



**DIN EN ISO 9001:2000
certified**



**ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER**



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

Function description

ADDICOUNT APCI-/CPCI-1710

SSI Monitor

2nd edition 03/2005

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.

1	DEFINITION OF APPLICATION	8
1.1	Intended use	8
1.2	Technical documentation	8
1.3	Function description	9
1.4	Abbreviations	9
1.5	Usage restrictions.....	9
2	SYNCHRONOUS SERIAL INTERFACE – MONITOR ..	10
2.1	SSI protocol	10
2.2	Technical data	11
2.2.1	Limit values:	11
2.2.2	Tolerances	11
2.3	Function description	12
2.3.1	General function.....	12
2.3.2	Monitoring mode	14
2.3.3	Optional settings	14
2.3.4	Block diagram of the SSI monitor	15
2.3.5	Typical applications	15
2.4	Used signals	16
2.5	Pin assignment for all modules with SSI monitor	17
2.6	Connection example	18
2.7	I/O assignment of the SSI monitor – interface	19
2.8	Description of the I/O functions.....	20
2.8.1	Function description	20
	General.....	20
2.8.2	READ REGISTER	20
	Status register (Base + 0).....	20
2.8.3	Write Register.....	23
3	STANDARD SOFTWARE	27
3.1	Define values.....	27
3.2	Interrupt mask	27
3.3	SSI monitor initialisation	29
	1) i_APCI1710_InitSSIMonitor (...).....	29
	2) i_APCI1710_InitSSIMonitorInterruptData (...)	30
	3) i_APCI1710_InitSSIMonitorInterruptError (...)	31
	4) i_APCI1710_InitSSIMonitorTriggerOutput (...).....	33
	5) i_APCI1710_InitSSIMonitorExternalGate (...).....	34
3.4	SSI monitor – Start/Stopp	35

1) i_APCI1710_StartSSIMonitor (...)	35
2) i_APCI1710_StopSSIMonitor (...)	36
3.5 SSI monitor reset.....	37
1) i_APCI1710_ResetSSIMonitorFIFO (...)	37
3.6 SSI monitor outputs.....	38
1) i_APCI1710_SetSSIMonitorDigitalOutputON (...)	38
2) i_APCI1710_SetSSIMonitorDigitalOutputOFF(...)	39
3.7 SSI monitor – read register	40
1) i_APCI1710_ReadSSIMonitorMeasuringStatus (...)	40
2) i_APCI1710_ReadSSIMonitorFIFOStatus (...)	42
3) i_APCI1710_ReadSSIMonitorAllDigitalInputs(...)	44
4) i_APCI1710_CheckSSIMonitor40MHzStatus (...)	45
5) i_APCI1710_ReadSSIMonitorMonoTime (...)	46
6) i_APCI1710_ReadSSIMonitorDelayTime (...)	47
7) i_APCI1710_ReadSSIMonitorFrequency (...)	48
8) i_APCI1710_ReadSSIMonitorData (...)	49
9) i_APCI1710_ReadSSIMonitorErrorInterruptStatus (...)	50
10) i_APCI1710_ReadSSIMonitorRestFIFOData (...)	52
4 TUTORIAL.....	53
4.1 Reading typical parameters	53
4.1.1 Step 1: Initialising the board	55
4.1.2 Step 2: Initialising the SSI monitor	55
4.1.3 Step 3: Starting the measurement.....	55
4.1.4 Step 4: Checking the status of the measurement	55
4.1.5 Step 5: Reading the results of the measurements.....	56
4.1.6 Step 6: Stopping the measurement	56
4.1.7 Step 7: Releasing the handle of the board	56
4.2 Reading all sensor data.....	57
4.2.1 Step 1: Initialising the board	59
4.2.2 Step 2: Initialising the interrupt routine.....	59
4.2.3 Step 3: Initialising the SSI monitor	59
4.2.4 Step 4: Initialising the data interrupt.....	60
4.2.5 Step 5: Starting the measurement.....	60
4.2.6 Step 6: Interrupt	60
4.2.7 Step 7: Stopping the measurement	60
4.2.8 Step 8: Reset board interrupt routine	61
4.2.9 Step 9: Releasing the handle of the board	61
4.3 Monitoring the control.....	62
4.3.1 Step 1: Initialising the board	64
4.3.2 Step 2: Initialising the interrupt routine.....	64
4.3.3 Step 3: Initialising the SSI monitor	64
4.3.4 Step 4: Initialising the error interrupt.....	65
4.3.5 Step 5: Starting the measurement.....	65

4.3.6	Step 6: Interrupt → Stopping the measurement	65
4.3.7	Step 7: Reading the interrupt status	65
4.3.8	Step 8: Reset board interrupt routine.....	65
4.3.9	Step 9: Releasing the handle of the board.....	66

Figures

Fig. 2-1: SSI diagram	10
Fig. 2-2: SSI diagram	10
Fig. 2-3: Function SSI monitor	12
Fig. 2-4: SSI data acquisition	13
Fig. 2-5: SSI data storage	13
Fig. 2-6: SSI parameter	14
Fig. 2-7: Block diagram of the SSI monitor	15
Fig. 2-8: Pin assignment of the 50-pin SUB-D connector X1	17
Fig. 2-9: Monitoring of an encoder (TWK CRE 58)	18

Tables

Table 1-1: Delivered manuals	8
Table 2-1: Used signals	16
Table 2-2: I/O assignment of the SSI monitor – interface	19
Table 2-3: Status register	20
Table 2-4: Output register	23
Table 2-5: Interrupt status register	23
Table 2-6: Reset register	24
Table 2-7: Initialisation register	24
Table 2-8: Output register	25
Table 2-9: Reset interrupt register	25
Table 3-1: Define value	27
Table 3-2: Interrupt mask of the functions „SSI monitor“	27

1 DEFINITION OF APPLICATION

1.1 Intended use

The **APCI-1710/CPCI-1710** boards must be inserted in a PC with PCI 5V/32-bit slots or Compact PCI/PXI PC with COMPACT PCI 5V/32 bit slots which is used as electrical equipment for measurement, control and laboratory pursuant to the norm IEC 61010-1, edition August 2002.

1.2 Technical documentation

The present technical description refers to the board **APCI-1710** and to the board **CPCI-1710/1711**. Please check that you also received the following:

- CD1 "Standard Software Drivers" with the ADDISET parameterizing program and the required software drivers.
- CD2 "Technical Manuals". This CD consists of
 - The manual **ADDICOUNT APCI-/CPCI-1710: Function programmable counter board for the PCI bus**, which contains general information about the operation of the board.
 - A manual or each function that you want to program on APCI-/CPCI-1710
- Yellow leaflet with safety precautions.

According to the used function, you will find the required pin and programming information in the respecting manuals.

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-1710/1711** is software compatible with the board **APCI-1710**. The programs ADDIREG and SET1710 do not make a difference between PCI boards and CompactPCI boards.

The API functions of the standard software are also identical.

1.3 Function description

The present manual contains besides a common description of the functions

- The pin assignment of the front connector
- A list of the used signals
- The I/O range
- A chapter with the supplied API functions of the standard software.
- Tutorials

1.4 Abbreviations

The signals on the 50 pin SUB-D connector refer all to one function module. The following abbreviations are used:

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

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

1.5 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.

2 SYNCHRONOUS SERIAL INTERFACE – MONITOR

2.1 SSI protocol

An external control generates a certain clock sequence for the control of the angle encoder. Then the numbering of required clocks and the clock break are generated.

The angle encoder returns the data serially to the control. Through the clock breaks the angle encoder is triggered and updates its positioning value.

The most important characteristics of the SSI transfer:

Fig. 2-1: SSI diagram

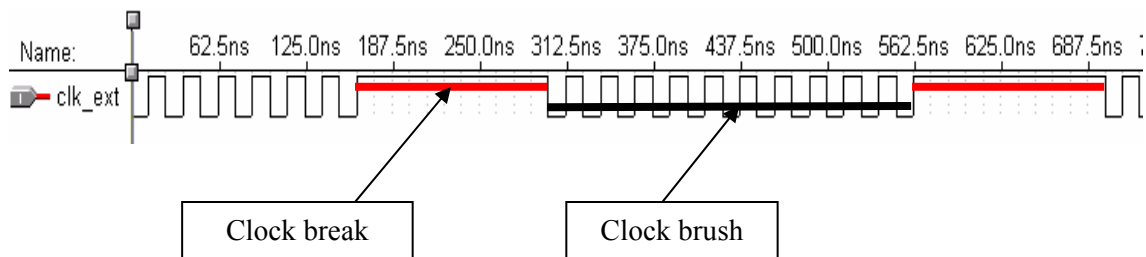
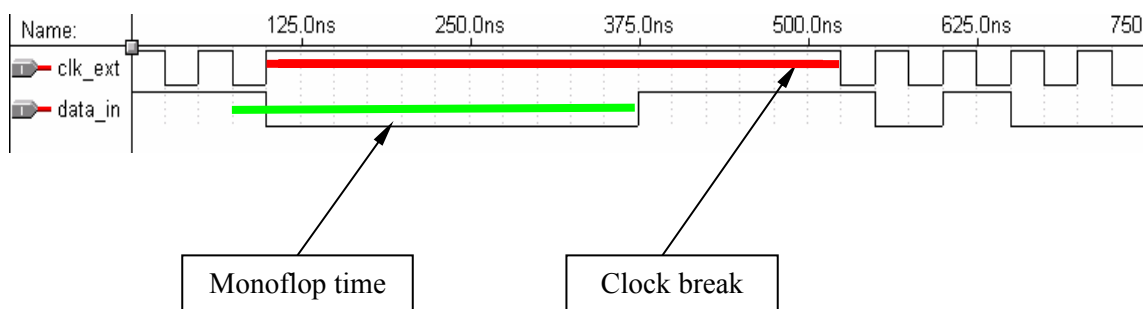


Fig. 2-2: SSI diagram



Further information you can obtain from the manufacturers of angle encoders, for example: www.stegmann.de

2.2 Technical data

2.2.1 Limit values:

The SSI monitor operates only with the updated APCI-/CPCI-1710 boards that contain a 40 MHz quartz.

