



DIN EN ISO 9001:2000  
certified



ADDI-DATA GmbH  
Dieselstraße 3  
D-77833 OTTERSWEIER



Technical support:  
+49 (0)7223 / 9493 – 0

## Technical description

### APCI-/CPCI-1710

**SSI: Synchronous Serial Interface  
(3 fold synchronous serial interface)**

Edition: 02.01 – 11/2006

## Product information

This manual contains the technical installation and important instructions for correct commissioning and usage, as well as production information according to the current status before printing. The content of this manual and the technical product data may be changed without prior notice. ADDI-DATA GmbH reserves the right to make changes to the technical data and the materials included herein.

## Warranty and liability

The user is not permitted to make changes to the product beyond the intended use, or to interfere with the product in any other way.

ADDI-DATA shall not be liable for obvious printing and phrasing errors. In addition, ADDI DATA, if legally permissible, shall not be liable for personal injury or damage to materials caused by improper installation and/or commissioning of the board by the user or improper use, for example, if the board is operated despite faulty safety and protection devices, or if notes in the operating instructions regarding transport, storage, installation, commissioning, operation, thresholds, etc. are not taken into consideration. Liability is further excluded if the operator changes the board or the source code files without authorisation and/or if the operator is guilty of not monitoring the permanent operational capability of working parts and this has led to damage.

## Copyright

This manual, which is intended for the operator and its staff only, is protected by copyright. Duplication of the information contained in the operating instructions and of any other product information, or disclosure of this information for use by third parties, is not permitted, unless this right has been granted by the product licence issued. Non-compliance with this could lead to civil and criminal proceedings.

## ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.

The software must only be used to set up the ADDI-DATA boards.

Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a board by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

## Trademarks

- ADDI-DATA is a registered trademark of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Insight Company.
- Microsoft C, Visual C++, Windows XP, 98, Windows 2000, Windows 95, Windows NT, EmbeddedNT and MS DOS are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DasyLab, Diadem are registered trademarks of National Instruments Corp.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems Inc.

# WARNING

The following risks result from improper implementation and from use of the board contrary to the regulations:



◆ Personal injury



◆ Damage to the board, PC and peripherals



◆ Pollution of the environment

◆ **Protect yourself, the others and the environment!**

◆ **Read carefully the safety precautions (yellow leaflet).**

If this leaflet is not with the documentation, please contact us and ask for it.

◆ **Observe the instructions of the manual.**

Make sure that you do not forget or skip any step. We are not liable for damages resulting from a wrong use of the board.

◆ **Used symbols:**



## **IMPORTANT!**

designates hints and other useful information.



## **WARNING!**

It designates a possibly dangerous situation.

If the instructions are ignored the board, PC and/or peripheral may be destroyed.

<b>1</b>	<b>DEFINITION OF APPLICATION .....</b>	<b>6</b>
1.1	Intended use .....	6
1.2	Usage restrictions.....	6
1.3	Technical description .....	6
1.4	Function description .....	7
1.5	Used abbreviations .....	7
<b>2</b>	<b>SYNCHRONOUS SERIAL INTERFACE .....</b>	<b>8</b>
2.1	Function description .....	8
2.1.1	Block diagram of the SSI .....	9
2.1.2	Typical applications .....	9
2.2	Used signals .....	9
2.3	Pin assignment for all modules with SSI.....	10
2.4	Connection examples.....	11
2.5	I/O mapping of the Synchronous Serial Interface (SSI) ..	13
2.6	Description of the I/O functions.....	14
2.6.1	Function description .....	14
2.6.2	FRQ Register (Base +0) .....	16
2.6.3	COUNTS Register (Base +4) .....	17
2.6.4	CONTROL Register (Base +12) .....	19
2.6.5	START Register (Base +8) .....	19
2.6.6	STATUS Register (Base +0).....	19
2.6.7	SHIFT register.....	20
2.6.8	OUTPUT register .....	20
2.6.9	VERSION register (base +60) .....	20
2.7	Operating with a SSI function.....	20
<b>3</b>	<b>STANDARD SOFTWARE .....</b>	<b>21</b>
3.1	Define values.....	21
3.2	SSI Initialization .....	22
3.3	Read SSI .....	25
3.4	Digital SSI inputs.....	36
3.5	Digital SSI output .....	40
3.6	Functions in the kernel mode.....	44

## Figures

Fig. 2-1: Block diagram of the SSI encoder .....	9
Fig. 2-2: Pin assignment of the 50-pin SUB-D connector X1 .....	10
Fig. 2-3: Connection to a shift encoder TWK CRE 58.....	11
Fig. 2-4: Switching principle of the inputs .....	12
Fig. 2-5: Switching principle of the outputs.....	12
Fig. 2-6: Maximum data rate .....	15
Fig. 2-7: Delay times .....	16
Fig. 3-1: SSI profile .....	23

## Tables

Table 1-1: Delivered manuals .....	7
Table 2-1: Used signals.....	9
Table 2-2: I/O range of the Synchronous Serial Interface (SSI).....	13
Table 2-3: Transmission for a 18 bit encoder.....	18
Table 2-4: Status register .....	19
Table 3-1: Define Value.....	21

# 1 DEFINITION OF APPLICATION

## 1.1 Intended use

The board **APCI-1710** must be inserted in a PC with PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory pursuant to the norm IEC 61010-1.

The board **CPCI-1710** must be inserted in a CompactPCI system with PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory pursuant to the norm IEC 61010-1

## 1.2 Usage restrictions

The board **APCI-/CPCI-1710** must not be used as safety related part for securing emergency stop functions

The board **APCI-/CPCI-1710** must not be used in potentially explosive atmospheres.

## 1.3 Technical description

This manual refers to the **APCI-1710** as well as to the **CPCI-1710** board. Make sure that you have received the following items:

- The CD 1 "Standard Software Drivers" with the ADDISET parameterizing program and the required software drivers.
- The CD 2 "Technical Manuals". This CD contains the following:

- 1) The technical description **ADDICOUNT APCI-1710 / CPCI-1710: Function-programmable counter board for the PCI bus** (containing general information on the operation of the board)
- 2) A function description for each function which you want to program on the board
- 3) The yellow leaflet "Safety precautions"

According to the function used you will find the required assignment and programming functions in the different manuals for each function:

Table 1-1: Delivered manuals

Function	PDF file (CD2 technical manuals)		Function description in SET1710	CFG file
	German	English		
Incremental counter	Inkr_zähler_d.pdf	Incr_counter_e.pdf	Incremental counter	inc_cpt.cfg
SSI	SSI_d.pdf	ssi_e.pdf	SSI	ssi.cfg
SSI monitor	SSI-Monitor_d	SSIMonitor_e.pdf	SSI_Monitor	ssi_mon.cfg
Chronos	chronos_d.pdf	chronos_e.pdf	Chronos	chronos.cfg
Counter/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
Digital I/O	dig_EA_d.pdf	dig_IO_e.pdf	Digital I/O	dig_IO.cfg
Pulse counter	Impulszähler_d.pdf	pulseCounter_e.pdf	Pulse counter	imp_cpt.cfg
ETM (Edge time measurement)	ETM_d.pdf	ETM_e.pdf	Edge time measurement	etm.cfg

**Please note:**

The board CPCI-1710 is compatible with the board APCI-1710 as far as the installation of the software is concerned. The ADDIREG and SET1710 programs make no difference between PCI and CompactPCI boards. The API functions of the standard software are also identical.

## 1.4 Function description

Apart from a global description of the functions this manual contains:

- the pin assignment of the front connector
- a list of the used signals
- the I/O mapping
- connection examples
- a chapter about the API software functions of the standard software.

## 1.5 Used abbreviations

The signals on the 50 pin SUB-D connector refer always to one function module.

Please note the used abbreviations:

- UAS: Interference signal
- CLK: Clock
- REF: Reference point logic
- ENA: Enable

**C1+** is a signal for **function module 1**.

## 2 SYNCHRONOUS SERIAL INTERFACE

### 2.1 Function description

The SSI function is an interface for absolute angle encoders (displacement measurement systems).

It allows the transmission of an absolute position information through a serial data transfer.

It stands out for applications in which high precision and reliability are necessary in a rough industrial environment.

**Advantages in comparison with the parallel interface:**

- Reduced cabling
- The cabling requirement and interface technology does not depend on the data word length
- The shielding capacity against interferences is reached through synchronous and symmetric clock and data signals via a cable equipped with twisted pair lines.
- The optical isolation is entirely completed by optical couplers to avoid any earth circuit.

**Properties:**

Connection of 1 to 3 encoders with SSI function

- The clock is common for all 3 interfaces.
- The clock frequency is adjustable through software to adapt the transfer of the control length.
- The number of data bits is programmable through software, which allows a flexible resolution.
- GRAY or BINARY conversion is possible
- 3 digital inputs and 1 digital output channels are available.

These channels do not affect the SSI functions; they can be used for an additional function.

The interface contains:

- Three from each other independent 32-bit SHIFT register that can be read out over the data bus.
- Clock and pulse generator
- Function and control logic

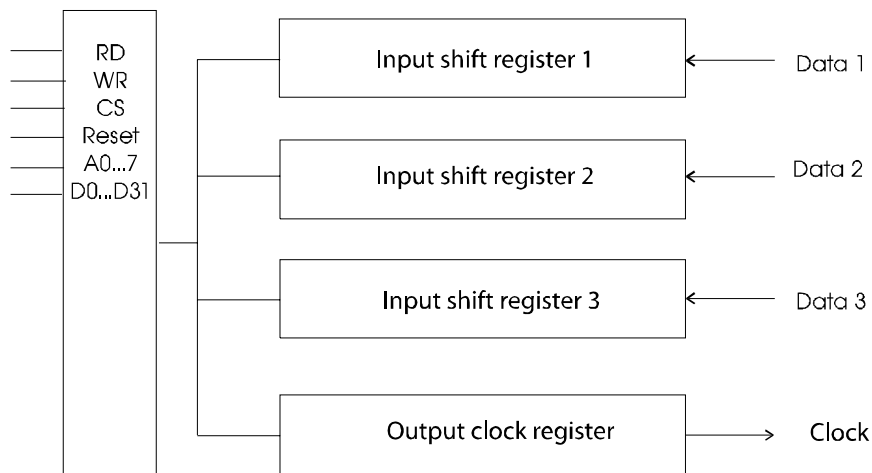


### 2.1.1 Block diagram of the SSI

Das Interface contains:

- 3x2 32-bit SHIFT register each, which can be read out via the data bus,
- Clock and pulse generator,
- Function and control logic.

**Fig. 2-1: Block diagram of the SSI encoder**



### 2.1.2 Typical applications

- Acquisition of displacement measuring systems
- X, Y, Z control
- Tolerance measurement

## 2.2 Used signals

The function "SSI" occupies **6 input channels (channel B to G) and 2 output channels (channel A and H)** of the respective function module on the board APCI-1710/CPCI-1710. On one board you can connect max. 12 absolute encoders or 3 per module.

