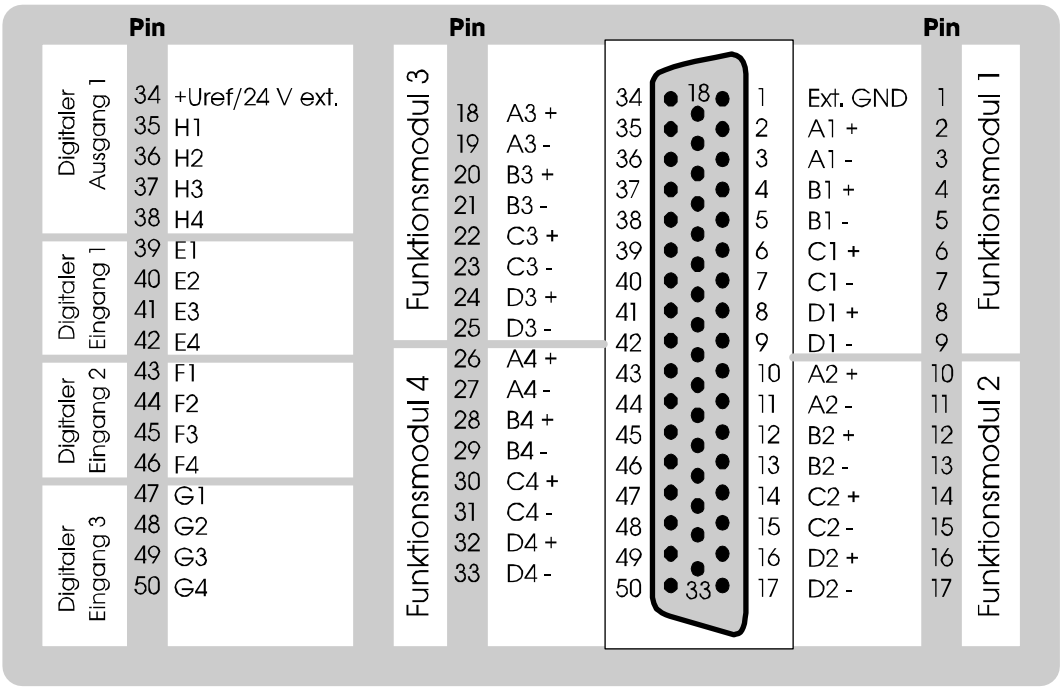
2.3 Pinbelegung des Frontsteckers

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.

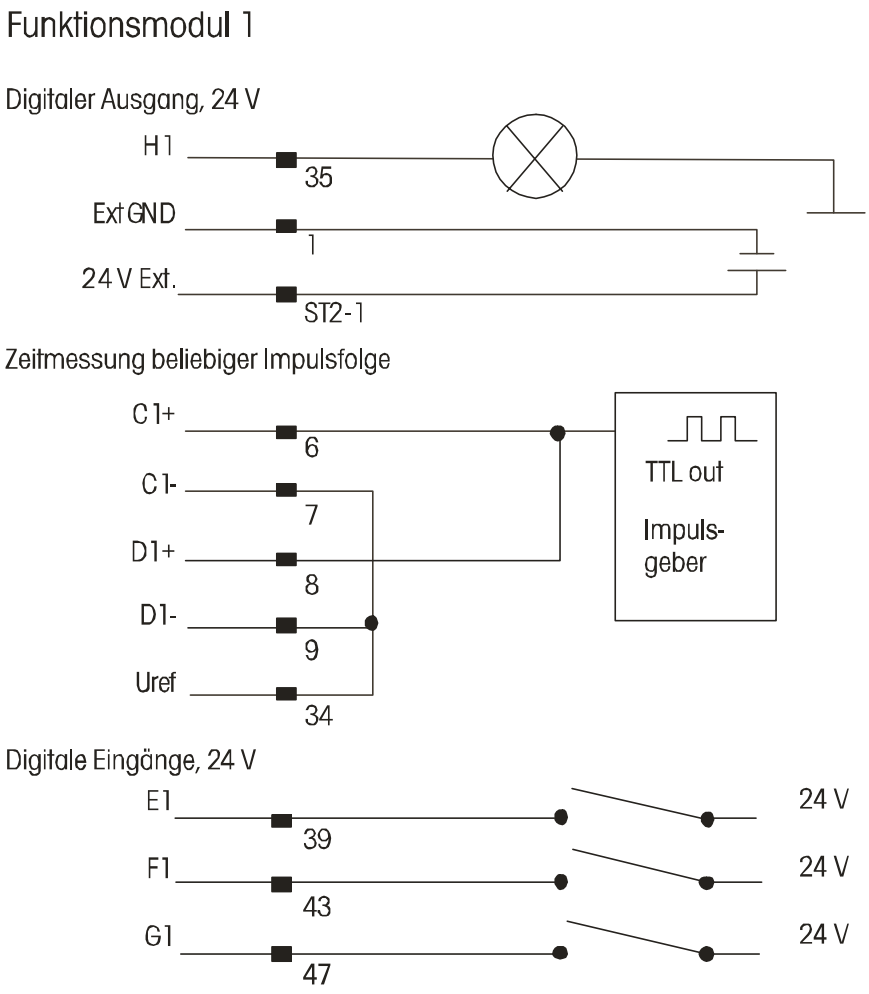
Die untere Abbildung ist ein Anschlussbeispiel: Die Funktion "Chronos" ist auf allen Funktionsmodulen implementiert.

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 Chronos-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	☑		**ZT-TIMER**	**ZT-TIMER**	**ZT-TIMER**	**ZT-TIMER**
BASE**x** + 8	☑		-	-	-	CHRONOMETER STATUS
BASE**x** + 12	☑		-	-	-	INTERRUPT STATUS
BASE**x** + 16		☑	-	-	-	CHRONOMETER CONFIG
BASE**x** + 20		☑	-	-	-	SET DIGITAL OUT0
BASE**x** + 24		☑	-	-	-	SET DIGITAL OUT1
BASE**x** + 28		☑	-	-	-	SET DIGITAL OUT2
BASE**x** + 32		☑	RESET INTERRUPT REQUEST			
BASE**x** + 36		☑	CLEAR MESUREAMENT			
			-	-	-	-
BASE**x** + 60	☑		FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

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

Die Zugriffe werden immer in 32-Bit breite gelesen oder geschrieben.

2.6 Beschreibung der E/A-Funktionen

2.6.1 Funktionsbeschreibung

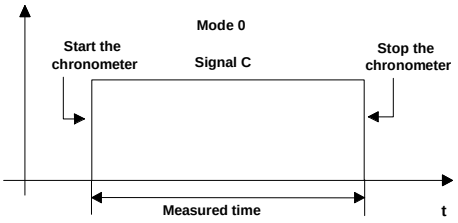
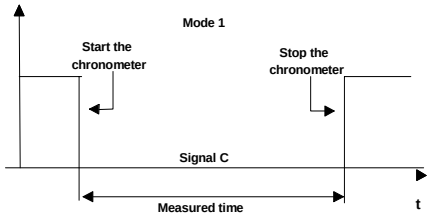
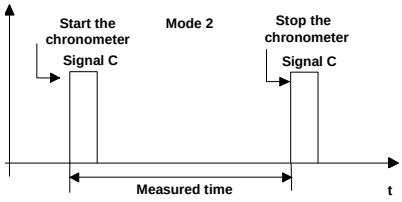
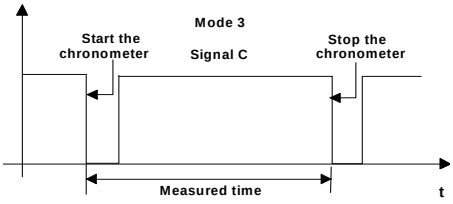
Die Funktion "Chronos" ist eine abgemagerte Version der Funktion Zähler/Timer. Grundsätzlich werden die Impulse aus Timer 0 gezählt, zwischen dem Start und Stop-Impulsen. Diese Anzahl steht im Zeitmessungstimer und kann durch E/A Zugriff gelesen werden.

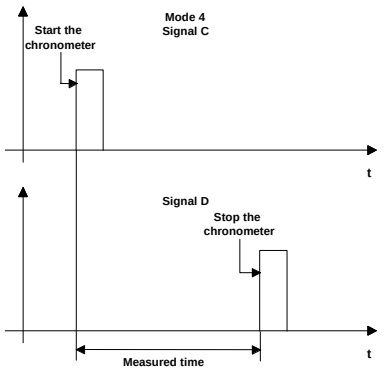
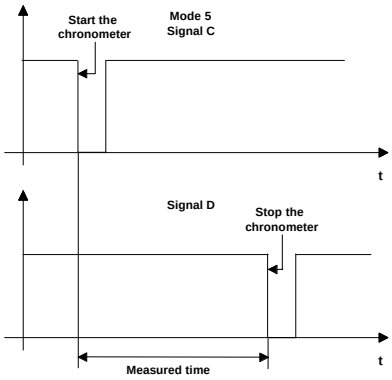
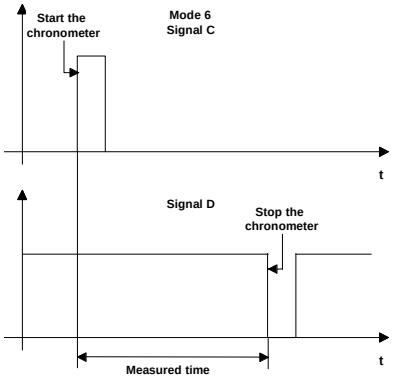
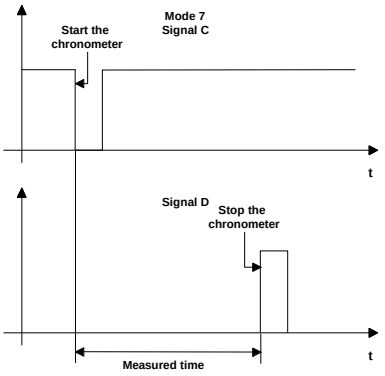
Der Timer 0 wird als Zeitreferenz-Generator benutzt.

Der Teilerfaktor wird in den Timer 0 geschrieben und legt die Zeitreferenz fest.
Der Eingangstakt ist der PCI-Bus Takt oder ein 40 MHz Oszillator auf der Karte.
Der Timer 0 ist mit dem Start "Event" synchronisiert.
Timer 0 kann jederzeit zurückgelesen werden.

Die Funktion "Chronos" kann in 8 verschiedenen Modi betrieben werden.

Tabelle 2-3: Modes der Chronos-Funktion

Mode	Modediagramm	Funktion des Signals C	Funktion des Signals D
0		Der High Pegel startet die Zeitmessung. Der Low Pegel stoppt die Messung.	Nicht benutzt
1		Der Low Pegel startet die Zeitmessung. Der High Pegel stoppt die Messung.	Nicht benutzt
2		Der High Pegel startet die Zeitmessung. Der nächste High Pegel stoppt die Zeitmessung.	Nicht benutzt
3		Der Low Pegel startet die Zeitmessung. Der nächste Low Pegel stoppt die Zeitmessung.	Nicht benutzt

Mode	Modediagramm	Funktion des Signals C	Funktion des Signals D
4		Der High-Pegel startet die Zeitmessung	Der High-Pegel stoppt die Zeitmessung.
5		Der Low-Pegel startet die Zeitmessung	Der Low-Pegel stoppt die Zeitmessung
6		Der High-Pegel startet die Zeitmessung	Der Low-Pegel stoppt die Zeitmessung
7		Der Low-Pegel startet die Zeitmessung	Der High-Pegel stoppt die Zeitmessung

Zeitmessung
Sobald ein **Start**-Event, (z.B.: 0→1) am Eingang C**x** erfolgt, wird der Timer für Zeitmessung zurückgesetzt. Er zählt dann die Impulse vom Timer 0.
Während des Vorgangs ist im Statusregister das Bit "Messung läuft" gesetzt.
Sobald ein **Stop**-Event (z.B.: 0→1) am Eingang D**x** erfolgt, wird der Timer angehalten und das Statusbit "Messung in Progress" zurückgesetzt.
Ein Interrupt kann ebenfalls generiert werden. Der Wert muß dann ausgelesen werden. Der zuletzt gemessenen Wert kann im Zeitmessung-Register gelesen werden. Der Zeitmessung-Timer kann jederzeit zurückgelesen werden.

