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

SSI
(3-fach Synchron-Serielles Interface)

5. 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	ttl_io.cfg
Digitale E/A	dig_EA_d.pdf	dig_IO_e.pdf	Digital I/O	dig_IO.cfg
Impulszähler	Impulszähler_d.pdf	pulse_counter_e.pdf	Pulse counter	imp_cpt.cfg
ETM	ETM_d.pdf	ETM_e.dpf	Edge time measurement	etm.cfg

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

Die API-Funktionen der Standardsoftware sind ebenfalls identisch.

1.4 Funktionsbeschreibung

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

1.5 Schriftvereinbarung

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

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

2 SYNCHRON-SERIELLES INTERFACE

2.1 Funktionsbeschreibung

Die Funktion SSI ist eine Schnittstelle für absolute Winkelcodierer (Wegmesssysteme).
Sie ermöglicht es, durch eine serielle Datenübertragung eine absolute Information über die Position zu erhalten.

Sie eignet sich besonders für Anwendungen, für die Zuverlässigkeit und Robustheit in industrieller Umgebung erforderlich sind.

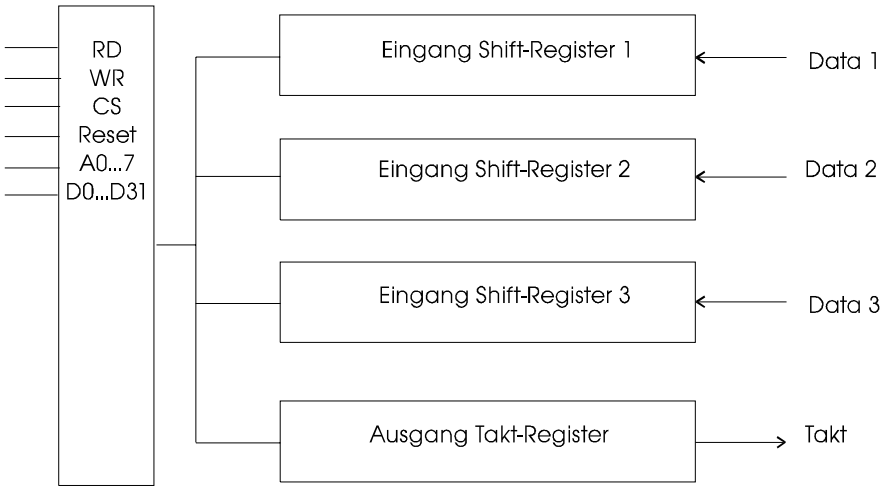
Vorteile im Vergleich zur parallelen Schnittstelle:
- Wesentlich geringerer Verkabelungsaufwand,
- Der Aufwand für Verkabelung und Interfacetechnik ist von der Länge des Datenwortes unabhängig
- Durch synchrone und symmetrische Takt- und Datensignale über paarig verdrilltes Kabel wird die Abschirmung gegen Störeinflüsse erreicht.
- Zur Vermeidung von Erdschleifen wird eine komplette galvanische Trennung durch Optokoppler herangezogen.

Eigenschaften:
- Anschluss von 1 bis zu 3 Gebern pro Modul mit der SSI-Funktion:
- Der Takt ist gemeinsam für alle drei Schnittstellen,
- Die Taktfrequenz ist per Software einstellbar, damit die Übertragung der Leitungslängen angepasst werden kann,
- Die Anzahl der Datenbits ist über Software programmierbar, so dass auch eine Flexibilität in Hinsicht der Auflösung möglich ist.
- GRAY zu BINÄR Wandlung möglich.
- 3 digitale Eingänge und 1 digitaler Ausgang stehen zur Verfügung.
  Diese Ein-/Ausgänge haben keinen Einfluss auf die SSI-Funktion, können aber für eine zusätzliche Funktion benutzt werden.

2.1.1 Blockdiagramm des SSI

Das Interface enthält:
- je drei voneinander unabhängige 32-Bit SHIFT Register, die über den Datenbus ausgelesen werden können,
- Takt- und Pulsgenerator,
- Funktions- und Kontrolllogik.

Abb. 2-1: Blockschaltbild des SSI



2.1.2 Typische Anwendungen

- Erfassung von Wegmesssystemen
- X-,Y-,Z-Steuerungen
- Toleranzmessung

2.2 Benutzte Signale

Die Funktion SSI belegt **6 Eingänge (Kanal B bis G) und 2 Ausgänge** (Kanal A und H) von dem entsprechenden Funktionsmodul der **APCI-/CPCI-1710**.
Auf einer Karte können Sie maximal 12 Absolutdrehgeber anschließen bwz. 3 pro Modul.

Tabelle 2-1: Benutzte Signale

SIGNALE	AM STECKER	POLARITÄT	FUNKTION
TAKT_x	A**x** +/ -	Diff.	TAKT Ausgang Signal für die SSI Geber
DATA1_x	B**x** +/ -	Diff. / OPT.24Vdiff.	DATEN Eingang für den ersten Geber
DATA2_x	C**x + / -**	Diff. / OPT. 24Vdiff.	DATEN Eingang für den zweiten Geber
DATA3_x	D**x + / -**	Diff. / OPT. 24Vdiff.	DATEN Eingang für den dritten Geber
Eingang1_x	E**x** +/-	24V / OPT. 5V	Digitaler Eingang
Eingang2_x	F**x +/-**	24V / OPT. 5V	Digitaler Eingang
Eingang3_x	G**x** +/-	24V / OPT. 5V	Digitaler Eingang
Ausgang_x	H**x** +/-	24V / OPT. 5V TTL	Digitaler Ausgang

x: Nummer des Funktionsmoduls.

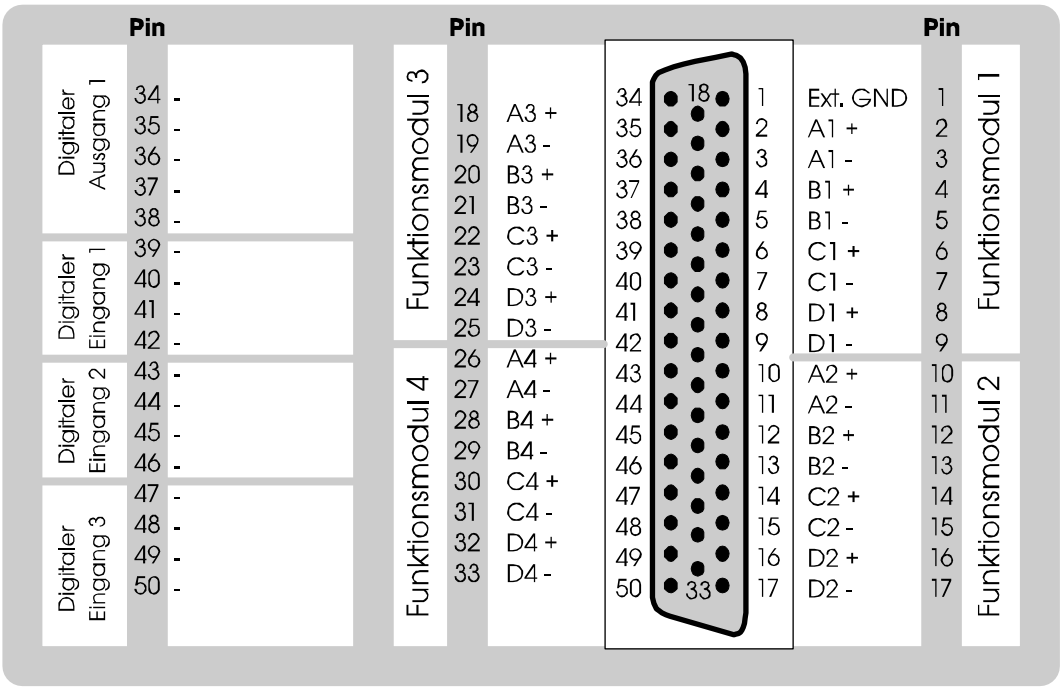
2.3 Steckerbelegung für alle Module mit SSI

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 "SSI" ist auf allen Funktionsmodulen implementiert.

Abb. 2-2: Pinbelegung des 50-pol. SUB-D Steckers X1



-: Nicht belegt

2.4 Anschlussbeispiel

Der Drehgeber CRE 58 von TWK ist an Funktionsmodul 1 der **APCI-/CPCI-1710** angeschlossen. Erste Schnittstelle.

