USB2.0 <=> I2C V4.4

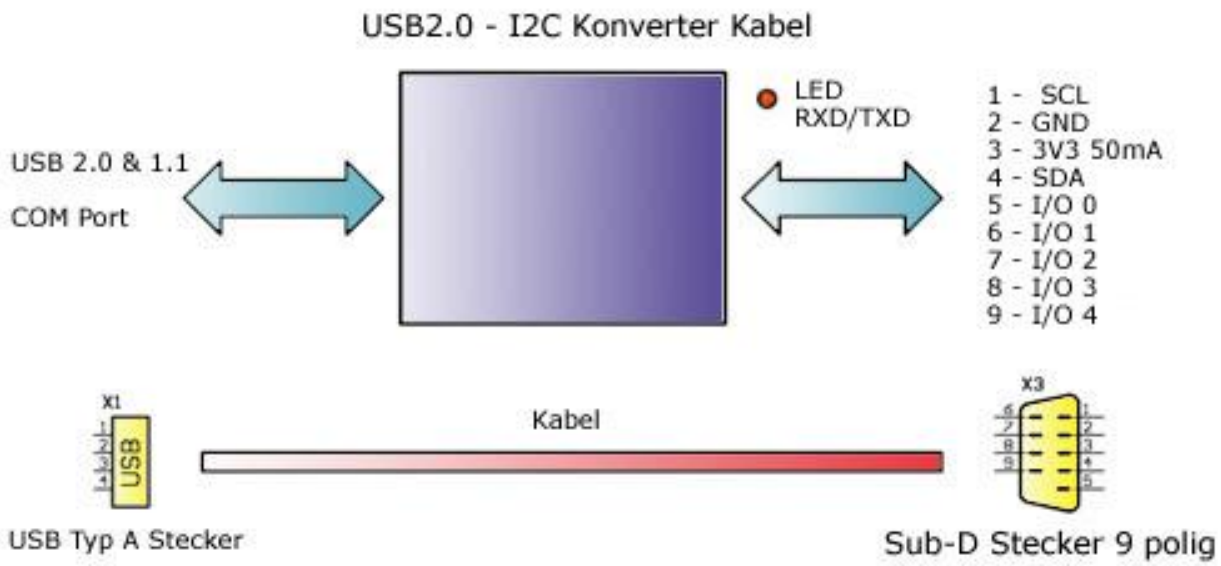
Konverter Kabel und Box mit Galvanischetrennung

USB 2.0 <=> I2C Konverter Kabel V4.4 (Prod. Nr. #210)

USB Modul:
Nach USB Spezifikation 2.0 & 1.1
Unterstützt automatisch "handshake mode"
Bis 3 Mb/s "data transfer rate"
Unterstützt "Remote wake-up" und power management
Einfachste Installation

I2C:
I2C-bus controller
5 x programmable I/O pins
High-speed I2C-bus: 400 kbit/s
Programmable baud rate generator
2.3 V and 3.6 V operation
5 V tolerance on the input pins
Sleep mode (power-down)
UART message format resembles I2C-bus transaction format
I2C-bus master functions
Multi-master capability

Applications:
Enable I2C-bus master support in a system
I2C-bus instrumentation and control
Industrial control
Cellular telephones
Handheld computers



Technische Daten

Produkt:	**USB2.0 <=> I2C Konverter Kabel**	**#210**
Driver:	**4N-GX.de**	
Installation:	**Plug & Play**	
Kabellänge:	**0,15 bis 5,0 m**	
USB-Interface:	**virtueller COM-Port**	
Anschluss 1:	**USB2.0 (1.1)**	
Übertragungsraten COM Port:	**183, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 11520, 230400, 460800, 921600 bps. bis 3000000 bps.**	
Anschluss 1 Belegung:	**Pin 1 - Vcc Pin 2 - D- Pin 3 - D+ Pin 4 - GND**	
Anschluss 2:	**I2C**	
Anschluss 2 Belegung:	**Pin 1 - SCL Pin 2 - GND Pin 3 - 3V3 max. 50 mA Pin 4 - SDA Pin 5 - I/O 0 Pin 6 - I/O 1 Pin 7 - I/O 2 Pin 8 - I/O 3 Pin 9 - I/O 4**	
Anschluss 2:	**SCL und SDA werden intern über 4K7 Pullup's auf 3V3 gezogen.**	
Übertragungsraten I2C:	**9600 bps ... 400 kbps**	
Zustandsanzeigen:	**TXD & RXD - 3mm LED rot**	
Betriebstemperatur:	**-5..+70°C**	
Treiber Software:	**Windows 98, Windows 98 SE Windows 2000, ME, XP, Vista Windows CE.NET - (Version 4.2 or greater) Apple OS-8, OS-9 und OS-X Linux Kernel ab Version 2.4.0 Open BSD ab Version 3.2 Free BSD ab Version 4.7**	