2.6.2 Timer0-REGISTER (Base +0)

32-Bit Register: der "Reload" Wert für den TIMER 0 wird geschrieben. Die Ausgangsfrequenz des Timer 0 wird durch den Teilerfaktor festgelegt.

0 = Teilerfaktor 2 → Frequenz max. 20 MHz bzw. PCI-Takt/2

Beim Lesen dieser Adresse wird der aktuelle gelatchte Wert des TIMER 0 gelesen.

2.6.3 Timer1-Register (Base +4)

32-Bit Register. Beim Lesen des Registers wird der aktuelle Wert (d.h. letzte Messung) der Zeitmessung gelesen.

2.6.4 Chronometer-Status-Register (Base +8)

32-Bit Register, das Statusinformationen über den Chronometer-Zustand zurückgibt. Das Register kann nur gelesen werden.

DQ0 : 0: Messung nicht gestartet
1: Messung gestartet; Start-Event ist eingetroffen,
Rücksetzen durch Schreiben auf Base + 36

DQ1 : 0: Messung nicht beendet
1: Messung beendet; Stop-Event ist eingetroffen,
Rücksetzen durch Schreiben auf Base + 36

DQ2 : 0: Messung läuft nicht
1: Messung läuft,
Rücksetzen durch Schreiben auf Base + 36

DQ3 : 0: Zeitmessungstimer ist nicht abgelaufen
1: Timer ist abgelaufen

DQ4 : 0: Single_Mode ist ausgewählt
Die Messung wird mit dem Stop-Event gestoppt
1: Continuous_Mode ist ausgewählt
Die Messung wird automatisch durch das nächste Start-Event gestartet

DQ5: 0 PCI-Bus Clock wird als Zeitbasis verwendet.
1 Interner Quarz wird als Zeitbasis verwendet (40MHz).

Die restlichen Bits werden beim Rücklesen auf 0 gesetzt.

2.6.5 Interrupt-Status-Register (Base +12)

Nur lesbar

DQ0 : Invertiertes Bild des digitalen Eingangs 0

DQ1 : Invertiertes Bild des digitalen Eingangs 1

DQ2 : Invertiertes Bild des digitalen Eingangs 2

DQ3 : =1: Interrupt bei Messungsende
(Reset der Interruptanforderung beim Schreiben auf Base + 32)

DQ31..4 immer auf 0 gesetzt.

2.6.6 Chronometer-Konfiguration-Register (Base +16)

Nicht rücklesbar.
DQ0: Modebit 0
DQ1: Modebit 1

Die Modebits legen die Messung fest:

B00: Zeit zwischen 2 Impulsen auf der C**x** Leitung
B01: Zeit des Impulses auf der C**x** Leitung
B10: Zeit zwischen Impulsen auf der C**x** und D**x** Leitung

Die Impulse können invertiert werden. Zusätzliche Modi (Siehe Tabelle 2-3)

DQ2: = 1 Invertierung des Signals auf der Cx Leitung,
= 0 nicht invertiert (nach Reset)

DQ3: = 1 Invertierung des Signals auf der Dx Leitung,
= 0 nicht invertiert (nach Reset)

DQ4: = 1 Softwarefreigabe des Chronometers,
= 0 Chronometer gesperrt (nach Reset)

DQ5: = 1 Interrupt generiert bei Messungsende
= 0 kein Interrupt (nach Reset)

DQ6: = 1 Continuous_Mode eingestellt,
= 0 Single_Mode eingestellt (nach Reset)

DQ31..7: keine Funktion

2.6.7 SET-OUT0 Register (Base +20)

DQ0 =1 bewirkt das Setzen des Ausgangs Hx: Strom fließt

DQ0 =0 bewirkt das Rücksetzen des Ausgangs Hx: Strom fließt nicht

2.6.8 SET-OUT1 Register (Base +24)

DQ0 =1 bewirkt das Setzen des Ausgangs Ax: Strom fließt

DQ0 =0 bewirkt das Rücksetzen des Ausgangs Ax: Strom fließt nicht

2.6.9 SET-OUT2 Register (Base +28)

DQ0 =1 bewirkt das Setzen des Ausgangs Bx : Strom fließt

DQ0 =0 bewirkt das Rücksetzen des Ausgangs Bx : Strom fließt nicht

2.6.10 Version Register (Base +60)

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

BASE + 60 "C" "H" "1" "1"

Bedeutet: Chronos Revision 1.1

2.7 Arbeiten mit der Chronos Funktion

- Anschluss des Signalgebers an die Karte
- Parametrierung der API Funktion (Taktauswahl, Zeitreferenz, Start/Stop-Event, Single oder Continuous Mode)
- Status der Messung per Polling oder Interrupt auswerten
- Zeitmessungs-Timer auslesen
- Wert aus dem Zeitmessungs-Timer und Zeitreferenz ergibt die Zeit zwischen Start-Event und Stop-Event

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_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28

3.2 Interruptmaske

Jede Impulszähler 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 "Chronos"

b_ModuleMask	ul_InterruptMask	Bedeutung
0000 000**1**	0000 0000 **1**000 0000	Interrupt auf Modul 0 ausgelöst
0000 00**1**0	0000 0000 **1**000 0000	Interrupt auf Modul 1 ausgelöst
0000 0**1**00	0000 0000 **1**000 0000	Interrupt auf Modul 2 ausgelöst
0000 **1**000	0000 0000 **1**000 0000	Interrupt auf Modul 3 ausgelöst

Tabelle 3-3: Rückgabetabelle für den Zählerwert

b_ModuleMask	ul_InterruptMask	Quelle	ul_CounterLatchValue
b_ModuleMask = 1	ul_InterruptMask = 128	Stop-Signal vom Modul 0. Messung wird gestoppt.	Der Chronometer hat einen Zählerwert gemessen.
b_ModuleMask = 2	ul_InterruptMask = 128	Stop-Signal vom Modul 1. Messung wird gestoppt.	Der Chronometer hat einen Zählerwert gemessen.
b_ModuleMask = 4	ul_InterruptMask = 128	Stop-Signal vom Modul 2. Messung wird gestoppt.	Der Chronometer hat einen Zählerwert gemessen.
b_ModuleMask = 8	ul_InterruptMask = 128	Stop-Signal vom Modul 3. Messung wird gestoppt.	Der Chronometer hat einen Zählerwert gemessen.

3.3 Chronos-Initialisierung

1) i_APCI1710_InitChrono (...)

Syntax:

```
<Return Wert> =  i_APCI1710_InitChrono
                                (BYTE     b_BoardHandle,
                                 BYTE     b_ModulNbr,
                                 BYTE     b_ChronoMode,
                                 BYTE     b_PCIInputClock,
                                 BYTE     b_TimeUnit,
                                 ULONG    ul_TimeInterval,
                                 PULONG   pul_RealTimeInterval)
```

Parameter:

- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_ChronoMode	Mode des Chronometers. (0 bis 7) Siehe Tabelle 2-2
BYTE	b_PCIInputClock	Auswahl des PCI Bus-Takts - APCI1710_30MHZ: Die Karte benutzt einen PCI Bus Takt von 30 MHz - APCI1710_33MHZ: Die Karte benutzt einen PCI Bus Takt von 33 MHz - APCI1710_40MHz: Die Karte benutzt den 40 MHz Quarz Takt.
BYTE	b_TimeUnit	Einheit der Zeitbasis (0 bis 4) 0: ns 1: µs 2: ms 3: s 4: min.
ULONG	ul_TimeInterval	Wert der Zeitbasis. Siehe Tabelle "Wert des Zeitbasis"

- Ausgabe:

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

Tabelle 3-4: Wert der Zeitbasis

PCI Bus-Takt	*b_TimeUnit*	*ul_TimeInterval* Minimum-Wert	*ul_TimeInterval* Maximum-Wert
APCI1710_30MHz	ns (0)	66	4294967295
	µs (1)	1	143165576
	ms (2)	1	143165
	s (3)	1	143
	mn (4)	1	2
APCI1710_33MHz	ns (0)	60	4294967295
	µs (1)	1	130150240
	ms (2)	1	130150
	s (3)	1	130
	mn (4)	1	2
APCI1710_40MHz	ns (0)	50	4294967295
	µs (1)	1	107374182
	ms (2)	1	107374
	s (3)	1	107
	mn (4)	1	2

Aufgabe:
Konfiguriert den Chronometer-Betriebsmode (*b_ChronoMode*) des ausgewählten Moduls (*b_ModulNbr*). Die Parameter *ul_TimeInterval* und *ul_TimeUnit* legen die Zeitbasis für die Messung fest. *pul_RealTimeInterval* gibt den richtigen Zeitwert zurück.
Diese Funktion soll aufgerufen werden, bevor Sie eine andere Funktion aufrufen, die auf den Chronometer zugreift.
Diese Funktion ermöglicht die Zeitmessung zwischen 2 Events.

Mode 0 und 1 sind für Periodenmessung geeignet.
Mode 2 und 3 sind für Frequenzmessung geeignet.
Mode 4 bis 7 sind für die Zeitmessung zwischen 2 Events geeignet.

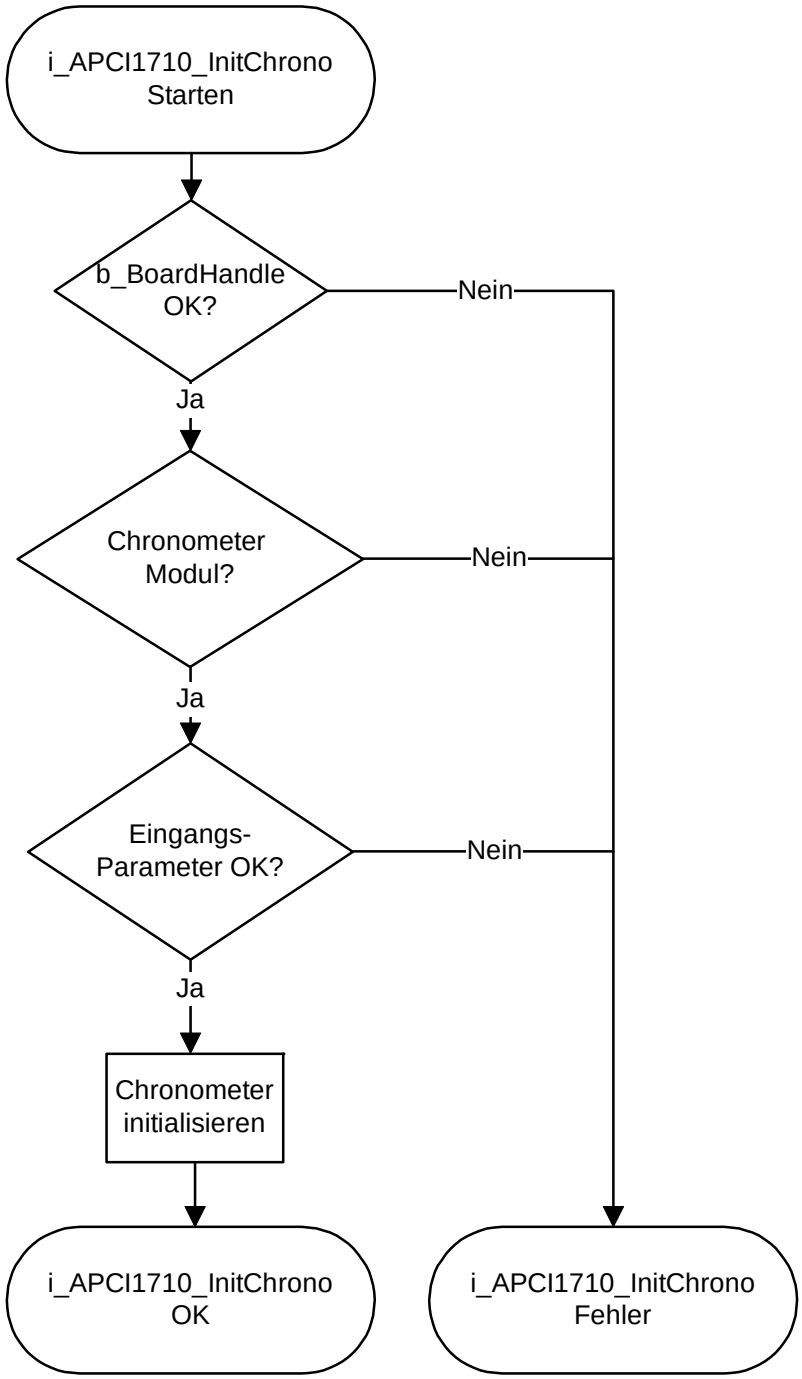
Funktionsaufruf:
ANSI C :

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

i_ReturnValue = i_APCI1710_InitChrono
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     APCI1710_33MHZ,
                                     1,
                                     100,
                                     &ul_RealTimeInterval);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Der ausgewählte Betriebsmode ist falsch.
-5: Der ausgewählte PCI Eingangstakt ist falsch.
-6: Die ausgewählte Zeiteinheit ist falsch.
-7: Die ausgewählte Zeitbasis ist falsch.