Abb. 2-3: Anschluss an einen Drehgeber TWK CRE 58

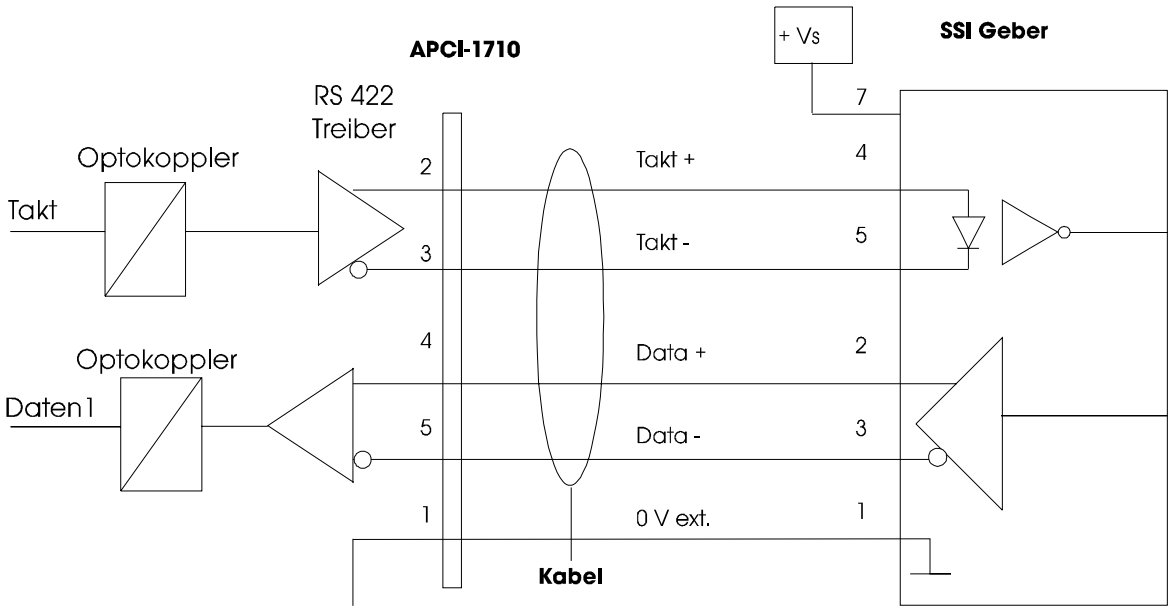


Abb. 2-4: Schaltungsprinzip der Eingänge

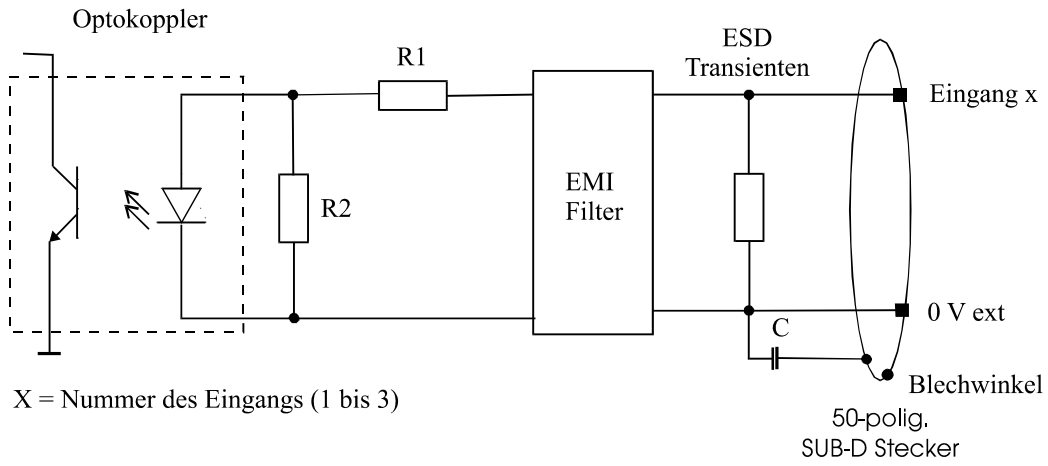
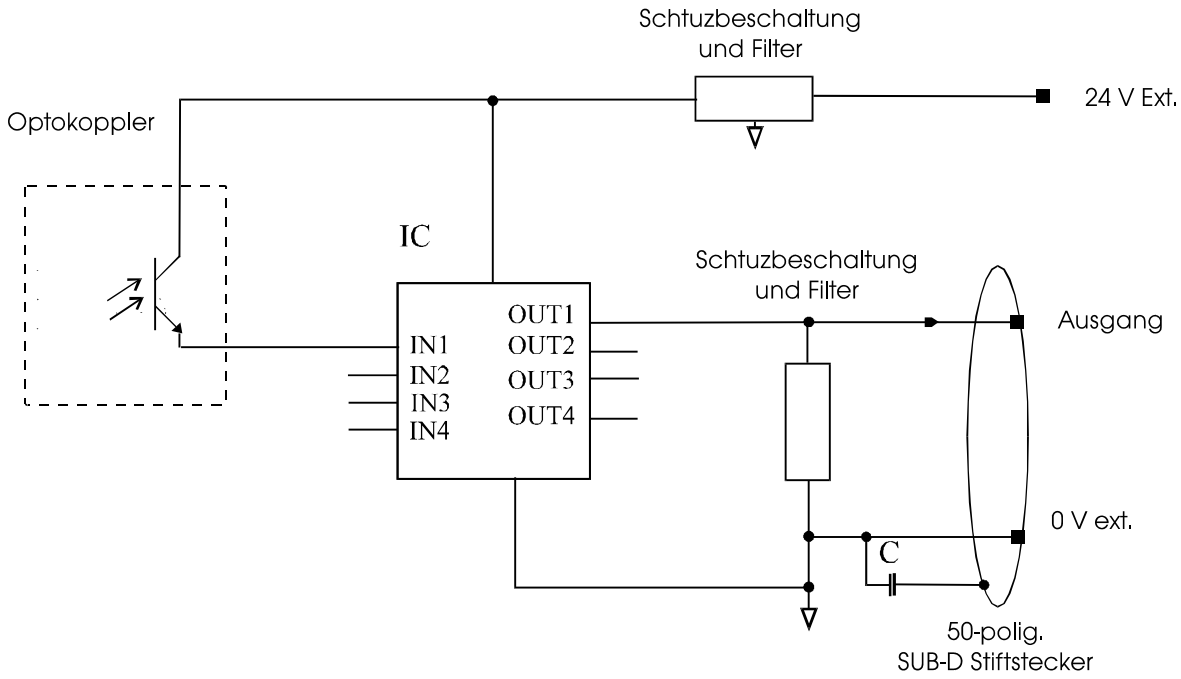


Abb. 2-5: Schaltungsprinzip der Ausgänge



2.5 E/A-Belegung der SSI-Schnittstelle

Tabelle 2-2: E/A-Belegung der SSI-Schnittstelle

	IORD			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHB	MIDLOWB	LOWBYTE
BASEx + 0	-	-	-	STATUS-REG
BASEx + 4	SHIFT1.3	SHIFT1.2	SHIFT1.1	SHIFT1.0
BASEx + 8	SHIFT2.3	SHIFT2.2	SHIFT2.1	SHIFT2.0
BASEx + 12	SHIFT3.3	SHIFT3.2	SHIFT3.1	SHIFT3.0
BASEx + 16	-	-	-	-
.....	-	-	-	-
BASEx + 60	FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

	IOWR			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE**x** + 0	-	-	FRQHIGH-REG	FRQLOW-REG
BASE**x** + 4	-	-	-	COUNT-REG
BASE**x** + 8	-	-	-	START-REG
BASE**x** + 12	-	-	-	CONTROL-REG
BASE**x** + 16	-	-	-	OUTPUT-REG
.....	-	-	-	-
BASE**x** + 60	-	-	-	-

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

Das SSI belegt 5 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

Allgemeines

Die im Winkelcodierer vorliegende parallele, absolute Winkelinformation wird durch einen internen Parallel/seriell-Wandler (Schieberegister) in eine serielle Information umgeformt. Sie wird synchron zu dem von der APCI-/CPCI-1710 gelieferten Takt an die Empfangselektronik übertragen.

Die synchrone Übertragung des Datenwortes wird mittels eines Taktsignals eingeleitet und gesteuert. Die Länge (bzw. Variable) der Taktsequenz wird durch ein internes 8-Bit Register (COUNT-REG) in der **APCI-/CPCI-1710** festgelegt, so daß die Länge des zu übertragenden Datenwortes beliebig von 0 bis 32 Bit änderbar ist. Zum Beispiel ein Winkelcodierer mit **Interface profile SSI-25 Bit** benötigt 26 Takte, um ausgelesen zu werden.

Die Taktfrequenz bestimmt die Geschwindigkeit der Datenübertragung. Diese Frequenz wird über ein internes 16-Bit Register (FRQ-REG) bestimmt.

Übertragungsprotokoll

Die nachfolgend genannten Logikpegel beziehen sich auf das Takt + bzw. Data + Signal. Im Bereitschaft- bzw. Ruhestand des SSI sind sowohl Takt- als auch Datenleitung (Clock +, Data +) Log 1. Die Empfangselektronik leitet die Datenübertragung durch den Wechsel des Taktsignals von Log 1 nach Log 0 ein. Mit dieser Änderung wird im Winkelcodierer ein retriggerbares Monoflop gesetzt, dessen Ausgang wiederum ein Schieberegister von parallel auf seriell umschaltet, wobei die im Gray-code parallel vorliegenden Daten gespeichert werden.

Mit dem nächsten Wechsel des Taktes von Log 0 nach Log 1 wird das höchstwertige Bit der Winkelinformation an den Datenausgang des Winkelcodierers gelegt. Jede weitere positive Flanke liefert dann das nächste darunterliegende Bit bis zum niederwertigsten Bit an den Ausgang. Gleichzeitig wird mit jeder negativen Flanke des Taktes das Monoflop retriggert.

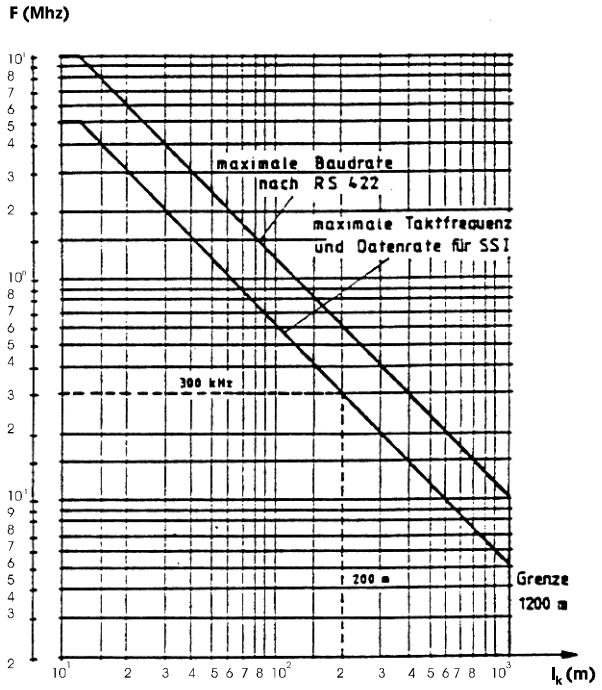
Die Monoflopzeit (z.B. 20 µs) bestimmt die Pause zwischen 2 Übertragungen und die minimale Taktfrequenz.

Maximale Datenrate

Vorbedingungen

Bedingt durch die RS485-Norm für die verwendeten Treiber, Empfänger und das Übertragungprotokoll ist die maximal erreichbare Datenrate (Taktfrequenz) vorgegeben. Die maximale Taktfrequenz beträgt die Hälfte der in der Norm genannten Werte für die Baudrate, und zwar: 5MHz.

Abb. 2-6: Maximale Datenrate



Übertragungsstrecke

Der prinzipielle Aufbau einer SSI-Übertragungsstrecke besteht aus Winkelcodierer, Übertragungskabel, und Taktsequenz- bzw. Empfangselektronik. Alle Einheiten sind naturgemäß mit einer Verzögerung (Laufzeit) der Signale behaftet. Die gesamte Verzögerungszeit führt nun dazu, daß die Daten auf der Empfangsseite zwar synchron zum Takt der Taktsequenz, aber um eben diese Laufzeit verzögert eintreffen.

Diese Verzögerungszeit ändert sich mit der Länge der Übertragungsstrecke, so daß die Taktrate der Strecke angepaßt werden muß.