1.1 Functional description

The converter is a bridge between a host (COM Port) and I2C-bus. The serial data format is fixed: one start bit, 8 data bits, and one stop bit. After reset the baud rate defaults to 9600 bit/s, and can be changed through the Baud Rate Generator (BRG) registers.

1.2 UART message format

The host initiates an I2C-bus data transfer, reads from and writes to converter internal registers through a series of ASCII commands. Table1 lists the ASCII commands. Unrecognized commands are ignored by the device.

To prevent the host from handing the converter due to an unfinished command sequence, the converter has a time-out feature. The delay between any two bytes of data coming from the host should be less than 655 ms. If this condition is not met, the USB2.0-I2C Converter will time-out and clear the receive buffer. The converter then starts to wait for the next command.

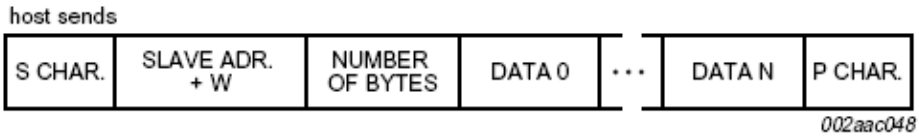
Table 1 – ASCII commands supported by USB2.0-I2C Converter

ASCII command	Hex valus	Command function
S	0x53	I^2C-bus START
P	0x50	I^2C-bus STOP
R	0x52	read internal converter register
W	0x57	write internal converter register
I	0x49	read GPIO port
O	0x4F	write to GPIO port
Z	0x5A	power down

1.3 Write N bytes to slave device

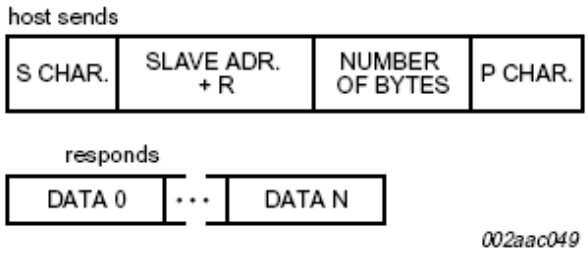
The host issues the write command by sending an S character followed by an I2C-bus slave device address, the total number of bytes to be sent, and I2C-bus data which begins with the first byte (DATA 0) and ends with the last byte (DATA N). The frame is then terminated with a P character. Once the host issues this command, the converter will access the I2C-bus slave device and start sending the I2C-bus data bytes.

Note that the second byte sent is the I2C-bus device slave address. The least significant bit (W) of this byte must be set to 0 to indicate this is an I2C-bus write command.



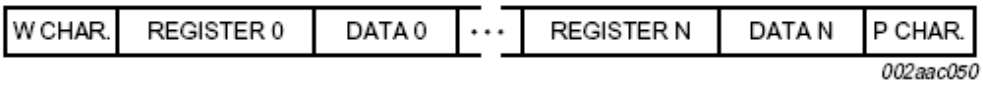
1.4 Read N byte from slave device

The host issues the read command by sending an S character followed by an I2C-bus slave device address, and the total number of bytes to be read from the addressed I2C-bus slave. The frame is then terminated with a P character. Once the host issues this command, the converter will access the I2C-bus slave device, get the correct number of bytes from the addressed I2C-bus slave, and then return the data to the host.
Note that the second byte sent is the I2C-bus device slave address. The least significant bit (R) of this byte must be set to 1 to indicate this is an I2C-bus write command.