Eingabe	
b_BoardHandle	b_PCIInputClock
b_ModulNbr	b_TimeUnit
b_ChronoMode	ul_TimeInterval

Funktion



Ausgabe

pul_RealTimeInterval

<Return Wert>

2) i_APCI1710_EnableChrono (...)

Syntax:
<Return Wert> = i_APCI1710_EnableChrono
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_CycleMode,
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_CycleMode	Auswahl des "Chronos"-Erfassungsmodes - APCI1710_SINGLE: Einfach-Mode. Der Chronometer mißt nur einen Zyklus. Diese Funktion wird für jede Zyklusmessung aufgerufen. - APCI1710_CONTINUOUS: Continuous-Mode. Jedes Start-Signal startet eine neue Messung
BYTE	b_InterruptEnable	Aktiviert oder deaktiviert den Interrupt APCI1710_ENABLE: Interrupt aktiviert APCI1710_DISABLE: Interrupt deaktiviert

- Ausgabe:
Es erfolgt keine Ausgabe.

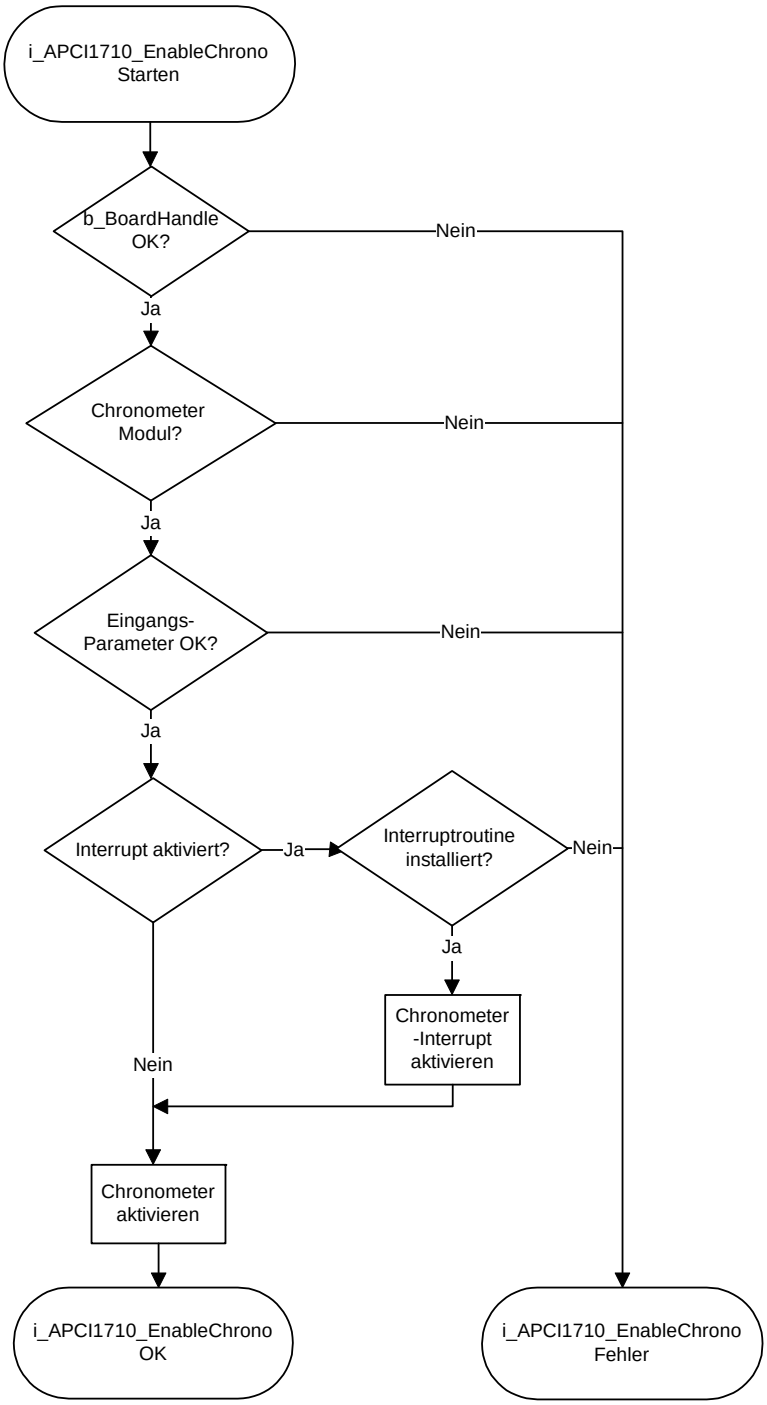
Aufgabe:
Aktiviert den Chronometer des ausgewählten Moduls (*b_ModulNbr*).
Die Funktion "i_APCI1710_InitChrono" soll als erste aufgerufen werden.
Wird der Chronos-Interrupt aktiviert, generiert der Chronometer einen Interrupt nach einem Stop-Signal. Siehe Funktion "i_APCI1710_SetBoardIntRoutineXX" und Tabelle 3-4.
Der *b_CycleMode* Parameter legt fest, ob eine oder mehrere Zyklen gemessen werden.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_EnableChrono
                (b_BoardHandle,
                 0,
                 APCI1710_SINGLE,
                 APCI1710_DISABLE);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".
-5: Der Zyklus des Chronos-Erfassungsmode ist falsch.
-6: Interrupt-Parameter ist falsch.
-7: Interrupt-Funktion nicht initialisiert.
Siehe Funktion "i_APCI1710_SetBoardIntRoutineXX".

Eingabe	
b_BoardHandle	b_CycleMode
b_ModulNbr	b_InterruptEnable
Funktion	



Ausgabe
<Return Wert>

3) i_APCI1710_DisableChrono (...)

Syntax:
<Return Wert> = i_APCI1710_DisableChrono
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

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

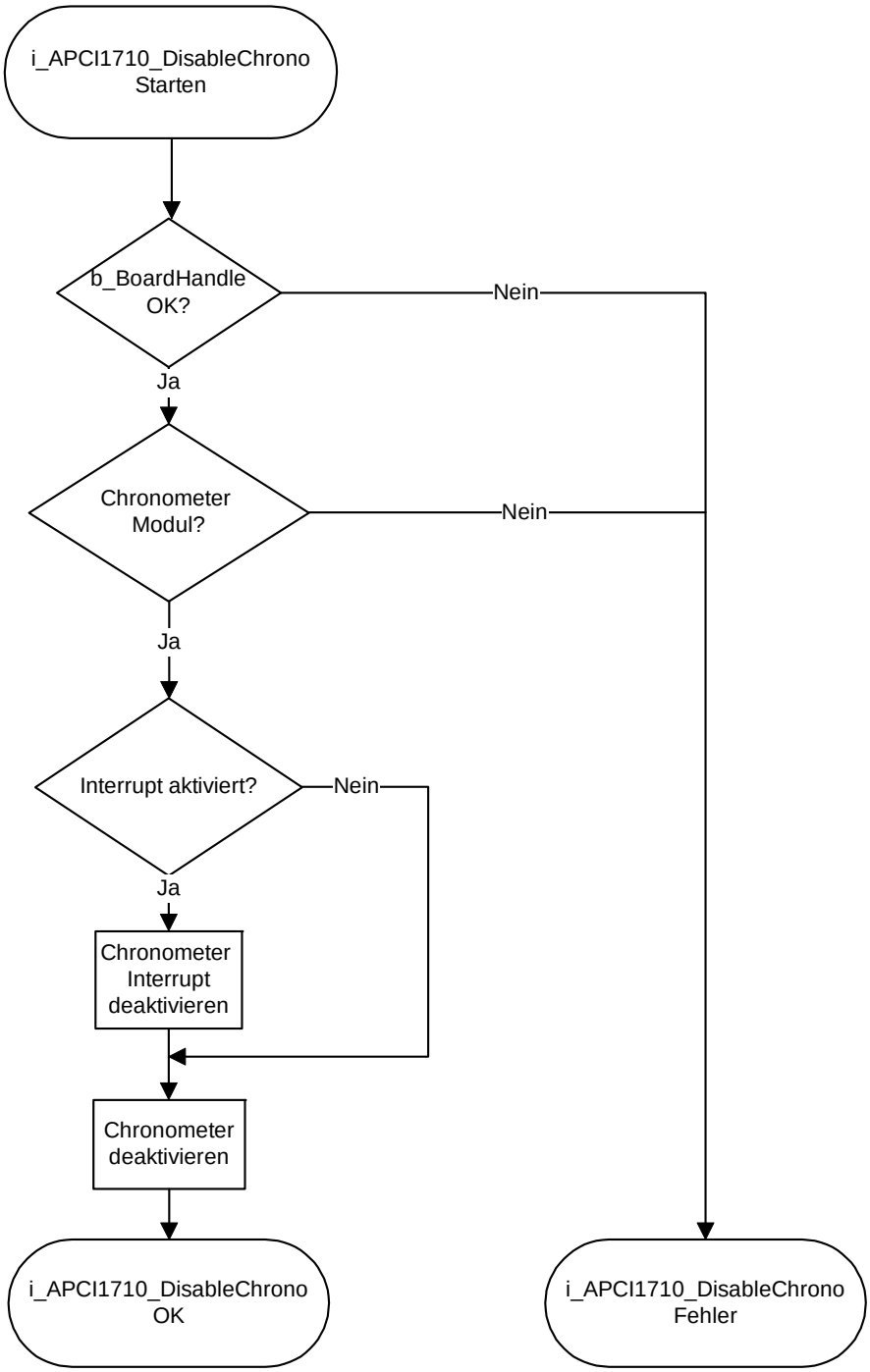
- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Deaktiviert den Chronometer des ausgewählten Moduls (*b_ModulNbr*). Wird der Chronometer nach einem Start-Signal deaktiviert und dann mit der Funktion "i_APCI1710_EnableChrono" wieder gestartet, wird dieses Start-Signal ignoriert, wenn es keinen Stop-Signal erfolgt.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_DisableChrono    (b_BoardHandle,
                                             0);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe
b_BoardHandle b_ModulNbr
Funktion

Ausgabe
<Return Wert>

3.4 Den Chronometer lesen

1) i_APCI1710_GetChronoProgressStatus (...)