**Table 2-1: Used signals**

SIGNALS	ON CONNECTOR	POLARITY	FUNCTION
TAKT_x	Ax +/-	Diff.	CLOCK output signal for the SSI encoder
DATA1_x	Bx +/-	Diff. / OPT. 24Vdiff.	DATA input for the first encoder
DATA2_x	Cx +/-	Diff. / OPT. 24Vdiff.	DATA input for the second encoder
DATA3_x	Dx +/-	Diff. / OPT. 24Vdiff.	DATA input for the third encoder
Input1_x	Ex +/-	24V / OPT. 5V	Digital input

Input2_x	Fx +/-	24V / OPT. 5V	Digital input
Input3_x	Gx +/-	24V / OPT. 5V	Digital input
Output_x	Hx +/-	24V / OPT. 5V TTL	Digital output

x = number of the function module

## 2.3 Pin assignment for all modules with SSI

**i**

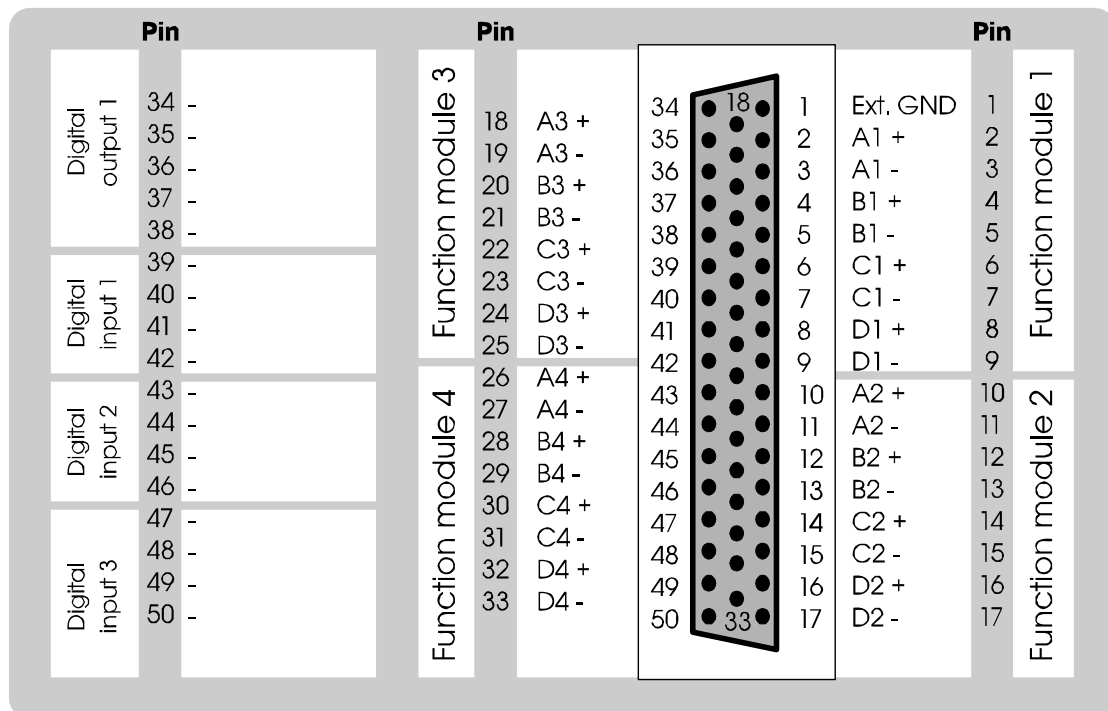
### IMPORTANT!

The function modules are defined differently in the hardware and software descriptions.

In the pin assignment (hardware) the modules 1 to 4 are numbered. In the SET1710 program or the software functions (software) the module numbering **BEGINS** with 0.

**The figure below is a connection example:** The function “SSI” is implemented on all function modules.

Fig. 2-2: Pin assignment of the 50-pin SUB-D connector X1



-: Not connected

## 2.4 Connection examples

The shift encoder CRE 58 by TWK is connected to the function module 1 of the APCI-/CPCI-1710 and CPCI-1710. First interface.

**Fig. 2-3: Connection to a shift encoder TWK CRE 58**

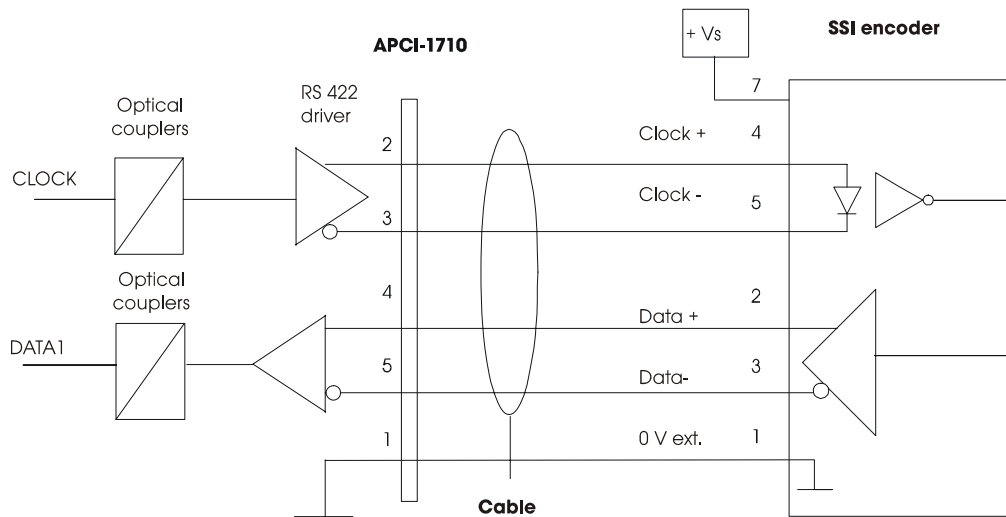


Fig. 2-4: Switching principle of the inputs

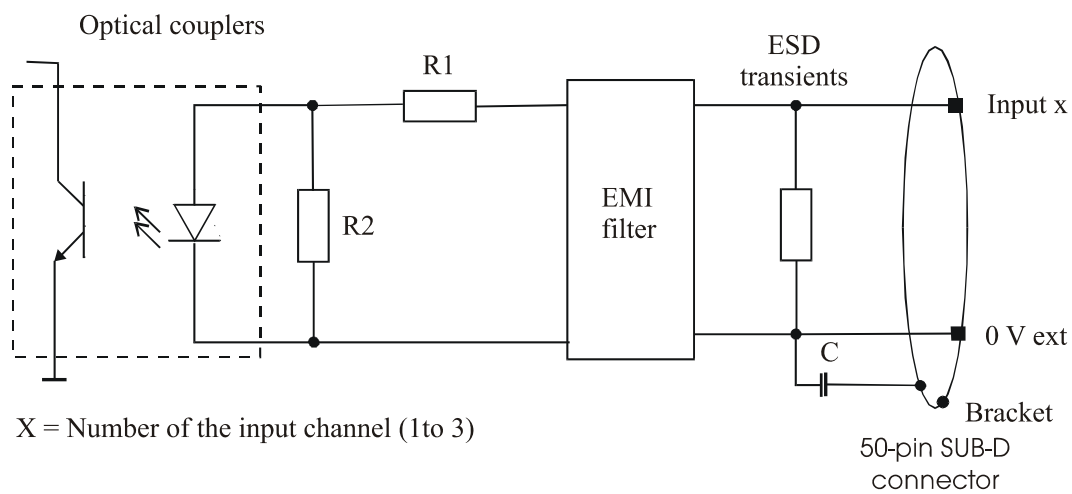
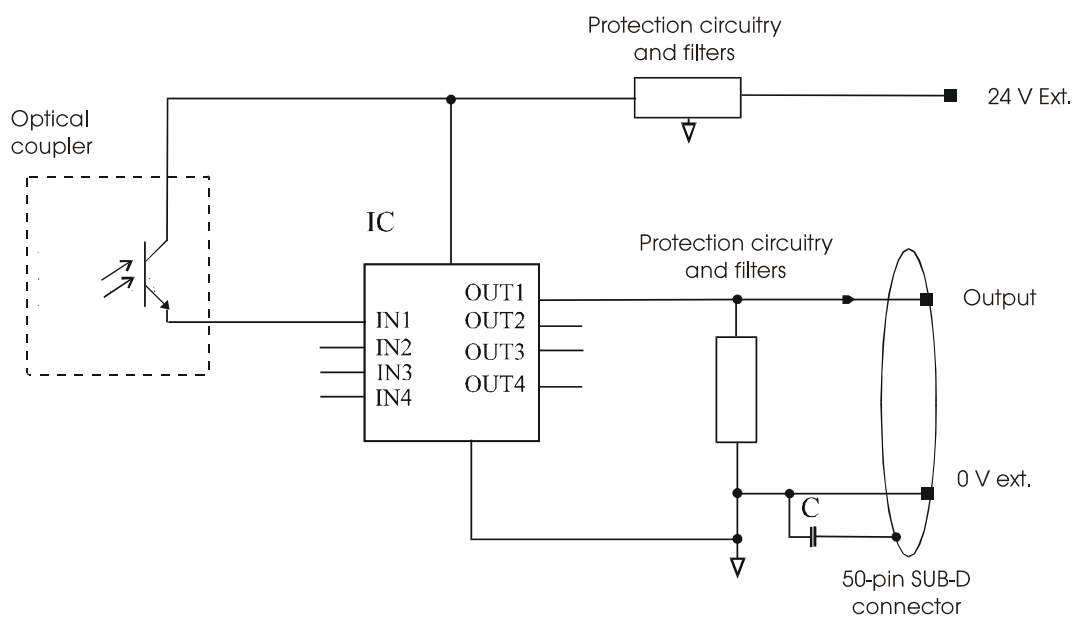


Fig. 2-5: Switching principle of the outputs



## 2.5 I/O mapping of the Synchronous Serial Interface (SSI)

Table 2-2: I/O range of the Synchronous Serial Interface (SSI)

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

	<b>IOWR</b>			
	D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE <sub>x</sub> + 0	-	-	FRQHIG-REG	FRQLOW-REG
BASE <sub>x</sub> + 4	-	-	-	COUNT-REG
BASE <sub>x</sub> + 8	-	-	-	START-REG
BASE <sub>x</sub> + 12	-	-	-	CONTROL-REG
BASE <sub>x</sub> + 16	-	-	-	OUTPUT-REG
.....	-	-	-	-
BASE <sub>x</sub> + 60	-	-	-	-

-: no function; y: no relevant data, **x**: number of the function module.

The SSI occupies 5 DWORDS in the I/O range of function module **x**.

The access are always read or written in 32 bit.

## 2.6 Description of the I/O functions

### 2.6.1 Function description

#### General

The parallel and absolute information of the shift encoder is converted into serial information by an internal parallel-serial converter (shift register).

It is then transmitted to the input channel in synchronisation with the clock emitted by the APCI-1710/CPCI-1710.

The synchronous transmission of the data word is introduced and controlled by a clock signal. The length (i.e. the variable) of the clock sequence is determined through an internal 8-bit register (COUNT-REG) in the APCI-/CPCI-1710, so that the length of the data word to be transmitted can be changed from 0 to 32-bit.

For example, a shift encoder with a **SSI 25-bit interface profile** requires 26 clock signals to be read.

The clock frequency determines the velocity of the data transmission.

This frequency is determined through an internal 16-bit-register (FRQ-REG).

#### Transmission protocol

The logic levels mentioned below refer to the TAKT+ (CLOCK+) or DATA+ signals. In operation or rest state of the SSI, the TAKT- (Clock) line and the data line (Clock+, DAT+) are Log1. The receive electronics transmits the data transfer via a conversion of the clock signal from Log1 to Log 0.

A retriggerable monoflop is set in the shift encoder through this change.

The output of this monoflop converts a parallel shift register into a serial one, whereas the parallel data which is present in gray-code is saved.

At the next change of the clock from Log 0 to Log1, the most significant bit of the shift information is set on the data output channel of the shift encoder.

Any other rising edge drives the following bit to the output channel until the least significant bit. The clock of the monoflop is simultaneously retriggered with any falling edge.

The monoflop period (e.g. 20 µs) determines the period between 2 transfers and the minimum clock frequency.