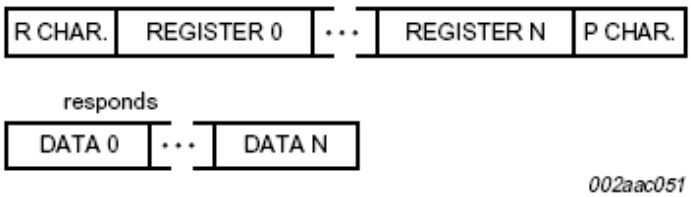
1.5 Write to converter internal register

The host issues the internal register write command by sending a W character followed by the register and data pair. Each register to be written must be followed by the data byte. The frame is then terminated with a P character.



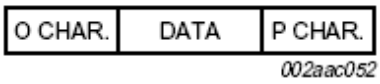
1.6 Read from converter internal register

The host issues the internal register read command by sending an R character followed by the registers to be read. The frame is then terminated with a P character.
Once the command is issued, converter will access its internal registers and returns the contents of these registers to the host.



1.7 Write to GPIO port

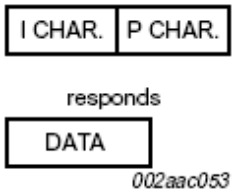
The host issues the output port write command by sending an O character followed by the data to be written to the output port. This command enables the host to quickly set any GPIO pins programmed as output without having to write to the converter internal IOState register.



1.8 Read from GPIO port

The host issues the input port read command by sending an I character. This command enables the host to quickly read any GPIO pins programmed as input without having to read the converter internal IOState register.

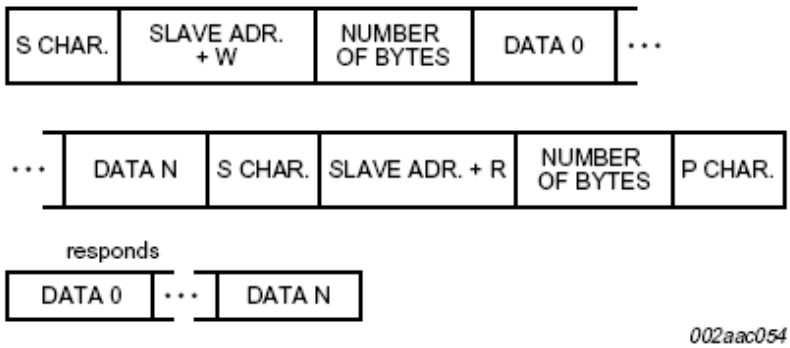
Once the command is issued, converter will read its internal IOState register and returns its content to the host.



1.9 Repeated START: read after write

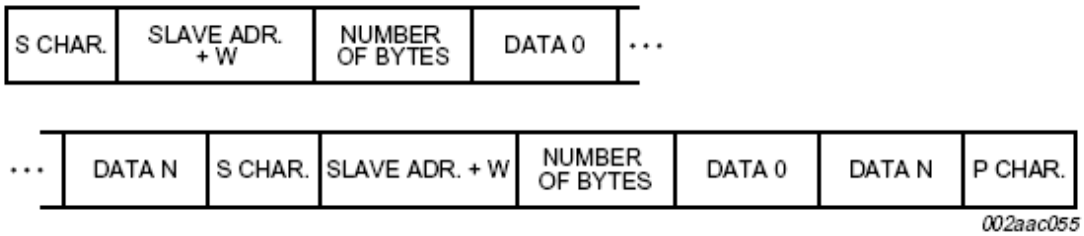
The converter also supports ‘read after write’ command as specified in the Philips’ I2C-bus specification. This allows a read command to be sent after a write command without having to issue a STOP condition between the two commands.

The host issues a write command as normal, then immediately issues a read command without sending a STOP (P) character after the write command.



1.10 Repeated START: write after write

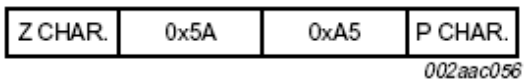
The converter also supports ‘write after write’ command as specified in the I2C-bus specification. This allows a write command to be sent after a write command without having to issue a STOP condition between the two commands. The host issues a write command as normal, then immediately issues a second write command without sending a STOP (P) character after the first write command.