Verzögerungszeiten

Die Gesamtverzögerung berechnet sich wie folgt:

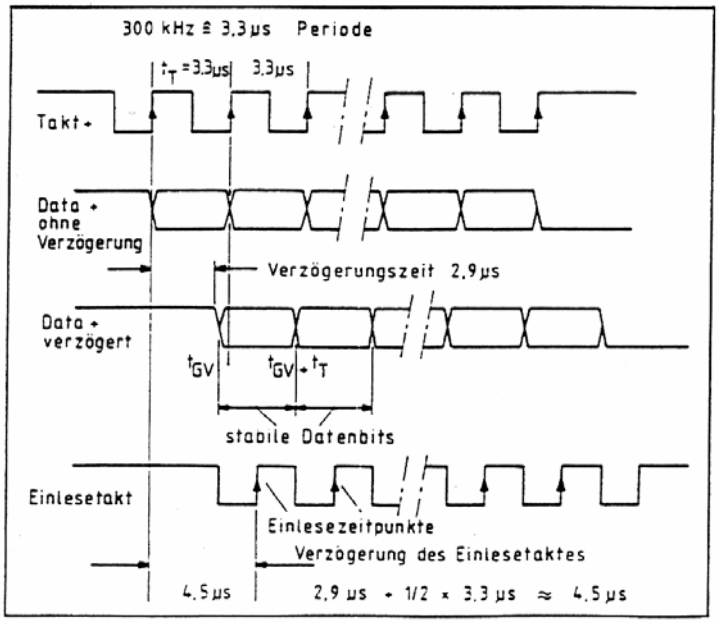
- Tgv = Tc + 2 x Tk + Te
- Tgv: Gesamtverzögerrungszeit,
- Tc: Verzögerungszeit durch die Elektronik des Winkelcodierers, maximal 150 ns
- Tk: Kabelverzögerung; Sie hängt von der Länge und von dem verwendeten Kabel ab. Bei Verwendung eines Kabels Typ LIYCY-OB mit 0,25 mm^2 Querschnitt beträgt die spezifische Laufzeit ca. 6,5 ns / m.
- Te: Verzögerungszeit der Empfangselektronik, max. 150 ns.

Beispiel: für eine Kabellänge von 200 m beträgt die Verzögerung:

Tgv(ns) = 300 + (2 x 6,5 x 200) = 2900 ns

Damit kann nur eine Taktfrequenz von 300 kHz bei 200 m Kabel gewählt werden.

Abb. 2-7: Verzögerungszeiten



2.6.2 FRQ-REGISTER (Base +0)

16-Bit Register zum Festlegung der Taktfrequenz. Dieses Register kann nur beschrieben werden.

Eingangsfrequenz = PCI-Bus Frequenz (zwischen 0 und 33 MHz).
Entnehmen Sie die Frequenz aus dem Handbuch Ihres Rechners.

Enthält dieses Register den Wert 0, wird diese Frequenz durch 2 dividiert,

Enthält das Register den Wert 50, wird diese Frequenz durch 100 dividiert.

Die Faktoren sind 2 bis 131070 in Schritten von 2, wobei eigentlich Frequenzen über 1MHz nicht Sinnvoll sind.

2.6.3 COUNTS-REGISTER (Base +4)

8-Bit-Register zur Festlegung der Bit-Anzahl für das SSI. Dieses Register kann nur beschrieben werden.

Enthält das Register den Wert n = 25, werden 26 Takte generiert;

Enthält das Register den Wert n= 32, werden 33 Takte generiert.

Übertragungsbeispiel für ein Winkelcodierer mit 18-Bit

Winkelcodierer mit 1024 Schritten / Umdrehungen (10 Bits im Monotour-Teil) und 256 Umdrehungen (8 Bits im Multitour-Teil).

Das Übertragungsprotokoll ist in der Standardausführung für ein Datenwort von 25 Bit ausgelegt. Davon sind 12 Bits für die Umdrehungen und 13 Bits für die Auflösung (Schritte/Umdrehungen). Da die Übertragung immer mit dem Multitour-Bit 12 beginnt, das Multitourteil in diesem Beispiel aber nur für 8 Bit ausdgelegt ist, werden zunächst 4 Leerstellen mit Log 0 übertragen und dann die bestückten 8 Bits des Multitourteils. Hierauf folgen die Bit des Monotourteils beginnend mit S10 bis S1. Die nicht bestückten 3 Bits werden hier auch mit Log 0 übertragen.

Tabelle 2-3: Übertragung für ein Winkelcodierer mit 18-Bit

Zahl der U.	Bit / U.	Multitour-Bit												Monotour-Bit													Schritte U.	Bit/U.
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25		
4096	12	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	8192	13
2048	11	0	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	0	4096	12
1024	10	0	0	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	0	0	2048	11
512	9	0	0	0	M9	M8	M7	M6	M5	M4	M3	M2	M1	**S10**	**S9**	**S8**	**S7**	**S6**	**S5**	**S4**	**S3**	**S2**	**S1**	**0**	**0**	**0**	**1024**	**10**
256	**8**	**0**	**0**	**0**	**0**	**M8**	**M7**	**M6**	**M5**	**M4**	**M3**	**M2**	**M1**	S9	S8	S7	S6	S5	S4	S3	S2	S1	0	0	0	0	512	9
128	7	0	0	0	0	0	M7	M6	M5	M4	M3	M2	M1	S8	S7	S6	S5	S4	S3	S2	S1	0	0	0	0	0	256	8
64	6	0	0	0	0	0	0	M6	M5	M4	M3	M2	M1	S7	S6	S5	S4	S3	S2	S1	0	0	0	0	0	0	128	7
32	5	0	0	0	0	0	0	0	M5	M4	M3	M2	M1	S6	S5	S4	S3	S2	S1	0	0	0	0	0	0	0	64	6
16	4	0	0	0	0	0	0	0	0	M4	M3	M2	M1	S5	S4	S3	S2	S1	0	0	0	0	0	0	0	0	32	5
8	3	0	0	0	0	0	0	0	0	0	M3	M2	M1	S4	S3	S2	S1	0	0	0	0	0	0	0	0	0	16	4
4	2	0	0	0	0	0	0	0	0	0	0	M2	M1	S3	S2	S1	0	0	0	0	0	0	0	0	0	0	8	3
2	1	0	0	0	0	0	0	0	0	0	0	0	M1	S2	S1	0	0	0	0	0	0	0	0	0	0	0	4	2

2.6.4 CONTROL-REGISTER (Base +12)

8-Bit Register, das die Konvertierung von GRAY-Code in BINÄR-Code ermöglicht.

Dieses Register kann nur beschrieben werden.

Die Freigabe wird über ein Bit pro SSI Modul betrieben. Sind diese 3 Bits nach Reset auf "0"gesetzt, wird keine Konvertierung der Daten vorgenommen.
Wird eines der 3 Bits auf "1" gesetzt, wird im entsprechenden SSI Modul eine Konvertierung von GRAY auf BINÄR Code vorgenommen.

2.6.5 START-REGISTER (Base +8)

Ein Schreiben auf die Basisadresse + 8 startet einen Zyklus.

Die Daten werden seriell übertragen.

2.6.6 STATUS-REGISTER (Base +0)

Ein Lesen auf die Basisadresse + 0 gibt Information über den aktuellen seriellen Datentransfer. Die Zustände der digitalen Eingänge können eingelesen werden.

Tabelle 2-4: Status-Register (Base + 0)

BIT D0	0	Tranfer ist beendet; Daten stehen in den SHIFT-REGISTER bereit
	1	Transfer
BIT D1	0	SSI-Daten Eingang ist offen
	1	SSI-Daten Eingang ist auf 24 V Pegel
BIT D2	0	SSI-Daten Eingang ist offen
	1	SSI-Daten Eingang ist auf 24 V Pegel
BIT D3	0	SSI-Daten Eingang ist offen
	1	SSI-Daten Eingang ist auf 24 V Pegel
BIT D4	0	Eingang1 ist auf 24V Pegel
	1	Eingang1 ist offen
BIT D5	0	Eingang2 ist auf 24V Pegel
	1	Eingang2 ist offen
BIT D6	0	Eingang3 ist auf 24V Pegel
	1	Eingang3 ist offen

2.6.7 SHIFT-REGISTER

Es befinden sich drei 32-Bit Register, in denen die Daten von der jeweiligen SSI gespeichert sind. Das Auslesen wird 32-Bit breit gemacht. Danach müssen aus dem 32-Bit Register die Daten gefiltert werden. (Siehe Tabelle 2-1).

2.6.8 OUTPUT-REGISTER

Wenn das Bit D0 gesetzt wird, wird der Ausgang gesetzt.
Wenn das Bit D0 zurückgesetzt ist (Zustand nach Reset) ist der Ausgang offen.

2.6.9 Versions-REGISTER (Base +60)

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

BASE + 60 "S" "I" "1" "0"

Bedeutet: Synchron serielles Interface Revision 1.0

2.7 Arbeiten mit der SSI Funktion

1. Das Übertragungsprofil des SSI Gebers legt die Anzahl der notwendigen Impulse zur Übertragung der Positioninformation fest, z.B.:
2. 25 Bit Profil benötigt 26 Takte → in COUNTS-REG sollen 25 Takte geschrieben werden.
3. Taktfrequenz:
   - Die minimale Frequenz wird durch die Monoflopzeit festgelegt. Für ca. 20 µs ist die minimale Frequenz 50kHz,
   - Die maximale Frequenz wird durch die Kabellänge festgelegt; der Teilerfaktor wird in das 16-Bit FRQ-REG geschrieben.
4. Die neuen Daten des SSI-Gebers werden über ein Dummy-Schreiben auf das START-REG angefordert.
5. Das Übertragungsende kann über das DONE-Bit im STATUS-REG abgefragt werden (polling).
6. Werte aus dem SHIFT-REGISTER einlesen.

3 STANDARDSOFTWARE

3.1 Define-Werte

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_DISABLE	0	0
APCI1710_ENABLE	1	1
ACPI1710_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28
APCI1710_BINARY_MODE	1	1
APCI1710_GRAY_MODE	0	0