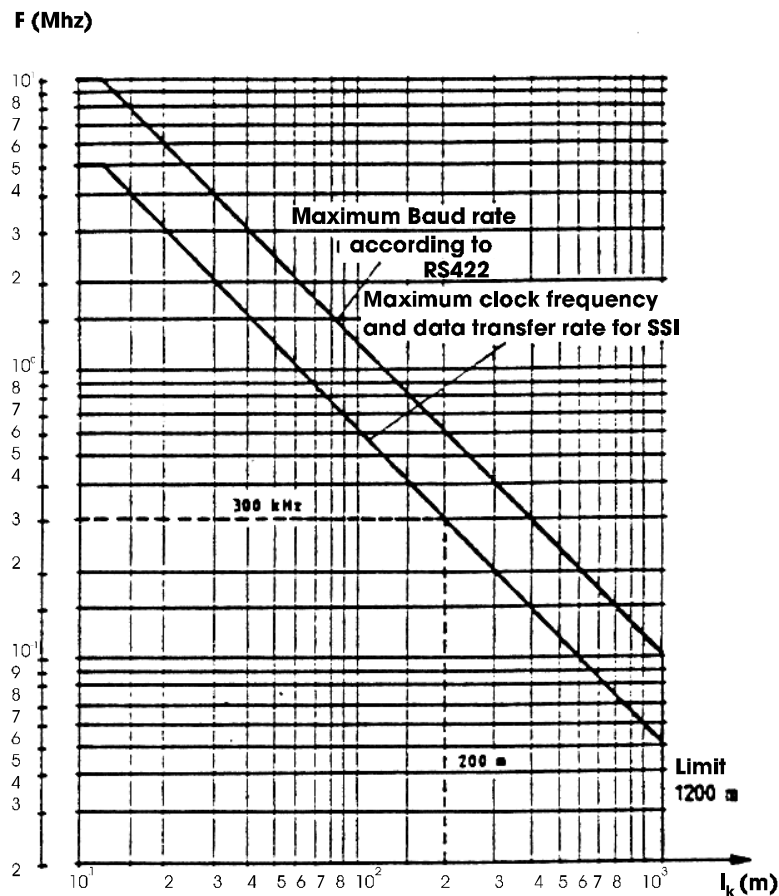
#### Maximum data rate

##### Conditions:

The maximum data rate (clock frequency) is determined by the RS485 norm for the used drivers, receivers and transmission protocol.

The maximum clock frequency amounts to half of the values of the norm for the baud rate, i.e.: 5 MHz.

Fig. 2-6: Maximum data rate



## Transfer line

The principal line of the SSI transfer line consists in a shift encoder, a transmission cable and clock sequence or receive electronics. Each unit naturally transmits the signals with a delay time (running period). As a consequence the data present on the receiver is transmitted synchronously to the clock of the clock sequence yet with a delay amounting to the running period.

This delay time varies according to the length of the transmission line. The clock rate is hence to be adapted to the line.

## Delay time

$$\text{Total delay time (tgv)} = T_{gv} = T_c + 2 \times T_k + T_e$$

-  $T_{gv}$  = total delay time

$T_c$ : Delay time due to the electronics of the shift encoder, max. 150 ns

$T_k$ : Cable delay: depends on the length of the cable. By using a cable of the type LIYCY-OB with  $0.25 \text{ mm}^2$  cross section the specific running period amounts approx. to  $6.5 \text{ ns/m}$ .

$T_e$ : Delay time of the receiver, max. 150 ns

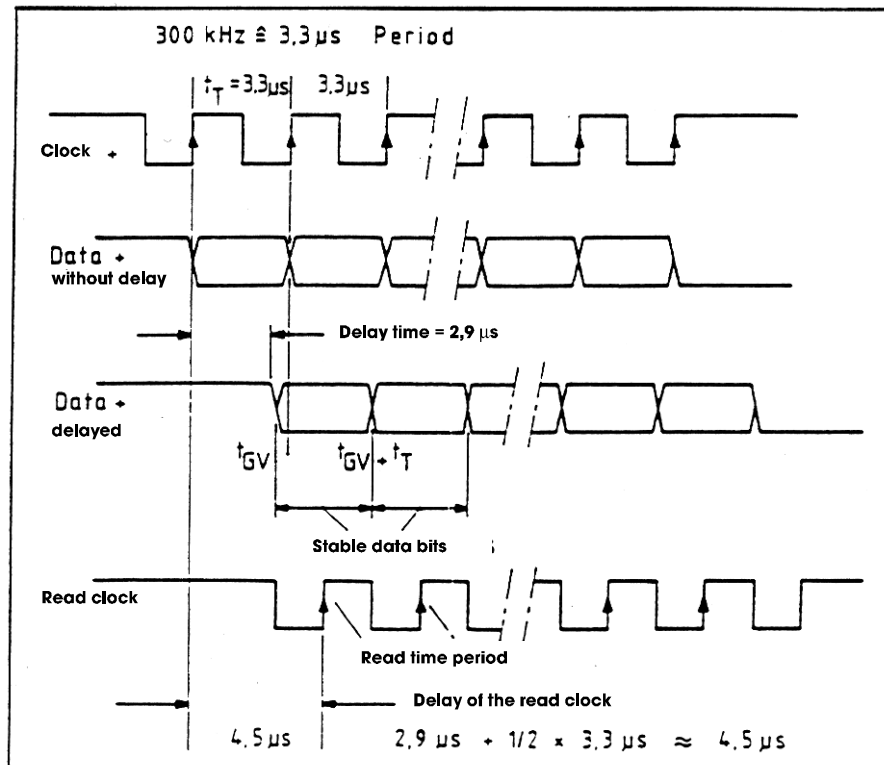
**Example:**

For a cable length of 200 m the delay time amounts to:

$$T_{gv}(\text{ns}) = 300 + (2 \times 6.5 \times 200) = 2900 \text{ ns}$$

The only selectable clock frequency is 300 kHz for a cable length of 200 m.

**Fig. 2-7: Delay times**



## 2.6.2 FRQ Register (Base +0)

16 bit register that determines the clock frequency. This register can only be written.

Input frequency = PCI bus frequency (between 0 and 33 MHz.) You will find the frequency in the manual of your PC.

If the register contains the value 0, the frequency is divided by 2. If the register contains the value 50, the frequency is divided by 100. These frequency values can be from 2 to 13070 in steps of 2. Frequency values can be from 2 to 13070 in steps of 2. Yet the frequency should not be higher than 1 MHz.



### 2.6.3 COUNTS Register (Base + 4)

8bit register that determines the number of bits for the SSI. This register can only be written.

If the register contains the value  $n = 25$ , 26 clock signals are generated;

If the register contains the value  $n = 32$ , 33 clock signals are generated.

#### Transmission example for a 18 bit shift encoder

Angle encoders with 1024 steps / turn (10 bits in monotour) and 256 turns (8 bits in multitour).

The transmission protocol is delivered in the standard version for a 25 bit data word. Hereof 12 bits are for the number of turns, 13 bits for the resolution (step/turn). As the transmission always begins with the multitour bit 12 and the multitour is determined for 8-bit in this example, 4 spaces are transmitted first with Lo0, then 8 bits of the multitour. The bits of the monotour (S10 to S1) are following. The empty 3 bits are also transmitted with Log0.

Table 2-3: Transmission for a 18 bit encoder

		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
Zahl der U.	Bit/ U.	Multitour-Bit												Monotour-Bit												Schritte U.	BitU.	

### 2.6.4 CONTROL Register (Base +12)

8-bit register that allows the conversion from GRAY code into BINARY code. This register can only be written.

The release occurs through a bit for each SSI module. If the 3 bits are set to "0" after reset no data conversion is completed.

If one of the three bits is set to „1“, in the corresponding SSI module the GRAY code is converted to the BINARY code.

### 2.6.5 START Register (Base +8)

By writing on the base address +8 a cycle begins.

The data is serially transmitted.

### 2.6.6 STATUS Register (Base +0)

The information about the current serial data transfer is read on the base address +0.

The state of the digital input channels can be read.

**Table 2-4: Status register**

BIT D0	0	Transfer is over; Data is ready in the SHIFT-REGISTER
	1	Transfer
BIT D1	0	SSI data input is open
	1	SSI data input is on 24 V level
BIT D2	0	SSI data input is open
	1	SSI data input is on 24 V level.
BIT D3	0	SSI data input is open.
	1	SSI data input is on 24 V level
BIT D4	0	Input1 is on 24 V level
	1	Input1 is open
BIT D5	0	Input2 is on 24 V level
	1	Input2 is open.
BIT D6	0	Input3 is on 24 V level
	1	Input 3 is open

### 2.6.7 SHIFT register

2x3 32-bit registers are available to save the data of the respective SSI. The reading operation occurs in 32-bit. The data from the 32-bit registers is then filtered.

### 2.6.8 OUTPUT register

When the bit D0 is set, the output channel is set.

When the bit D0 is reset (State after reset), the output channel is open.

### 2.6.9 VERSION register (base + 60)

The function and the revision are recognized (reading command, ASCII format)

**BASE + 60**   "S"   "I"   "1"   "0"

This means: Synchronous serial interface revision 1.0

## 2.7 Operating with a SSI function

1. The transfer profile of the SSI encoder determines the number of necessary pulses for the transfer of the position information, e.g.:
2. 25-bit profile needs 26 clocks → in COUNTS-REG 25 clocks shall be written.
3. Clock frequency:
  - The min. frequency is determined by the monoflop time.  
For approx. 20 µs the min. frequency is 50 kHz,
  - The max. frequency is determined by the length of the cable; the division factor is written into the 16-bit FRQ-REG
4. The new data of the SSI encoder are demanded through a dummy write to the START-REG.
5. The end of transmission can be demanded through the DONE-Bit in the STATUS-REG (polling).
6. Read the value in the SHIFT-REGISTER.

## 3 STANDARD SOFTWARE

### 3.1 Define values



#### **IMPORTANT!**

Please note the following writing conventions in the text:

Function: "i\_APCI1710\_SetBoardInformation"