Syntax:
<Return Wert> = i_APCI1710_GetChronoProgressStatus
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_ChronoStatus)

Parameter:
- Eingabe:

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

- Ausgabe:

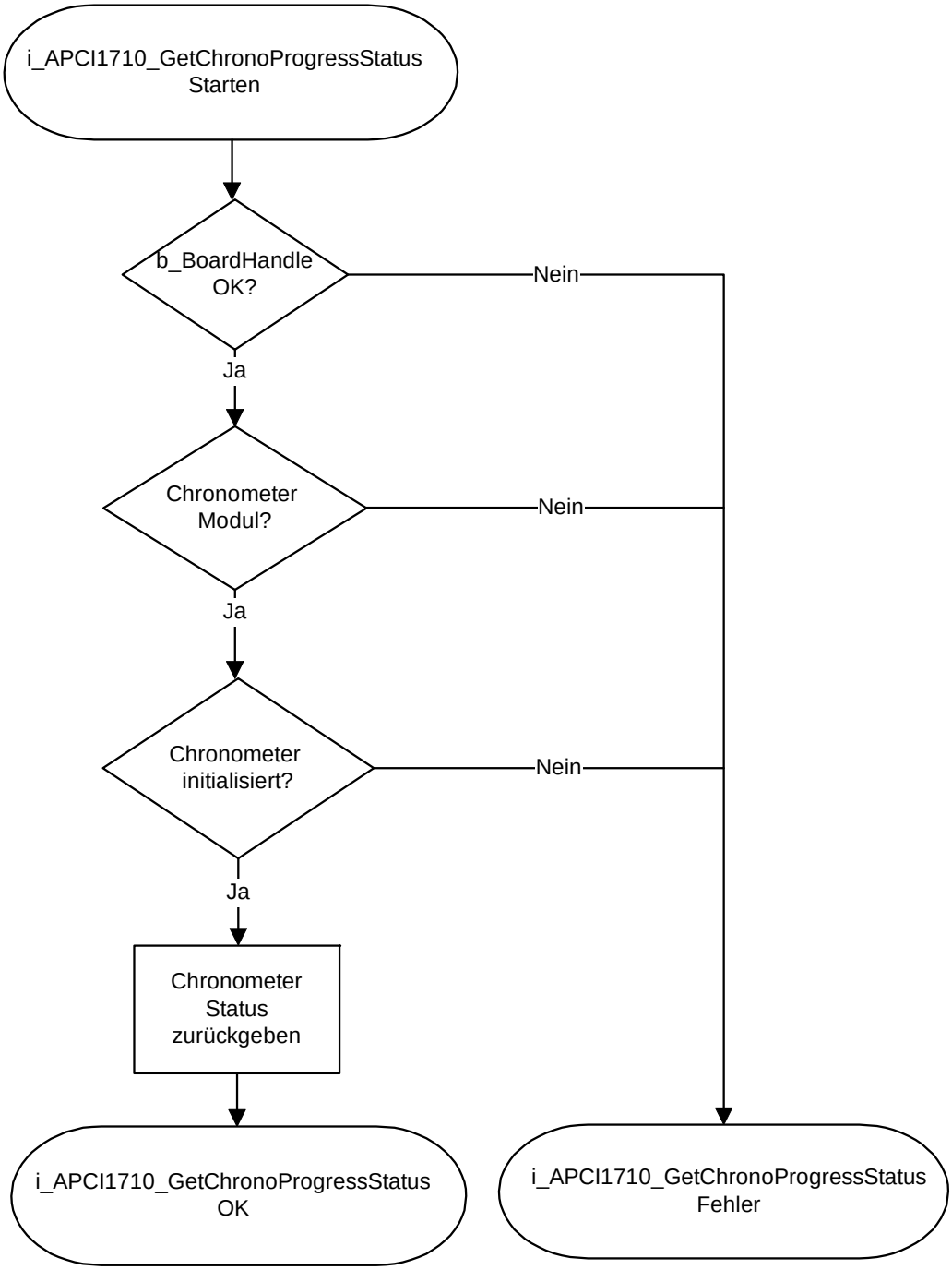
PULONG	pb_ChronoStatus	Rückgabe des Chronometer-Status. 0: Messung nicht gestartet. Kein Start-Signal ist eingetroffen. 1: Messung gestartet. Ein Start-Signal ist eingetroffen. 2: Messung gestoppt. Ein Stop-Signal ist eingetroffen. Die Messung ist beendet. 3: Ein Überlauf ist erfolgt. Bitte die Zeitbasis mit der Funktion "i_APCI1710_InitChrono" ändern

Aufgabe:
Gibt den Chronometer-Status (*pb_ChronoStatus*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_ChronoStatus;

i_ReturnValue = i_APCI1710_GetChronoProgressStatus
                (b_BoardHandle,
                 0,
                 &pb_ChronoStatus);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe
b_BoardHandle b_ModulNbr
Funktion

Ausgabe
pb_ChronoStatus
<Return Wert>

2) i_APCI1710_ReadChronoValue (...)

Syntax:
<Return Wert> = i_APCI1710_ReadChronoValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
UINT ui_TimeOut,
PBYTE pb_ChronoStatus,
PULONG pul_ChronoValue)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
UINT	ui_TimeOut	Auswahl des Timeouts (0 bis 65535) 0: Timeout nicht benutzt. Die Funktion gibt den Chronometer-Status zurück und mißt den Zählerwert, wenn ein Stop-Signal eingetroffen ist. 1 bis 65535: legt das Timeout in ms fest. Die Funktion wird nach einem Timeout oder einem Stop-Signal gestoppt.

- Ausgabe:

PULONG	pb_ChronoStatus	Rückgabe des Chronometer-Status. 0: Messung nicht gestartet. Kein Start-Signal ist eingetroffen. 1: Messung gestartet. Ein Start-Signal ist eingetroffen. 2: Messung gestoppt. Ein Stop-Signal ist eingetroffen. Die Messung ist beendet und *pul_ChronoValue* gibt die Chronometer-Zeit zurück. 3: Ein Überlauf ist erfolgt. Bitte Zeitbasis mit der Funktion "i_APCI1710_InitChrono" ändern 4: Timeout ist erfolgt.
PULONG	pul_ChronoValue	Chronometer-Zeitwert.

Aufgabe:
Rückgabe des Chronometer-Status (*pb_ChronoStatus*) und des Zeitwerts (*pul_ChronoValue*) nach einem Stop-Signal auf das ausgewählte Modul (*b_ModulNbr*). Diese Funktion ist nur verfügbar, wenn Sie die Interrupt-Funktion deaktiviert haben. Siehe Funktion "i_APCI1710_EnableChrono" und Tabelle 3-4.
Der Chronometer-Status kann mit der Funktion "i_APCI1710_GetChronoProgressStatus" getestet werden.
Der durch *pul_ChronoValue* zurückgegebene Wert ist nicht der richtige Zeitwert. Benutzen Sie die "i_APCI1710_ConvertChronoValue" Funktion.
Sonst gilt das folgende Formel, um den richtigen Zeitwert auszurechnen:
Zeitwert = *pul_ChronoValue* x *pul_RealTimeInterval*.

pul_RealTimeInterval ist der rückgegebene Wert von "i_APCI1710_InitChrono". die Zeiteinheit ist durch die Variable *b_TimeUnit* von der Funktion "i_APCI1710_InitChrono".

Funktionsaufruf:
ANSI C :

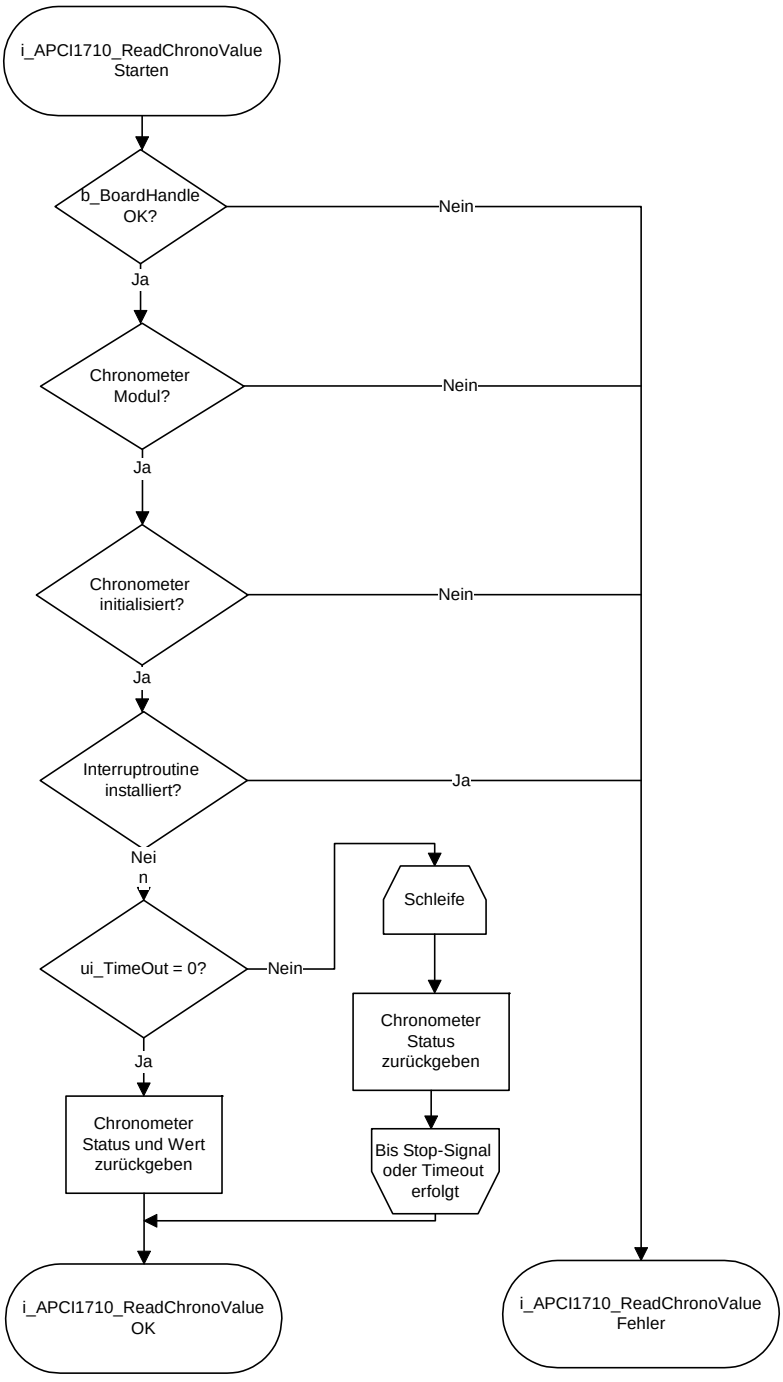
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_ChronoStatus;
unsigned long   ul_ChronoValue;

i_ReturnValue = i_APCI1710_ReadChronoValue
                                (b_BoardHandle,
                                 0,
                                 0,
                                 &pb_ChronoStatus,
                                 &ul_ChronoValue);
```

Return-Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".
-5: Der Timeout Parameter ist falsch (0 bis 65535).
-6: Interruptroutine installiert. Die gemessene Chronometer-Zeit kann nicht direkt gelesen werden.

Eingabe

b_BoardHandle ui_TimeOut
b_ModulNbr

Funktion



Ausgabe

pb_ChronoStatus
pul_ChronoValue

<Return Wert>

3) i_APCI1710_ConvertChronoValue (...)

Syntax:
<Return Wert> = i_APCI1710_ConvertChronoValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
ULONG ul_ChronoValue,
PULONG pul_Hour,
PBYTE pb_Minute,
PBYTE pb_Second,
PUINT pui_MilliSecond,
PUINT pui_MicroSecond,
PUINT pui_NanoSecond)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
ULONG	ul_ChronoValue	Chronometer-Zeitwert. Siehe "i_APCI1710_ReadChronoValue"

- Ausgabe:

PULONG	pul_Hour	Zeitmessung in Stunden
PBYTE	pb_Minute	Zeitmessung in Minuten
PBYTE	pb_Second	Zeitmessung in Sekunden
PUINT	pui_MilliSecond	Zeitmessung in Millisekunden
PUINT	pui_MicroSecond	Zeitmessung in Microsekunden
PUINT	pui_NanoSecond	Zeitmessung in Nanosekunden

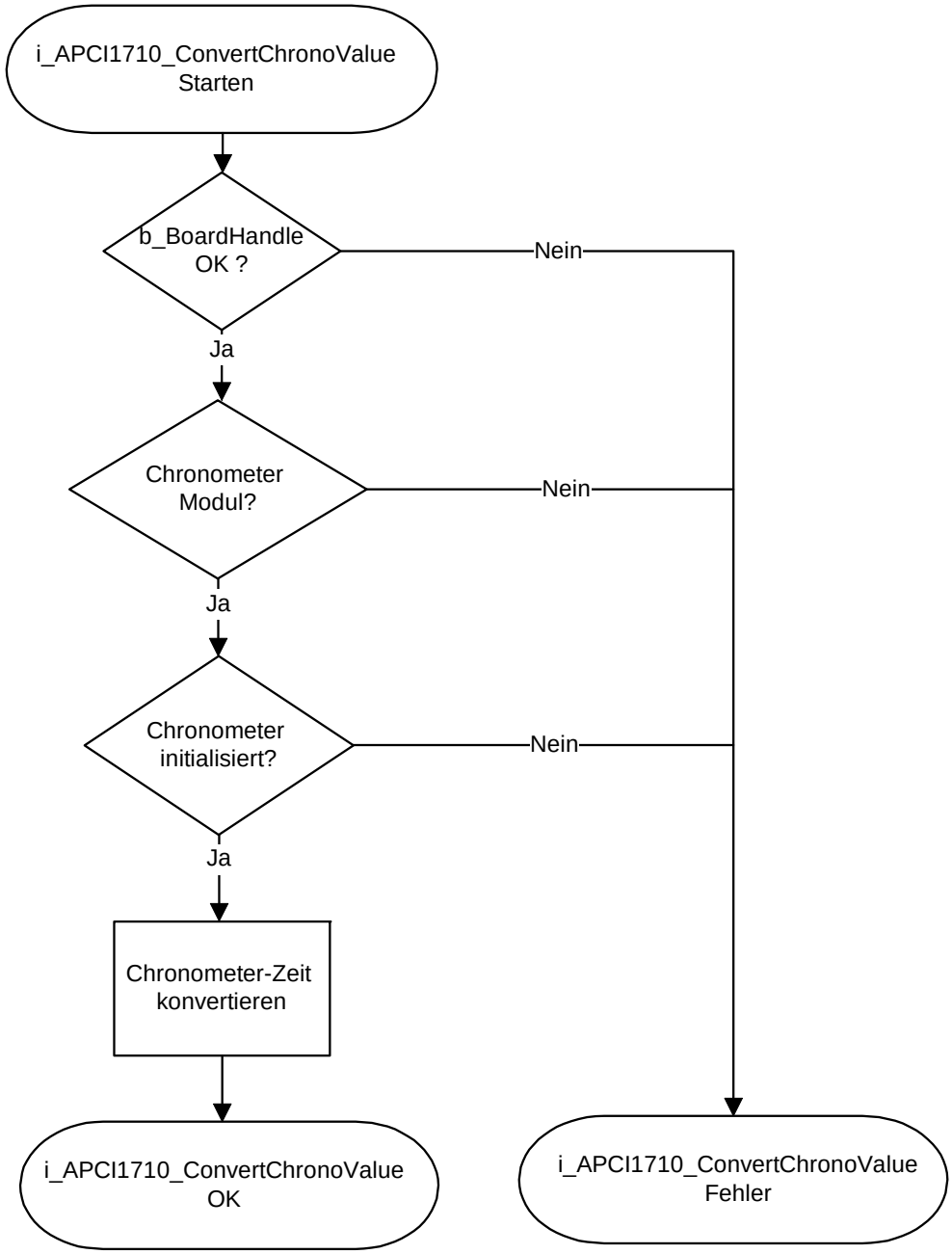
Aufgabe:
Konvertierung der gemessenen Chronometer-Zeit (*ul_ChronoValue*) in h, mn, s, ms, µs und ns.

Funktionsaufruf:
ANSI C :

