I²C communication and cybersecurity

XENSIV™ hydrogen gas sensors

About this document

Scope and purpose

This application note assists the developer in correctly configuring I²C communication between the host controller and XENSIV™ TCI-x hydrogen (H_2) gas sensors (TCI-x stands for TCI and TCI-B). It also includes a cybersecurity declaration. Code examples for the host controller are provided at the end of this application note.

Intended audience

The intended audiences for this document are design engineers, technicians, and developers of electronic systems.

About this product family

Product family

XENSIV™ hydrogen gas sensors based on the integrated thermal conductivity principle.

Target applications

- Automotive battery monitoring systems (thermal runaway detection)
- Automotive fuel cell hydrogen leakage measurement
- Industrial hydrogen leakage measurement (e.g., electrolyzers, energy storage systems, stationary fuel cells)

Table of contents

About this document **1**
About this product family **1**
Table of contents **2**
1 Application circuit **3**
2 I²C communication **5**
2.1 General I²C data exchange 5
2.2 I²C communication with TCI-x 5
2.2.1 TCI-B I²C interface limitations 5
2.2.2 Write/read communication cycle 6
2.2.1 Triggered operating mode 6
2.2.2 Autonomous operating mode (TCI-B only) 7
2.2.3 AS program flow 8
2.2.4 Pull-up resistor calculation 9
3 Cybersecurity declaration (TCI-B) **11**
3.1 Preface 11
3.2 Declaration 11
4 Code examples **12**
References **16**
Revision history **17**
Disclaimer **18**

1 Application circuit

Figure 1 shows a typical application circuit. XENSIV™ TCI-x is connected to a host controller (MCU) via I²C and the optional signals BUSY, WAKEHOST, and NACTIVE (TCI-B) or STDBY (TCI). The host controller is also connected to a pressure sensor (p) and a humidity sensor (RH). To obtain a pressure- and humidity-compensated concentration reading, the host controller must read a pressure value from the p-sensor and a humidity and temperature value from the RH-sensor and then send these three values to the TCI-x via an I²C write command. TCI-x executes a concentration measurement and then uses the external sensor values for running a compensation algorithm. After a certain measurement and processing time, the host controller can fetch the compensated concentration from the TCI via an I²C read command.

The I²C pull-up resistors R_p depend on the chosen I²C speed and the capacitive load at the I²C lines, see Section 2.2.4.