The external control and the angle encoders must have the following values:

Number of bits of the angle encoder:	2-36
Number of bits of the control:	2-64
Clock break:	7.6 μ s – 1.66 s
Monoflop time:	15 μ s – 100 μ s
Frequency:	70 kHz – 1.5 MHz
Min. level length for the external start:	1 μ s

2.2.2 Tolerances

Max. error through input driver:

Monoflop time:	\pm 500 ns
Duration of period:	\pm 500 ns
Clock break	\pm 500 ns

Typical error through input driver:

Monoflop time:	\pm 60 ns
Duration of period:	\pm 60 ns
Clock break:	\pm 60 ns

Max. error through reference clock:

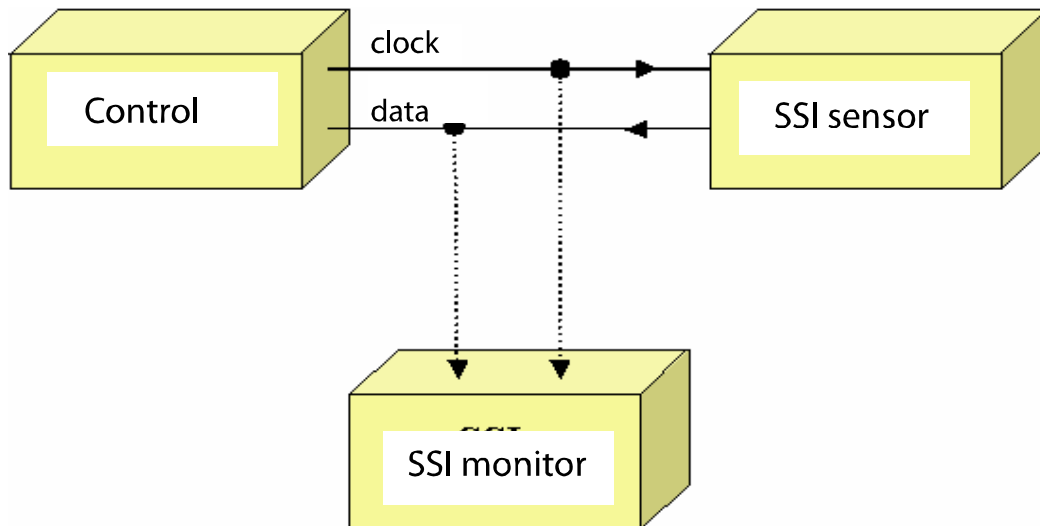
Monoflop time:	\pm 100 ppm
Duration of period:	\pm 100 ppm
Clock break:	\pm 100 ppm

The error of 100 ppm occurs because of the component tolerance of the 40 MHz quartz that is used for time measurement.

2.3 Function description

2.3.1 General function

Fig. 2-3: Function SSI monitor



The function SSI monitor is a tool for the data acquisition and the control of an angle encoder. The angle encoder, which is to be monitored, can be operated with any type of controls (for example PC or SPS) as long as the valid SSI transfer protocol as well as the defined limit values of the monitor module are kept.

In order to monitor the angle encoder, the clock line and the data line must be connected with the APCI-/CPCI-1710 or with the monitor.

Before every measurement, the user must do an initialisation through the software function. Here the user must define the bit length of the angle encoder and the number of the function module of the board.

A firmly saved reference value of 7.5 μs is used in order to look for a clock break. With each rising flank of the clock line the measurement is started. With the next negative flank the measurement is stopped. If the measured time exceeds the reference value (7.5 μs), this is a clock break. The measurement requires that the frequency of the clock brush is in a certain range. When this frequency is not reached, each clock would be identified as a clock break.

When with this procedure a clock break was found, the following bits of the data line is read and stored in the FIFO. The number of bits will be entered at initialisation (number of bits of the angle encoder). The saving of data is done in the same sequence as the reading. The bits are stored without any empty bits in the FIFO, without any empty bits. Then the next clock break is waited for. There is a function available for reading out the data, with which 32 bit values can be read. After reading the FIFO with this function, there can be max. 31-bit data in the memory. This remaining data is be read with another function.

Fig. 2-4: SSI data acquisition

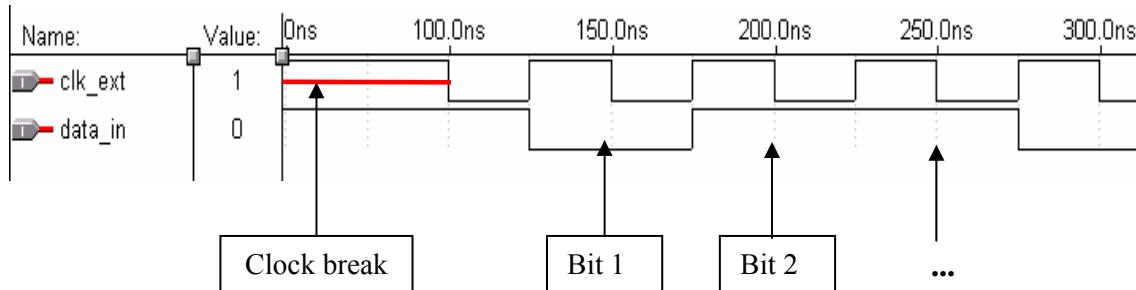
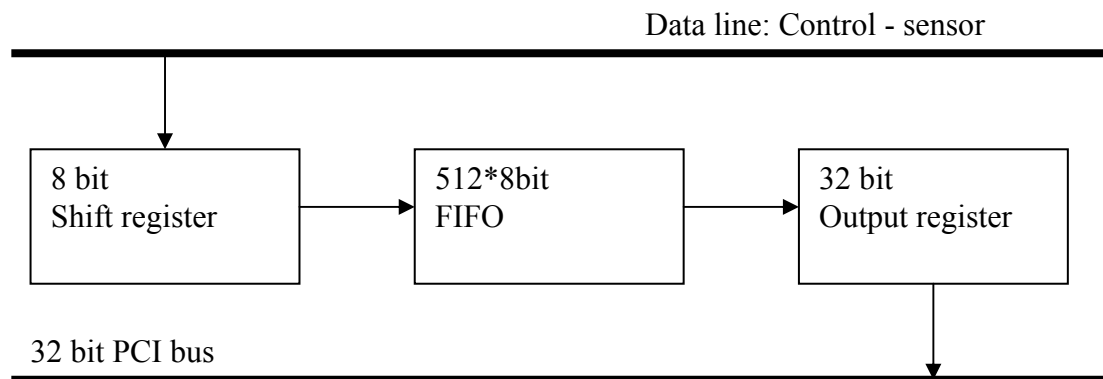
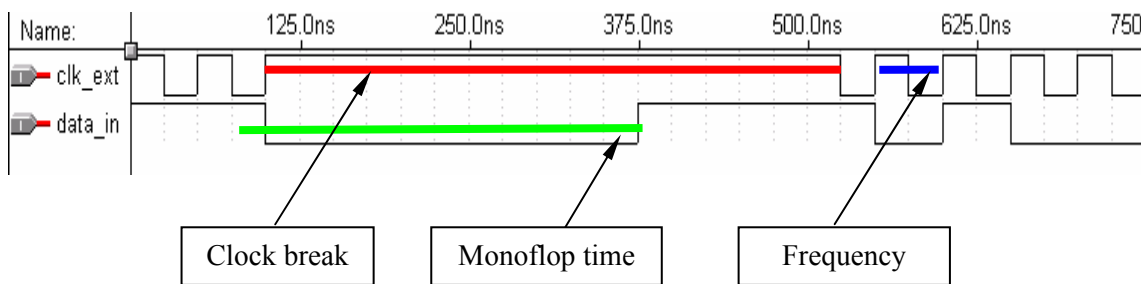


Fig. 2-5: SSI data storage



The clock break, the monoflop time and the frequency are measured and saved in a register after each clock brush. The registers are updated permanently, whereas the overwritten value is not saved. Herewith only current values can be read out. For the measurement of the frequency, the first clock after the clock break will be measured. So the frequency measurement is merely a sampling measurement and does not serve for the evaluation of the whole clock brush.

The monoflop time measurement ends usually with a rising edge of the data line. In the case of an error the level of the clock line goes to "0" before the level of the data line sets to "1". In this case the clock break would be too short and the monoflop time measurement would be also stopped and the value saved.

Fig. 2-6: SSI parameter

The 40 MHz clock, which is generated by the quartz on the board, is used for all time measurements.

2.3.2 Monitoring mode

As option there is a monitoring or ERROR mode available. This mode can be used only as interrupt and is used to identify and to signalise possible control errors.

During initialisation of this mode the min. required clock break and the number of bits are entered. If too many or not enough clocks are generated within a clock brush or if the clock break is not kept, the measurement is stopped and an interrupt occurs. The min required clock break can be max. 102 μ s.