```
int                i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned int    ui_MilliSecond;
unsigned int    ui_MicroSecond;
unsigned int    ui_NanoSecond;
unsigned char   b_Second;
unsigned char   b_Minute;
i_ReturnValue = i_APCI1710_ConvertChronoValue
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     &b_Minute,
                                     &b_Second,
                                     &ui_MilliSecond,
                                     &ui_MicroSecond,
                                     &ui_NanoSecond);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe	
b_BoardHandle b_ModulNbr	ul_ChronoValue
Funktion	



Ausgabe	
pul_Hour pb_Minute pb_Second	pui_MilliSecond pui_MicroSecond pui_NanoSecond
<Return Wert>	

3.5 Auf einen digitalen Ausgang schreiben

1) i_APCI1710_SetChronoChlOn (...)

Syntax:
<Return Wert> = i_APCI1710_SetChronoChlOn
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_OutputChannel)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_OutputChannel	Auswahl eines digitalen Ausgangs (0 bis 2) 0: Ausgang H 1: Ausgang A 2: Ausgang B

- Ausgabe:
Es erfolgt keine Ausgabe.

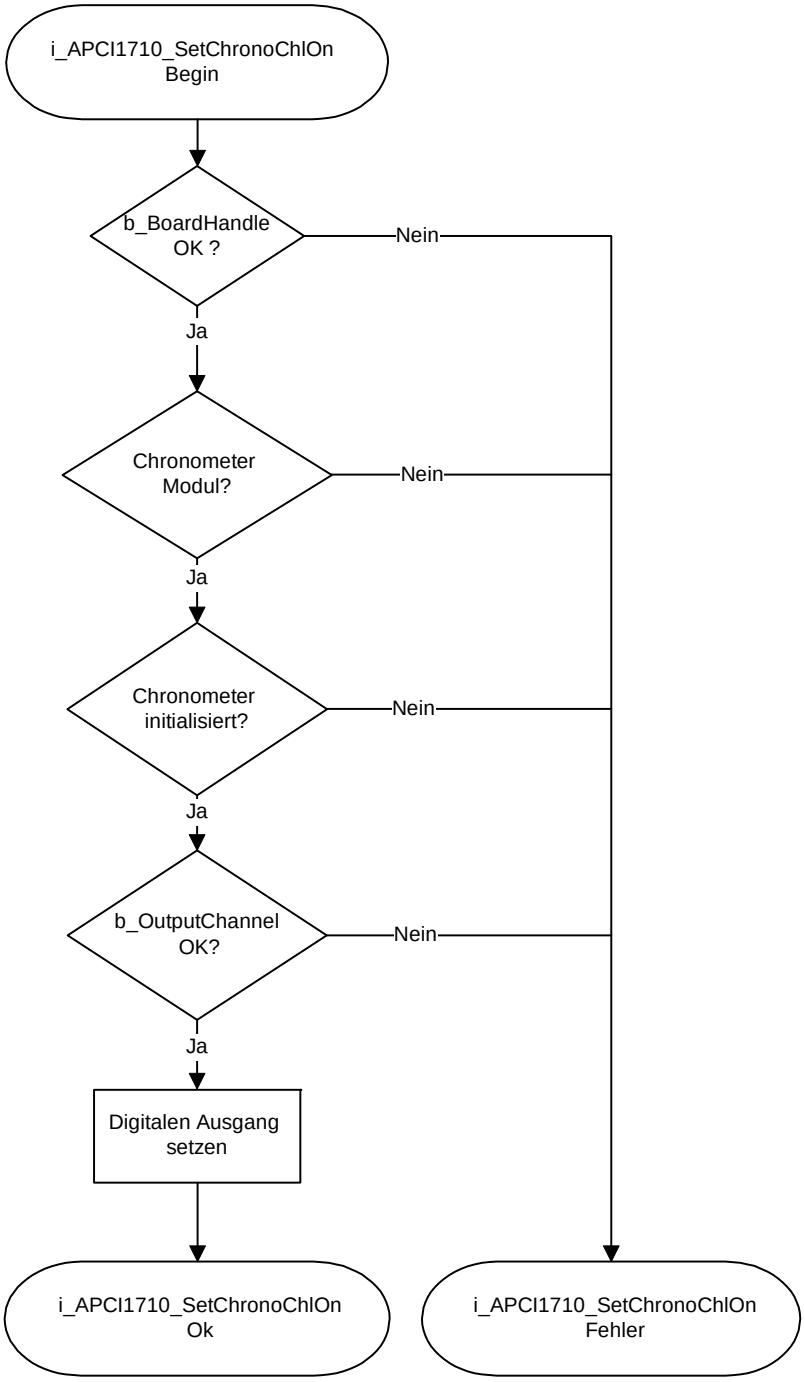
Aufgabe:
Setzt den Ausgang, der von b_OutputChannel durchgegeben wird. Einen Ausgang setzen bedeutet den Ausgang auf High setzen.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetChronoChlOn
                (b_BoardHandle,
                 0,
                 0);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Der ausgewählte digitale Ausgang ist falsch.
-5: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe	
b_BoardHandle b_ModulNbr	b_OutputChannel
Funktion	



Ausgabe
<Return Wert>

2) i_APCI1710_SetChronoChlOff (...)

Syntax:
<Return Wert> = i_APCI1710_SetChronoChlOff
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_OutputChannel)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_OutputChannel	Auswahl eines digitalen Ausgangs (0 bis 2) 0: Ausgang H 1: Ausgang A 2: Ausgang B

- Ausgabe:
Es erfolgt keine Ausgabe.

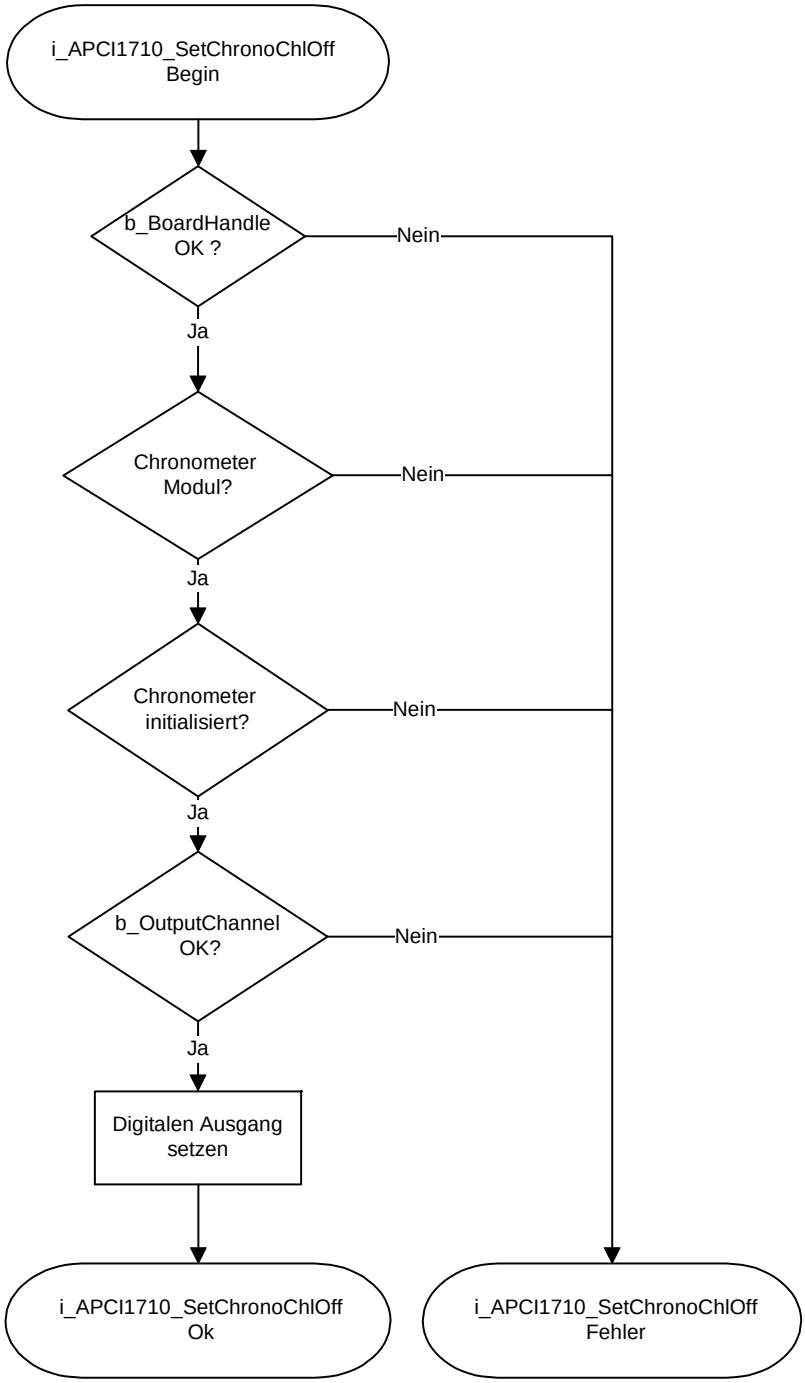
Aufgabe:
Setzt den Ausgang zurück, der von b_OutputChannel durchgegeben wird. Einen Ausgang zurücksetzen bedeutet den Ausgang auf Low setzen.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_APCI1710_SetChronoChlOff (b_BoardHandle,
                                            0,
                                            0);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Der ausgewählte digitale Ausgang ist falsch.
-5: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono"

Eingabe	
b_BoardHandle b_ModulNbr	b_OutputChannel
Funktion	



Ausgabe
<Return Wert>

3.6 Einen digitalen Eingang lesen

1) i_APCI1710_ReadChronoChlValue (...)