1.11 Power-down mode

The converter can be placed in a low-power mode. In this mode the internal oscillator is stopped and converter will no longer respond to the host messages. Enter the Power-down mode by sending the power-down character Z (0x5A) followed by the two defined bytes, which are 0x5A and followed by 0xA5. If the exact message is not received, the device will not enter the power-down state.

Upon entering the power-down state, converter places the WAKEUP pin in a HIGH state. To have the device leave the power-down state, the WAKEUP pin should be brought LOW. A 1 kΩ resistor must be connected between the WAKEUP pin and V_{DD}.



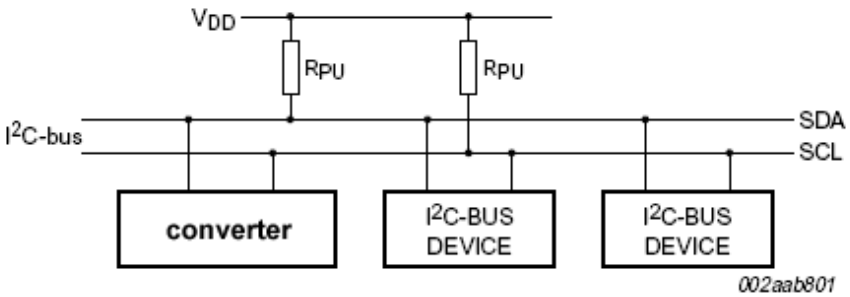
2.0 I2C-bus serial interface

The I2C-bus uses two wires (SDA and SCL) to transfer information between devices

connected to the bus, and it has the following features:

- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.

A typical I2C-bus configuration is shown in Figure. The converter device provides a byte-oriented I2C-bus interface that supports data transfers up to 400 kHz.



3. Internal registers available

3.1 Register summary

Register address	Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	R/W
General register set										
0x00	BRG0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	R/W
0x01	BRG1	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	R/W
0x02	PortConf1	GPIO3.1	GPIO3.0	GPIO2.1	GPIO2.0	GPIO1.1	GPIO1.0	GPIO0.1	GPIO0.0	R/W
0x03	PortConf2	GPIO7.1	GPIO7.0	GPIO6.1	GPIO6.0	GPIO5.1	GPIO5.0	GPIO4.1	GPIO4.0	R/W
0x04	IOState	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	R/W
0x05	reserved	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	-
0x06	I2CAdr	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	R/W
0x07	I2CClkL	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	R/W
0x08	I2CClkH	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	R/W
0x09	I2CTO	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TE	R/W
0x0A	I2CStat	1	1	1	1	I2CStat[3]	I2CStat[2]	I2CStat[1]	I2CStat[0]	R

3.2 Register descriptions

3.2.1 Baud Rate Generator (BRG)

The baud rate generator is an 8-bit counter that generates the data rate for the transmitter and the receiver. The rate is programmed through the BRG register and the baud rate can be calculated as follows:

$$\text{Baud rate} = \frac{7.3728 \times 10^6}{16 + (\text{BRG1, BRG0})}$$

Remark: To calculate the baud rate the values in the BRG registers must first be converted from hex to decimal.

Remark: For the new baud rate to take effect, both BRG0 and BRG1 must be written in sequence (BRG0, BRG1) with new values. The new baud rate will be in effect once BRG1 is written.

3.2.2 Programmable port configuration (PortConf1 and PortConf2)

GPIO port 0 to port 7 may be configured by software to one of four types. These are: quasi-bidirectional, push-pull, open-drain, and input-only. Two bits are used to select the desired configuration for each port pin. PortConf1 is used to select the configuration for GPIO3 to GPIO0, and PortConf2 is used to select the configuration for GPIO7 to GPIO4. A port pin has Schmitt triggered input that also has a glitch suppression circuit.