2.3.3 Optional settings

- External start possibility through a pulse at input 1 or input 2 of the board
- Trigger pulse (1 μ) at the beginning of each clock brush at output 1
- Manual switching of output 1
- Manual switching of output 2
- Interrupt at FIFO full or half full
- Interrupt at error of the control (monitoring mode)
- Trigger signal at output 1 at error of the control (monitoring mode)

Caution:

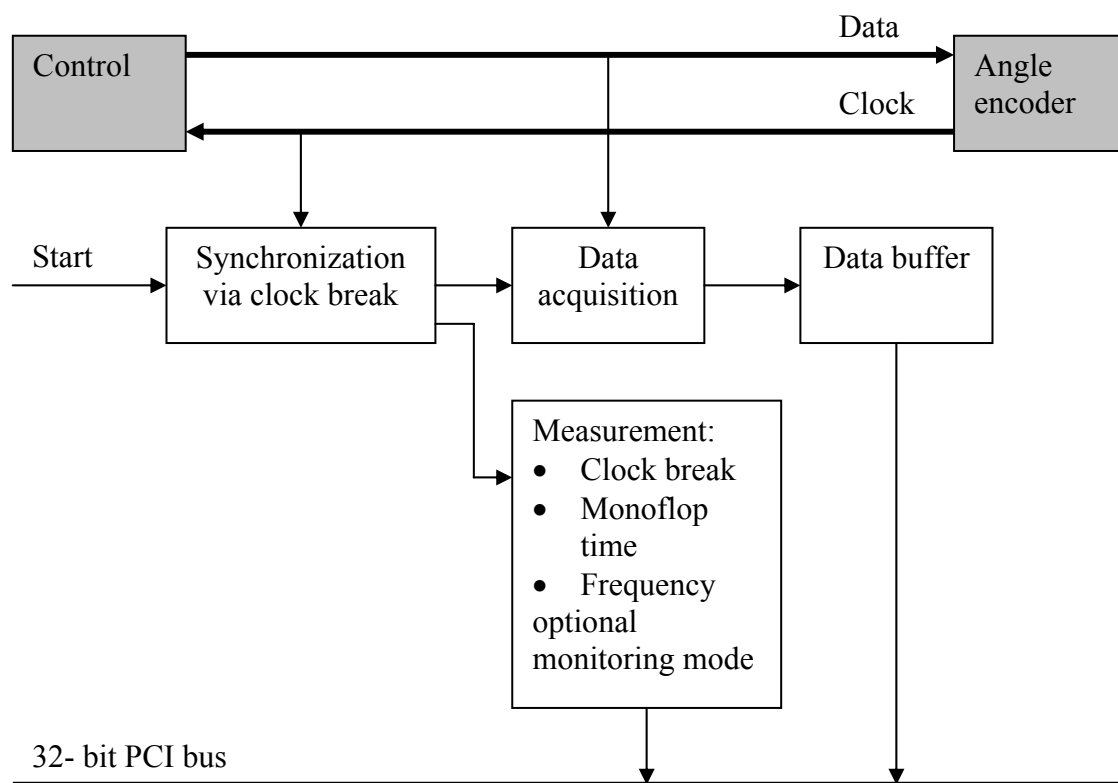
Sometimes the same input or output is used for different functions of the board. Therefore each input or output shall be initialised always only in one functions in order to avoid signal overlappings.

2.3.4 Block diagram of the SSI monitor

The interface contains:

- One module in order to find the clock break and to measure it
- One module in order to measure the monoflop time
- One module in order to measure the frequency of the clock line
- One 512 * 8 bit FIFO for data storage
- One module for the monitoring of the control
- Function and control logic

Fig. 2-7: Block diagram of the SSI monitor



2.3.5 Typical applications

- Acquisition of serial data
- Measurement of the following parameters:
 1. Clock break of the control
 2. Monoflop time of the angle encoder
 3. Frequency of the control
- Monitoring of the control

2.4 Used signals

The function SSI monitor uses **6 inputs (channels A and C to G) and 2 outputs** (channel B and H) of the respecting function module of the **APCI-/CPCI-1710**.

On one board you can monitor max. 4 absolute encoders or one absolute encoder per module:

Table 2-1: Used signals

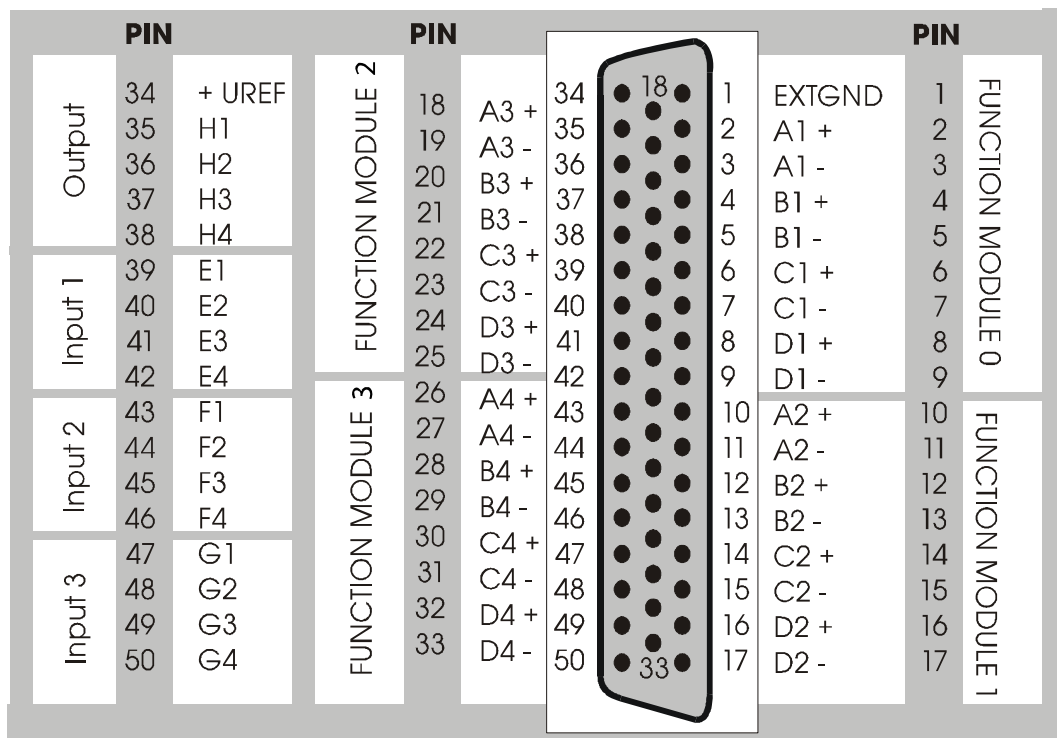
SIGNALS	AT THE CONNECTOR	POLARITY	FUNCTION
Input1_x	Ax +/-	Diff.	Digital input 1 (optional: external start)
Output1_x	Bx +/-	Diff. / OPT. 24Vdiff.	Digital output 1 (optional: Trigger output or error output in the monitoring mode)
Clock_x	Cx + / -	Diff. / OPT. 24Vdiff.	Clock line of the angle encoder
Data_x	Dx + / -	Diff. / OPT. 24Vdiff.	Data line of the angle encoder
Input2_x	Ex +/-	24V / OPT. 5V	Digital input 2 (optional: External start)
Input3_x	Fx +/-	24V / OPT. 5V	Digital input 3
Input4_x	Gx +/-	24V / OPT. 5V	Digital input 4
Output	Hx +/-	24V / OPT. 5V	Digital output 2

X: Number of the function module.

2.5 Pin assignment for all modules with SSI monitor

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

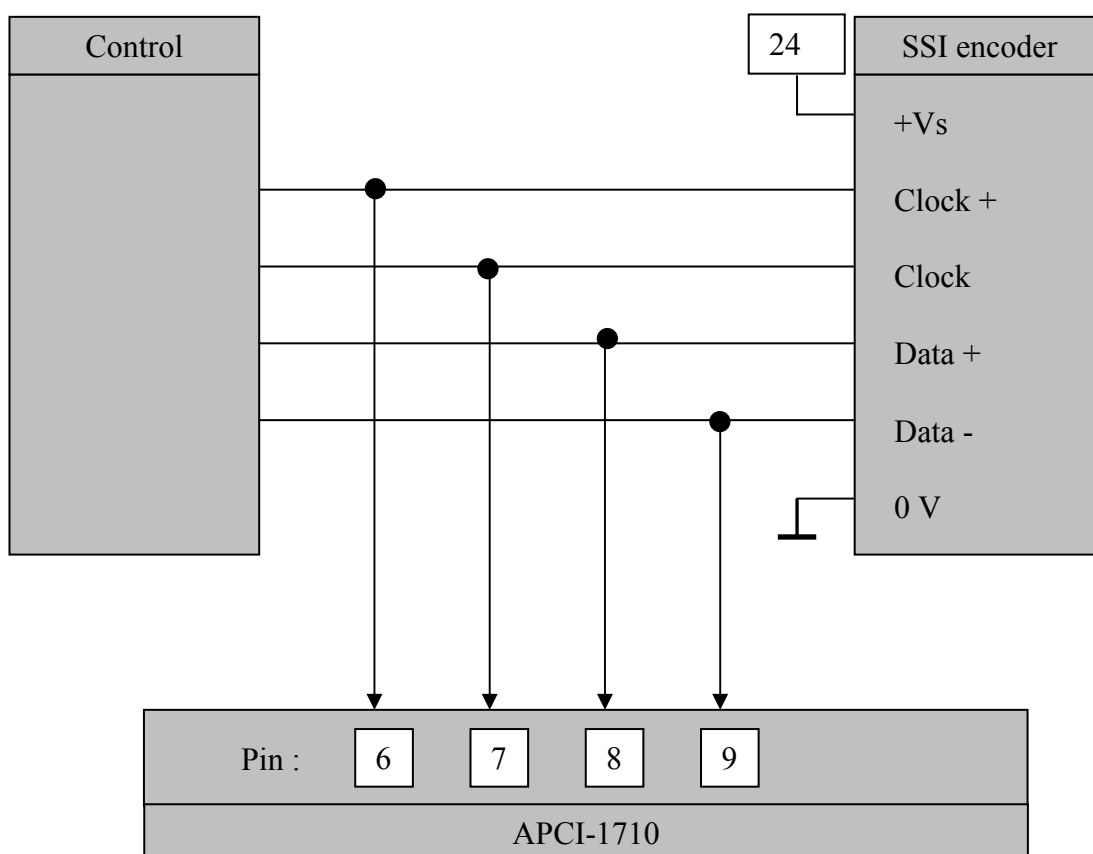
Fig. 2-8: Pin assignment of the 50-pin SUB-D connector X1



2.6 Connection example

The encoder CRE 58 of TWK is connected to an external control and is monitored with a SSI monitor in function module No. 1 of the APCI-/CPCI-1710.