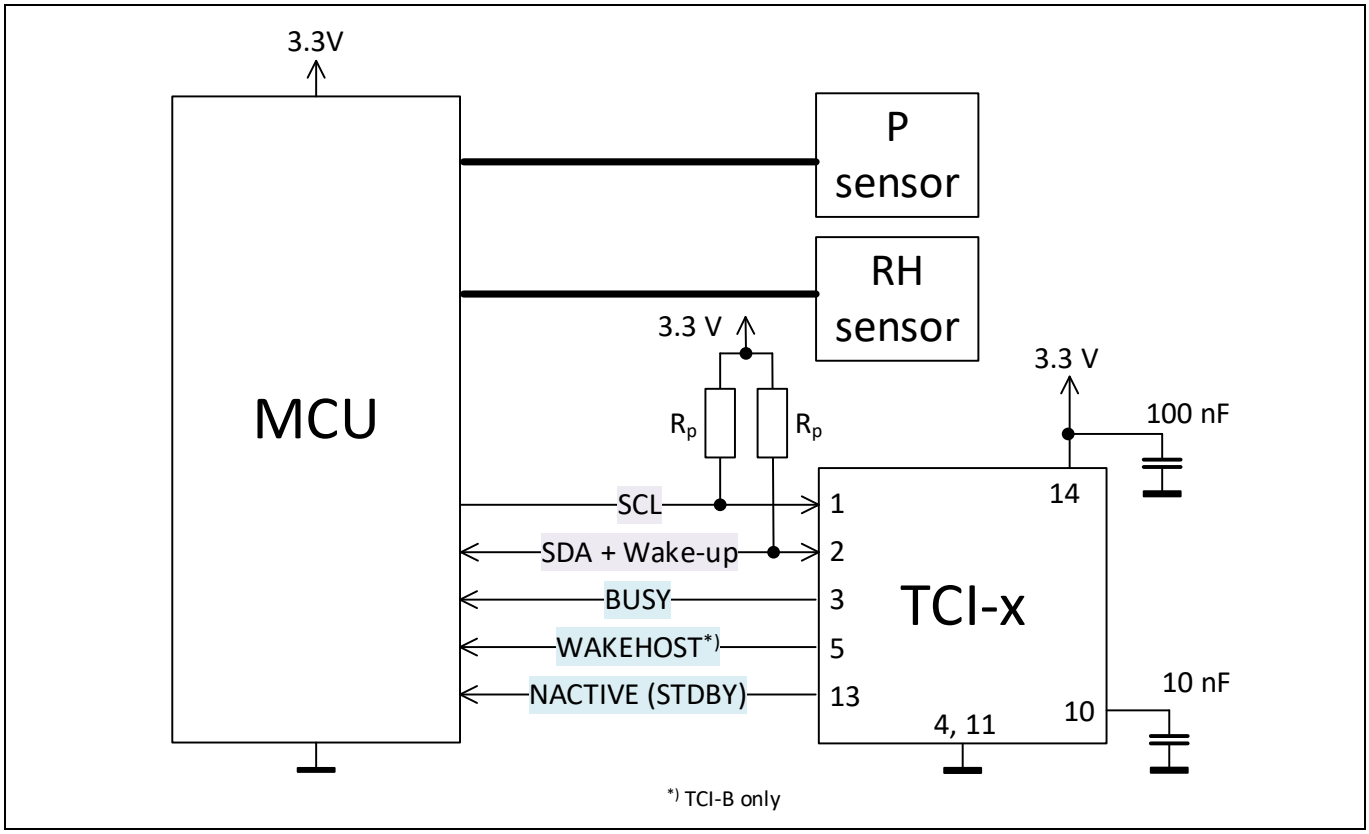


Figure 1 Application circuit

Note: The TCI-x pin configuration is the default configuration after power-on.

Table 1 shows the digital signals of TCI-x. The bold printed pin numbers in this table are the default pins for each signal. Pins can be changed with the Configuration command. Signals can share the same pin if assigned to the same pin by configuration.

Note: Only TCI-B: If Pin 3 is used as the wake-up pin, the BUSY signal is automatically routed to Pin 13 and shared with the NACTIVE signal.

Table 1 TCI-x digital pins

Signal	TCI-x in/out	TCI pin #	TCI-B pin #	Comment
I²C SCL	Input	**1**	**1**	
I²C SDA	In/out	**2**	**2**	In the default configuration, the SDA and the wake-up functions share Pin 2. If assigned to Pin 2, the wake-up signal is always active LOW
Wake-up	Input	**2** or 3	**2** or 3	
WAKEHOST	Output	-	**5**	This signal goes HIGH when the TCI-B is operating autonomously and generates an alarm
NACTIVE (= STDBY on TCI)	Output	3 or 5 or **13**	3 or **13**	This signal goes HIGH when the device is on Stand-by or in Autonomous state
BUSY	Output	**3** or 5 or 13	**3** or 13	This signal indicates command execution and I²C is inactive

2 I²C communication

Note that TCI-x follows the I²C bus specification [3]. For better understanding, the timing diagram is shown here.

2.1 General I²C data exchange

Figure 2 shows the I²C data transfer diagram. The clock signal (SCL) is generated by the master device (host controller); the data signal (SDA) is either generated by the master or by the slave device (TCI-x).

The “Data Write” communication in the upper part of Figure 2 is used to send data from the master device to the slave device. The slave device generates an acknowledge bit after each data byte.

The “Data Read” communication in the lower part of Figure 2 is used to read data from the slave device. The slave device generates the acknowledge bit after the address byte. However, the acknowledge after each data byte is now generated by the master device.

During the data exchange, the SDA line may change only while the SCL line is LOW. If the SDA level changes while the SCL line is HIGH, this is interpreted as a start or stop condition. The start and stop conditions are generated by the master device.