Variable *ui\_Address*

**Table 3-1: Define Value**

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 Initialization

### 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_PCInputClock,
                                ULONG     ul_SSIOutputClock,
                                BYTE      b_SSICountingMode)
```

#### Parameter:

##### - Input:

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_SSIProfile	Selection of the SSI profile length (2 to 32). (See Fig. 3-1)
BYTE	b_PositionTurnLength	Selection of the SSI position data length. (1 to 31) (See Fig. 3-1)
BYTE	b_TurnCptLength	Selection of the SSI data length for the turns (1 to 31). If it is a single turn impulse encoder, this parameter must be set to 0. Otherwise see Fig. 3-1
BYTE	b_PCInputClock	Selection of the PCI bus clock - APCI1710_30MHZ: The PC has a PCI Clock of 30 MHz - APCI1710_33MHZ: The PC has a PCI bus clock of 33 MHz
ULONG	ul_SSIOutputClock	Selection of the SSI output clock. - Between 229 and 5 000 000 Hz for 30 MHz - Between 252 and 5 000 000 Hz for 33 MHz
BYTE	b_SSICountingMode	Selection of the SSI counter mode. - APCI1710_BINARY_MODE: Binary code. If the SSI encoder supplies Gray values, the values of the program will be returned as binary values. With the APCI1710_BINARY_MODE a conversion is realized. - APCI1710_GRAY_MODE: Graycode. For binary SSI encoder this mode is to be used to read return values in binaries. This mode returns Gray values for the Gray

SSI encoder.

There occurs no conversion with  
APCI1710\_GRAY\_MODE

**- Output:**

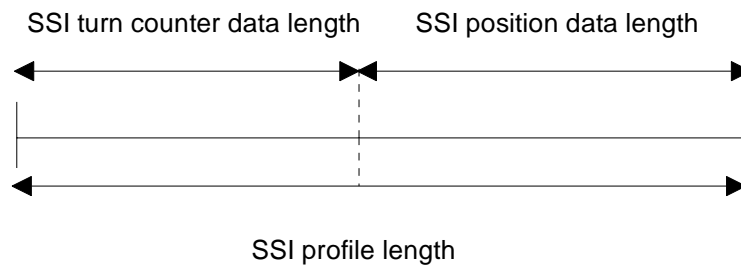
No output signal.

**Task:**

Configures the SSI operating mode of the selected module (*b\_ModulNbr*).

Call up this function before you call up any other function that accesses the SSI.

**Fig. 3-1: SSI profile**



**Calling convention:**

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 value:**

- 0: No error
- 1: Handle parameter of the board is wrong.
- 2: The selected module is wrong.
- 3: The module is no SSI module
- 4: The selected SSI profile length is wrong.
- 5: The selected data length of the SSI position is wrong.
- 6: The selected SSI turn counter data length is wrong.
- 7: The selected PCI input clock is wrong
- 8: The selected SSI output clock is wrong.
- 9: The selected PCI counter mode is wrong.

**Input**

b\_BoardHandle

b\_ModulNbr

b\_SSIProfil

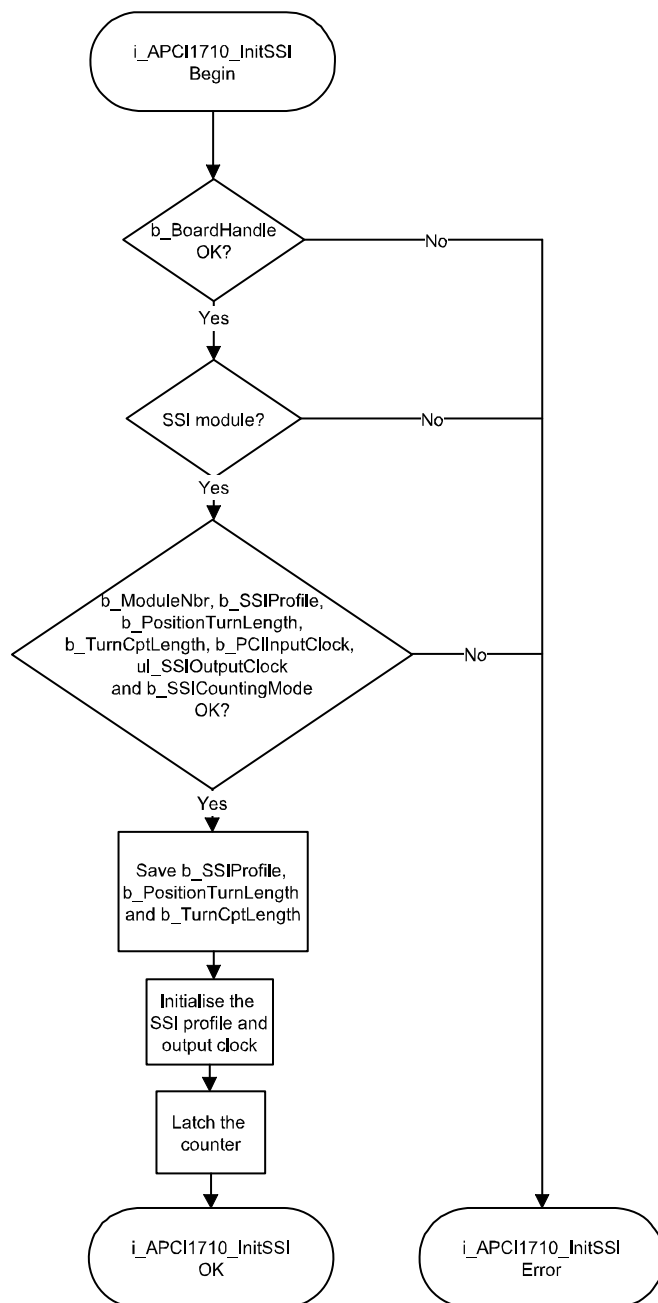
b\_PositionTurnLength

b\_TurnCptLength

b\_PCInputClock

ul\_SSIOutputClock

b\_SSICountingMode

**Output**

&lt;Return Value&gt;



**2) i\_APCI1710\_InitSSIRawData (...)****Syntax:**

```
<Return value> = i_APCI1710_InitSSIRawData
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                BYTE      b_SSIProfile,
                                BYTE      b_PCInputClock
                                ULONG      ul_SSIOutputClock)
```

**Parameter:****-Input:**

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_SSIProfile	Selection of the SSI profile length (2 to 32).
BYTE	b_PCInputClock	Selection of the PCI bus clock - APCI1710_30MHZ: The PC has a PCI clock of 30 MHz - APCI1710_33MHZ: The PC has a PCI bus clock of 33 MHz
ULONG	ul_SSIOutputClock	Selection of the SSI output clock. - Between 229 and 5 000 000 Hz for 30 MHz - Between 252 and 5 000 000 Hz for 33 MHz

**- Output:**

There is no output.

**Task:**

Configures the SSI operation mode of the selected module (*b\_ModulNbr*).  
Firstly call up this function, before calling up other functions that access to the SSI.

**Callin convention:**

ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitSSIRawData (b_BoardHandle,
                                             0,
                                             25,
                                             APCI1710_33MHZ,
                                             150000);
```

**Return value:**

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module is wrong.

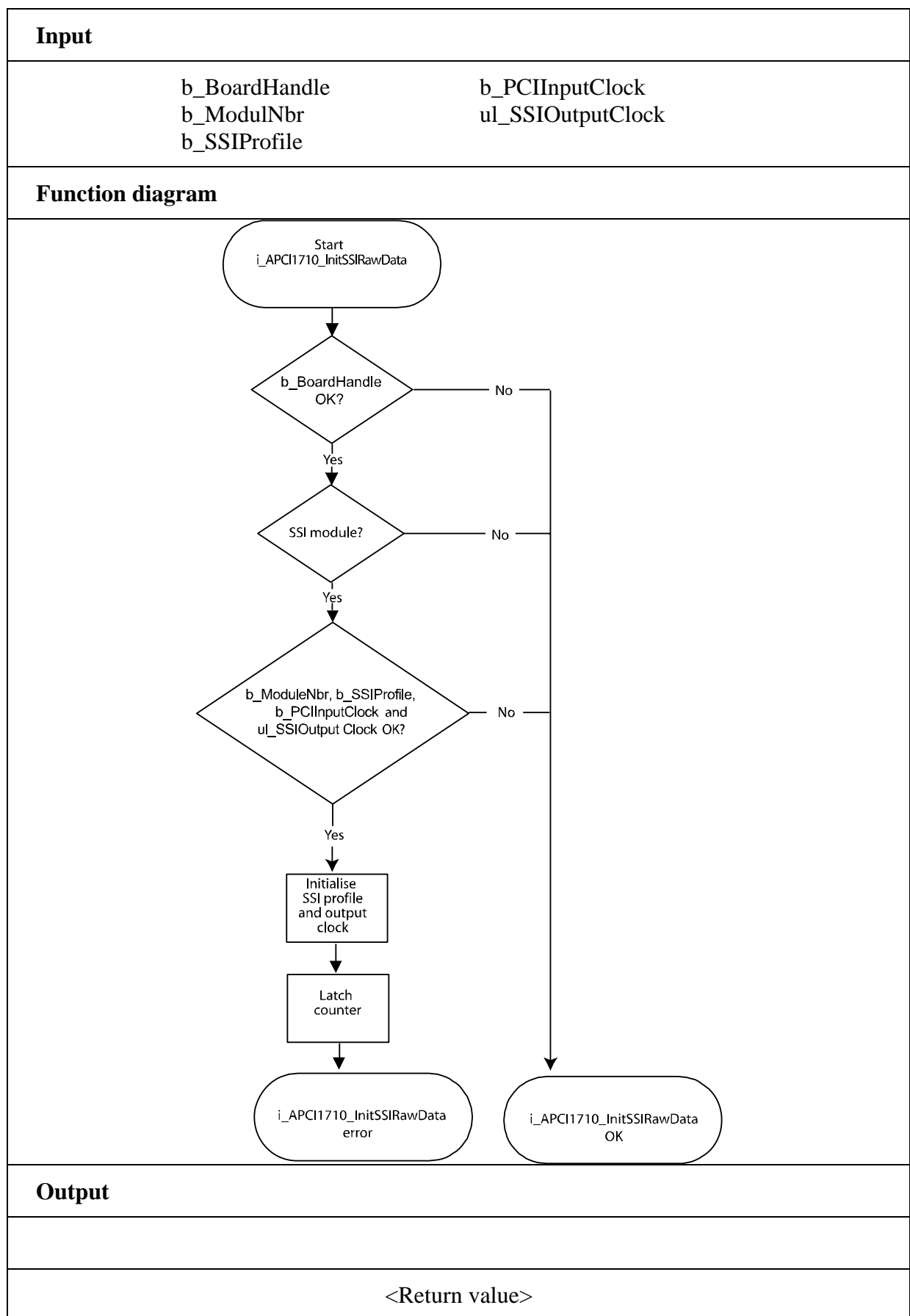
-3: The module is no SSI module

-4: The module does not support the profile length

-5: The selected SSI profile length is wrong

-6: The selected PCI input clock is wrong

-7: The selected SSI output clock is wrong



### 3.3 Read SSI

#### 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:

###### - Input:

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_SelectedSSI	Selection of the SSI counter (0 to 2)

###### - Output:

PULONG	pul_Position	SSI position in the turns
PULONG	pul_TurnCpt	Number of turns

##### Task:

Reads the SSI counter (b\_SelectedSSI) of the configured module (b\_ModulNbr).

##### Calling convention:

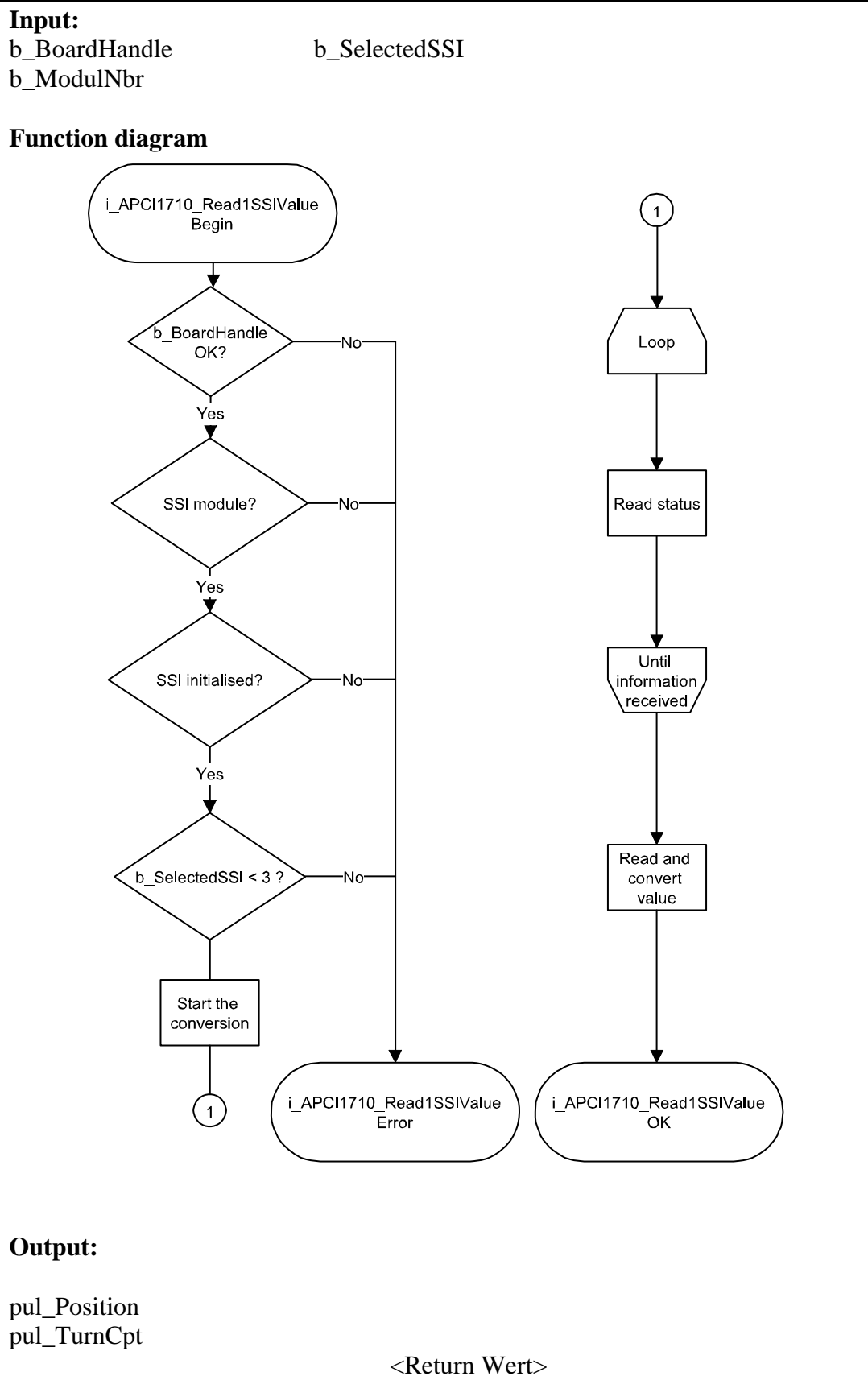
ANSI C:

```
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: No error
- 1: Handle parameter of the board is wrong.
- 2: The selected module is wrong.
- 3: The module is no SSI module.
- 4: SSI is not initialized. See function "i\_APCI1710\_InitSSI"
- 5: The selected SSI is wrong.



## 2) i\_APCI1710\_Read1SSIRawDataValue (...)

### Syntax:

```
<Return value> = i_APCI1710_Read1SSIRawDataValue
                                (BYTE    b_BoardHandle,
                                BYTE    b_ModulNbr,
                                BYTE    b_SelectedSSI,
                                PULONG  pul_ValueArray
                                BYTE    b_ValueArraySize)
```

### Parameter:

#### - Input:

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_SelectedSSI	Selection of the SSI counter (0 to 2)
BYTE	b_ValueArraySize	Size of pul_ValueArray in dword

#### - Output:

PULONG	pul_ValueArray	Array of raw data of SSI counter
--------	----------------	----------------------------------

### Task:

Reads the selected SSI counter (*b\_SelectedSSI*) of the selected module (*b\_ModulNbr*).

### Calling convention:

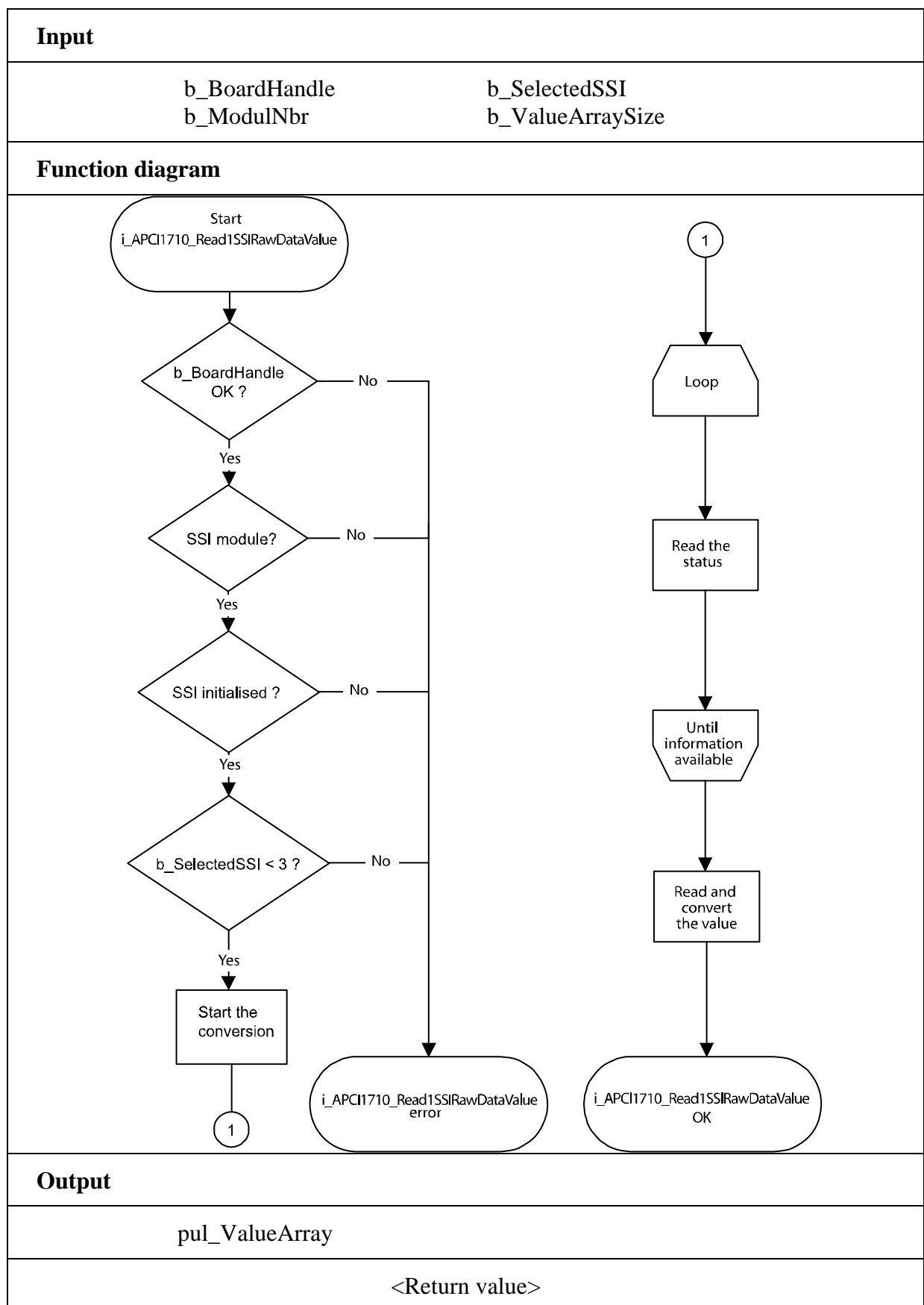
ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_ValueArray[2]
unsigned char b_ValueArraySize=2
```

```
i_ReturnValue = i_APCI1710_Read1SSIRawDataValue
                                (b_BoardHandle,
                                0,
                                0,
                                ul_ValueArray
                                b_ValueArraySize);
```

### Return value:

0: No error  
 -1: Handle parameter of the board is wrong  
 -2: Module parameter is wrong  
 -3: The module is no SSI module  
 -4: SSI is not initialised. See function "i\_APCI1710\_InitSSI"  
 -5: The selected SSI is wrong  
 -6: Parameter b\_ValueArraySize is wrong



### 3) i\_APCI1710\_ReadAllSSIValue (...)

**Syntax:**

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

**Parameter:**
**- Input:**

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

**- Output:**

PULONG	pul_Position	SSI position in the turns
PULONG	pul_TurnCpt	Number of turns

**Task:**

Reads all SSI counters of the selected module (*b\_ModulNbr*).

pul\_Position [0]: SSI Position of the SSI counter 0  
 pul\_Position [1]: SSI Position of the SSI counter 1  
 pul\_Position [2]: SSI Position of the SSI counter 2  
 pul\_TurnCpt [0]: Number of turns for SSI counter 0  
 pul\_TurnCpt [1]: Number of turns for SSI counter 1  
 pul\_TurnCpt [2]: Number of turns for SSI counter 2

**Calling convention:**

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);
```

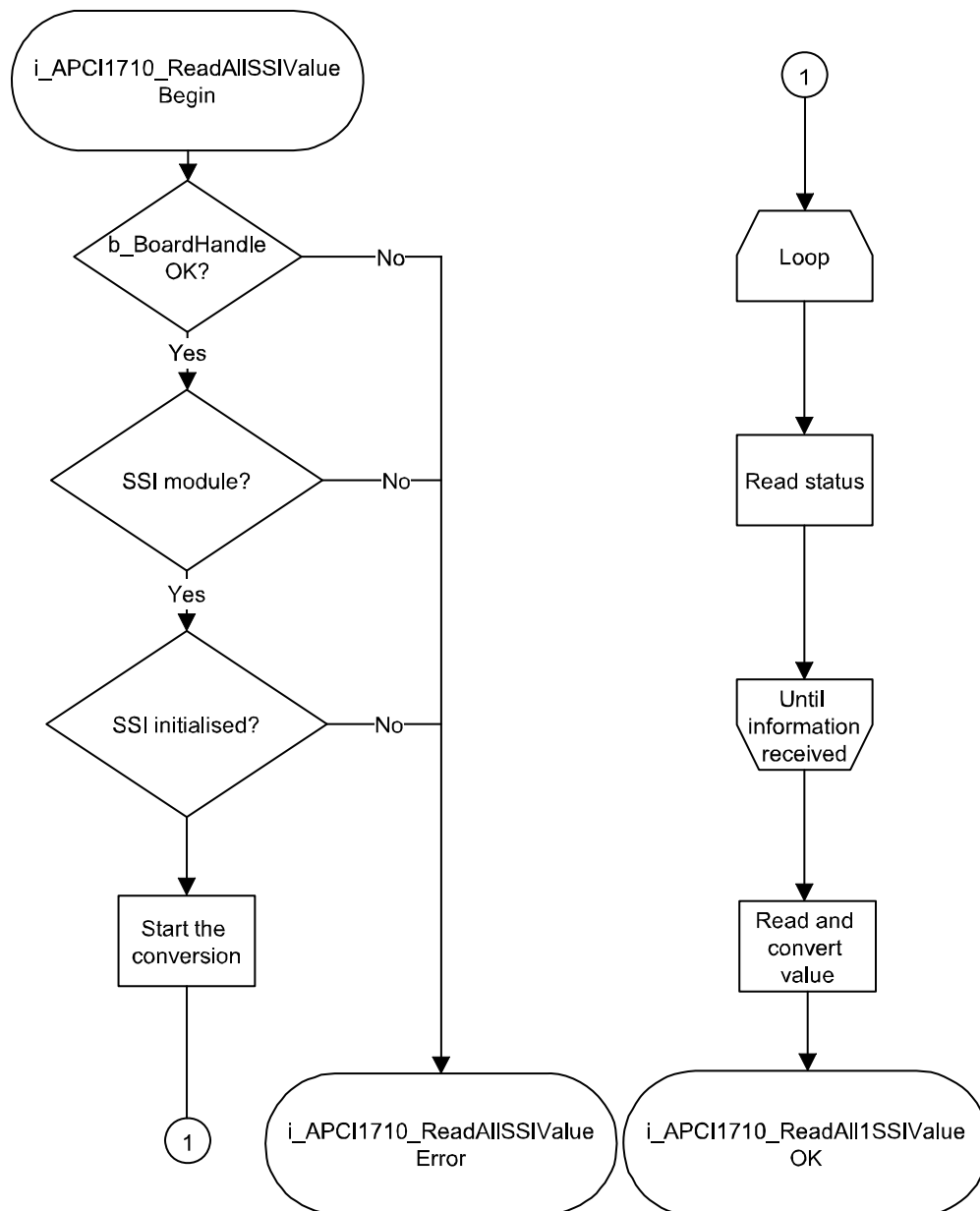
**Return value:**

0: No error  
 -1: Handle parameter of the board is wrong  
 -2: The selected module is wrong  
 -3: The module is no SSI module  
 -4: SSI is not initialized. See function "i\_APCI1710\_InitSSI"



**Input:**

b\_BoardHandle  
b\_ModulNbr

**Function diagram****Output:**

pul\_Position  
pul\_TurnCpt

<Return Value>

#### 4) i\_APCI1710\_ReadAllSSIRawDataValue (...)

**Syntax:**