Syntax:
<Return Wert> = i_APCI1710_ReadChronoChlValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_InputChannel,
PBYTE pb_ChannelStatus)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_InputChannel	Auswahl eines digitalen Eingangs (0 bis 2) 0: Eingang E 1: Eingang F 2: Eingang G

- Ausgabe:

PBYTE	pb_ChannelStatus	Status des digital Eingangs. 0: Kanal aktiv 1: Kanal inaktiv

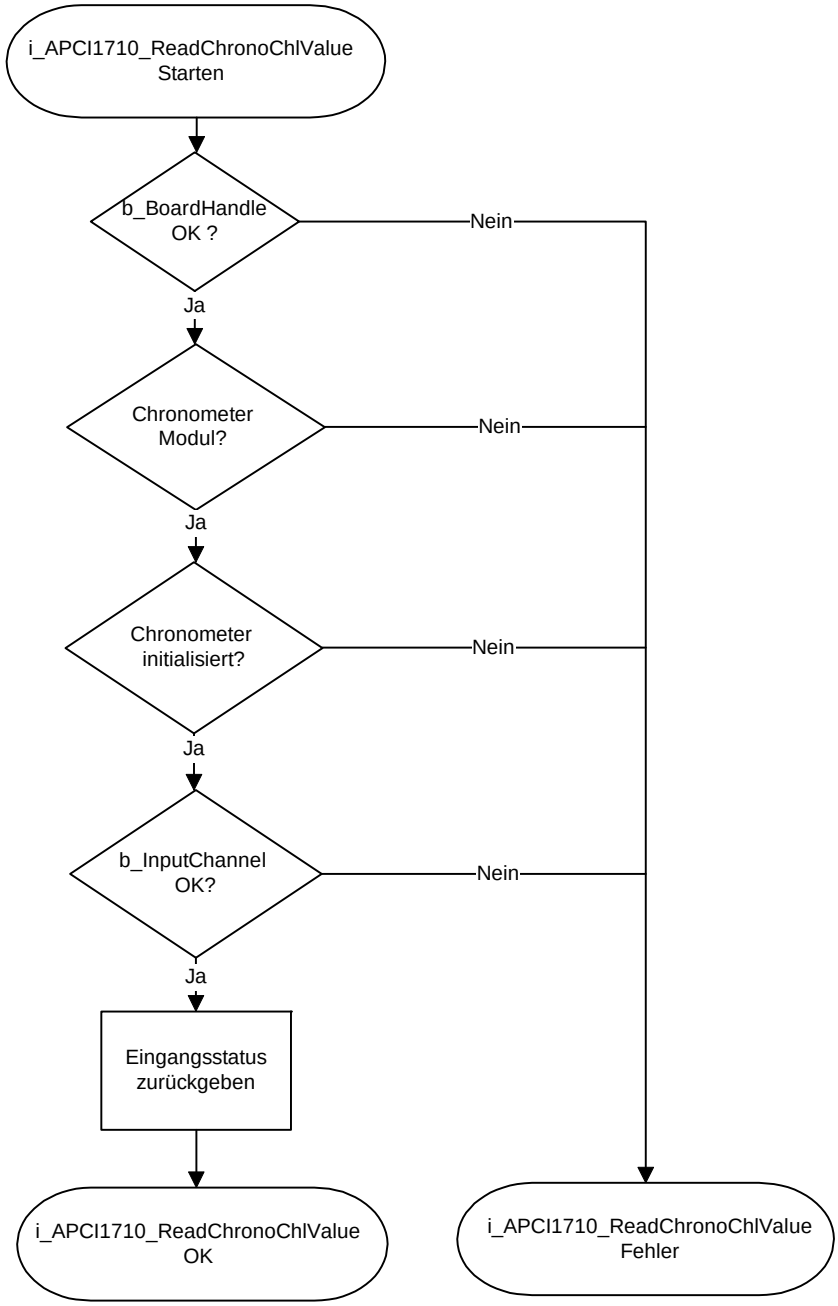
Aufgabe:
Rückgabe des Status des digitalen Eingangs (*b_InputChannel*) in dem ausgewählten Modul (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

```
int                             i_ReturnValue;
unsigned char                   b_BoardHandle;
unsigned char                   b_ChannelStatus;

i_ReturnValue = i_APCI1710_ReadChronoChlValue
                                (b_BoardHandle,
                                 0,
                                 0,
                                 &b_ChannelStatus);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Der ausgewählte digitale Eingang ist falsch.
-5: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe	
b_BoardHandle b_ModulNbr	b_InputChannel
Funktion	



Ausgabe
pb_ChannelStatus
<Return Wert>

2) i_APCI1710_ReadChronoPortValue (...)

Syntax:
<Return Wert> = i_APCI1710_ReadChronoPortValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_PortValue)

Parameter:
- Eingabe:

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

- Ausgabe:

PBYTE	pb_PortValue	Status des digitalen Eingangsports

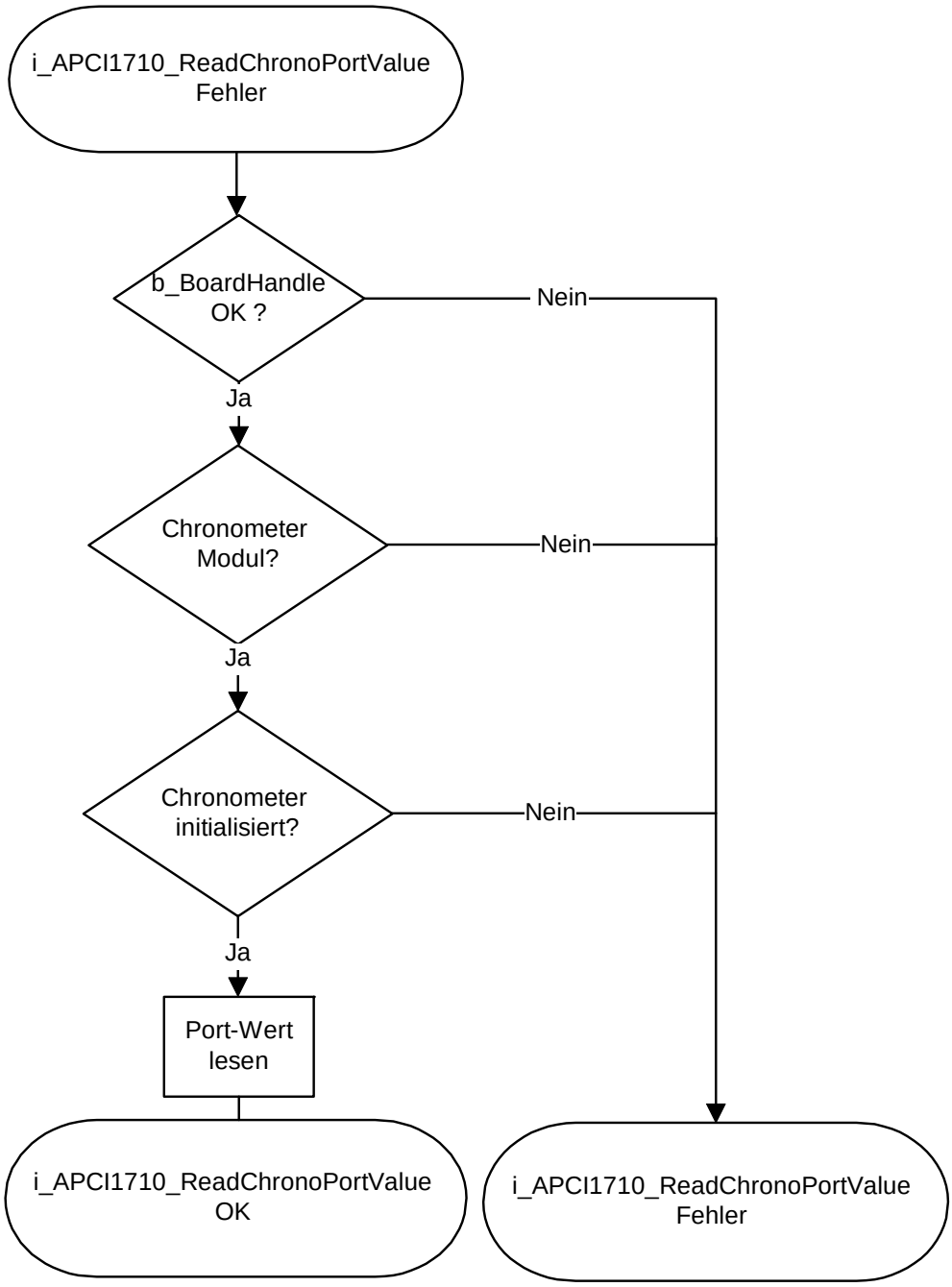
Aufgabe:
Rückgabe des Status des digitalen Eingangsports im ausgewählten Modul (*b_ModulNbr*).

D0	**D1**	**D2**
Eingang E	Eingang F	Eingang G

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
 unsigned char  b_PortValue;

i_ReturnValue = i_APCI1710_ReadChronoPortValue
                                (b_BoardHandle,
                                0,
                                &b_PortValue);
```

Return-Wert
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Die ausgewählte Modulnummer ist falsch.
-3: Das ausgewählte Modul ist kein "Chronometer"-Modul.
-4: Chronometer nicht initialisiert. Siehe Funktion "i_APCI1710_InitChrono".

Eingabe
b_BoardHandle b_ModulNbr
Funktion

Ausgabe
pb_PortValue
<Return Wert>

3.7 Funktionen in 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"

3.7.1 Chronometer lesen

1) i_APCI1710_KRNL_GetChronoProgressStatus (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_GetChronoProgressStatus
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_ChronoStatus)

Parameter:
- Eingabe:

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

- Ausgabe:

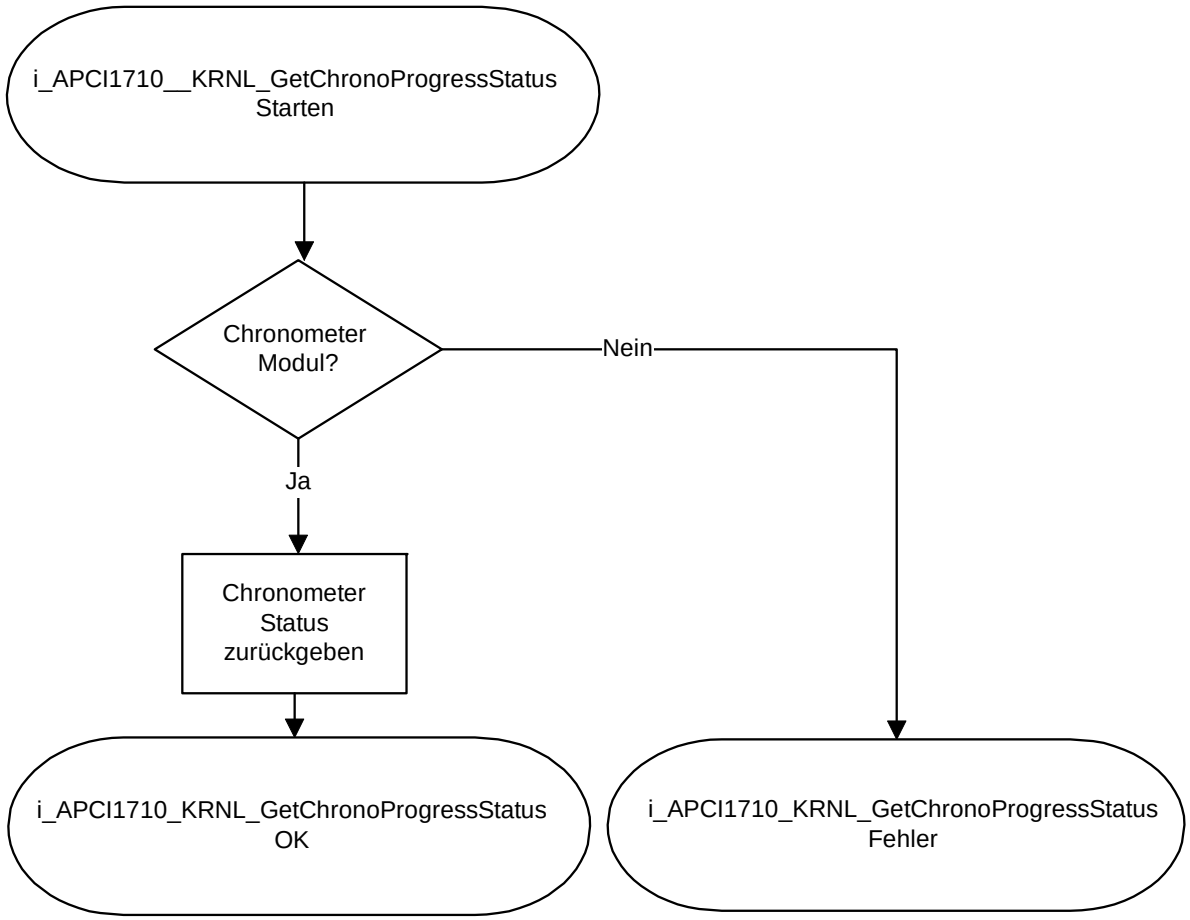
PULONG	pb_ChronoStatus	Rückgabe des Chronometer-Status. 0: Messung nicht gestartet. Kein Start-Signal ist eingetroffen. 1: Messung gestartet. Ein Start-Signal ist eingetroffen 2: Messung gestoppt. Ein Stop-Signal ist eingetroffen. Die Messung ist beendet. 3: Ein Überlauf ist erfolgt. Bitte die Zeitbasis mit der Funktion "i_APCI1710_InitChrono" ändern