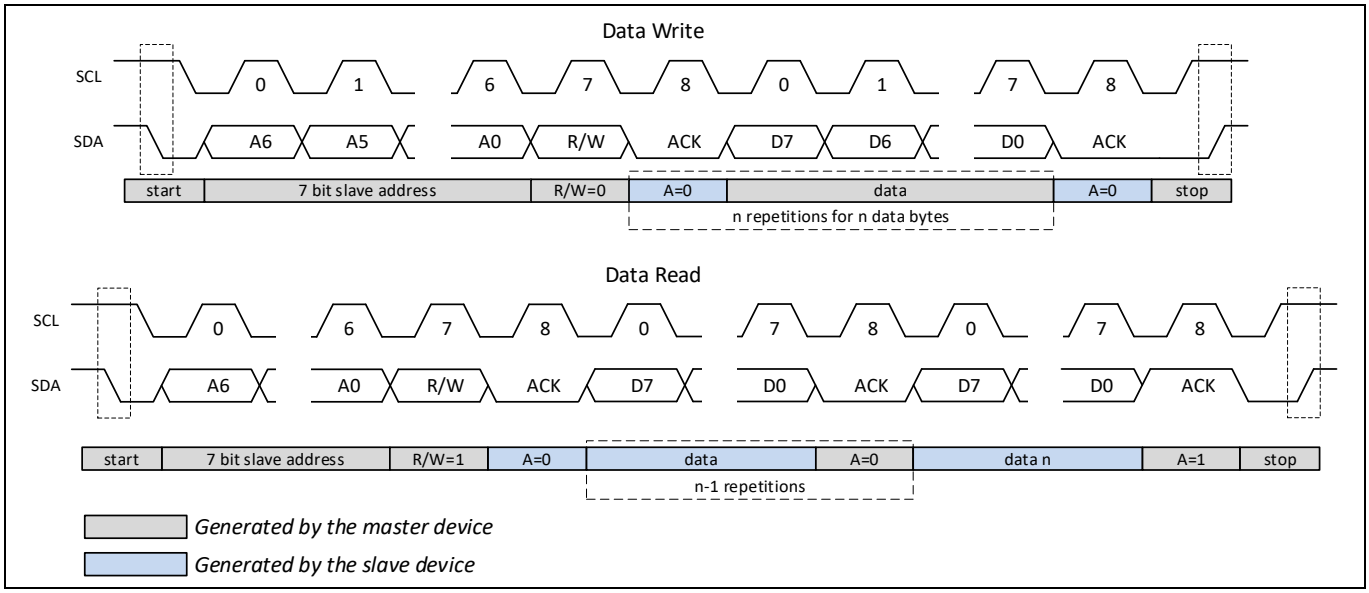


Figure 2 I²C data exchange diagram

2.2 I²C communication with TCI-x

2.2.1 TCI-B I²C interface limitations

Due to current limitations of the SDA and SCL pins, the minimum pull-up resistors are limited. This in turn means that a maximum capacitance at the SDA or SCL pin must not exceed because that would violate the I²C timing specifications. See Section 2.2.4 for details. Also note that other devices on the I²C bus may limit the minimum pull-up resistors because their output stages can have even lower current specifications than TCI-x. A high inner resistance of the output stage of some devices on the I²C bus may limit the minimum pull-up resistors.

TCI-x has an optional spike filter for the I²C interface. The spike filter can lower the transmission error rate if interference is expected on the SDA or SCL line in the application. The spike filter is disabled by default because for short I²C connections on a PCB it is not required. However, it can be enabled via the Configuration command. If the spike filter is enabled, the maximum allowed clock rate is 800 kHz, i.e., Fast mode plus is not available.

2.2.2 Write/read communication cycle

Communication between the host controller and TCI-x essentially consists of a series of I²C write and read commands. Between each pair of write/read commands, an execution time for the command must be allowed. During the command execution, the I²C interface of TCI-x is inactive, i.e., no command will be acknowledged. This execution phase is signaled by the BUSY signal. The write/read cycle is shown in Figure 3.

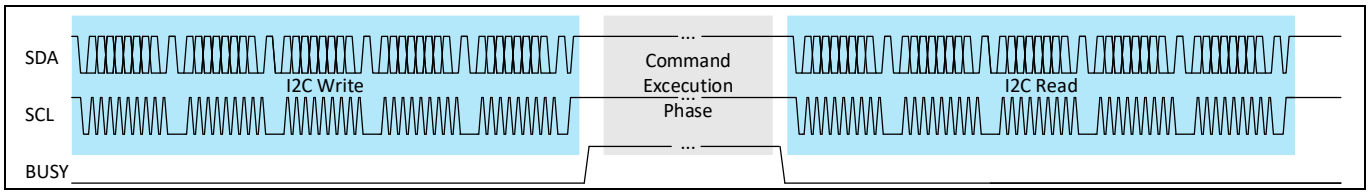


Figure 3 I²C write/read cycle

2.2.1 Triggered operating mode

In triggered operating mode, the host controller must send an I²C trigger command for each single concentration or temperature measurement. After each trigger, the host controller must wait for the measurement execution time and then the result must be fetched with an I²C read command.

