131 E. Brokaw Road | San Jose CA 95112 | +1 (408) 392-0989 | info@posifatech.com | www.posifatech.com

I²C Specification for PGS1000 Hydrogen Sensor

1 Interface Connection

The PGS1000 module includes a two-wire I²C digital interface with a bidirectional data line (SDA) and a clock line (SCL). The two lines are open-drain and connected to the power supply (Vdd) via two pull-up resistors (Rp). In a system with a master-slave configuration, the Posifa sensor module is the slave.

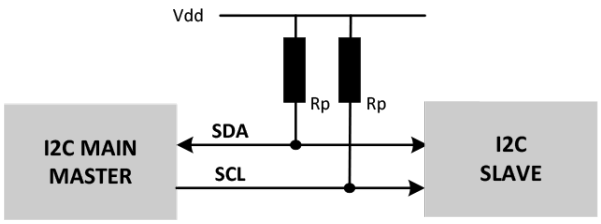


Figure 1: I²C master-slave configuration

The recommended pull-up resistor (Rp) values depend on the system implementation, but a value between 1 kΩ and 10 kΩ can be used for prototyping. Please refer to NXP's I²C specification for more information.

The capacitive load on both the SDA and SCL should be the same, so the signal lengths should be similar to avoid asymmetry. Using shielded cable is recommended for wire lengths above 10 cm and I²C buffers should be used if signal paths are longer than 30 cm.

The I²C clock speed is 100 kHz.

2 I²C Address

The PGS1000 module uses a 7-bit addressing scheme. The address is always followed by a read (1) or write (0) bit. The module's default I²C address is 0x50.

131 E. Brokaw Road | San Jose CA 95112 | +1 (408) 392-0989 | info@posifatech.com | www.posifatech.com

3 I²C Communication

Each I²C transaction consists of a start bit, followed by the 7-bit address and a read or write bit. At the end of a transmission, a stop bit is sent from the master to terminate the communication. An acknowledgement is expected from the slave in between each byte (8 bits) in a transmission.

3.1 Transmission START Condition (S)

The START condition is used to initiate I²C communication by the master. A HIGH to LOW transition on the SDA line while the SCL is HIGH indicates the beginning of a transmission.

3.2 Transmission STOP Condition (P)

The STOP condition is used to stop I²C communication by the master. A LOW to HIGH transition on the SDA line while the SCL is HIGH indicates the end of a transmission. The bus is free after a STOP condition.

3.3 Acknowledge (ACK) / Not Acknowledge (NACK)

The master expects an ACK back from the slave after each byte is transmitted over the I²C bus. The slave pulls the SDA low to indicate that it has received a byte and then it frees the I²C bus again. If the slave does not initiate an ACK, it is considered a NACK.

3.4 Data Transfer Format

Data is transferred in byte packages, i.e., in frames of 8-bit length. Each byte is followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first.

The data transfer format is shown in the figure “I²C Read and Write.” A data transfer sequence is initiated by the master producing the START condition (S) and sending a header byte. The I²C header is made up of the 7-bit I²C device address and a data direction bit (R/_W).