```
<Return value> = i_APCI1710_ReadAllSSIRawDataValue
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     PULONG    pul_ValueArray,
                     BYTE      b_ValueArraySize)
```

**Parameter:**
**-Input:**

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_ValueArraySize	Size of pul_ValueArray in dword

**- Output:**

PULONG	pul_ValueArray	Array of raw data of the SSI counter
--------	----------------	--------------------------------------

**Task:**

Reads all SSI counters (*b\_SelectedSSI*) of the selected module (*b\_ModulNbr*).

**Calling convention:**

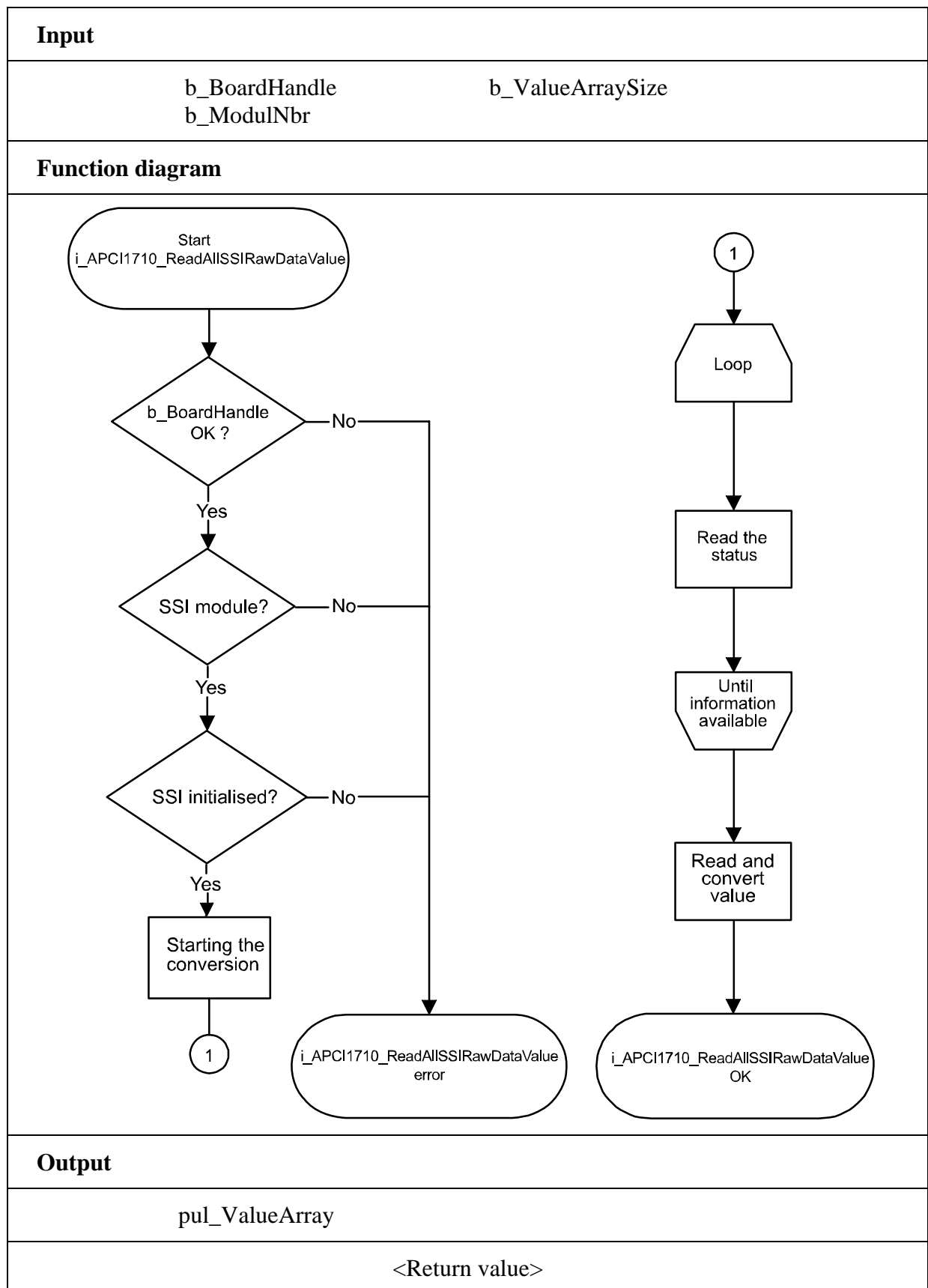
ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_ValueArray[6]
unsigned char b_ValueArraySize=6
```

```
i_ReturnValue = i_APCI1710_ReadAllSSIValue (b_BoardHandle,
                                             0,
                                             ul_ValueArray
                                             b_ValueArraySize);
```

**Return value:**

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: Module parameter is wrong
- 3: The module is no SSI module
- 4: SSI is not initialised. See function "i\_APCI1710\_InitSSI"
- 5: Parameter b\_ValueArraySize is wrong



### 3.4 Digital SSI inputs

#### 1) i\_APCI1710\_ReadSSI1DigitalInput (...)

##### Syntax:

```
<Return Wert> = i_APCI1710_ReadSSI1DigitalInput
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     BYTE      b_InputChannel,
                     PBYTE     pb_ChannelStatus)
```

##### Parameter:

###### - Input

BYTE	b_BoardHandle	Handle of the <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_InputChannel	Selection of the digital input (0 to 2).

###### - Ausgabe:

PBYTE	pb_ChannelStatus	Status of the digital input. 0: not activ 1: input is not active
-------	------------------	--

##### Task:

Returns the status of the selected digital input to the SSI module. Variable *b\_InputChannel* and module *b\_ModulNbr*.

##### Calling convention:

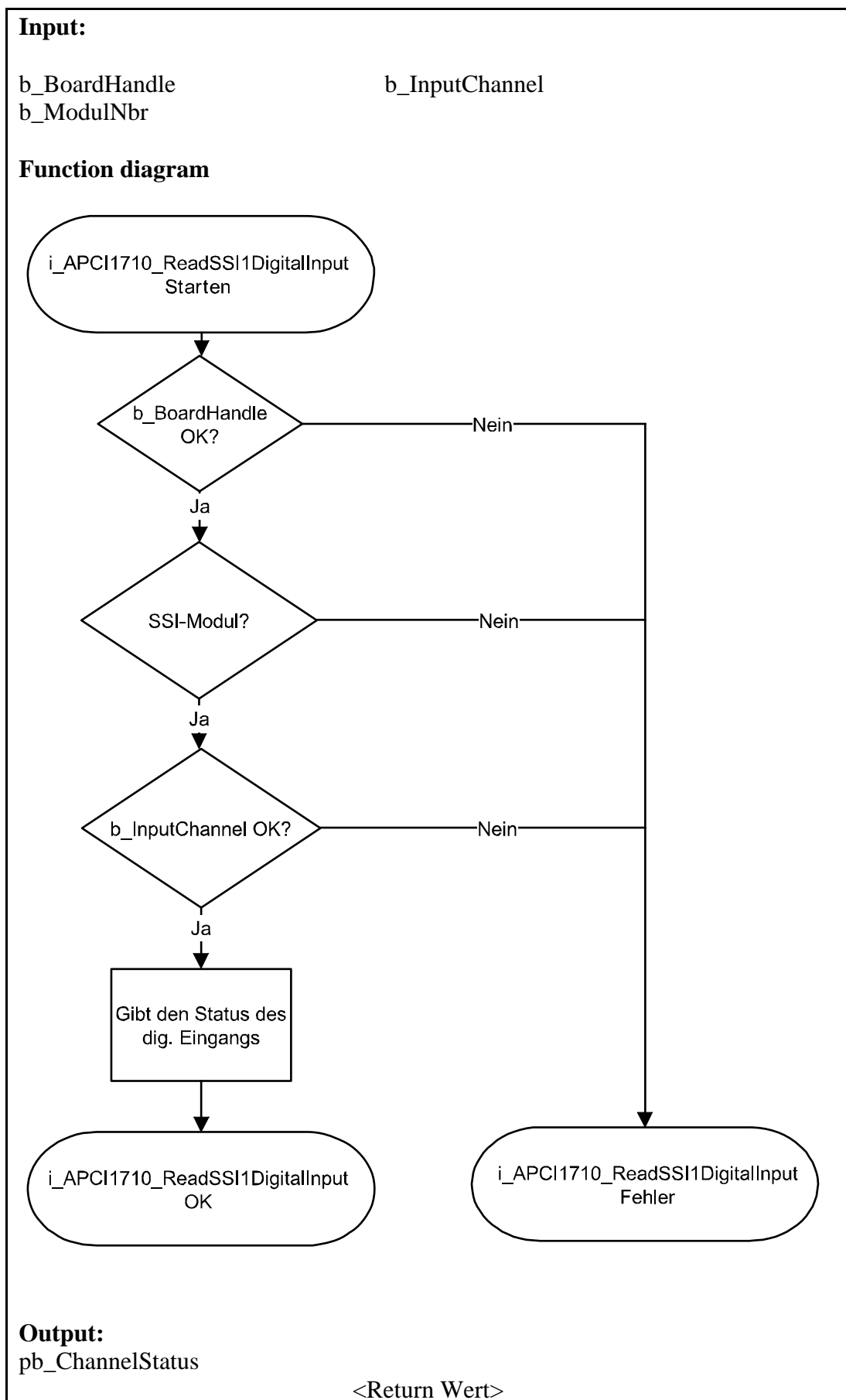
ANSI C:

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

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

##### Return value:

0: No error  
 -1: The handle parameter of the board is wrong  
 -2: The selected module is wrong  
 -3: The module is no SSI module  
 -4: Selection of the digital SSI input is wrong.



## 2) i\_APCI1710\_ReadSSIAIldigitalInput (...)

### Syntax:

```
<Return Wert> = i_APCI1710_ReadSSIAIldigitalInput
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     PBYTE     pb_InputStatus)
```

### Parameter:

#### - Input:

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

#### - Ouput

PBYTE	pb_InputStatus	Status of the digital inputs
-------	----------------	------------------------------

### Task:

Returns the status of all digital SSI inputs. (*b\_ModulNbr*).