There are two methods to account for the instruction execution time:

- Wait for the maximum execution time specified in the datasheet. In this case, the BUSY signal can be ignored; see Figure 4 ①
- Query the BUSY signal and wait for its falling edge; see Figure 4 ② and ③

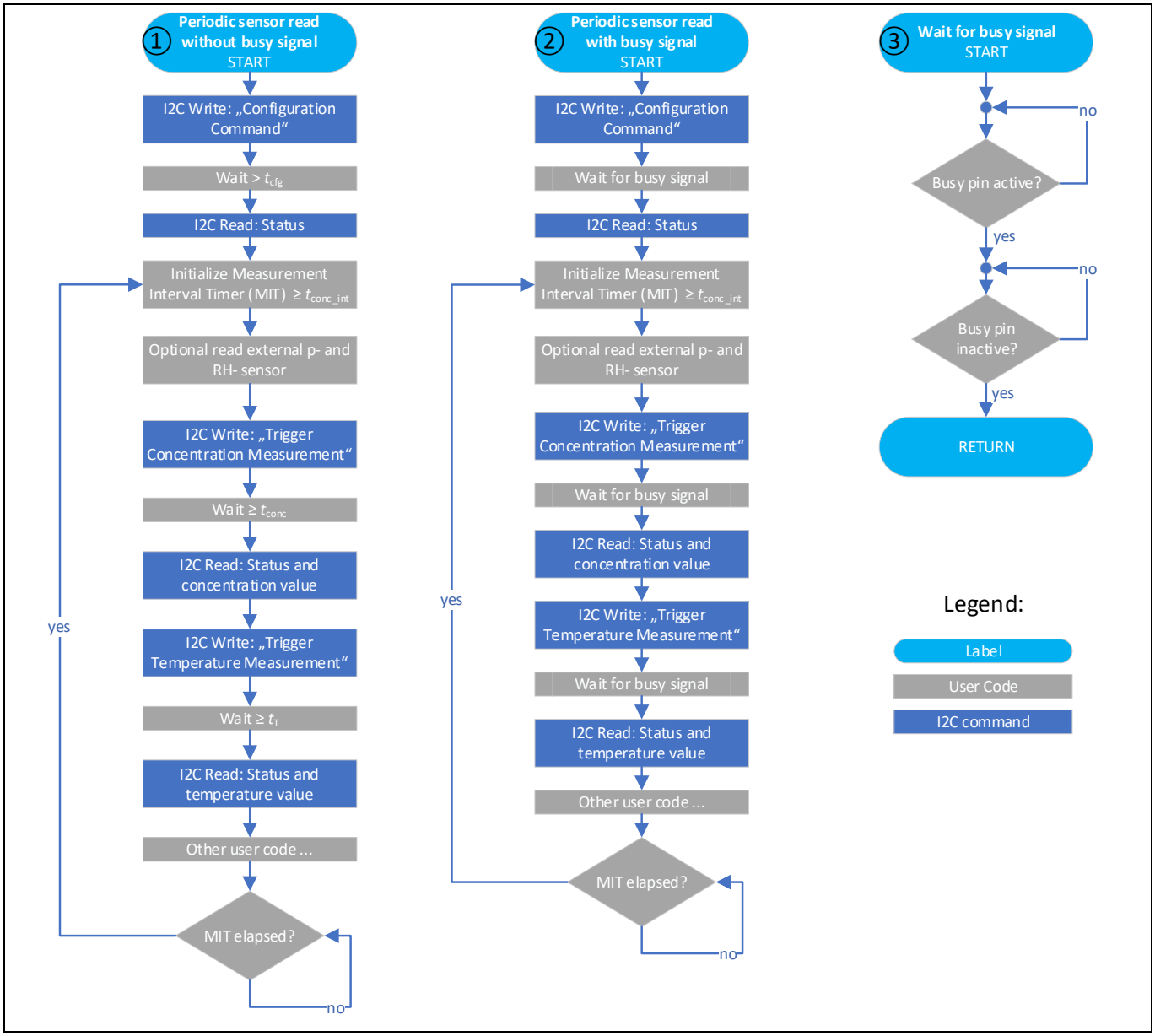


Figure 4 Example program flow in triggered operating mode

2.2.2 Autonomous operating mode (TCI-B only)

In autonomous operating mode, TCI-B is set into autonomous state (AS); the host controller can go to a sleep state. In AS, TCI-B carries out concentration measurements automatically and wakes up the host controller in case the measured concentration exceeds a configurable threshold. For this purpose, the host controller needs to be connected to the WAKEHOST pin of TCI-B.

The WAKEHOST pin is active HIGH, i.e., the host controller must be configured to wake up on HIGH level. The WAKEHOST pin goes HIGH in any alarm situation, either if a concentration threshold is exceeded or if an internal diagnosis function detects a malfunction of TCI-B. After such an alarm, the WAKEHOST pin stays HIGH until the next valid I²C command is received from the master device.