Fig. 2-9: Monitoring of an encoder (TWK CRE 58)



2.7 I/O assignment of the SSI monitor – interface

Table 2-2: I/O assignment of the SSI monitor – interface

	IORD	IOWR
BASEx + 0	D12..D0 Status register	D5..D0 Number of bits of the sensor D12..D6 Number of bits of the control
BASEx + 4	D11..D0 Monoflop time minus neg. level D20..D12 neg. level before clock break D21 Counter overflow	D8..D0 min. time in order to look for clock breaks D27..D16 min. clock break for error mode
BASEx + 8	D25..D0 Clock break D26 Counter overflow	Start measurement
BASEx + 12	D9...D0 Frequency D10 Counter overflow	Stop measurement
BASEx + 16	D8..D0 Filling status of the FIFO D9 Output register D13..D10 Filling status of the shift register before FIFO	D1..D0 Reset
BASEx + 20	D31..D0 32 bit FIFO data	Shift pulse for shift register
BASEx + 28	D11..D0 Initialisation register	D11..D0 Initialisation register
BASEx + 32	D1..D0 Output register	D1..D0 Output register
BASEx + 36	D4..D0 Interrupt status	D4..D0 Interrupt reset
BASEx + 60	D31..D0 Module identification	-

-: No function: **x**: Number of the function module.
The accesses are always read or written in 32 bit range

2.8 Description of the I/O functions

2.8.1 Function description

General

All time measurements of the SSI monitor are realised with the 40 MHz clock of the APCI-/CPCI-1710 board. The counter status of a time measurement indicates the number of 40 MHz clocks (25 ns) within the measured time window.

Please consider the tolerances of the different measurements shown in chapter 2.3.

2.8.2 READ REGISTER

Status register (Base +0)

Table 2-3: Status register

BIT D0	0	Monoflop time measurement not ended
	1	Monoflop time measurement ended
BIT D1	0	Clock break measurement not ended
	1	Clock break measurement ended
BIT D2	0	Frequency measurement not ended
	1	Frequency measurement ended
BIT D3	0	Less than 32 bit data in the buffer
	1	At least 32 bit data in the buffer
BIT D4	0	FIFO empty
	1	FIFO not empty
BIT D5	0	FIFO less than half filled
	1	FIFO at least half filled
BIT D6	0	FIFO not filled
	1	FIFO filled
BIT D7	0	Low level at input 1
	1	High level at input 1
BIT D8	0	Low level at input 2
	1	High level at input 2
BIT D9	0	Low level at input 3
	1	High level at input 3

BIT D10	0	Low level at input 4
	1	High level at input 4
BIT D11	0	40 MHz quartz not available
	1	40 MHz quartz not available
BIT D12	0	Measurement not active
	1	Measurement active

Monoflop time - register (Base + 4)

Reading the base address + 4 returns the monoflop time.

Only recommended, when bit 0 (measurement ended) is set in the status register.

Bit 0-11: Monoflop time minus the negative level before the clock break
 Bit 12-20: Negative level before the clock break
 Bit 21: Indicates if a counter overflow occurred during the measurement
 0 : No overflow
 1 : Overflow

In order to obtain the value of the monoflop time, the vale (bit 0-11) must be added to the value (bit 12-20). The result shows the number of system clocks (40 MHz) within the measured time window.

→ Monoflop time = Sum * 25 ns

If a counter overflow occurred during the measurement, then the data (bit 0-20) are invalid. The measurement is not ended automatically when a counter overflow occurred.

Firstly at the next start of a measurement bit 21 will be reset.

Clock break – register (Base + 8)

Reading the base address + 8 returns the clock break.

Only recommended, when bit 1 (measurement ended) is set in the status register.

Bit 0-25: Clock break of the control
 Bit 26: Indicates if a counter overflow occurred during the measurement.
 0 - No overflow
 1 - Overflow

The obtained value (bit 0-25) indicates the number of system clocks (40 MHz) within the measured time window.

→ Clock break = Value (bit 0-25) * 25 ns

If a counter overflow occurred during the measurement, then the data (bit 0-25) are invalid. The measurement is not ended automatically when a counter overflow occurred.

Bit 26 will be reset firstly at the next start of a measurement.

Frequency register (Base + 12)

Reading the base address + 12 returns the period duration of the sensor clock.
Only recommended, when bit 2 (measurement ended) is set in the status register.

Bit 0-9:	Period duration of a clock
Bit 10:	Indicates if a counter overflow occurred during the measurement.
0	- No overflow
1	- Overflow

The obtained value (bit 0-9) shows the number of system clocks (40 MHz) within the measured time window.

→ Period duration = Value (bit 0-9) * 25 ns

→ Frequency = 1 / period duration

If a counter overflow occurred during the measurement, then the data (bit 0-9) are invalid. The measurement is not ended automatically when a counter overflow occurred.

Bit 10 will be reset firstly at the next measurement.

FIFO_value – register (Base + 16)

Reading on the base address + 16 reads out the filling status of the buffer.

D8..D0	Filling status of the FIFO from 0 to 511
	Related to 8-bit values
D9	32-bit output register
0	- empty
1	- full
D13..D10	Filling status of the shift register
	Values from 0 to 7
	Related to single bits

Read_FIFO - register (Base + 20)

Reading on the base address + 20 reads 32-bit data from the FIFO.

Is only recommended, when at least 32-bit data are in the FIFO (or output register).

Can be checked through bit 3 in the status register.

Initialisation register (Base + 28)

Reading on the base address + 28 reads the initialisation register (Description: See above)

Output register (Base + 32)

Reading on the base address + 32 reads the initialisation of the outputs. Related on the manual switching and not on the trigger output or output in the monitoring mode.

Table 2-4: Output register

BIT D0	0	Output 1 not set
	1	Output 1 set
BIT D1	0	Output 2 not set
	1	Output 2 set

Interrupt-Status register (Base + 36)**Table 2-5: Interrupt status register**

BIT D0	0	Interrupt „FIFO half full“ not generated
	1	Interrupt „FIFO half full“ generated
BIT D1	0	Interrupt „FIFO full“ not generated
	1	Interrupt „FIFO full“ generated
BIT D2	0	Interrupt „not enough clocks“ not generated
	1	Interrupt „not enough clocks“ generated
BIT D3	0	Interrupt „too many clocks“ not generated
	1	Interrupt „too many clocks“ generated
BIT D4	0	Interrupt „Clock break too short“ not generated
	1	Interrupt „Clock break too short“ generated

Identification register (Base + 60)

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

BASE + 60 "S" "M" "1" "0"

Meaning: SSI (Synchronous serial interface) Monitor Revision 1.0

2.8.3 Write Register**Countsregister (Base + 0)**

13 bit register for determination of bit numbers

- Bit 0-5: Number of bits of the angle encoder
Determines the number of bits to be read after a clock break.
- Bit 6-12: Number of bits of the control
Is only required in the monitoring mode. The monitorin mode identifies a wrong number of bits of the control between two clock breaks. The entered number (bit 6-12) serves as reference value.

Timeregister (Base + 4)

28-bit register for the determination of times

Bit 0-8 min. time for a clock break (referential value to find a clock break)

Bit 16-27 min. clock break to be kept (is only required for the error monitoring mode)

Start - register (Base + 8)

The measurement is started by writing a dummy on the address (Base + 8).

Stopp - register (Base + 12)

By writing a dummy on the address (Base + 12), the measurement is stopped.

Reset – register (Base + 16)

At the reset of the whole design, the measurement is ended and registers of the monitor are reset. After the reset all already measured data are lost and before the next measurement all must be initialised newly.

Through a reset of the FIFO, the shift register, the FIFO and the output register are reset (see Fig. 2-5).

Table 2-6: Reset register

BIT D0	0	-
	1	Reset of the whole design
BIT D1	0	-
	1	Reset of the FIFO

Shift pulse – register (Base + 20)

By writing a dummy on the address (Base + 20), a shift pulse is generated for the shift register so that the remaining data of the shift register can be read after a measurement.

Initialisation register (Base + 28)**Table 2-7: Initialisation register**

BIT D0	0	-
	1	-
BIT D1	0	No interrupt at FIFO full or half full
	1	Interrupt at FIFO full or half full
BIT D2	0	No interrupt at error (in the monitoring mode)
	1	Interrupt at error (in the monitoring mode)
BIT D3	0	Monitoring mode OFF
	1	Monitoring mode ON

BIT D4	0	Start of a measurement through pulse at input 1 not possible.
	1	Start of a measurement through pulse at input 1 possible
BIT D5	0	Start of a measurement through pulse at input 2 not possible.
	1	Start of a measurement through pulse at input 2 possible.
BIT D6	0	At error (in the monitoring mode) does not set output 1
	1	At error (in the monitoring mode) sets output 1
BIT D7	0	-
	1	-
BIT D8	0	-
	1	-
BIT D9	0	-
	1	-
BIT D10	0	No trigger pulse (1µs) at the beginning of each clock brush
	1	Trigger pulse (1µs) at the beginning of each clock brush (output 1)

Output register (Base +32)

Table 2-8: Output register

BIT D0	0	Resets output 1
	1	Sets output 1
BIT D1	0	Resets output 2
	1	Sets output 2

Reset-Interrupt-Register (Base + 36)

Table 2-9: Reset interrupt register

BIT D0	0	-
	1	Resets the interrupt „FIFO half full“
BIT D1	0	-
	1	Resets the interrupt „FIFO full“
BIT D2	0	-
	1	Resets the interrupt „not enough clocks“
BIT D3	0	-
	1	Resets the interrupt „too many clocks“
BIT D4	0	-

	1	Resets the interrupt „clock break too short“
--	---	--

3 STANDARD SOFTWARE

3.1 Define values



IMPORTANT!

Observe the following style 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

3.2 Interrupt mask

Each SSI monitor can generate different interrupts. The following table shows the module number and the type of the generated interrupt.