3.2 SSI-Initialisierung

1) i_APCI1710_InitSSI (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitSSI
                          (BYTE    b_BoardHandle,
                           BYTE    b_ModulNbr,
                           BYTE    b_SSIProfile,
                           BYTE    b_PositionTurnLength,
                           BYTE    b_TurnCptLength,
                           BYTE    b_PCIInputClock,
                           ULONG   ul_SSIOutputClock,
                           BYTE    b_SSICountingMode)
```

Parameter:

- Eingabe:

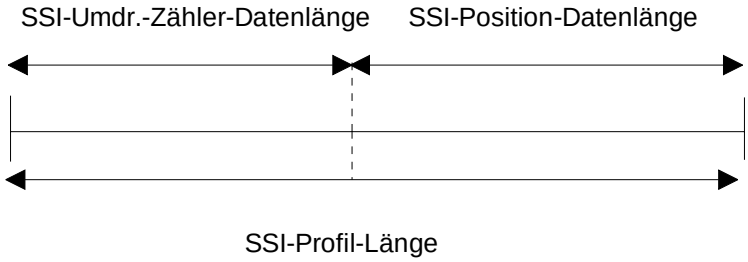
BYTE	b_BoardHandle	Handle der Karte APCI-/CPCI-1710
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_SSIProfile	Auswahl der SSI Profillänge (2 bis 32). Siehe Abbildung 3-1.
BYTE	b_PositionTurnLength	Auswahl der SSI Position-Datenlänge. (1 bis 31) Siehe Abbildung 3-1.
BYTE	b_TurnCptLength	Auswahl der SSI Datenlänge für die Umdrehungen (1 bis 31). Falls es sich um einen Single-Turn Drehgeber handelt, muss dieser Parameter auf 0 gesetzt werden. Andernfalls siehe Abb. 3-1.
BYTE	b_PCIInputClock	Auswahl des PCI Bus Takts - APCI1710_30MHZ: Das PC verfügt über einen PCI Takt Clock von 30 MHz - APCI1710_33MHZ: Das PC verfügt über einen PCI Bus Takts von 33 MHz
ULONG	ul_SSIOutputClock	Auswahl des SSI Ausgangstakts. - Zwischen 229 und 5 000 000 Hz für 30 MHz - Zwischen 252 und 5 000 000 Hz für 33 MHz

BYTE	b_SSICountingMode	Auswahl des SSI Zählermodes. - APCI1710_BINARY_MODE: Binärcode. Falls der SSI Drehgeber Gray-Werte liefert, werden die Werte vom Programm als binäre Werte zurückgegeben. Mit APCI1710_BINARY_MODE wird eine Konvertierung durchgeführt. - APCI1710_GRAY_MODE: Graycode. Für binäre SSI Drehgeber soll dieser Mode vewendet werden, um Return-Werte in binär zu lesen Dieser Mode gibt Gray-Werte für Gray SSI Drehgeber zurück. Mit APCI1710_GRAY_MODE erfolgt keine Konvertierung.

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Konfiguriert den SSI-Betriebsmode des ausgewählten Moduls (*b_ModulNbr*). Rufen Sie zuerst diese Funktion auf, bevor Sie eine andere Funktion aufrufen, die auf das SSI zugreift.

Abb. 3-1: SSI-Profil



Funktionsaufruf:
ANSI C:

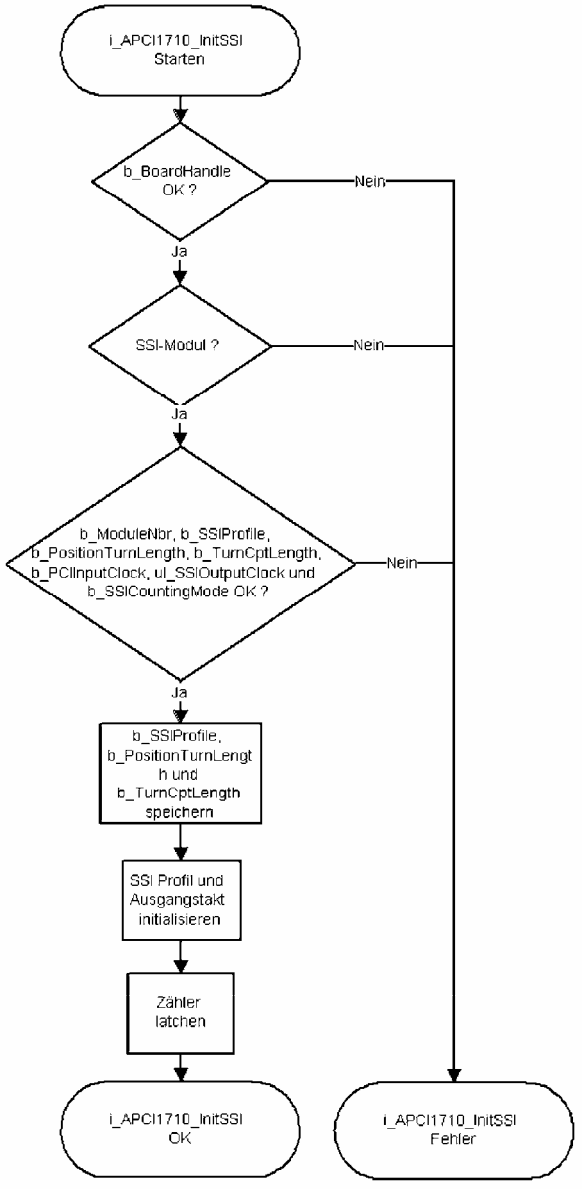
```
int                 i_ReturnValue;
unsigned char       b_BoardHandle;

i_ReturnValue = i_APCI1710_InitSSI      (b_BoardHandle,
                                         0,
                                         25,
                                         12,
                                         12,
                                         APCI1710_33MHZ,
                                         150000,
                                         APCI1710_BINARY_MODE);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul
-4: Die ausgewählte SSI Profillänge ist falsch
-5: Die ausgewählte Datenlänge der SSI Position ist falsch
-6: Die ausgewählte SSI Datenlänge der Umdrehungen ist falsch
-7: Der ausgewählte PCI Eingangstakt ist falsch
-8: Der ausgewählte SSI Ausgangstakt ist falsch
-9: Der ausgewählte SSI Zählermode ist falsch

Eingabe

b_BoardHandle	b_TurnCptLength
b_ModulNbr	b_PCIInputClock
b_SSIProfile	ul_SSIOutputClock
b_PositionTurnLength	b_SSICountingMode

Funktionsablauf



Ausgabe

<Return Wert>

3.3 SSI lesen

1) i_APCI1710_Read1SSIValue (...)

Syntax:
<Return Wert> = i_APCI1710_Read1SSIValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
BYTE b_SelectedSSI,
PULONG pul_Position,
PULONG pul_TurnCpt)

Parameter:
- Eingabe:

BYTE	b_BoardHandle	Handle der Karte **APCI-/CPCI-1710**
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_SelectedSSI	Auswahl des SSI Zählers (0 bis 2)

- Ausgabe:

PULONG	pul_Position	SSI Position in den Umdrehungen
PULONG	pul_TurnCpt	Anzahl der Umdrehungen

Aufgabe:
Liest den SSI Zähler (*b_SelectedSSI*) des ausgewählten Moduls (*b_ModulNbr*).

Funktionsaufruf:
ANSI C:

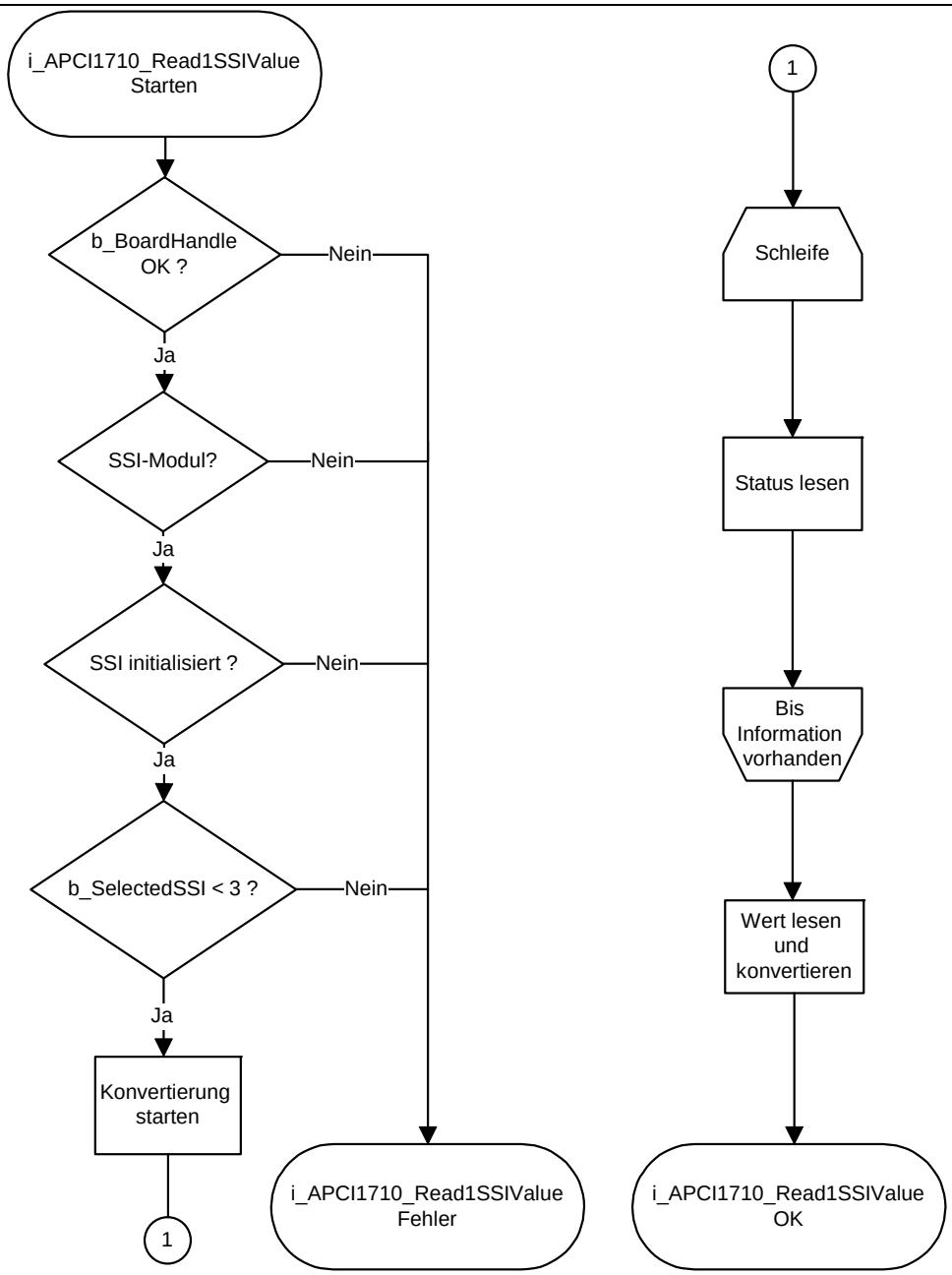
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned long   ul_Position;
unsigned long   ul_TurnCpt;

i_ReturnValue = i_APCI1710_Read1SSIValue
                                    (b_BoardHandle,
                                     0,
                                     0,
                                     &ul_Position,
                                     &ul_TurnCpt);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul
-4: SSI nicht initialisiert. Siehe Funktion "i_APCI1710_InitSSI"
-5: Das ausgewählte SSI ist falsch.

Eingabe

b_BoardHandle b_SelectedSSI
b_ModulNbr

Funktionsablauf



Ausgabe

pul_Position
pul_TurnCpt

<Return Wert>

2) i_APCI1710_ReadAllSSIValue (...)