The WAKEHOST pin also goes HIGH after a reset of TCI-B and will go LOW only after the first valid I²C command is received.

2.2.3 AS program flow

The following assumptions are made for the flow diagram in Figure 5

- TCI-B is configured to use SDA line for wake-up
- The host controller is periodically woken up by an internal timer. The time intervals are long compared to the TCI-B concentration measurement sampling time and short compared to the time required for a significant change in ambient pressure and humidity
- TCI-B is configured to stop AS at alarm

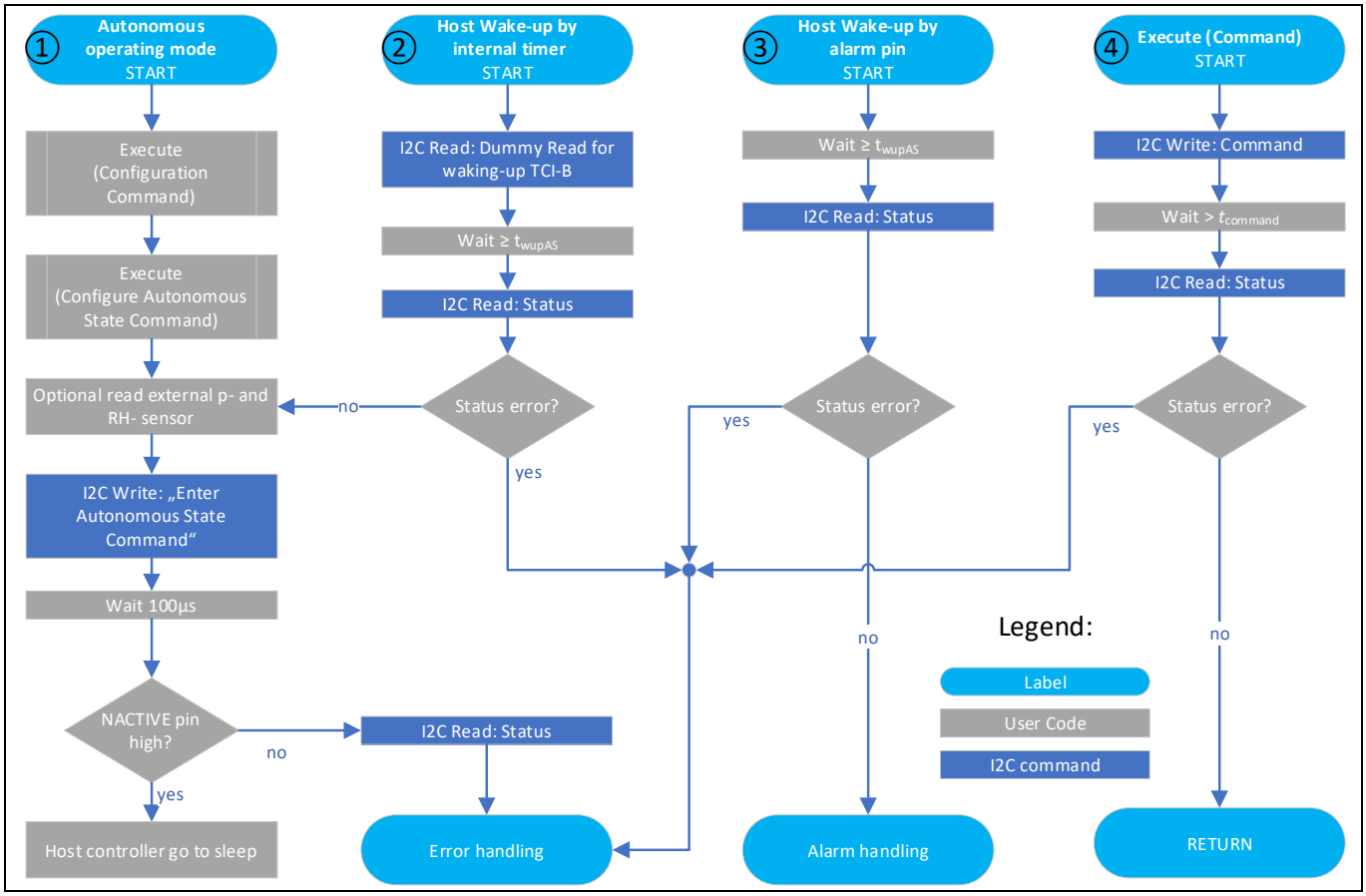


Figure 5 Example program flow in AS

The flow in Figure 5 ① carries out the TCI-B configurations, then reads the external pressure and humidity values, and finally sends these values to TCI-B and sets it into AS. After 100 µs, the host controller tests the NACTIVE pin to check if TCI-B indeed entered AS.