Table 3-2: Interrupt mask of the functions „SSI monitor“

b_ModuleMask	ul_InterruptMask	Meaning
0000 0001	0x80000000	FIFO half full - Interrupt, module 0
0000 0001	0x80000001	FIFO full- Interrupt, module 0
0000 0001	0x80000002	Error - Interrupt, not enough clocks, module 0
0000 0001	0x80000003	Error - Interrupt, too many clocks, module 0
0000 0001	0x80000004	Error - Interrupt, clock break too short, module 0
0000 0010	0x80000000	FIFO half full - Interrupt, module 1
0000 0010	0x80000001	FIFO full- Interrupt, module 1

0000 0010	0x80000002	Error - Interrupt, not enough clocks, module 1
0000 0010	0x80000003	Error - Interrupt, too many clocks, module 1
0000 0010	0x80000004	Error - Interrupt, clock break too short, module 1
0000 0100	0x80000000	FIFO half full - Interrupt, module 2
0000 0100	0x80000001	FIFO full - Interrupt, module 2
0000 0100	0x80000002	Error - Interrupt, not enough clocks, module 2
0000 0100	0x80000003	Error - Interrupt, too many clocks, module 2
0000 0100	0x80000004	Error - Interrupt, clock break too short, module 2
0000 1000	0x80000000	FIFO half full - Interrupt, module 3
0000 1000	0x80000001	FIFO full - Interrupt, module 3
0000 1000	0x80000002	Error - Interrupt, not enough clocks, module 3
0000 1000	0x80000003	Error - Interrupt, too many clocks, module 3
0000 1000	0x80000004	Error - Interrupt, clock break too short, module 3

3.3 SSI monitor initialisation

1) i_APCI1710_InitSSIMonitor (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitSSIMonitor
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 BYTE      b_SSIBitNbr_in)
```

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_SSIBitNbr_in	Selection of the bit of the bit of the angle encoder (2 to 36)

-Output:

There is no output

Task:

Configures the SSI monitor of the selected module (*b_ModulNbr*).
Call this function first before calling any other function, which accesses the SSI monitor. When calling this function, the whole design is reset.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_APCI1710_InitSSI    (b_BoardHandle,
                                     0,
                                     25);
```

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 monitor module
 -4: The 40 MHz quartz is not available
 -5: The selected bit length of the angle encoder is wrong

2) i_APCI1710_InitSSIMonitorInterruptData (...)**Syntax:**

```
<Return Wert> = i_APCI1710_InitSSIMonitorInterruptData
                    (BYTE    b_BoardHandle,
                     BYTE    b_ModulNbr,
                     BYTE    b_SSIInterrupt_Data)
```

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_SSIInterrupt_Data	Selection of the data interrupt
	0:	APCI1710_DISABLE
	1:	APCI1710_ENABLE

-Output:

There is no output.

Task:

Initialises the data interrupt at FIFO full or FIFO half full

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitSSIMonitorInterruptData
                    (b_BoardHandle,
                     0,
                     APCI1710_ENABLE);
```

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 monitor module
 -4: The selection ENABLE or DISABLE is wrong

3) i_APCI1710_InitSSIMonitorInterruptError (...)**Syntax:**

```
<Return Wert> = i_APCI1710_InitSSIMonitorInterruptError
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                BYTE      b_SSIIInterrupt_Error,
                                BYTE      b_SSIIInterrupt_ErrorOutput_5V,
                                BYTE      b_extBitNbr_in,
                                BYTE      b_SSIDelayTime_min_in)
```

Parameter:**- Input**

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_SSIIInterrupt_Error	Selection of the error interrupt (monitoring mode) 0: APCI1710_DISABLE 1: APCI1710_ENABLE
BYTE	b_SSIIInterrupt_ErrorOutput_5V	Selection of the error output (output 1) 0: APCI1710_DISABLE 1: APCI1710_ENABLE
BYTE	b_extBitNbr_in	Selection of the bit length of the control (2 to 64)
BYTE	b_SSIDelayTime_min_in	Selection of the min. clock break in µs (2 to 100)

-Output:

There is no output.

Task:

Initialises the error interrupt (monitoring mode) when:

- Clock break too short
- Too many clocks
- Not enough clocks

You can select to set the output 1 with the error. The number of bits to be entered of the control is used as referential value for the message “too many clocks” or “not enough clocks”.

The min. clock break to be entered is used as a reference value for the error “clock break too short”.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitSSIMonitorInterruptError  
                (b_BoardHandle,  
                 0,  
                 APCI1710_ENABLE,  
                 APCI1710_ENABLE,  
                 25,  
                 40);
```

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 monitor module
- 4: The selection ENABLE or DISABLE is wrong
- 5: The selection of the bit length of the control is wrong
- 6: The bit length of the control is shorter than the bit length of the angle encoder.
- 7: The selection of the clock break is wrong

4) i_APCI1710_InitSSIMonitorTriggerOutput (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIAllDigitalInput
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 BYTE      b_SSITriggerOutput)
```

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_SSITriggerOutput	Selection of the trigger output 0: APCI1710_DISABLE 1: APCI1710_ENABLE

- Output

There is no output

Task:

Initialisation of the trigger output (output 1). At the beginning of each clock brush a trigger pulse of 1 µs can be generated (optionally).

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitSSIMonitorTriggerOutput
                (b_BoardHandle,
                 0,
                 APCI1710_ENABLE);
```

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 monitor module
- 4: The selection ENABLE or DISABLE is wrong

5) i_APCI1710_InitSSIMonitorExternalGate (...)**Syntax:**

```
<Return Wert> = i_APCI1710_InitSSIMonitorExternalGate
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 BYTE      b_SSIExternalGate,
                 BYTE      b_SSIExternalGateNbr)
```

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_SSIExternalGate	Selection of the external start conditions 0: APCI1710_DISABLE 1: APCI1710_ENABLE
BYTE	b_SSIExternalGateNbr	1: Input 1 2: Input 2

- Output

There is no output

Task:

Initialisation of the external start possibilities. If DISABLE is selected, then the both start possibilities (input 1 and input 2) are reset.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
```

```
i_ReturnValue = i_APCI1710_InitSSIMonitorExternalGate
                (b_BoardHandle,
                 0,
                 APCI1710_ENABLE,
                 1);
```

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 monitor module
-4: The selection ENABLE or DISABLE is wrong
-5: The selection of the input number is wrong

3.4 SSI monitor – Start/Stopp

1) i_APCI1710_StartSSIMonitor (...)

Syntax:

```
<Return Wert> = i_APCI1710_StartSSIMonitor
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr)
```

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)

- Output

There is no output.

Task:

Starts the measurement of the SSI monitor. Before the start the FIFO of the monitor is deleted.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_StartSSIMonitor
                    (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 monitor module
- 4: The interrupt routine is not installed, although an interrupt mode is ENABLE.

2) i_APCI1710_StopSSIMonitor (...)**Syntax:**

```
<Return Wert> = i_APCI1710_StopSSIMonitor
                        (BYTE      b_BoardHandle,
                        BYTE      b_ModulNbr)
```

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

- Output

There is no output.

Task:

Stops the measurement of the SSI monitor, no matter how the monitor was started (per software function or external start possibility)

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_StopSSIMonitor
                        (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 monitor module

3.5 SSI monitor reset

1) i_APCI1710_ResetSSIMonitorFIFO (...)

Syntax:

```
<Return Wert> = i_APCI1710_ResetSSIMonitorFIFO
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr)
```

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)

- Output

There is no output

Task:

Deletes the whole memory of the memory (shift register, FIFO and output register)

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ResetSSIMonitorFIFO
                                (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 monitor module

3.6 SSI monitor outputs

1) i_APCI1710_SetSSIMonitorDigitalOutputON (...)

Syntax:

```
<Return Wert> = i_APCI1710_SetSSIMonitorDigitalOutputON
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     BYTE      b_SSIDigitalOutputNbr)
```

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_SSIDigitalOutputNbr	1: Output 1 2: Output 2

- Output

There is no output

Task:

Sets the outputs of the board. You can choose between output 1 and output 2.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_SetSSIMonitorDigitalOutputON
                    (b_BoardHandle,
                     0,
                     1);
```

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 monitor module
 -4: The selection of the output is wrong

2) i_APCI1710_SetSSIMonitorDigitalOutputOFF(...)**Syntax:**

```
<Return Wert> = i_APCI1710_SetSSIMonitorDigitalOutputOFF
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 BYTE      b_SSIDigitalOutputNbr)
```

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_SSIDigitalOutputNbr	1: Output 1 2: Output 2

- Output

There is no output

Task:

Resets the outputs of the board. It can be selected between output 1 and output 2.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_SetSSIMonitorDigitalOutputON
                (b_BoardHandle,
                 0,
                 1);
```

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 monitor module
 -4: The selection of the output is wrong

3.7 SSI monitor – read register

1) i_APCI1710_ReadSSIMonitorMeasuringStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_ReadSSIMonitorMeasuringStatus
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 PBYTE     pb_SSIMeasuring_ON,
                 PBYTE     pb_SSIEndSearch_MonoTime,
                 PBYTE     pb_SSIEndSearch_DelayTime,
                 PBYTE     pb_SSIEndSearch_Frequency,)
```

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)

-Output:

PBYTE	pb_SSIMeasuring_ON	Status of the measurement 0: not active 1: active
PBYTE	pb_SSIEndSearch_MonoTime	Status of the monoflop time measurement 0: No monoflop time measured 1: Monoflop time measured at least once
PBYTE	pb_SSIEndSearch_DelayTime	Status of the clock break measurement 0: No clock break measured 1: Clock break measured at least once
PBYTE	pb_SSIEndSearch_Frequency	Status of the frequency measurement 0: No frequency measured 1: Frequency measured at least once.

Task:

Reads the status of the measurement.