Syntax:
<Return Wert> = i_APCI1710_ReadAllSSIValue
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PULONG pul_Position,
PULONG pul_TurnCpt)

Parameter:
- Eingabe:

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

- Ausgabe:

PULONG	pul_Position	SSI Position in den Umdrehungen
PULONG	pul_TurnCpt	Anzahl der Umdrehungen

Aufgabe:
Liest alle SSI Zähler des ausgewählten Moduls (*b_ModulNbr*).
pul_Position [0]: SSI Position des SSI Zählers 0
pul_Position [1]: SSI Position des SSI Zählers 1
pul_Position [2]: SSI Position des SSI Zählers 2
pul_TurnCpt [0]: Anzahl der Umdrehungen für SSI Zähler 0
pul_TurnCpt [1]: Anzahl der Umdrehungen für SSI Zähler 1
pul_TurnCpt [2]: Anzahl der Umdrehungen für SSI Zähler 2

Funktionsaufruf:
ANSI C:

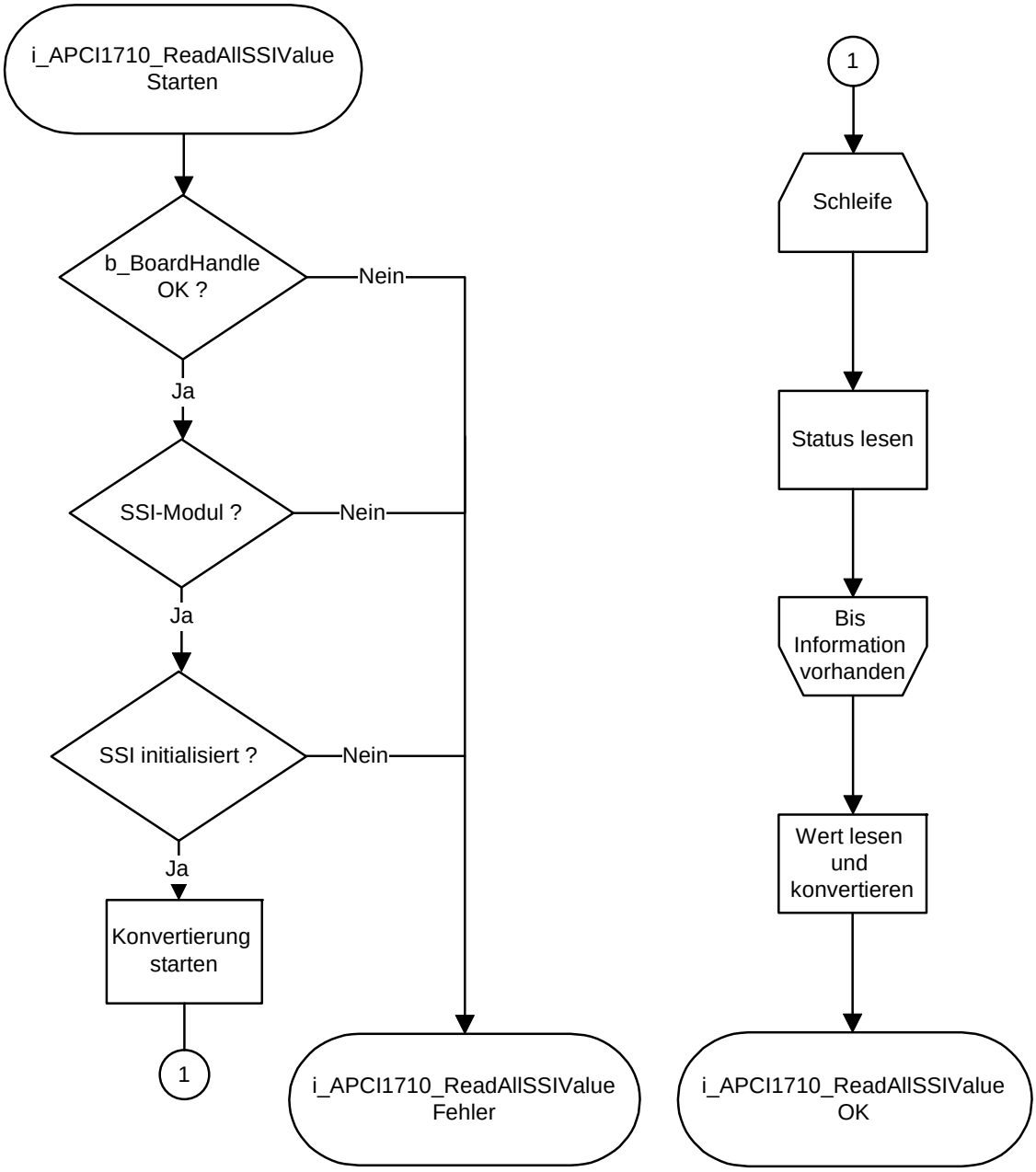
```
int             i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_Position   [3];
unsigned long ul_TurnCpt  [3];

i_ReturnValue = i_APCI1710_ReadAllSSIValue   (b_BoardHandle,
                                               0,
                                               ul_Position,
                                               ul_TurnCpt);
```

Return Wert:
0: Kein Fehler
-1: Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul
-4: SSI nicht initialisiert. Siehe Funktion "i_APCI1710_InitSSI"

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

pul_Position
pul_TurnCpt

<Return Wert>

3.4 Digitale SSI Eingänge

1) i_APCI1710_ReadSSI1DigitalInput (...)

Syntax:
<Return Wert> = i_APCI1710_ReadSSI1DigitalInput
(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 zu konfigurierenden Moduls (0 to 3)
BYTE	b_InputChannel	Auswahl des digitalen Eingangs (0 to 2).

- Ausgabe:

PBYTE	pb_ChannelStatus	Status des digitalen Eingangs. 0: nicht aktiv 1: Eingang ist aktiv

Aufgabe:
Gibt den Status des ausgewählten digitalen Eingangs auf das SSI Modul zurück. Variable *b_InputChannel* und Modul *b_ModulNbr*.

Funktionsaufruf:
ANSI C:

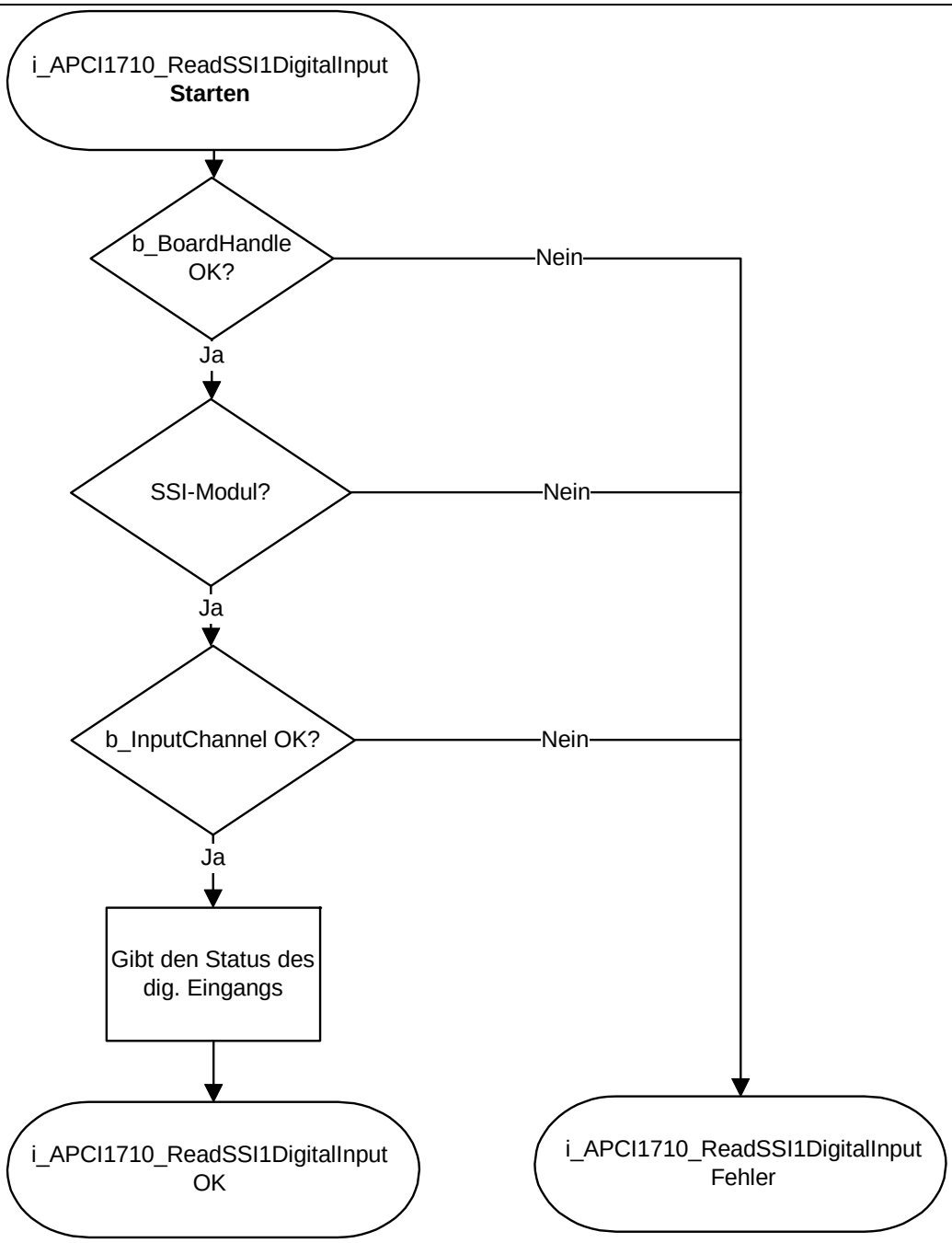
```
int             i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_ChannelStatus;

i_ReturnValue = i_APCI1710_ReadSSI1DigitalInput
                (b_BoardHandle,
                 0,
                 0,
                &b_ChannelStatus);
```

Return Wert:
0: Kein Fehler
-1: Der Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul
-4: Auswahl des digitalen SSI Eingangs ist falsch

Eingabe

b_BoardHandle b_InputChannel
b_ModulNbr

Funktionsablauf



Ausgabe

pb_ChannelStatus

<Return Wert>

2) i_APCI1710_ReadSSIAllDigitalInput (...)

Syntax:
<Return Wert> = i_APCI1710_ReadSSIAllDigitalInput
(BYTE b_BoardHandle,
BYTE b_ModulNbr,
PBYTE pb_InputStatus)

Parameter:
- Eingabe:

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

- Ausgabe

PBYTE	pb_InputStatus	Status der digitalen Eingänge.

Aufgabe:
Gibt den Status aller digitalen SSI Eingänge zurück. (*b_ModulNbr*).