The upper part of Figure 5 ② describes the wake-up of the host controller triggered by the internal timer. After waking up and checking the status of TCI-B, the sensor is sent back to AS.

Figure 5 ③ shows the flow which is executed after a wake-up of the host controller triggered by the alarm pin of TCI-B. The host controller must read the TCI-B status to check if the alarm is caused by a sensor error or due to exceeding a threshold. In the latter case, the host controller enters the alarm handling routine. In this routine, the host controller can read the past concentration measurements from TCI-B or trigger a new concentration measurement to verify the alarm.

Figure 5 ④ is just a general subroutine for sending a command to TCI-B and checking the status afterwards. It is called twice from flow ①.

2.2.4 Pull-up resistor calculation

The I²C interface requires a suitable pull-up resistor R_p on both the SDA and SCL lines, because the I²C connections of all devices on an I²C bus are of the open-drain type (see the application circuit, Figure 1).

The I²C standard specifies a maximum fall and rise times of the signals on the SCL and SDA lines. Fall and rise times are different for the three I²C operating speeds: Standard mode, Fast mode, and Fast mode Plus.

Both fall and rise times are defined as the time for the signal to change between 30% of VDD to 70% of VDD.

Table 1 Rise and fall times vs. mode

Parameter	Standard mode	Fast mode	Fast mode Plus
Max. rise time t_r	1000 ns	300 ns	120 ns
Max. fall time t_f	300 ns	300 ns	120 ns

The rise time depends on the capacitive load on the SDA or SCL line (C_{load}) and the pull-up resistance R_p.

The fall time mainly depends on the resistance R_{DSon} of the open-drain transistor at the SDA/SCL line and is lower than the rise time in most cases because $R_{DSon} \ll R_{p_t}$.

The minimum allowed R_{p_t} depends on the maximum current I_{DS_max} through the transistors. For TCI-x, the current $I_{DS_max} = 2$ mA. Therefore, the minimum allowed R_{p_t} is 1650 Ω.

The maximum total pull-up resistor R_{p_t} time is given by the following formula:

$$R_{p_t} = \frac{1.18 t_r}{C_{load}}$$

Note that R_{p_t} is composed of all pull-up resistors at the SDA/SCL line, i.e., R_p and possible internal pull-up resistors of all other devices 1 to *n* on the bus:

$$R_{p_t} = \left[\frac{1}{R_p} + \frac{1}{R_{int_TCI}} + \frac{1}{R_{int_1}} + \cdots + \frac{1}{R_{int_n}}\right]^{-1}$$

Equation 1

The maximum R_{int_TCI} is 11 kΩ.

If all other devices on the bus have no internal pull-up resistor, then:

If $R_{p_t} \geq 11k\Omega$: R_p is not required.

If $R_{p_t} < 11k\Omega$: $R_p = \left[\frac{1}{R_{p_t}} - \frac{1}{R_{int_TCI}}\right]^{-1}$

Example 1

$C_{load} = 30$ pF, $t_r = 300$ ns (Fast Mode) → $R_{p_t} = 11.8\ k\Omega$.

Since the required R_{p_t} is larger than R_{int_TCI} no external pull-up is required.

Example 2

$C_{load} = 80$ pF, $t_r = 120$ ns (Fast Mode Plus) → $R_{p_t} = 1.77$ kΩ.

In this case, an external pull-up of $R_p = 2.1$ kΩ is necessary.

3 Cybersecurity declaration (TCI-B)

This section refers to the user flash memory feature of TCI-B (see Write and Read User Flash Memory Commands in the XENSIV™ TCI-B datasheet [1]).

3.1 Preface

A TARA (Threat Analysis and Risk Assessment) has been performed as the primary analysis method to determine the cybersecurity relevance of TCI-B. According to ISO/SAE 21434:2021 clause 15, the TARA has the objective to determine the extent to which a road user can be impacted by a threat scenario. The TARA is a method that involves the identification of assets, definitions of damage scenarios and their impact ratings, determination of threat scenarios including their attack paths with feasibility ratings and eventually risk value determination for each threat scenario. Based on each determined risk, corresponding risk treatment decisions are made in terms of risk avoidance, reduction, sharing or retention.

Cybersecurity claims are specified where cybersecurity risks are shared or retained. Cybersecurity risks which are rated to be reduced would lead to cybersecurity requirements and corresponding cybersecurity activities according to an ISO/SAE 21434 lifecycle. However, during the TARA for the product(s) in scope of this document, no cybersecurity requirements have been identified and, hence, no further cybersecurity activities have been triggered.