Port configurations

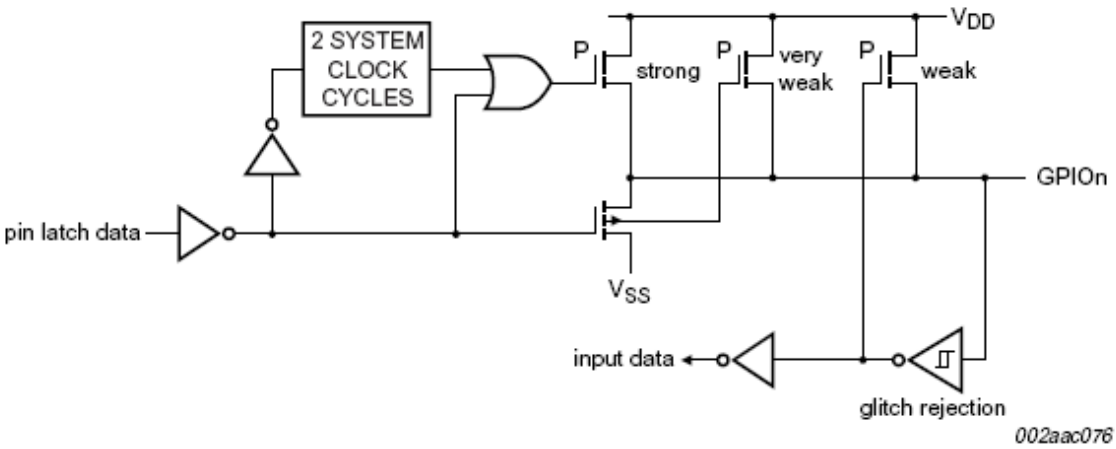
GPIOx.1	GPIOx.0	Port configuration
0	0	quasi-bidirectional output configuration
0	1	input-only configuration
1	0	push-pull output configuration
1	1	open-drain output configuration

3.2.2.1 Quasi-bidirectional output configuration

Quasi-bidirectional output type can be used as both an input and output without the need to reconfigure the port. This is possible because when the port outputs a logic HIGH, it is weakly driven, allowing an external device to pull the pin LOW. When the pin is driven LOW, it is driven strongly and able to sink a fairly large current. These features are somewhat similar to an open-drain output except that there are three pull-up transistors in the quasi-bidirectional output that serve different purposes.

The converter is a 3 V device, but the pins are 5 V tolerant. In quasi-bidirectional mode, if a user applies 5 V on the pin, there will be a current flowing from the pin to V_{DD}, causing extra power consumption. Therefore, applying 5 V in quasi-bidirectional mode is discouraged.

A quasi-bidirectional port pin has a Schmitt triggered input that also has a glitch suppression circuit.

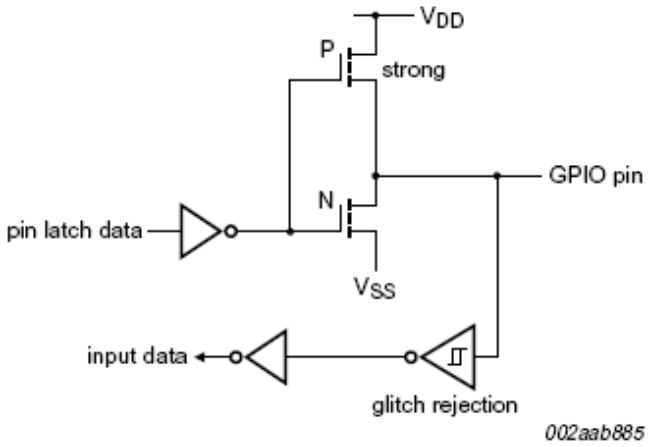


3.2.2.2 Input-only configuration

The input-only port configuration has no output drivers. It is a Schmitt triggered input that also has a glitch suppression circuit.