Calling convention:

ANSI C:

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



```
unsigned char b_SSIEndSearch_MonoTime;
unsigned char b_SSIEndSearch_DelayTime;
unsigned char b_SSIEndSearch_Frequency;

i_ReturnValue = i_APCI1710_ReadSSIMonitorMeasuringStatus
    (b_BoardHandle,
     0,
     &b_SSIMeasuring_ON,
     &b_SSIEndSearch_MonoTime,
     &b_SSIEndSearch_DelayTime,
     &b_SSIEndSearch_Frequency);
```

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 monitor module

2) i_APCI1710_ReadSSIMonitorFIFOStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_ReadSSIMonitorFIFOStatus
                                (BYTE    b_BoardHandle,
                                 BYTE    b_ModulNbr,
                                 PBYTE   pb_SSIFIFO_newvalue,
                                 PBYTE   pb_SSIFIFO_nempty,
                                 PBYTE   pb_SSIFIFO_hfull,
                                 PBYTE   pb_SSIFIFO_full,
                                 PULONG  pul_SSIFIFO_value)
```

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)

-Output:

PBYTE	pb_SSIFIFO_newvalue	At least 32 bit data in the buffer 0: No 1: Yes
PBYTE	pb_SSIFIFO_nempty	Buffer empty 0: Yes 1: No
PBYTE	pb_SSIFIFO_hfull	Memory at least half full 0: No 1: Yes
PBYTE	pb_SSIFIFO_full	Buffer full 0: No 1: Yes
PULONG	pul_SSIFIFO_value	Fillling status of the buffer in x 8bit values From 0: Empty to 515: Full

Task:

Reads the status of the buffer.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_SSIFIFO_newvalue;
unsigned char b_SSIFIFO_nempty;
unsigned char b_SSIFIFO_hfull;
unsigned char b_SSIFIFO_full;
unsigned long ul_SSIFIFO_value;
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorFIFOStatus
                (b_BoardHandle,
                 0,
                 &b_SSIFIFO_newvalue,
                 &b_SSIFIFO_nempty,
                 &b_SSIFIFO_hfull,
                 &b_SSIFIFO_full,
                 &ul_SSIFIFO_value);
```

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 monitor module

3) i_APCI1710_ReadSSIMonitorAllDigitalInputs(...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorAllDigitalInputs
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 PBYTE     pb_SSIInput_Status)
```

Parameter:**- Input**

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

-Output:

PBYTE	pb_SSIInputStatus	Status of the inputs	0: Low 1: High
		D0:	Input 1
		D1:	Input 2
		D2:	Input 3
		D3:	Input 4

Task:

Reads in the level of the digital inputs.

Calling covention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorAllDigitalInputs
                (b_BoardHandle,
                 0,
                 &b_SSIInput_Status);
```

Return value:

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

4) i_APCI1710_CheckSSIMonitor40MHzStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_CheckSSIMonitor40MHzStatus
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     PBYTE     pb_SSI40MHz_Status)
```

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)

- Output

PBYTE	pb_SSI40MHz_Status	Status of the 40 MHz quartz
	0:	Quartz not available
	1:	Quartz available

Task:

Reads in the status of the 40 MHz quartz of the board.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_SSI40MHz_Status
```

```
i_ReturnValue = i_APCI1710_CheckSSIMonitor40MHzStatus
                (b_BoardHandle,
                0,
                &b_SSI40MHz_Status);
```

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 monitor module

5) i_APCI1710_ReadSSIMonitorMonoTime (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorMonoTime
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     double *   pd_SSIMonoTime_out)
```

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)

- Output

double*	pd_SSIMonoTime_out	Value of the monoflop time in μ s At counter overflow the value FFFFFFFF is returned
---------	--------------------	--

Task:

Reads the measured monoflop time of the angle encoder.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
double       d_SSIMonoTime_out;
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorMonoTime
                    (b_BoardHandle,
                     0,
                     &d_SSIMonoTime_out);
```

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 monitor module
- 4: The measurement of the monofloptime is not ended yet.

6) i_APCI1710_ReadSSIMonitorDelayTime (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorDelayTime
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                double *   pd_SSIDelayTime_out)
```

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)

- Output

double*	pd_SSIDelayTime_out	Value of the clock break in μ s At counter overflow the value is returned in FFFFFFFF
---------	---------------------	---

Task:

Reads the measured clock break of the control

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
double       d_SSIDelayTime_out;
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorDelayTime
                                (b_BoardHandle,
                                0,
                                &d_SSIDelayTime_out);
```

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 monitor module
- 4: The measurement of the clock break is not ended

7) i_APCI1710_ReadSSIMonitorFrequency (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorFrequency
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     double *   pd_SSIFrequency_out)
```

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)

- Output

double*	pd_SSIFrequency_out	Value of the frequency in kHz at counter overflow the value is returned in FFFFFFFF
---------	---------------------	---

Task:

Reads the measured frequency of the control.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
double       d_SSIFrequency_out;
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorFrequency
                    (b_BoardHandle,
                     0,
                     &d_SSIFrequency_out);
```

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 monitor module
- 4: The measurement of the frequency is not ended

8) i_APCI1710_ReadSSIMonitorData (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorData
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                PULONG    pul_FIFOData_out)
```

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)

- Output

PULONG	pul_FIFOData_out	32-bit data in the buffer
--------	------------------	---------------------------

Task:

Reads 32-bit data from the buffer of the monitor. Reading data is only recommended, when at least 32-bit data are stored in the FIFO. This can be checked with the function „i_APCI1701_ReadSSIMonitorFIFOStatus“.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorData
                                (b_BoardHandle,
                                0,
                                &ul_FIFOData_out);
```

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 monitor module

9) i_APCI1710_ReadSSIMonitorErrorInterruptStatus (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorErrorInterruptStatus
                (BYTE    b_BoardHandle,
                 BYTE    b_ModulNbr,
                 PBYTE   pb_SSError_Register,
                 PULONG  pul_SSIFIFO_errvalue,
                 double*  pd_SSIDelayTime_out)
```

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)

-Output:

PBYTE	pb_SSError_Register	Type of error D0: Not enough clocks within the clock brush D1: Too many clocks within the clock brush D2: Clock break too short
PULONG	pul_SSIFIFO_errvalue	Filling status of the buffer when the error occurs From 0: Empty To 515: Full
double*	pd_SSIDelayTime_out	At error „Clock break too short“, the clock break, which has caused the error is returned. Value in µs.

Task:

Reads the error status, which was saved when the error occurred. Only possible if the error interrupt (monitoring mode) is ENABLE.

Calling convention:

ANSI C:

int	i_ReturnValue;
unsigned char	b_BoardHandle;
unsigned char	b_SSError_Register;
unsigned long	ul_SSIFIFO_errvalue;
double	d_SSIDelayTime_out;

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorErrorInterruptStatus  
                (b_BoardHandle,  
                 0,  
                 &b_SSIError_Register,  
                 &ul_SSIFIFO_errvalue,  
                 &d_SSIDelayTime_out);
```

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 monitor module
- 4: Error interrupt mode is DISABLE

10) i_APCI1710_ReadSSIMonitorRestFIFOData (...)**Syntax:**

```
<Return Wert> = i_APCI1710_ReadSSIMonitorRestFIFOData
                    (BYTE    b_BoardHandle,
                     BYTE    b_ModulNbr,
                     PULONG  pul_FIFOData_out,
                     PBYTE   pb_UsefulRestBit)
```

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)

-Output:

PULONG	pul_FIFOData_out	32-bit value with data
PBYTE	pb_UsefulRestBit	Number of bits, which contain data. The remaining bits of the 32-bit value are not defined.

Task:

Reads the 32-bit value from the buffer. The number of useful bits is indicated by a second return parameter. The function can be used only when less than 32-bit data is available in the buffer and when the measurement is ended. After reading the data, the function deletes the whole buffer of the monitor.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_FIFOData_out;
unsigned char b_UsefulRestBit;
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorRestFIFOData
                    (b_BoardHandle,
                     0,
                     &ul_FIFOData_out,
                     &b_UsefulRestBit);
```

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 monitor module
- 4: The measurement is still active
- 5: There are still more than 32 bit data in the buffer

4 TUTORIAL

The present tutorial shows the application possibilities of the SSI monitor with a few examples. Here, the application is created step by step parallel with the user in order to ease the first use of the function module. The present tutorial can be extended with any SSI monitor function that is available.

The used functions are not described in detail in order to keep the tutorials brief and clearly structured. Please refer firstly to the technical description of the board APCI-/CPIC-1710 as well as to the function description of the SSI monitor.

The tutorials are based on the programming language C. Furthermore, the following requirements should be fulfilled in advance:

- Successful installation of the board
- Loading of the function module
- Connection of the monitor to an existing SSI system

The installation of the board and the loading of the function module are described in the technical description of the board. A connection example of the monitor is explained in chapter 2.6 of the function description.

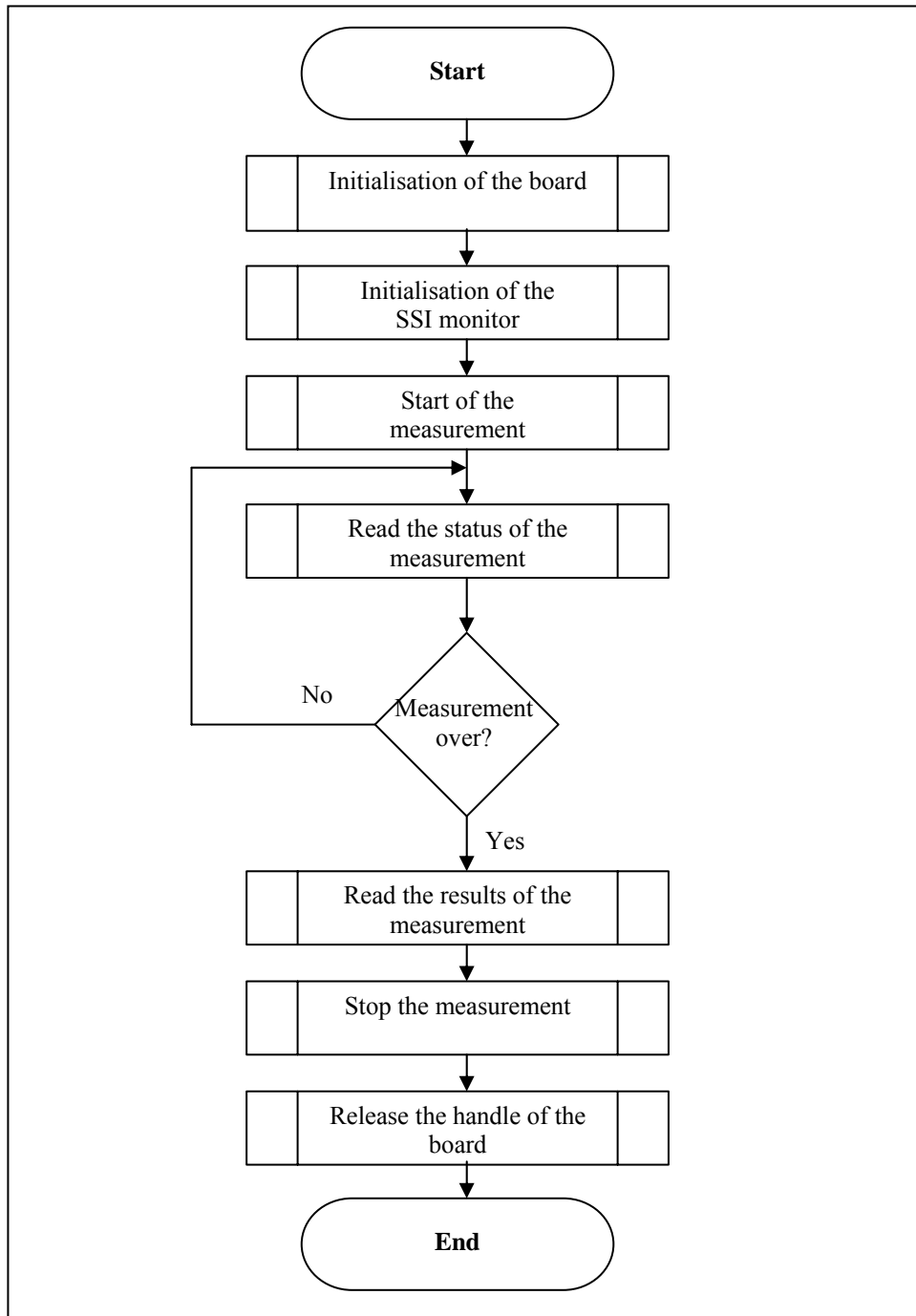
4.1 Reading typical parameters

The typical parameters of existing SSI systems consist of:

- Clock break of the control
- Frequency of the control
- Monofloptime of the angle encoder

The following description refers to C-Samples Sample03, Sample04 and Sample05, with which the above mentioned parameters can be read.

The used functions are described more detailed in the technical description of the board or in the function description of the SSI monitor.

Diagram – Reading typical parameters

4.1.1 Step 1: Initialising the board

Firstly, the initialisation of the SSI monitor must be realised. In order to structure it more clearly, a function is created that calls the functions, which are provided by ADDI-DATA. In the C-Samples the function “Initialisation” is shown at the beginning. The following steps are required at the initialisation:

1. Selecting the compiler:

With the following function the used compiler is indicated, in the present example it is a C-compiler.