3.2 Declaration

ISO 21434 is not in scope of the TCI-B project. No cybersecurity requirements or measures have been considered or implemented during the lifecycle of the project. The responsibility for the confidentiality and integrity of the data stored by customers in the FLASH area allocated for the Customer FLASH storage feature goes directly to the customer with no Infineon involvement.

This feature or the exclusion of ISO 21434 from the scope of the product has no impact with regard to the fulfillment of the ISO 26262 TLSRs or the overall Functional Safety of the product.

4 Code examples

This section presents C code that runs on the host controller to read concentration and temperature values from TCI-x. The hardware-specific functions are placeholders and must be implemented by the user for the specific hardware.

Code listing 1 Code example

```
//Defines
//define the pin of your host controller which is connected to the BUSY pin
#define BUSY_PIN tbd //defined by user

// Hardware-specific functions; must be defined for your hardware
// I²C Write Command
void I2C_Write(uint8_t addr, uint8_t *data, uint8_t length);
// I2C Read Command
void I2C_Read(uint8_t addr, uint8_t *data, uint8_t length);
// Delay function in milliseconds
void delayMS(uint32_t time);
// Delay function in microseconds
void delayUS(uint32_t time);
// Return the state of the pin PIN: 1 if HIGH, 0 if LOW
uint8_t readPinState(uint8_t PIN);

// Definition of global variables and constants
// TCI I2C Address
const uint8_t TCI_I2C_ADDR = 0x36;
// Relative Humidity, Temperature and Pressure Values
int32_t RH_Value, T_Value, P_Value;
// Concentration Measurement Result from TCI
int16_t concentrationResult_TCI;
// Temperature Measurement Result from TCI
int8_t temperatureResult_TCI;

// Function definitions
// CRC Computing function
uint16_t CRC16_CCITT_FALSE(uint8_t *pData, int length)
{
   uint8_t i;
   uint16_t wCrc = 0xffff;
   while (length--) {
       wCrc ^= *(unsigned char *)pData++ << 8;
       for (i=0; i < 8; i++)
           wCrc = wCrc & 0x8000 ? (wCrc << 1) ^ 0x1021 : wCrc << 1;
   }
   return wCrc & 0xffff;
}

// Check status of Busy pin. Function shall return 1 if BUSY pin is “H”.
uint8_t checkBusyPin()
{
   return readPinState(BUSY_PIN);
}

// Configuration Command. The settings used in this example set the sensor to
// its default settings (= settings after power on). Note that TCI and TCI-B
// have different configuration options (see [1] and [2]).
void configurationCommand()
{
```

```
    uint8_t I2C_CMD[6];
    // Command ID
    I2C_CMD[0] = 0xC4;
    // Spike filter disabled. Pin 2 (SDA) used as wake-up pin. BUSY signal on
    // pin 3. Automatic Stand-By disabled. Stand-by signal on pin 13.
    I2C_CMD[1] = 0x00; // Cfg_1
    I2C_CMD[2] = 0x01; // Cfg_2. Needed for TCI only, set 0 for TCI-B
    I2C_CMD[3] = 0x03; // Cfg_3. Needed for TCI only, set 0 for TCI-B
    // Calculate CRC for the first 4 bytes and append to command string
    uint16_t CRC = CRC16_CCITT_FALSE(I2C_CMD, 4);
    I2C_CMD[4] = (uint8_t)(CRC >> 8);
    I2C_CMD[5] = (uint8_t)CRC;
    // Send the I2C_CMD array to TCI
    I2C_Write(TCI_I2C_ADDR, I2C_CMD, 6);
}

// Check configuration command; function returns 0 if everything is ok
uint8_t checkConfigSuccessful()
{
    uint8_t I2C_Response[3];
    // Read the response from configurationCommand()
    I2C_Read(TCI_I2C_ADDR, I2C_Response, 3);
    // Check if configuration command was successful
    if (I2C_CMD[0] || CRC16_CCITT_FALSE(I2C_Response, 3))
        return 1;
    else
        return 0;
}

// Trigger concentration measurement
void triggerConcentrationMeasurement(uint8_t RH, int8_t T, uint8 _t P)
{
    uint8_t I2C_CMD[7];
    // Trigger Concentration Measurement Command ID
    I2C_CMD[0] = 0xA8;
    // Configuration Byte for fully compensated H2 measurement and field
    // contamination check enabled.
    I2C_CMD[1] = 0x20;
    // Relative humidity value
    I2C_CMD[2] = RH;
    // Temperature value (from humidity sensor)
    I2C_CMD[3] = (uint8_t)T;
    // Pressure Data
    I2C_CMD[4] = P;
    // Calculate CRC for the first 5 bytes and append to command string
    uint16_t CRC = CRC16_CCITT_FALSE(I2C_CMD, 5);