Aufgabe:
Gibt den Chronometer-Status (*pb_ChronoStatus*) des ausgewählten Moduls (*b_ModulNbr*) zurück.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_ChronoStatus;

i_ReturnValue = i_APCI1710_KRNL_GetChronoProgressStatus
                (ui_BaseAddress,
                 0,
                 &pb_ChronoStatus);
```

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

Eingabe
ui_BaseAddress b_ModulNbr
Funktion

Ausgabe
pb_ChronoStatus
<Return Wert>

2) i_APCI1710_KRNL_ReadChronoValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ReadChronoValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_ChronoStatus,
PULONG pul_ChronoValue)

Parameter:
- Eingabe:

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

- Ausgabe:

PBYTE	pb_ChronoStatus	Rückgabe des Chronometer-Status. 0: Messung nicht gestartet. Kein Start-Signal ist eingetroffen. 1: Messung gestartet. Ein Start-Signal ist eingetroffen. 2: Messung gestoppt. Ein Stop-Signal ist eingetroffen Die Messung ist beendet und *pul_ChronoValue* gibt die Chronometer-Zeit zurück. 3: Ein Überlauf ist erfolgt. Bitte Zeitbasis mit der Funktion "i_APCI1710_InitChrono" ändern
PULONG	pul_ChronoValue	Chronometer-Zeitwert.

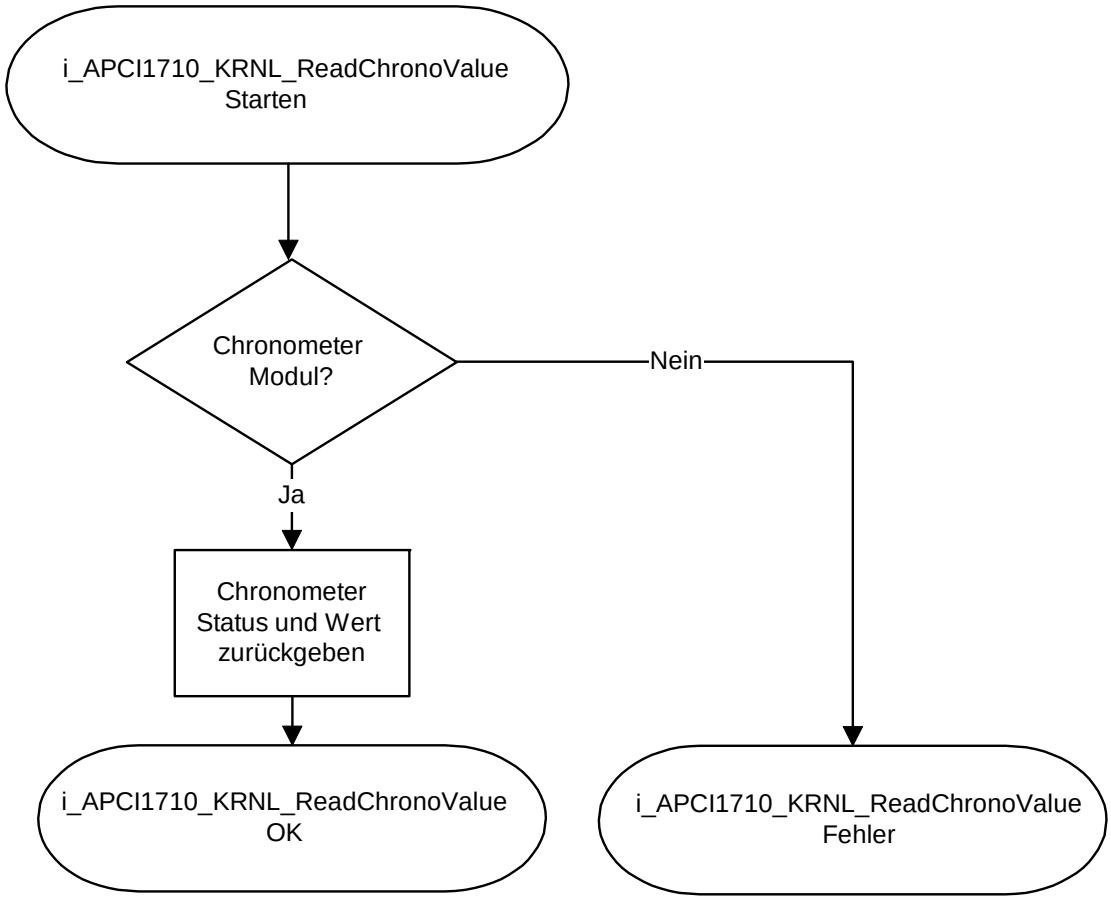
Aufgabe:
Rückgabe des Chronometer-Status (*pb_ChronoStatus*) und des Zeitwerts (*pul_ChronoValue*) nach einem Stop-Signal auf das ausgewählte Modul (*b_ModulNbr*). Diese Funktion ist nur verfügbar, wenn Sie die Interrupt-Funktion deaktiviert haben. Siehe Funktion "i_APCI1710_EnableChrono" und Tabelle 3-4. Der Chronometer-Status kann mit der "i_APCI1710_KRNL_GetChronoProgressStatus" Funktion getestet werden.

Der durch *pul_ChronoValue* zurückgegebene Wert ist nicht der richtige Zeitwert. Benutzen Sie die "i_APCI1710_ConvertChronoValue" Funktion.
Sonst gilt das folgende Formel, um den richtigen Zeitwert auszurechnen:
Zeitwert = *pul_ChronoValue* x *pul_RealTimeInterval*.
pul_RealTimeInterval ist der rückgegebene Wert von "i_APCI1710_InitChrono". die Zeiteinheit ist durch die Variable *b_TimeUnit* von der Funktion "i_APCI1710_InitChrono".

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_ChronoStatus;
unsigned long   ul_ChronoValue;

i_ReturnValue = i_APCI1710_KRNL_ReadChronoValue
                                (ui_BaseAddress,
                                 0,
                                 &pb_ChronoStatus,
                                 &ul_ChronoValue);
```

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

Eingabe
ui_BaseAddress b_ModulNbr
Funktion

Ausgabe
pb_ChronoStatus pul_ChronoValue
<Return Wert>

3.7.2 Auf einen digitalen Ausgang schreiben

3) i_APCI1710_KRNL_SetChronoChlOn (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetChronoChlOn
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_OutputChannel)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710** Karte
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_OutputChannel	Auswahl eines digitalen Ausgangs (0 bis 2) 0: Ausgang H 1: Ausgang A 2: Ausgang B

- Ausgabe:
Es erfolgt keine Ausgabe.

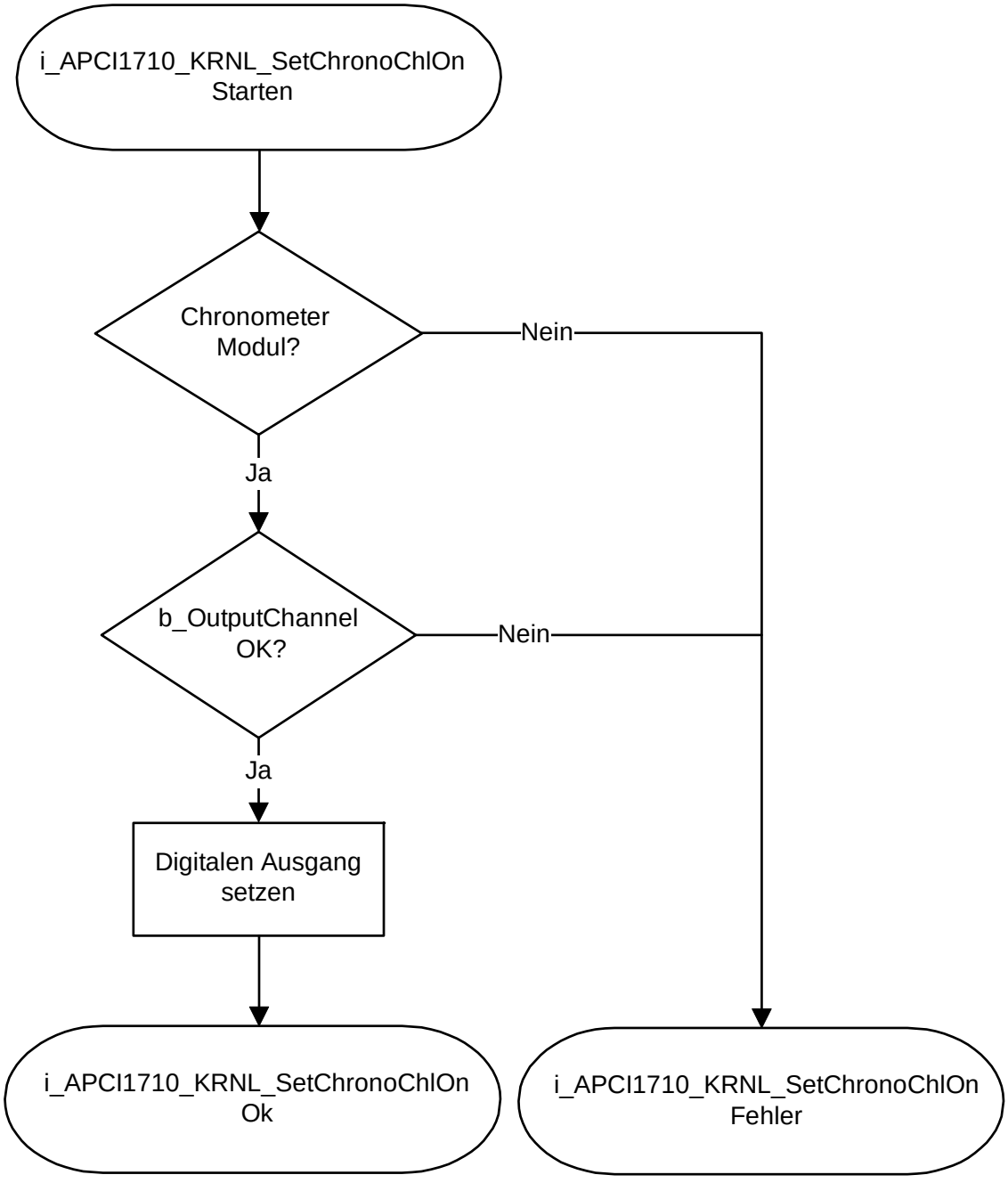
Aufgabe:
Setzt den Ausgang, der von b_OutputChannel durchgegeben wird. Einen Ausgang setzen bedeutet den Ausgang auf High setzen.