```
i_ReturnValue = i_APCI1710_InitCompiler (DLL_COMPILER_C);
```

2. Determining the slot number:

```
i_ReturnValue = i_APCI1710_CheckAndGetPCISlotNumber  
(b_SlotNumberArray);
```

3. Checking the slot number and determining the handle of the board:

```
i_ReturnValue = i_APCI1710_SetBoardInformation (b_SlotNumberArray [0],  
                                                pb_BoardHandle);
```

4.1.2 Step 2: Initialising the SSI monitor

Before you can use the functions of the SSI monitor, the monitor must be initialised. Here the bit length of the SSI angle encoder, which shall be monitored, is transferred.

```
i_ReturnValue = i_APCI1710_InitSSIMonitor (    b_BoardHandle,  
                                                b_ModuleNbr,  
                                                b_SSIBitNbr_in);
```

4.1.3 Step 3: Starting the measurement

```
i_ReturnValue = i_APCI1710_StartSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.1.4 Step 4: Checking the status of the measurement

After starting the monitor successfully, the status of the measurement is read until the measurement of the parameter to be read is realised at least once successful.

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorMeasuringStatus(b_BoardHandle,  
                                                         b_ModuleNbr,  
                                                         &b_SSIMeasuring_ON,  
                                                         &b_SSIEndSearch_MonoTime,  
                                                         &b_SSIEndSearch_DelayTime,
```

&b_SSIEndSearch_Frequency);

4.1.5 Step 5: Reading the results of the measurements

When the first measured value of the required measurement is available, the corresponding function can be read with

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorMonoTime (b_BoardHandle,  
                                                    b_ModuleNbr,  
                                                    &b_SSIMonoTime_out);
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorDelayTime (b_BoardHandle,  
                                                    b_ModuleNbr,  
                                                    &b_SSIDelayTime_out);
```

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorFrequency (b_BoardHandle,  
                                                    b_ModuleNbr,  
                                                    &b_SSIFrequency_out);
```

4.1.6 Step 6: Stopping the measurement

When all required results of measurements are read, the measurement of the SSI monitor can be ended.

```
i_ReturnValue = i_APCI1710_StopSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.1.7 Step 7: Releasing the handle of the board

Before the user program is ended, the handle of the board must be released with the following function:

```
i_ReturnValue = i_APCI1710_CloseBoardHandle (b_BoardHandle);
```


4.2 Reading all sensor data

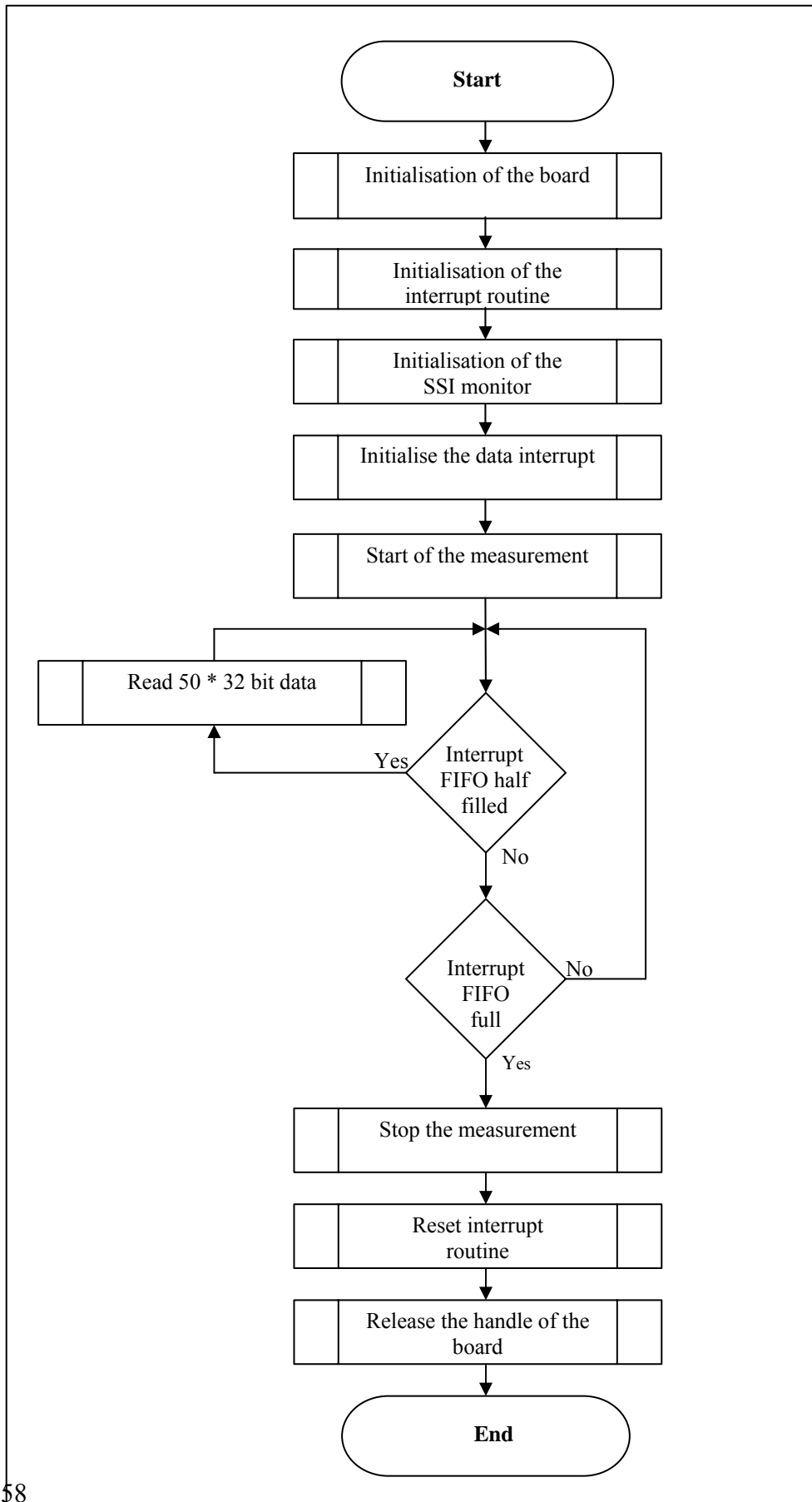
A further application of the SSI monitor is the acquisition and storage of serial data of the SSI angle encoder. The data acquisition can be realised with different methods. Either in the polling mode through the status register or with the data interrupt at FIFO half full or FIFO full. Hereto the following C-samples are available:

Sample14	->	Interrupt (asynchronous mode)
Sample15	->	Interrupt (synchronous mode)
Sample16	->	Polling mode

The following tutorial refers to Sample 14, in which the data are reported via interrupt in the asynchronous mode and then are read out in a loop.

The differences between asynchronous and synchronous mode as well as the used functions are described more detailed in the technical description of the board or in the function description of the SSI monitor.

Diagram – Reading all data of the sensor



4.2.1 Step 1: Initialising the board

Firstly, the initialisation of the SSI monitor must be realised. To structure it more clearly, a function is created, which calls the functions that are provided by ADDI-DATA. In the C-Samples the function “Initialisation” is shown at the beginning. The following steps are required at initialisation:

4. Selecting a compiler:

The following function indicates the used compiler, in this example it is a C-compiler

```
i_ReturnValue = i_APCI1710_InitCompiler (DLL_COMPILER_C);
```

5. Determining the slot number:

```
i_ReturnValue = i_APCI1710_CheckAndGetPCISlotNumber  
(b_SlotNumberArray);
```

6. Checking the slot number and determining the handle of the board:

```
i_ReturnValue = i_APCI1710_SetBoardInformation (b_SlotNumberArray [0],  
                                                pb_BoardHandle);
```

4.2.2 Step 2: Initialising the interrupt routine

Sample calling for the function Windows NT and Windows 9x. The function is transferred under another name to the user interrupt routine. If an interrupt is generated, it will be called by the system.

The interrupt can be generated in the synchronous and in the asynchronous mode as well as the shared memory can be used in the synchronous mode. A more detailed description of the interrupt routine can be found in the technical description of your board.