D2	D1	D0
INPUT 2	INPUT 1	INPUT 0

0: Input is active

1: Input is not active

### Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadSSIAIldigitalInput
                (b_BoardHandle,
                 0,
                 &b_InputStatus);
```

### Return value:

0: No error

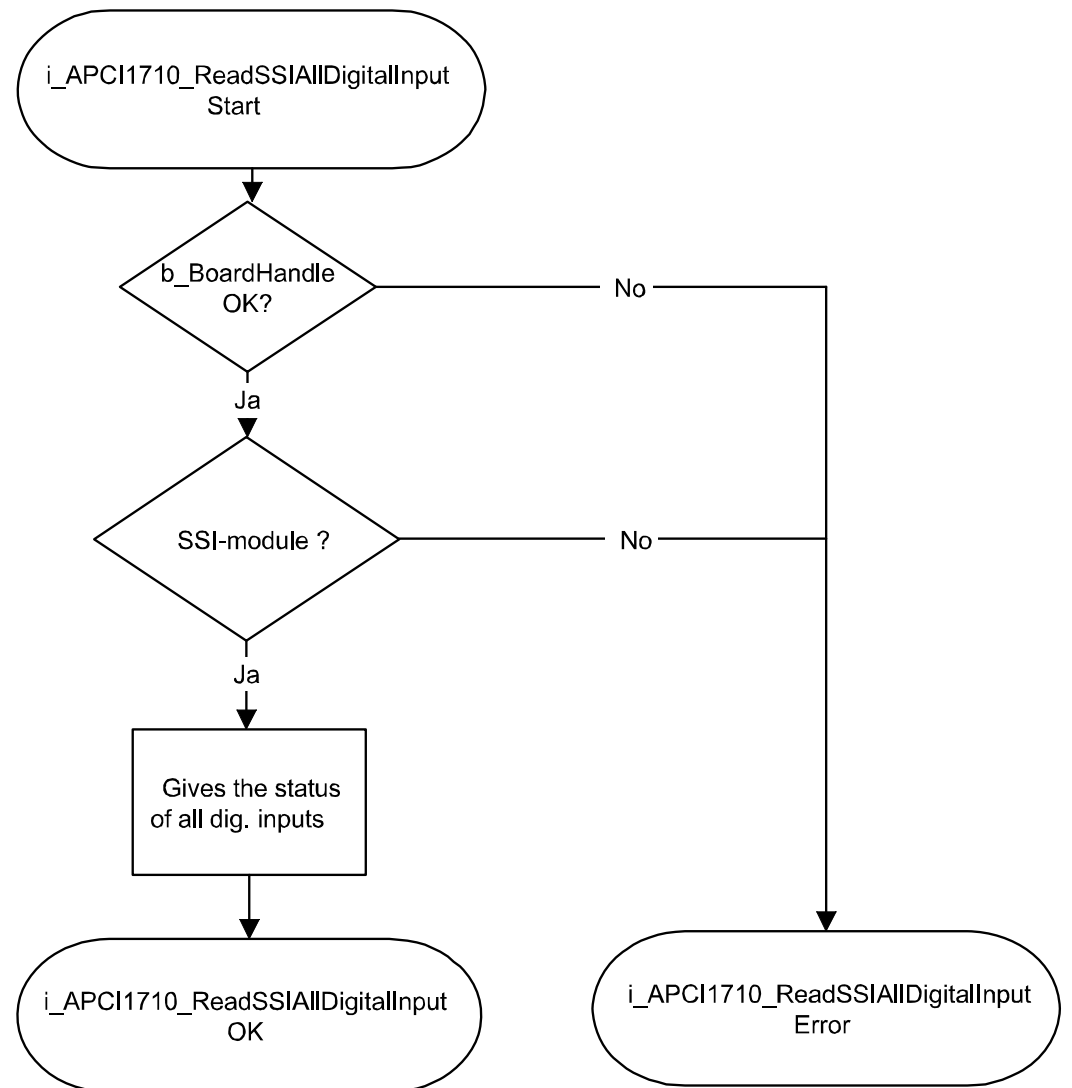
-1: The handle parameter of the board is wrong

-2: The selected module is wrong

-3: The module is no SSI module

**Input**

b\_BoardHandle  
b\_ModulNbr

**Function diagram****Output:**

<Return Value>

## 3.5 Digital SSI output

### 1) i\_APCI1710\_SetSSIDigitalOutputOn (...)

**Syntax:**

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

**Parameter:**

**- Eingabe:**

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

**- Output**

No output signal.

**Task:**

Switches on the digital output of the selected SSI module (b\_ModulNbr).

**Calling convention:**

ANSI C:

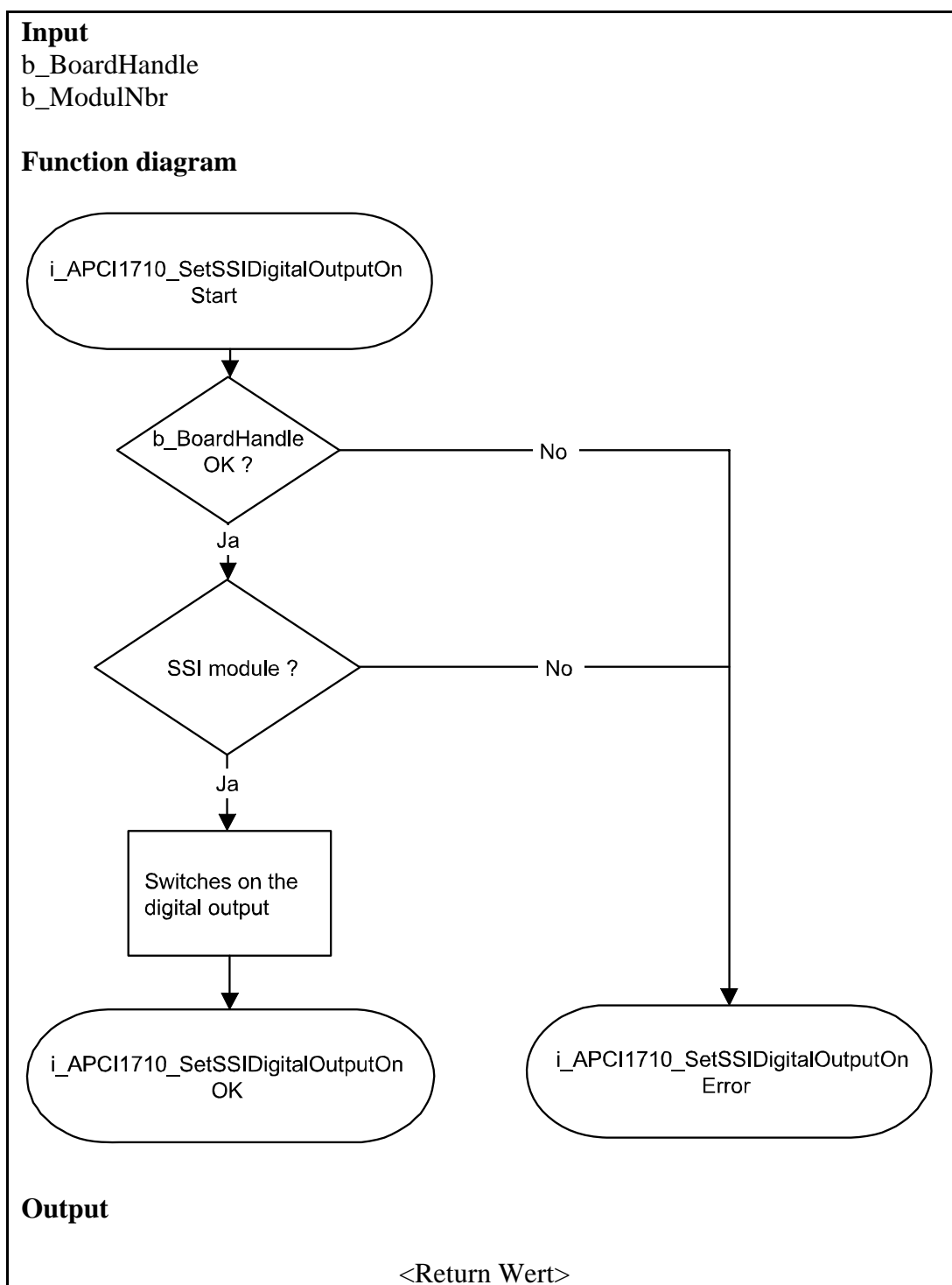
```
int          i_ReturnValue;
unsigned char b_BoardHandle;
```

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

**Return value:**

0: No error  
 -1: The handle parameter of the board is wrong  
 -2: The selected module is wrong.  
 -3: The module is no SSI module





## 2) i\_APCI1710\_SetSSIDigitalOutputOff (...)

### Syntax:

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

### Parameter:

#### - Input:

BYTE	b_BoardHandle	Handle of the board <b>APCI-/CPCI-1710</b>
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

#### - Output

No output signal has occurred

### Task:

Switches off the digital output of the selected SSI Modul (b\_ModulNbr).

### Calling convention:

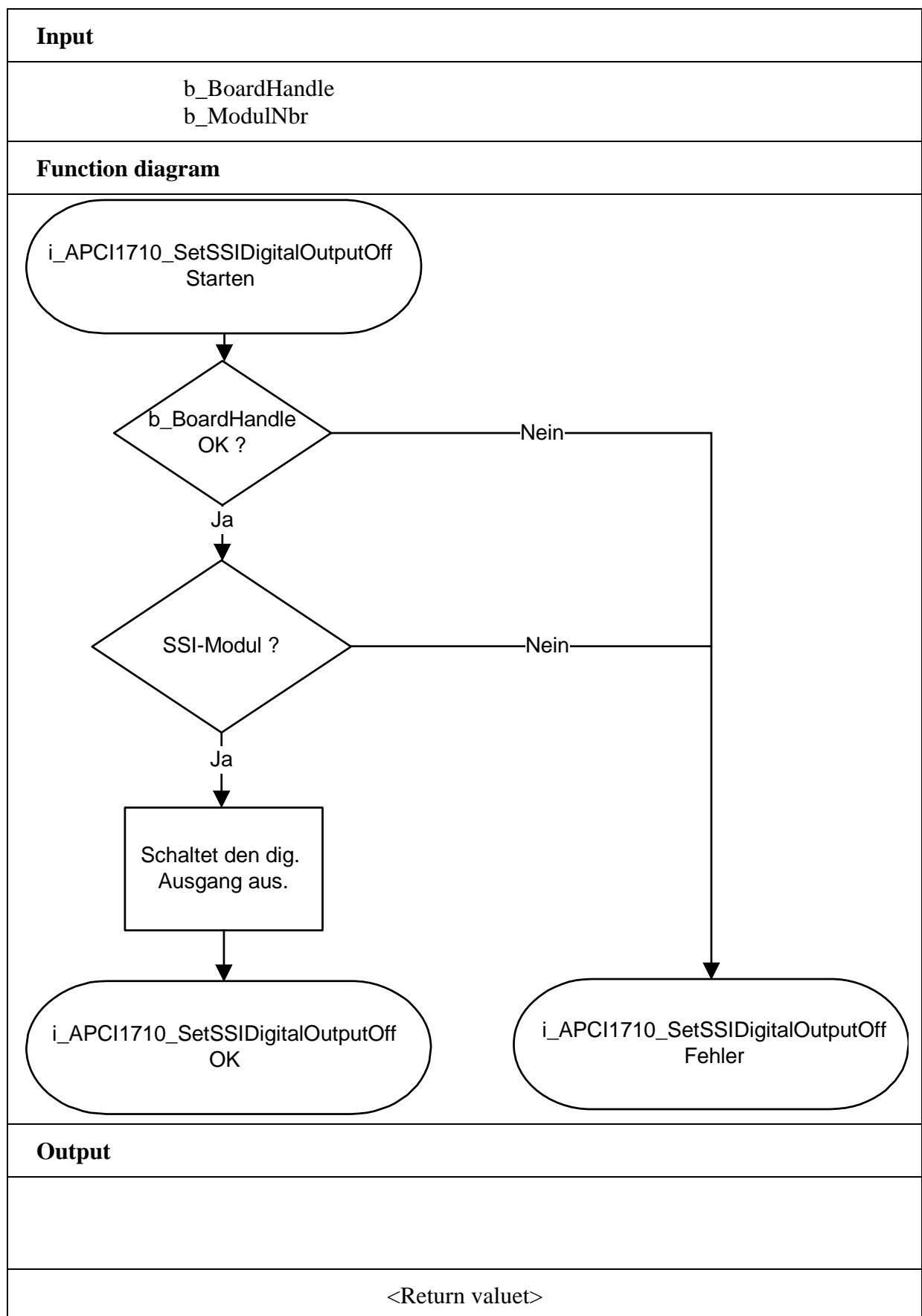
ANSI C:

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

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

### Return value:

0: No error  
-1: The handle parameter of the board is wrong  
-2: The selected module is wrong  
-3: The module is no SSI module



## 3.6 Functions in the kernel mode

**i**

### IMPORTANT!

These functions are only available for the user of the interrupt routine in Windows NT and Windows 95 in the synchronous mode. See function "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:

##### - Input:

UINT	ui_BaseAddress	Base address of the <b>APCI-1710</b> . See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_InputChannel	Selection of the digital input (0 to 2).

##### - Output:

PBYTE	pb_ChannelStatus	Status of the digital input. 0: Input is active. 1: Input is not active.
-------	------------------	--

##### Task:

Returns the status of the selected digital input (b\_InputChannel) of the SSI module (b\_ModulNbr).

##### Calling convention:

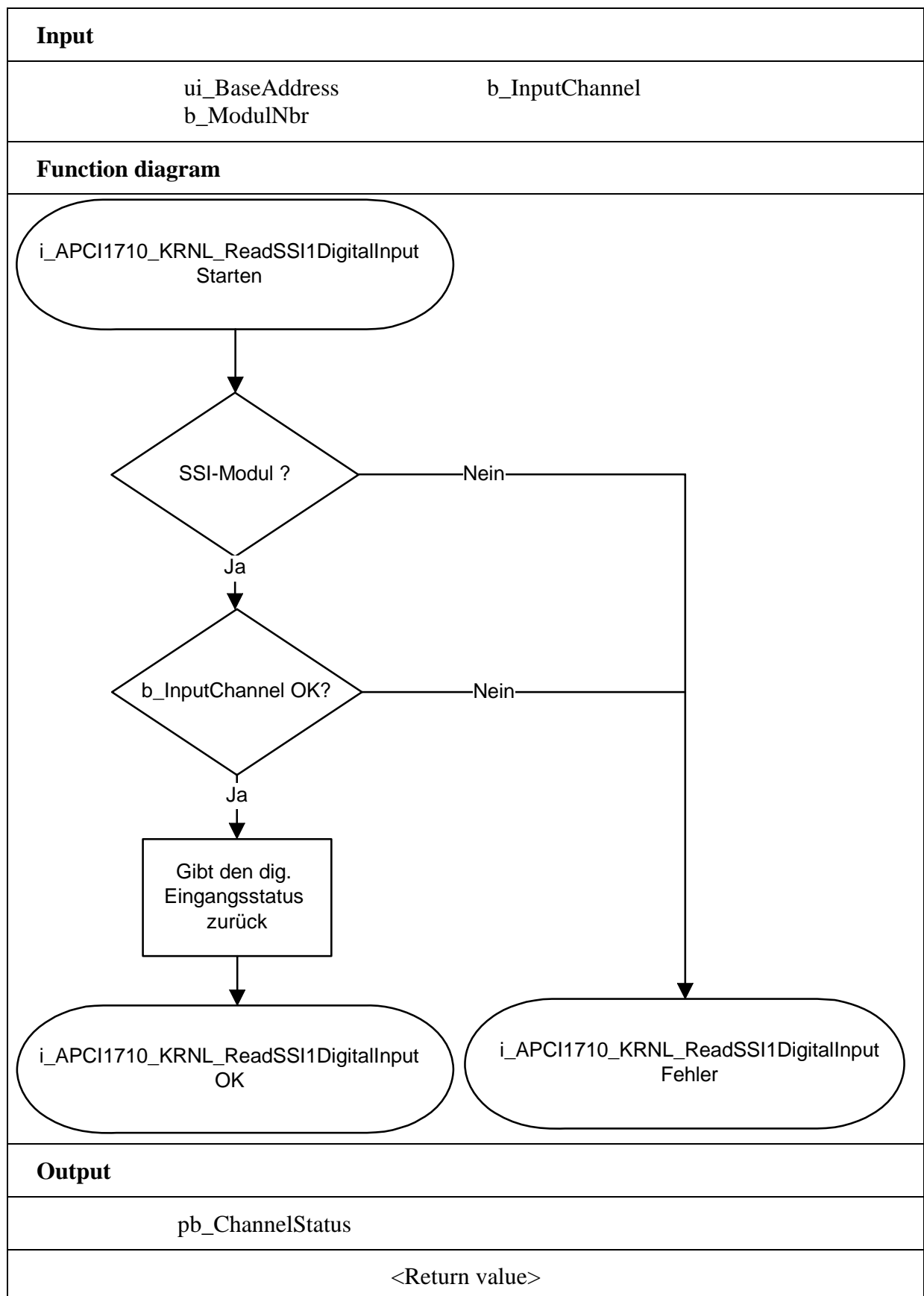
ANSI C:

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

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

##### Return value:

0: No error  
-1: The selected module is wrong  
-2: The module is no SSI module  
-3: Selection of the digital SSI input is wrong



## 2) i\_APCI1710\_KRNL\_ReadSSIAIldigitalInput (...)

### Syntax:

```
<Return Wert> = i_APCI1710_KRNL_ReadSSIAIldigitalInput
                    (UINT      ui_BaseAddress,
                     BYTE      b_ModulNbr,
                     PBYTE     pb_InputStatus)
```

### Parameter:

#### - Input:

UINT ui\_BaseAddress Base address of the **APCI-1710**. See "i\_APCI1710\_GetHardwareInformation"

BYTE b\_ModulNbr Number of the module to be configured (0 to 3)

#### - Ausgabe

PBYTE pb\_InputStatus Status of the digital inputs. See table 3-2.

### Task:

Returns the status of all digital SSI inputs. (*b\_ModulNbr*).

D2	D1	D0
INPUT 2	INPUT 1	INPUT 0

0 : Channel is not active.

1 : Channel is active.

### Calling convention:

#### ANSI C:

```
int i_ReturnValue;
unsigned int ui_BaseAddress;
unsigned char b_InputStatus;
```

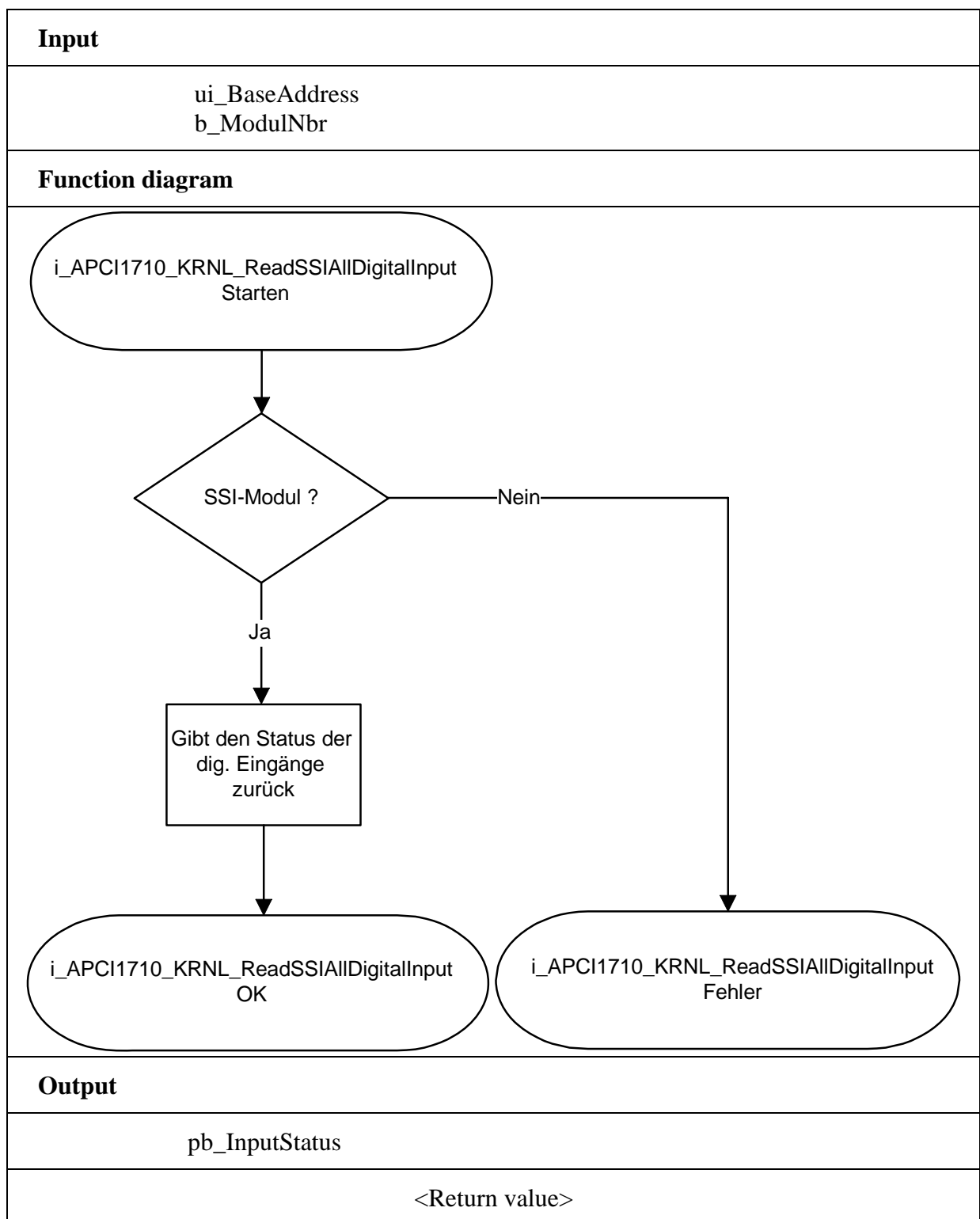
```
i_ReturnValue = i_APCI1710_KRNL_ReadSSIAIldigitalInput
                    (ui_BaseAddress,
                     0,
                     &b_InputStatus);
```

### Return value:

0: No error

-1: The selected module is wrong

-2: The module is no SSI module



**3) i\_APCI1710\_KRNL\_SetSSIDigitalOutputOn (...)****Syntax:**

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

**Parameter:****- Input:**

UINT	ui_BaseAddress	Base address of the <b>APCI-1710</b> . See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

**- Output:**

No output signal has occurred.

**Task:**

Switches on the digital SSI output of the module b\_ModulNbr

**Calling convention:**

ANSI C:

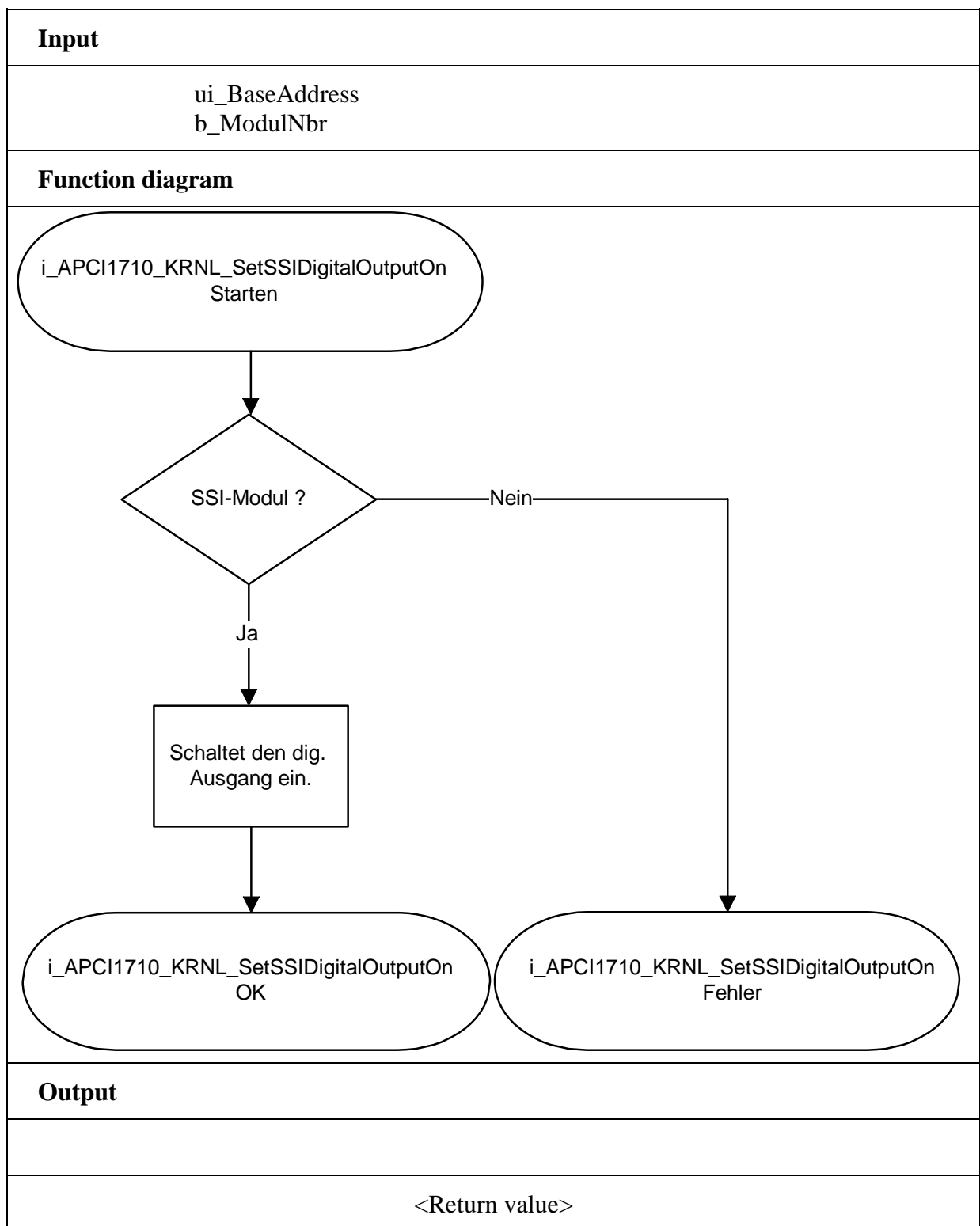
```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
```

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

**Return value:**

0: No error  
 -1: The selected module is wrong.  
 -2: The module is no SSI module





**4) i\_APCI1710\_KRNL\_SetSSIDigitalOutputOff (...)****Syntax:**

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

**Parameter:****- Input:**

UINT	ui_BaseAddress	Base address of the APCI-1710. See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

**- Output:**

No output signal has occurred.

**Task:**

Switches off the digital SSI input of the module b\_ModulNbr.

**Calling convention:**

ANSI C:

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

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

**Return value:**

0: No error  
 -1: The selected module is wrong  
 -2: The module is no SSI module