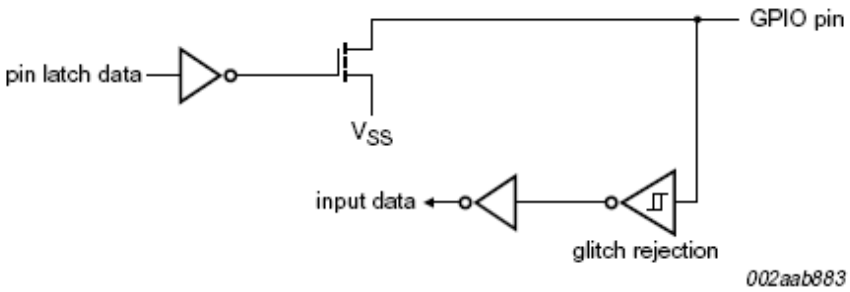
3.2.2.3 Push-pull output configuration

The push-pull output configuration has the same pull-down structure as both the open-drain and the quasi-bidirectional output modes, but provides a continuous strong pull-up when the port latch contains a logic 1. The push-pull mode may be used when more source current is needed from a port output. A push-pull port pin has a Schmitt triggered input that also has a glitch suppression circuit.



3.2.2.4 Open-drain output configuration

The open-drain output configuration turns off all pull-ups and only drives the pull-down transistor of the port driver when the port latch contains a logic 0. To be used as a logic output, a port configured in this manner must have an external pull-up, typically a resistor tied to V_{DD}. An open-drain port pin has a Schmitt triggered input that also has a glitch s uppression circuit.



3.2.3 Programmable I/O pins state register (IOState)

When read, this register returns the actual state of all I/O pins. When written, each register bit will be transferred to the corresponding I/O pin programmed as output.

IOState - Programmable I/O pins state register (address 0x04h) bit description

Bit	Symbol	Description
7:0	IOLevel	Set the logic level on the outputs pins. Write to this register:

		logic 0 = set output pin to zero logic 1 = set output pin to one Read this register returns states of all pins.

3.2.4 I2C-bus address register (I2CAdr)

The contents of the register represents the device's own I2C-bus address. The most significant bit corresponds to the first bit received from the I2C-bus after a START condition. A logic 1 in I2CAdr corresponds to a HIGH level on the I2C-bus, and a logic 0 corresponds to a LOW level on the I2C-bus. The least significant bit is not used, but should be programmed with a '0'.

I2CAdr is not needed for device operation, but should be configured so that its address does not conflict with an I2C-bus device address used by the bus master.

3.2.5 I2C-bus clock rates (I2CClk)

This register determines the serial clock frequency. The various serial rates table.

The frequency can be determined using the following formula:

$$\text{bit frequency} = \frac{7.3728 \times 10^6}{2 \times (\text{I2CClkH} + \text{I2CClkL})}$$

I2CClkH determines the SCL HIGH period, and I2CClkL determines the SCL LOW period.

I2C-bus clock frequency

I2CCIk (I2CCIkH+I2CCIkL)	I²C-bus clock frequency (kHz)
10 (minimum)	369
15	246
25	147
30	123
50	74
60	61
100	37

Remark: The numbers used in the formulas are in decimal, but the numbers to program I2CClkH and I2CClkL are in hex.

3.2.6 I2C-bus time-out (I2CTO)

The time-out register is used to determine the maximum time that SCL is allowed to be LOW before the I2C-bus state machine is reset.

When the I2C-bus interface is running, I2CTO is loaded after each I2C-bus state transition.

I2CTO - I2C-bus time-out register (address 0x09h) bit description

Bit	Symbol	Description
7:1	TO[7:1]	time-out value
0	TE	enable/disable time-out function logic 0 = disable logic 1 = enable

The least significant bit of I2CTO (TE bit) is used as a time-out enable/disable. A logic 1

$$\text{time-out period} = \frac{\text{I2CTO}[7:1] \times 256}{57600} \text{ seconds}$$

will enable the time-out function. The time-out period can be calculated as follows:

The time-out value may vary, and this is an approximate value.

3.2.7 I2C-bus status register

This register reports the I2C-bus transmit and receive frame status, whether the frame transmit correctly or not.

I2C-bus status

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	I²C-bus Statusbeschreibung
1	1	1	1	0	0	0	0	I2C_OK
1	1	1	1	0	0	0	1	I2C_NACK_ON_ADDRESS
1	1	1	1	0	0	1	0	I2C_NACK_ON_DATA
1	1	1	1	1	0	0	0	I”C_TIME_OUT

Trademarks