D2	D1	D0
EINGANG 2	EINGANG 1	EINGANG 0

0: Eingang ist aktiv
1: Eingang ist nicht aktiv

Funktionsaufruf:
ANSI C:

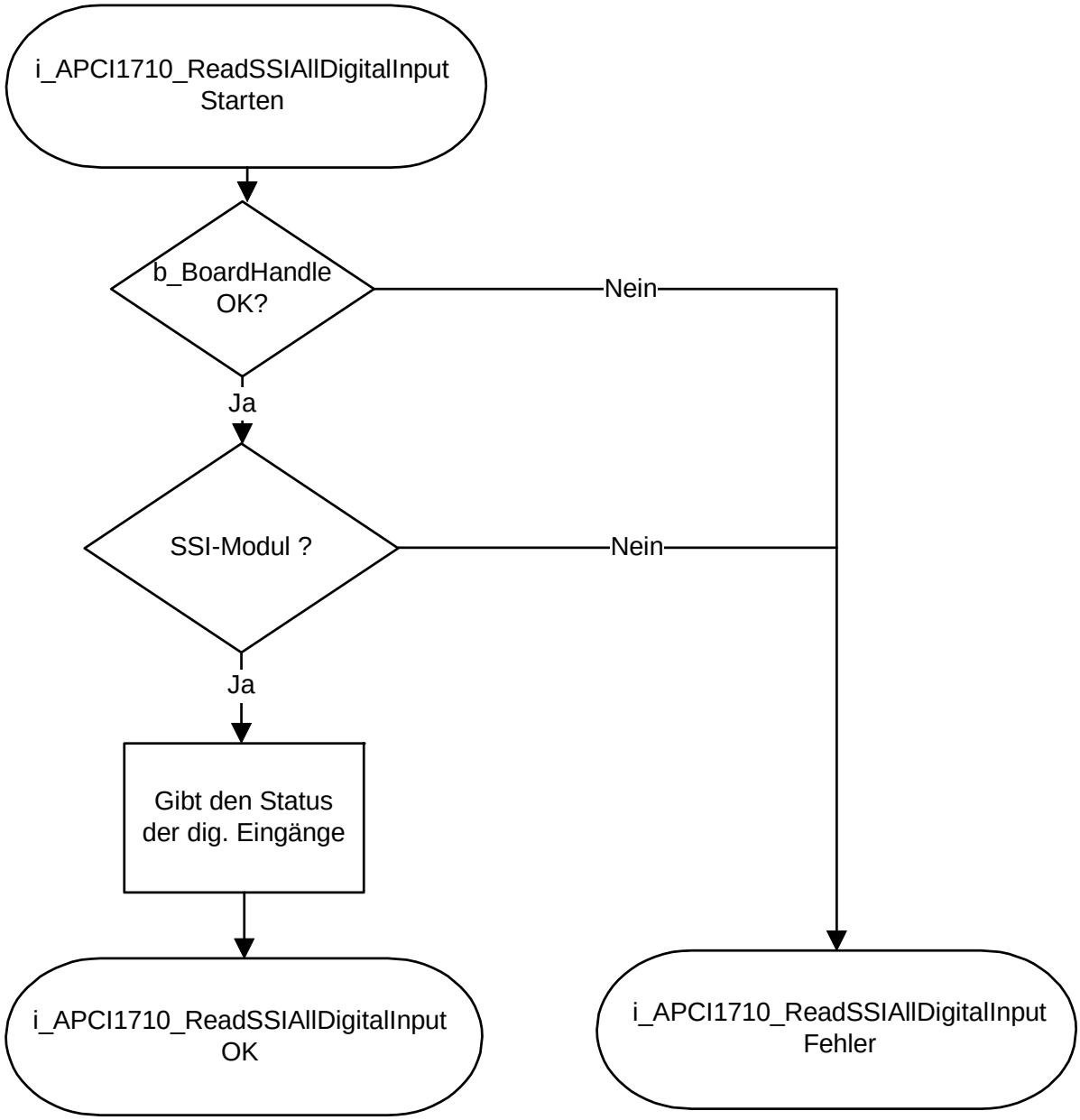
```
int             i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_InputStatus;

i_ReturnValue = i_APCI1710_ReadSSIAllDigitalInput
                                    (b_BoardHandle,
                                     0,
                                     &b_InputStatus);
```

Return Wert:
0: Kein Fehler
-1: Der Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

pb_InputStatus

<Return Wert>

3.5 Digitaler SSI Ausgang

1) i_APCI1710_SetSSIDigitalOutputOn (...)

Syntax:
<Return Wert> = i_APCI1710_SetSSIDigitalOutputOn
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

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

- Ausgabe
Es erfolgt keine Ausgabe.

Aufgabe:
Schaltet den digitalen Ausgang des ausgwählten SSI Moduls (b_ModulNbr) ein.

Funktionsaufruf:
ANSI C:

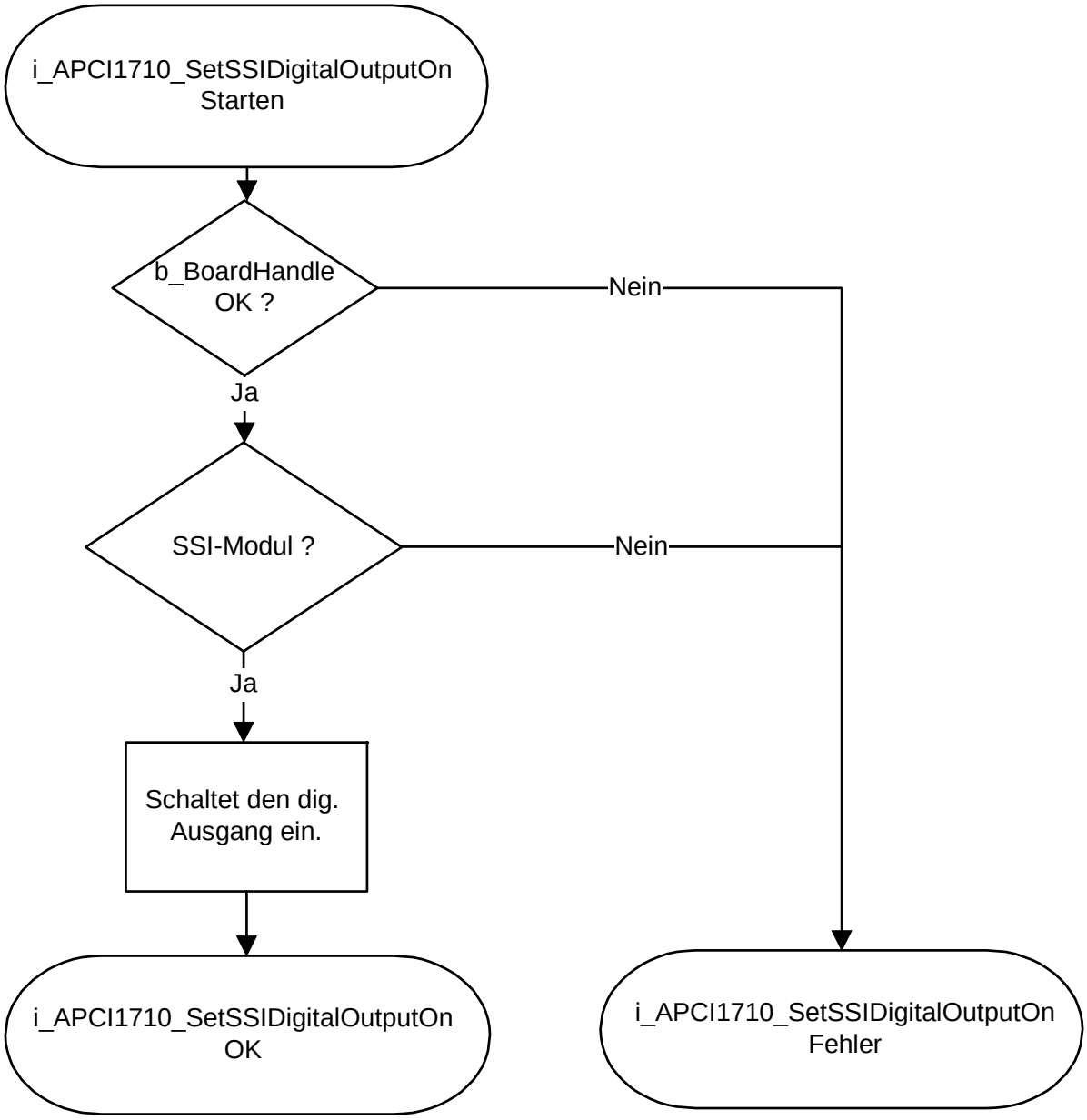
```
int             i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_APCI1710_SetSSIDigitalOutputOn
                                        (b_BoardHandle,
                                         0);
```

Return Wert:
0: Kein Fehler
-1: Der Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

2) i_APCI1710_SetSSIDigitalOutputOff (...)

Syntax:
<Return Wert> = i_APCI1710_SetSSIDigitalOutputOff
(BYTE b_BoardHandle,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

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

- Ausgabe
Es erfolgt keine Ausgabe.

Aufgabe:
Schaltet den digitalen Ausgang des ausgwählten SSI Moduls (b_ModulNbr) aus.

Funktionsaufruf:
ANSI C:

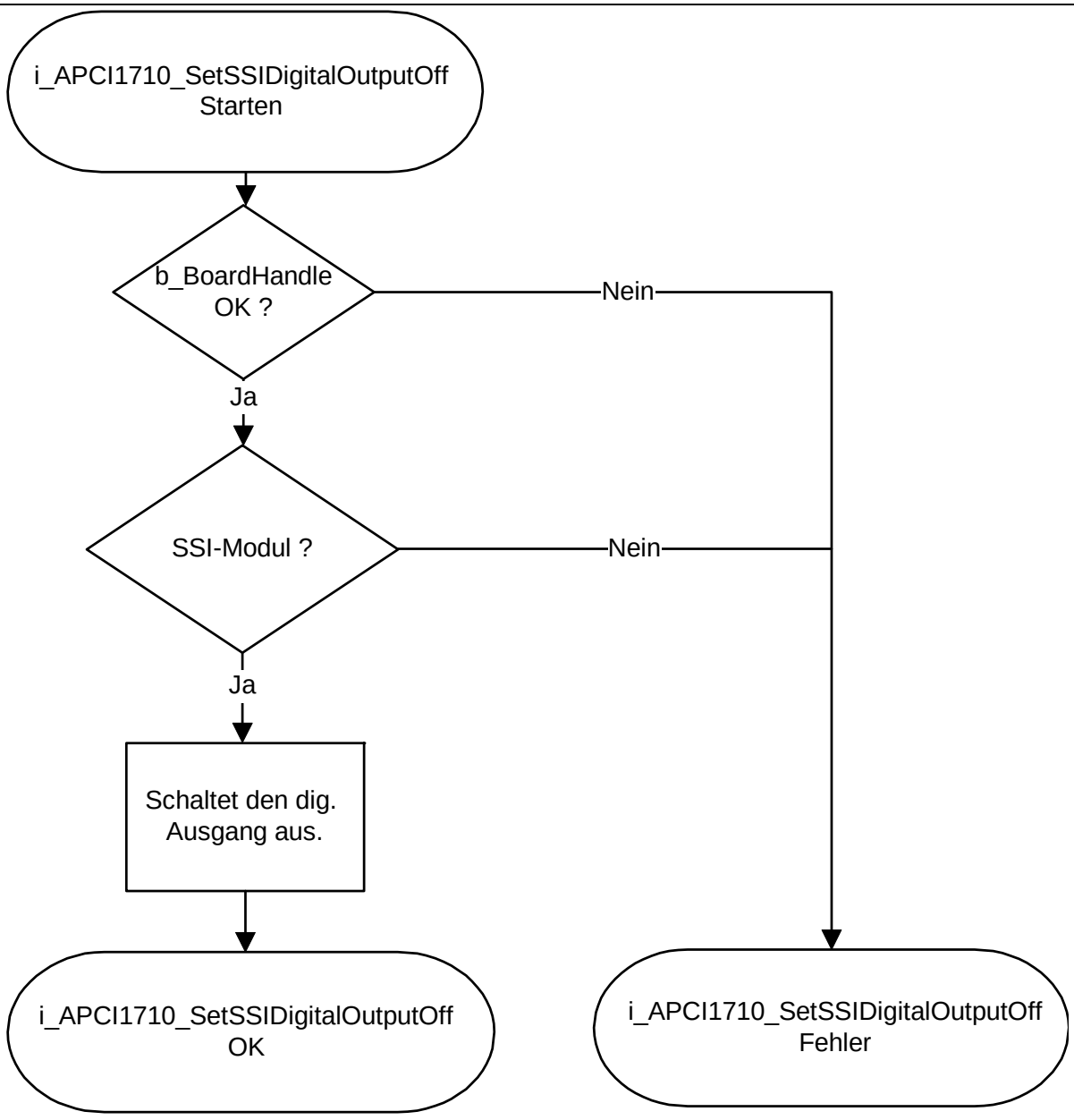
```
int             i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_APCI1710_SetSSIDigitalOutputOff
                                        (b_BoardHandle,
                                         0);
```