    I2C_CMD[5] = (uint8_t)(CRC >> 8);
    I2C_CMD[6] = (uint8_t)CRC;
    // Send the I2C_CMD array to TCI
    I2C_Write(TCI_I2C_ADDR, I2C_CMD, 7);
}

// Read concentration meas. result. Function returns 0 if everything is ok.
int8_t readConcentrationMeasurement(int16_t *Conc)
{
    uint8_t I2C_Response[5];
    I2C_Read(TCI_I2C_ADDR, I2C_Response, 5);
    // Check if measurement command was successful
    if (I2C_CMD[0] || CRC16_CCITT_FALSE(I2C_Response, 5))
```

```
        return 1;
    else
        {
            // concatenate Conc_H and Conc_L fields
            *Conc = (uint16_t)I2C_Response[1] << 8 | I2C_Response[2];
            return 0;
        }
    }

    // Trigger temperature measurement
    void triggerTemperatureMeasurement()
    {
        uint8_t I2C_CMD[3];
        // Trigger Temperature Measurement Command ID
        I2C_CMD[0] = 0xA9;
        // Calculate CRC of command ID and append to command string
        uint16_t CRC = CRC16_CCITT_FALSE(I2C_CMD, 1);
        I2C_CMD[1] = (uint8_t)(CRC >> 8);
        I2C_CMD[2] = (uint8_t)CRC;
        // Send the I2C_CMD array to TCI
        I2C_Write(TCI_I2C_ADDR, I2C_CMD, 3);
    }

    // Read temperature meas. Result. Function returns 0 if everything is ok.
    int8_t readTemperatureMeasurement(int8_t *Temp)
    {
        uint8_t I2C_Response[4];
        I2C_Read(TCI_I2C_ADDR, I2C_Response, 4);
        // Check if measurement command was successful
        if (I2C_CMD[0] || CRC16_CCITT_FALSE(I2C_Response, 4))
            return 1;
        else
        {
            *Temp = I2C_Response[1];
            return 0;
        }
    }

    // Configure TCI by waiting for the maximal command execution time (mode=0)
    // or by polling the BUSY signal (mode > 0).
    uint8_t configTCI(uint8_t mode)
    {
        configurationCommand();
        if (mode == 0)
            delayUS(100);
        else
        {
            delayUS(40);
            while (checkBusyPin())
                ;
        }
        return checkConfigSuccessful();
    }

    // Execute measurements by waiting for the maximal command execution time
    // (mode=0) or by polling the BUSY signal (mode > 0).
    // Function returns 0 is everything is ok, returns 1 if concentration
    // measurement  is nok, and returns 2 if temperature measurement is nok.
    uint8_t measurementTCI(uint8_t mode)
```

```
    {
        uint8_t status;
        // Gas concentration measurement
        triggerConcentrationMeasurement(RH_Value, T_Value, P_Value);
        if (mode == 0)
            delayMS(30);
        else
        {
            delayUS(40);
            while (checkBusyPin())
                ;
        }
        status = readConcentrationMeasurement(&concentrationResult_TCI);
        // Temperature Measurement
        triggerTemperatureMeasurement();
        if (mode == 0)
            delayMS(1);
        else
        {
            delayUS(40);
            while (checkBusyPin())
                ;
        }
        status += 2 * readTemperatureMeasurement(&temperatureResult_TCI);
        return status;
    }
```

References

[1] Infineon Technologies AG: *XENSIV™ TCI-B hydrogen gas sensor datasheet;* Available online

[2] Infineon Technologies AG: *XENSIV™ TCI hydrogen gas sensor datasheet;* Available online

[3] NXP Semiconductors: *UM10204: I²C-bus specification and user manual*; Available online

Revision history

Document revision	Date	Description of changes
1.00	2026-11-06	Initial release

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition: 2026-11-06

Published by
Infineon Technologies AG
Am Campeon 1-15
85579 Neubiberg, Germany

© 2026 Infineon Technologies AG.
All rights reserved

Do you have a question about this document?
Email: erratum@infineon.com

Document reference:
AN120630

Important Notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.

Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.

Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).

If the Product includes security features:
Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:
The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved.

Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit https://www.infineon.com.