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetChronoChlOn
                                    (ui_BaseAddress,
                                     0,
                                     0);
```

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

Eingabe	
ui_BaseAddress b_ModulNbr	pb_OutputChannel
Funktion	



Ausgabe
<Return Wert>

4) i_APCI1710_KRNL_SetChronoChlOff (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetChronoChlOff
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_OutputChannel)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710** Karte
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_OutputChannel	Auswahl eines digitalen Ausgangs (0 bis 2) 0: Ausgang H 1: Ausgang A 2: Ausgang B

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Setzt den Ausgang zurück, der von b_OutputChannel durchgegeben wird. Einen Ausgang zurücksetzen bedeutet den Ausgang auf Low setzen.

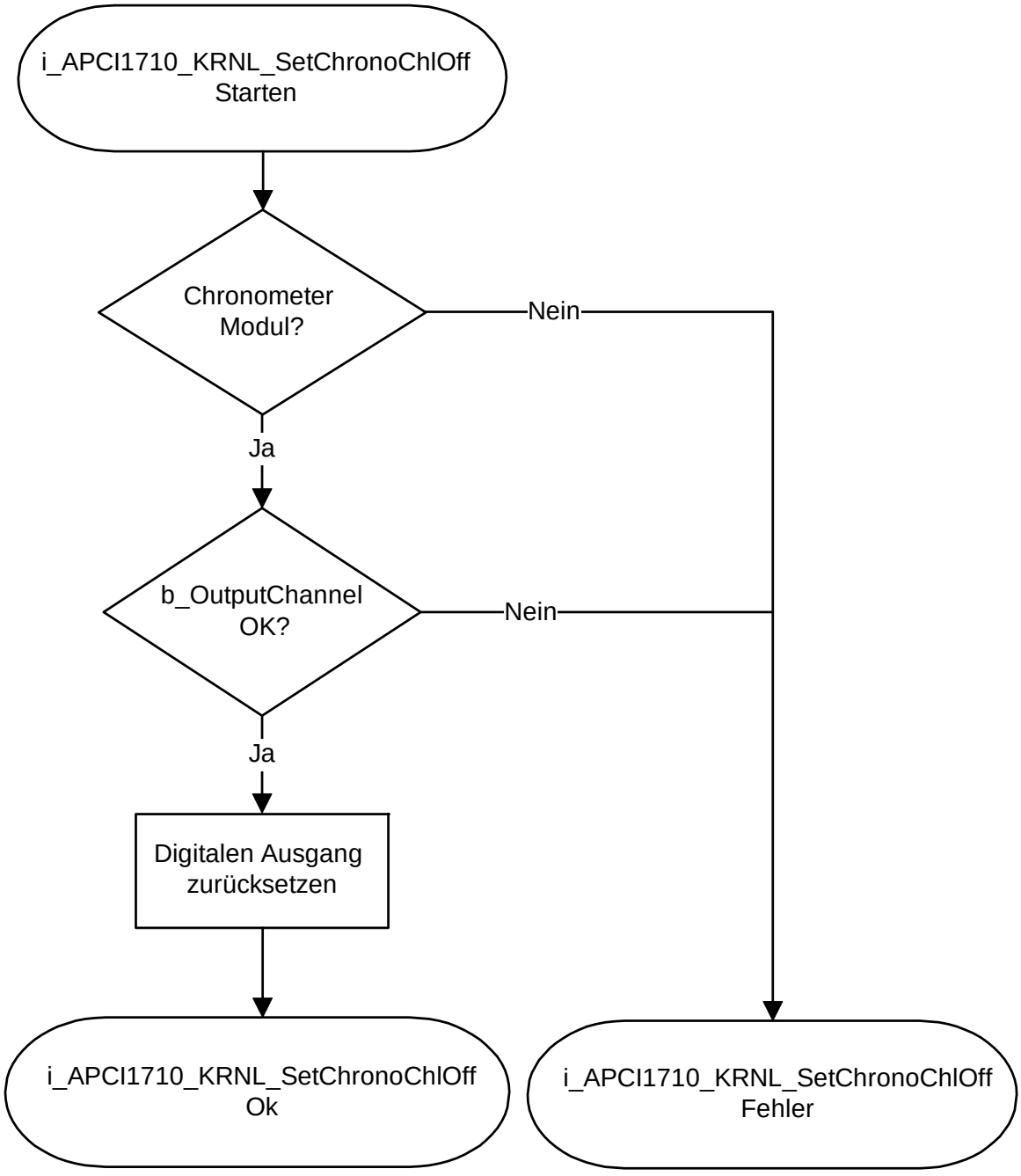
Funktionsaufruf:

ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetChronoChlOff
                                    (ui_BaseAddress,
                                     0,
                                     0);
```

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

Eingabe	
ui_BaseAddress b_ModulNbr	b_OutputChannel
Funktion	



Ausgabe
<Return Wert>

3.7.3 Einen digitalen Eingang lesen

5) i_APCI1710_KRNL_ReadChronoChlValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ReadChronoChlValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
BYTE b_InputChannel,
PBYTE pb_ChannelStatus)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-/CPCI-1710** Karte
BYTE	b_ModulNbr	Nummer des Moduls zu konfigurieren (0 bis 3)
BYTE	b_InputChannel	Auswahl eines digitalen Eingangs (0 bis 2) 0: Eingang E 1: Eingang F 2: Eingang G

- Ausgabe:

PBYTE	pb_ChannelStatus	Status des digital Eingangs. 0: Kanal aktiv 1: Kanal inaktiv

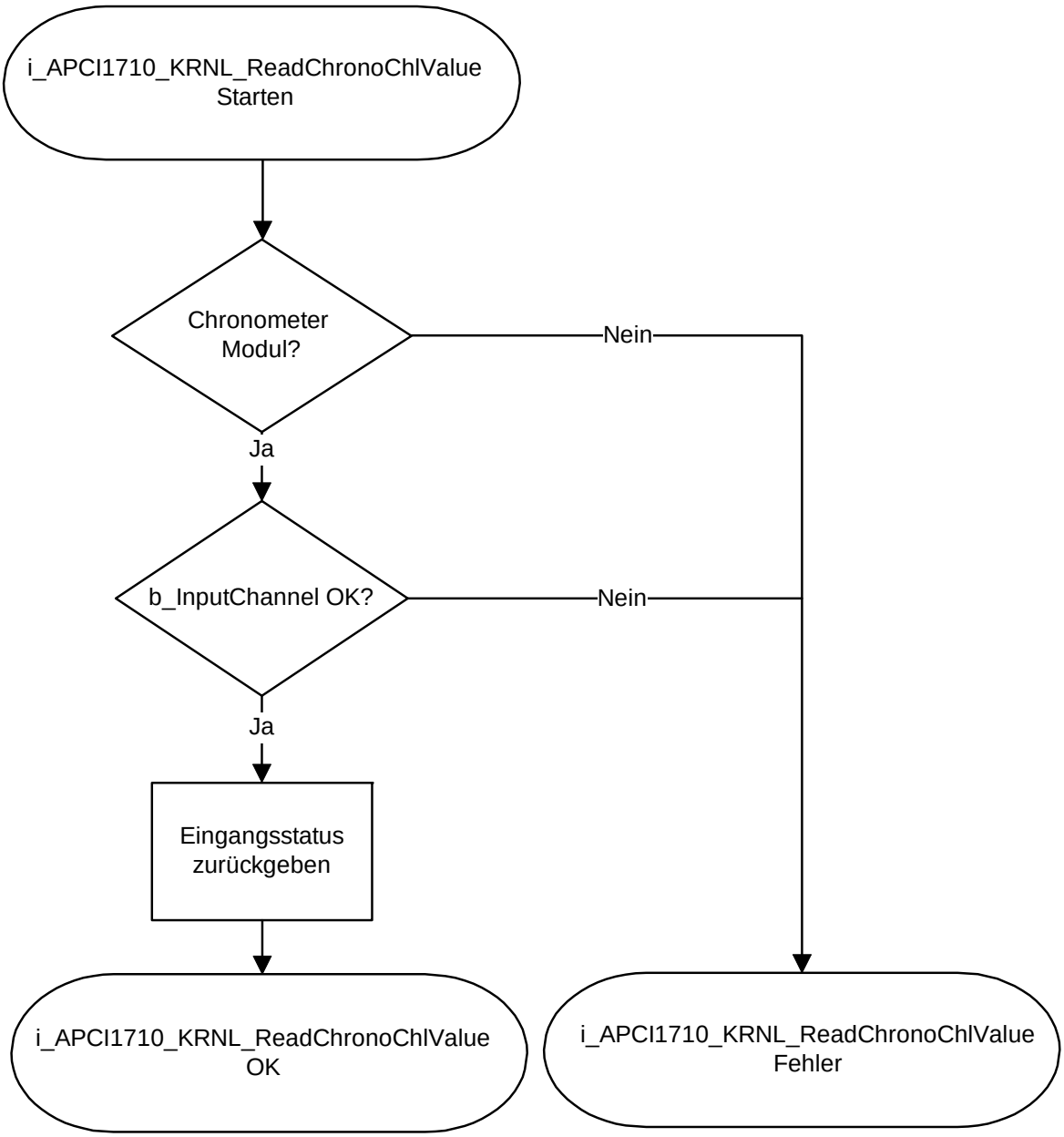
Aufgabe:
Rückgabe des Status des digitalen Eingangs (*b_InputChannel*) in dem ausgewählten Modul (*b_ModulNbr*).

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_ChannelStatus;

i_ReturnValue = i_APCI1710_KRNL_ReadChronoChlValue
                                (ui_BaseAddress,
                                 0,
                                 0,
                                 &b_ChannelStatus);
```

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

Eingabe	
ui_BaseAddress b_ModulNbr	b_InputChannel
Funktion	



Ausgabe
pb_ChannelStatus
<Return Wert>

6) i_APCI1710_KRNL_ReadChronoPortValue (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ReadChronoPortValue
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_PortValue)

Parameter:
- Eingabe:

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

- Ausgabe:

PBYTE	pb_PortValue	Status des digitalen Eingangsports

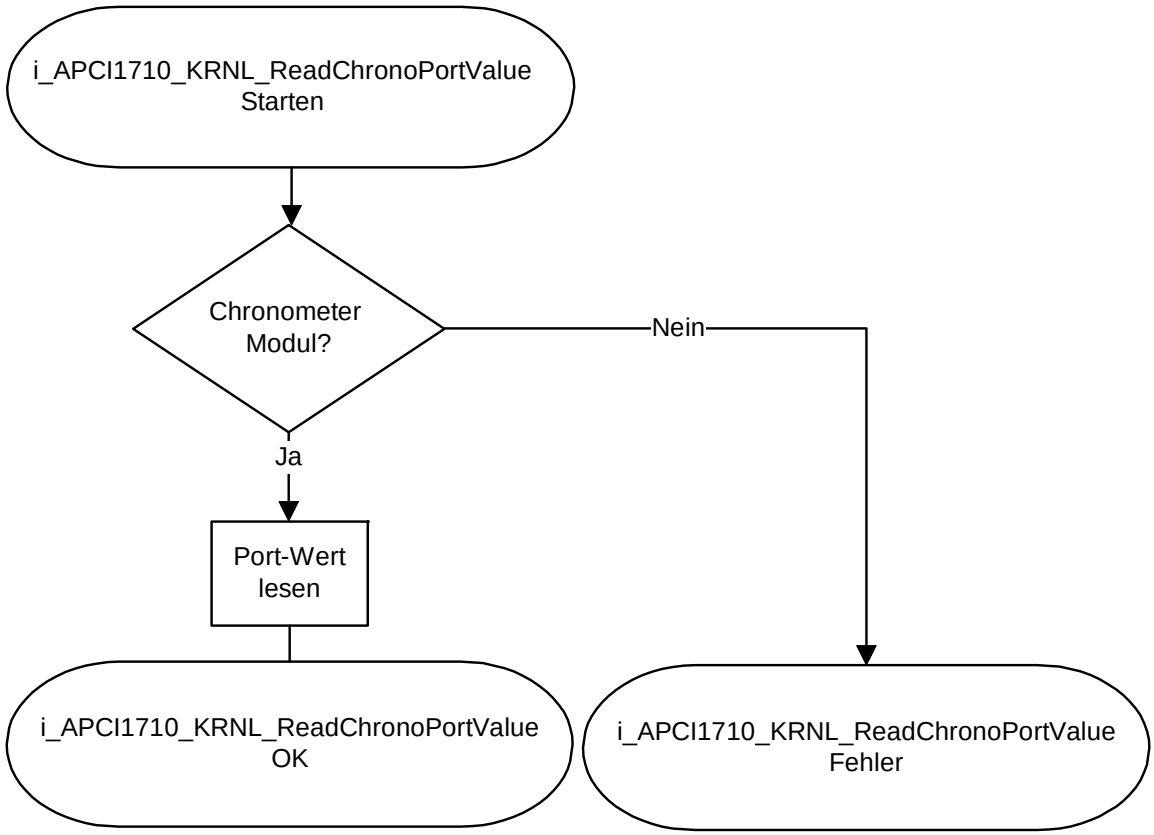
Aufgabe:
Rückgabe des Status des digitalen Eingangsports im ausgewählten Modul (*b_ModulNbr*).

D0	**D1**	**D2**
Eingang E	Eingang F	Eingang G

Funktionsaufruf:
ANSI C :

```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
 unsigned char  b_PortValue;

i_ReturnValue = i_APCI1710_KRNL_ReadChronoPortValue
                                    (ui_BaseAddress,
                                     0,
                                     &b_PortValue);
```

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

Eingabe
ui_BaseAddress b_ModulNbr
Funktion

Ausgabe
pb_PortValue
<Return Wert>