Return Wert:
0: Kein Fehler
-1: Der Handle Parameter der Karte ist falsch
-2: Das ausgewählte Modul ist falsch
-3: Das Modul ist kein SSI Modul

Eingabe

b_BoardHandle
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

3.6 Funktionen im Kernel-Mode

WICHTIG!
Diese Funktionen sind nur für die Benutzer Interruptroutine in Windows NT und Windows 95 im synchronen Mode verfügbar. Siehe Funktion "i_APCI1710_SetBoardIntRoutineWin32"

1) i_APCI1710_KRNL_ReadSSI1DigitalInput (...)

Syntax:
```
<Return Wert> =  i_APCI1710_KRNL_ReadSSI1DigitalInput
        (UINT ui_BaseAddress,
         BYTE         b_ModulNbr,
         BYTE b_InputChannel
         PBYTE        pb_ChannelStatus)
```

Parameter:

- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-1710.** See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)
BYTE	b_InputChannel	Auswahl des digitalen Eingangs (0 to 2).

- Ausgabe:

PBYTE	pb_ChannelStatus	Status des digitalen Eingangs. 0: Eingang ist aktiv 1: Eingang ist nicht aktiv

Aufgabe:
Gibt den Status des ausgewählten digitalen Eingangs (b_InputChannel) vom SSI Modul (b_ModulNbr) zurück.

Funktionsaufruf:
ANSI C:

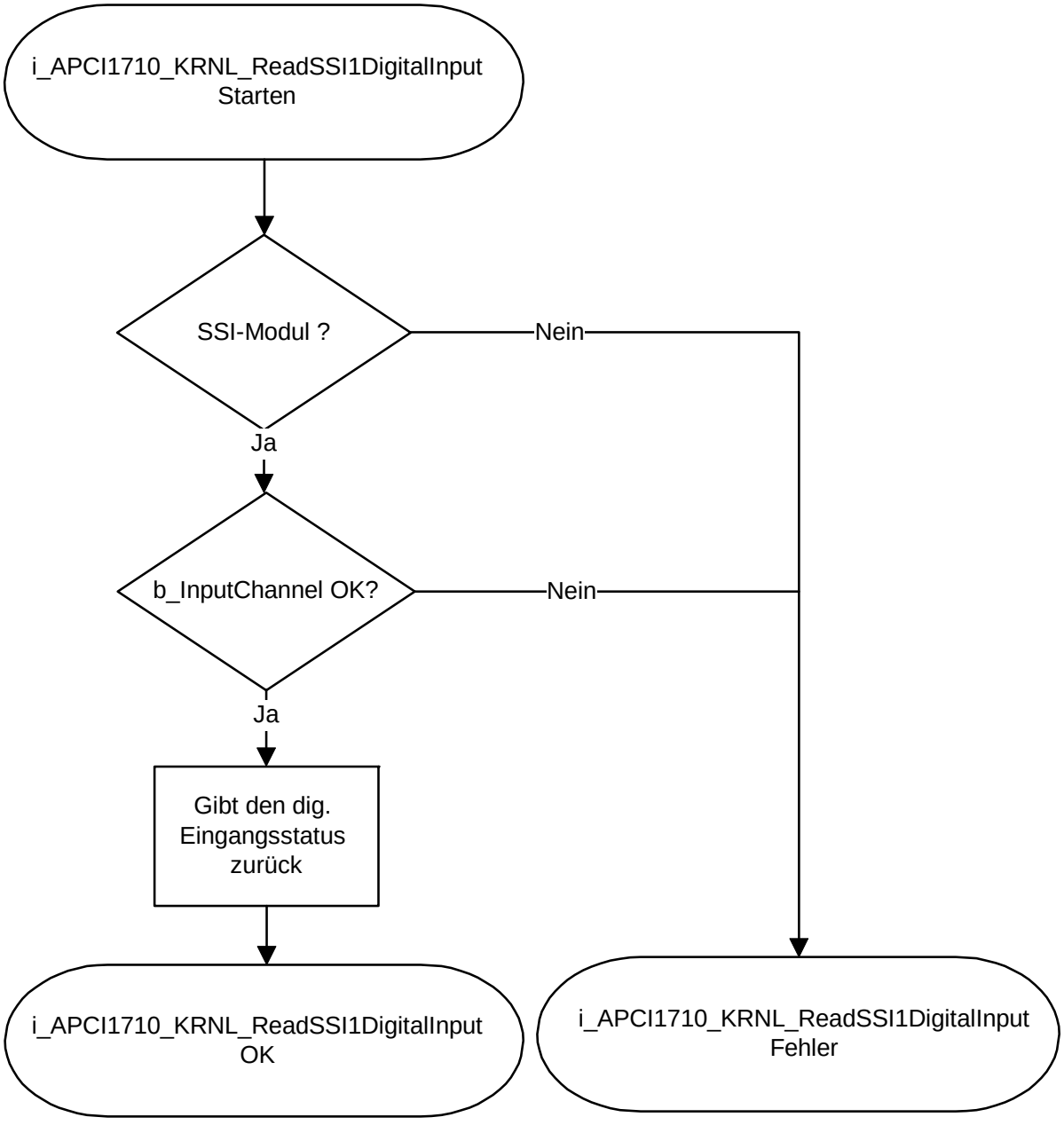
```
int               i_ReturnValue;
unsigned int      ui_BaseAddress;
unsigned char     b_ChannelStatus;

i_ReturnValue = i_APCI1710_KRNL_ReadSSI1DigitalInput
                                        (ui_BaseAddress,
                                         0,
                                         0,
                                        &b_ChannelStatus);
```

Return Wert:
0: Kein Fehler
-1: Das ausgewählte Modul ist falsch
-2: Das Modul ist kein SSI Modul
-3: Auswahl des digitalen SSI Eingangs ist falsch

Eingabe

ui_BaseAddress b_InputChannel
b_ModulNbr

Funktionsablauf



Ausgabe

pb_ChannelStatus

<Return Wert>

2) i_APCI1710_KRNL_ReadSSIAllDigitalInput (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_ReadSSIAllDigitalInput
(UINT ui_BaseAddress,
BYTE b_ModulNbr,
PBYTE pb_InputStatus)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-1710.** See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe

PBYTE	pb_InputStatus	Status der digitalen Eingänge. Siehe Tabelle 3-2.

Aufgabe:
Gibt den Status aller digitalen SSI-Eingänge zurück. (*b_ModulNbr*).

D2	D1	D0
INPUT 2	INPUT 1	INPUT 0

0 : Kanal ist nicht aktiv
1 : Kanal ist aktiv

Funktionsaufruf:

ANSI C:
int i_ReturnValue;
unsigned int ui_BaseAddress;
unsigned char b_InputStatus;

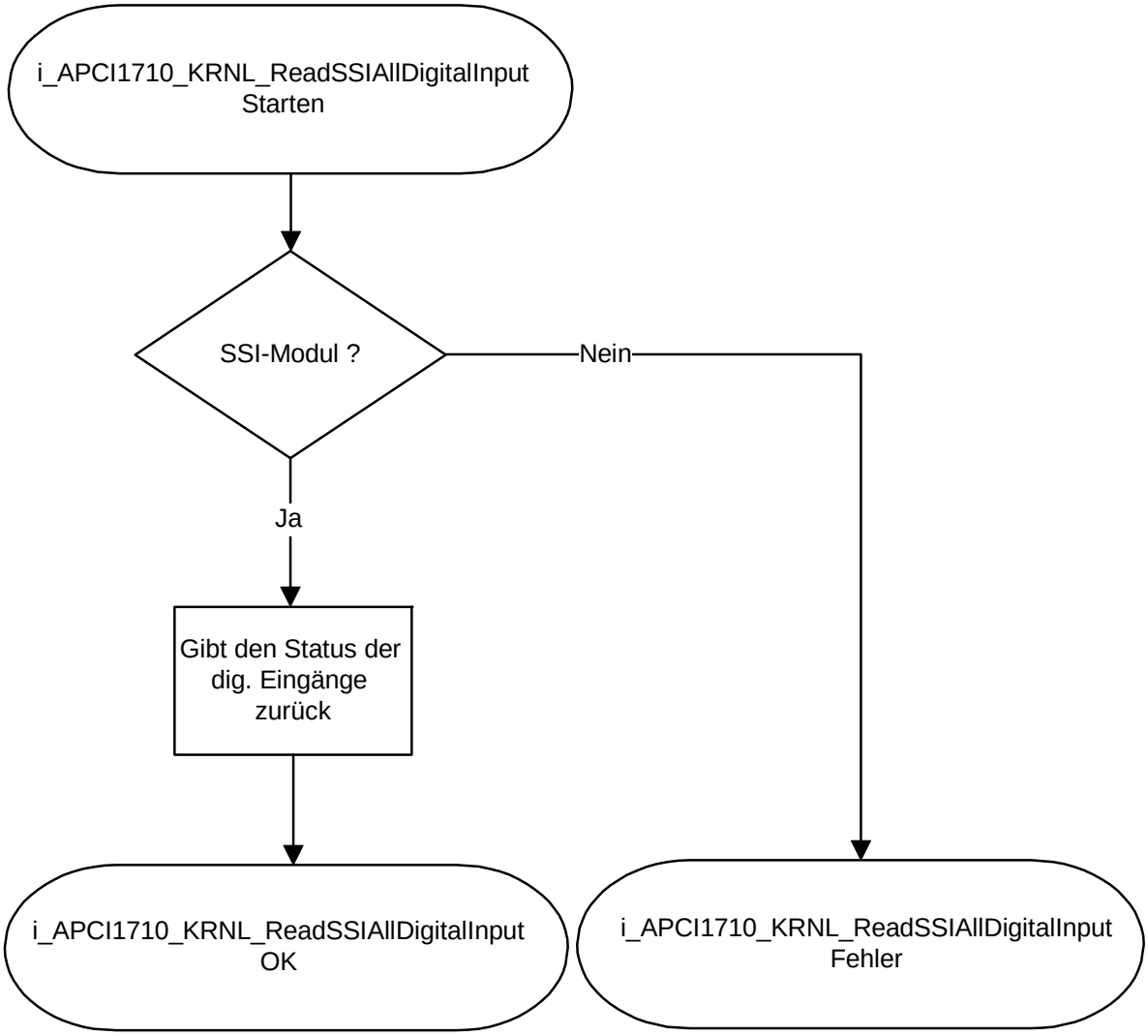
i_ReturnValue = i_APCI1710_KRNL_ReadSSIAllDigitalInput
(ui_BaseAddress,
0,
&b_InputStatus);