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin32 (b_BoardHandle,  
                                                     APCI1710_ASYNCHRONOUS_MODE,  
                                                     0,  
                                                     NULL,  
                                                     v_InterruptFunktion);
```

4.2.3 Step 3: Initialising the SSI monitor

Before the functions of the SSI monitor can be used, the monitor must be initialised. Here the bit length of the SSI angle encoder to be monitored is transferred:

```
i_ReturnValue = i_APCI1710_InitSSIMonitor (    b_BoardHandle,  
                                              b_ModuleNbr,  
                                              b_SSIBitNbr_in);
```

4.2.4 Step 4: Initialising the data interrupt

This function must be called in order to initialise the function of the data interrupt:

```
i_ReturnValue = i_APCI1710_InitSSIMonitorInterruptData (b_BoardHandle,  
                                                         b_ModuleNbr,  
                                                         b_SSIInterrupt_Data);
```

4.2.5 Step 5: Starting the measurement

```
i_ReturnValue = i_APCI1710_StartSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.2.6 Step 6: Interrupt

FIFO half full → **Read data**

When the interrupt FIFO *half full* occurs, the user interrupt routine is called in which the data must be read. This can be realised for example with a FOR loop, in which the following functions are called:

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorData (b_BoardHandle,  
                                                         b_ModuleNbr,  
                                                         pul_FIFOData_out);
```

In Sample 14 the function is called 50 times. This means that $50 * 32$ bit data (1600 bits) are read from the FIFO. At this time the FIFO is at least half filled
→ $256 * 8$ bit = 2048 bit.

It is recommended to set the number of reading accesses per interrupt high in order to reduce the number of interrupts.

The number of reading accesses may not exceed 64 per interrupt, as in this case $64 * 32$ bit = 2048 bit are read and at least this number of bits is available in the FIFO.

FIFO full → **end**

When the interrupt FIFO full occurs or when the program is interrupted by the user, the monitor shall be stopped, the interrupt routine should be reset and the handle of the board shall be released. These steps are described below:

4.2.7 Step 7: Stopping the measurement

The following function stops the measurement of the monitor:

```
i_ReturnValue = i_APCI1710_StopSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.2.8 Step 8: Reset board interrupt routine

This function stops the interrupt management of the board.

```
i_ReturnValue = i_APCI1710_ResetBoardIntRoutine (b_BoardHandle);
```

4.2.9 Step 9: Releasing the handle of the board

Before the user program is ended, the handle of the board must be released with the following function:

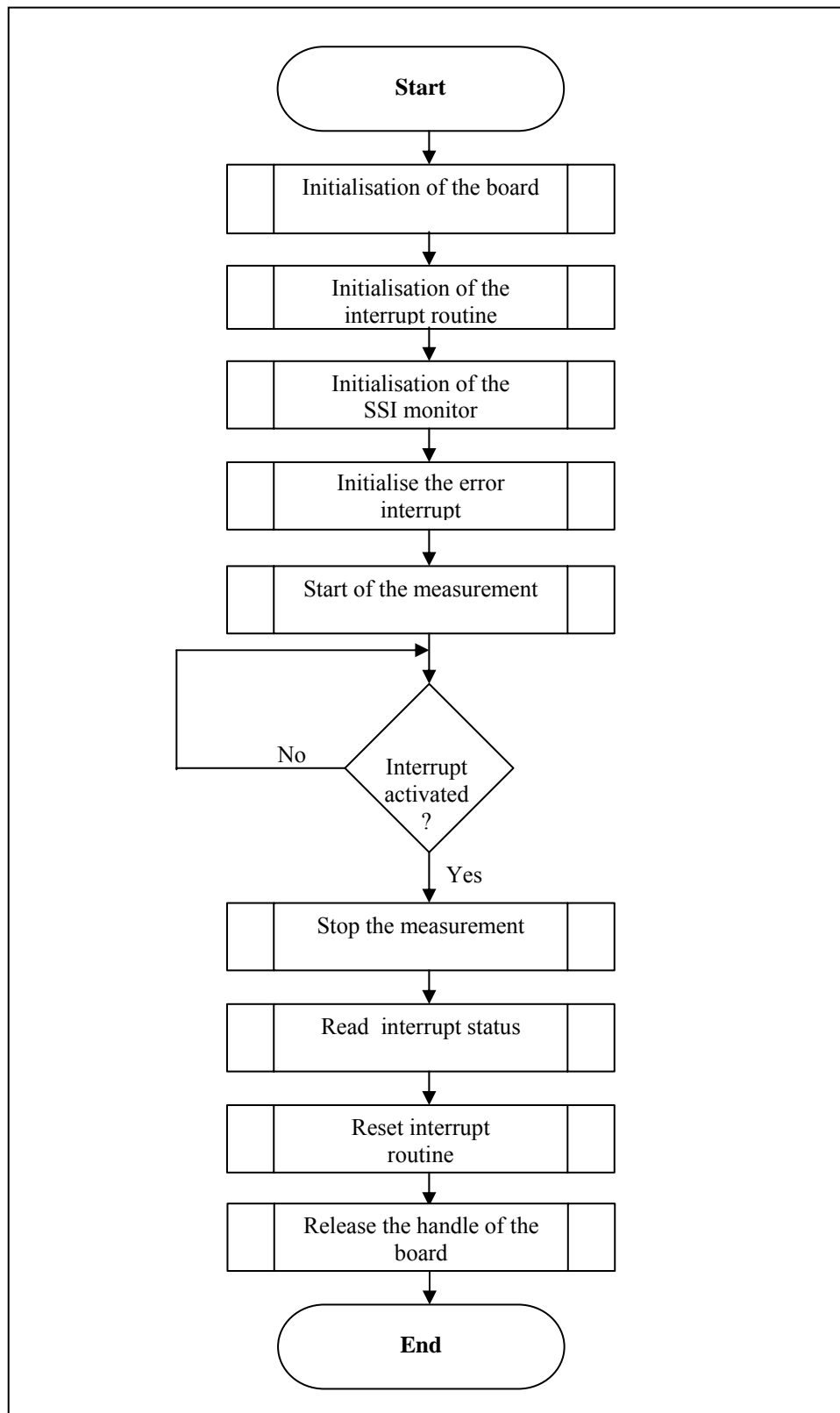
```
i_ReturnValue = i_APCI1710_CloseBoardHandle (b_BoardHandle);
```

4.3 Monitoring the control

A further application possibility of the SSI monitor is the monitoring of the external control, which operates the SSI angle encoder. Here the number of the clocks and the clock break between the clock brushes, which are generated by the control, is monitored.

The following description refers to the C sample Sample13, with which the above mentioned function can be tested.

The used functions are described more detailed in the technical description of the board or in the function description of the SSI monitor.

Diagram-Monitoring the control

4.3.1 Step 1: Initialising the board

Firstly, the initialisation of the SSI monitor must be realised.

To structure it more clearly, a function is created, which calls the functions that are provided by ADDI-DATA. In the C-Samples the function “Initialisation” is shown at the beginning. The following steps are required at the initialisation:

7. Selecting the compiler:

The following function shows the used compiler, in this example it is a C-compiler.

```
i_ReturnValue = i_APCI1710_InitCompiler (DLL_COMPILER_C);
```

8. Determining the slot number:

```
i_ReturnValue = i_APCI1710_CheckAndGetPCISlotNumber  
(b_SlotNumberArray);
```

9. Checking the slot number und determining the handle of the board:

```
i_ReturnValue = i_APCI1710_SetBoardInformation (b_SlotNumberArray [0],  
                                                pb_BoardHandle);
```

4.3.2 Step 2: Initialising the interrupt routine

Sample calling for the function Windows NT and Windows 9x. The function is transferred under an other name to the user interrupt routine. If an interrupt is generated, it will be called by the system.

The interrupt can be generated in the synchronous and in the asynchronous mode and the shared memory can be used in the synchronous mode. A more detailed description of the interrupt routine can be found in the technical description of your board.

```
i_ReturnValue = i_APCI1710_SetBoardIntRoutineWin32 (b_BoardHandle,  
                                                    APCI1710_ASYNCHRONOUS_MODE,  
                                                    0,  
                                                    NULL,  
                                                    v_InterruptFunktion);
```

4.3.3 Step 3: Initialising the SSI monitor

Before the functions of the SSI monitor can be used, the monitor must be initialised. Here the bit length of the SSI angle encoder to be monitored is transferred:

```
i_ReturnValue = i_APCI1710_InitSSIMonitor (    b_BoardHandle,  
                                              b_ModuleNbr,  
                                              b_SSIBitNbr_in);
```


4.3.4 Step 4: Initialising the error interrupt

This function must be called in order to initialise the function of the error interrupt (monitoring mode of the control):

```
i_ReturnValue = i_APCI1710_InitSSIMonitorInterruptError (b_BoardHandle,
                                                         b_ModuleNbr,
                                                         b_SSIIInterrupt_Error,
                                                         b_SSIIInterrupt_ErrorOutput_5V,
                                                         b_extBitNbr_in,
                                                         b_SSIDelayTime_min_in);
```

4.3.5 Step 5: Starting the measurement

```
i_ReturnValue = i_APCI1710_StartSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.3.6 Step 6: Interrupt → Stopping the measurement

When the first interrupt occurs, the user interrupt routine is called. In this routine a flag can be set for example, which generates further events in the main program. The function to be called at first after the interrupt would be in this example the stopping of the measurement:

```
i_ReturnValue = i_APCI1710_StopSSIMonitor (b_BoardHandle, b_ModuleNbr);
```

4.3.7 Step 7: Reading the interrupt status

By reading the interrupt state the type of error, the filling status of the FIFO at the time of the error as well as, if necessary, the length of the causing clock break are shown:

```
i_ReturnValue = i_APCI1710_ReadSSIMonitorErrorInterruptStatus
(b_BoardHandle,
 b_ModuleNbr,
 &b_SSIError_Register,
 &ul_SSIFIFO_errvalue,
 &d_SSIDelayTime_out);
```

4.3.8 Step 8: Reset board interrupt routine

This function stops the interrupt management of the board.

```
i_ReturnValue = i_APCI1710_ResetBoardIntRoutine (b_BoardHandle);
```

4.3.9 Step 9: Releasing the handle of the board

Before the user program is ended, the handle of the board must be released with the following function:

```
i_ReturnValue = i_APCI1710_CloseBoardHandle (b_BoardHandle);
```