Return Wert:
0: Kein Fehler
-1: Das ausgewählte Modul ist falsch
-2: Das Modul ist kein SSI Modul

Eingabe

ui_BaseAddress
b_ModulNbr

Funktionsablauf



Ausgabe

pb_InputStatus

<Return Wert>

3) i_APCI1710_KRNL_SetSSIDigitalOutputOn (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetSSIDigitalOutputOn
(UINT ui_BaseAddress,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

UINT	ui_BaseAddress	Basisadresse der **APCI-1710.** See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Nummer des zu konfigurierenden Moduls (0 bis 3)

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Schaltet den digitalen SSI Ausgang vom Modul b_ModulNbr ein.

Funktionsaufruf:
ANSI C:

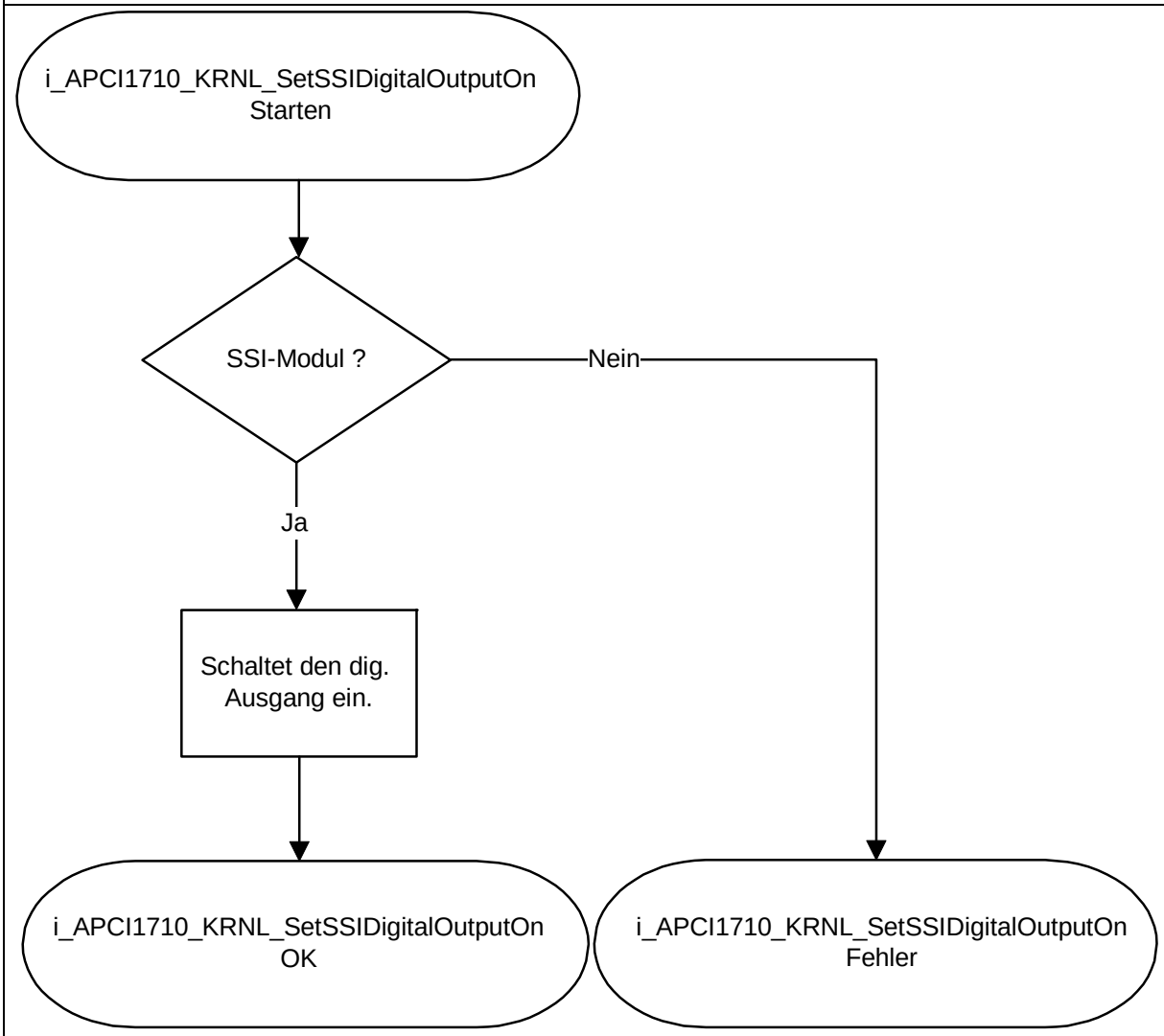
```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetSSIDigitalOutputOn
                                        (ui_BaseAddress,
                                         0);
```

Return Wert:
0: Kein Fehler
-1: Das ausgewählte Modul ist falsch
-2: Das Modul ist kein SSI Modul

Eingabe

ui_BaseAddress
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>

4) i_APCI1710_KRNL_SetSSIDigitalOutputOff (...)

Syntax:
<Return Wert> = i_APCI1710_KRNL_SetSSIDigitalOutputOff
(UINT ui_BaseAddress,
BYTE b_ModulNbr)

Parameter:
- Eingabe:

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

- Ausgabe:
Es erfolgt keine Ausgabe.

Aufgabe:
Schaltet den digitalen SSI Eingang vom Modul b_ModulNbr aus.

Funktionsaufruf:
ANSI C:

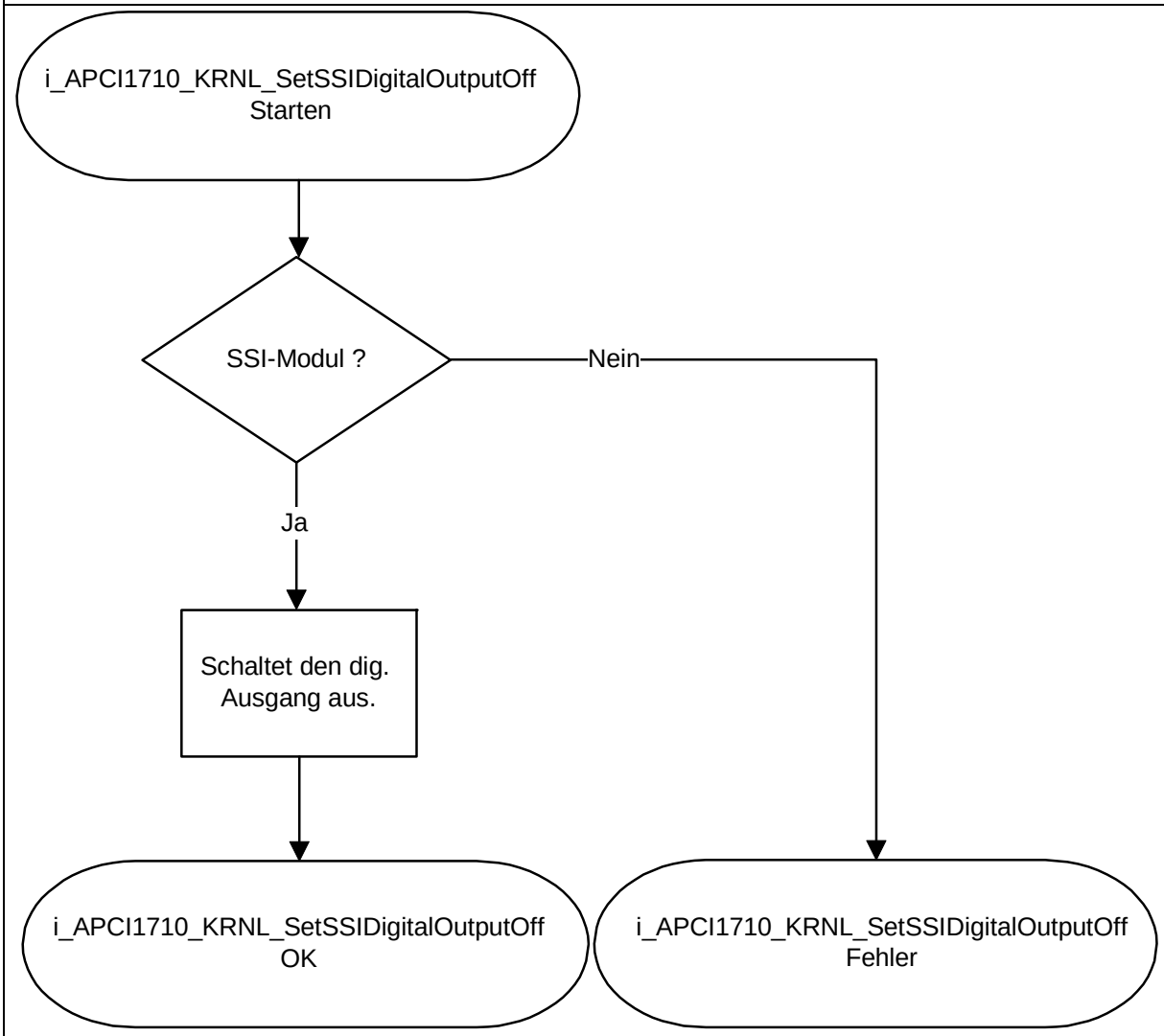
```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;

i_ReturnValue = i_APCI1710_KRNL_SetSSIDigitalOutputOff
                                                (ui_BaseAddress,
                                                 0);
```

Return Wert:
0: Kein Fehler
-1: Das ausgewählte Modul ist falsch
-2: Das Modul ist kein SSI Modul

Eingabe

ui_BaseAddress
b_ModulNbr

Funktionsablauf



Ausgabe

<Return Wert>