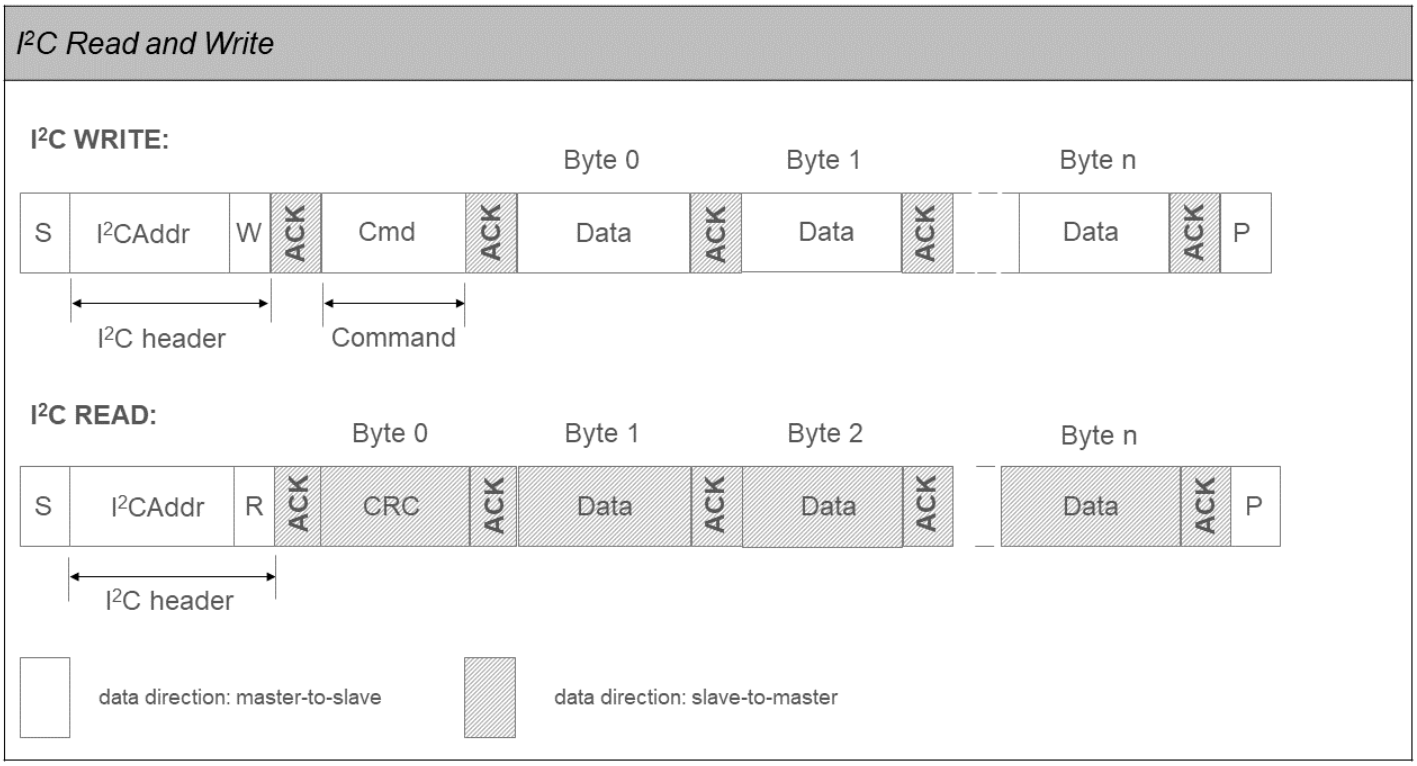
The value of the R/_W bit in the header determines the data direction for the entire rest of the data transfer sequence: if R/_W = 0 (WRITE) the direction remains master-to-slave; if R/_W = 1 (READ) the direction changes to slave-to-master after the header byte.

3.5 Data Types and Representation

Unless otherwise specified, only unsigned, 16-bit integers are used.

An unsigned, 16-bit integer is represented as two bytes in either a big-endian or little-endian format, depending on the sensor command. Please refer to section 4.1 for the data representation for each command.

131 E. Brokaw Road | San Jose CA 95112 | +1 (408) 392-0989 | info@posifatech.com | www.posifatech.com



3.6 Checksum

Posifa sensors implement a slave-to-master checksum. We highly recommend implementing the checksum in your code and checking each answer from the sensor for the correct checksum. The checksum is the first byte following the 7-bit I²C device address and the data direction bit (R/_W = 1) in the slave-to-master data transfer.

The checksum is the 2's complement (negative) of the 256-modulo (8-bit) sum of the data bytes (does not include the I²C address). This can be calculated using:

checksum = 1 + ~(sum)

Example:

If the I²C payload bytes from a normal read operation are { 0xC9, 0x0B, 0x28, 0x04, 0x00 }, the 256-modulo (8-bit) sum is calculated as:

sum = 0x0B + 0x28 + 0x04 + 0x00 = 0x37

Then the checksum is calculated as:

131 E. Brokaw Road | San Jose CA 95112 | +1 (408) 392-0989 | info@posifatech.com | www.posifatech.com

checksum = 0x01 + ~(0x37) = 0x01 + 0xC8 = 0xC9

Validating the data payload is done by calculating the sum and adding it to the checksum. If the result is 0x00, then the data is valid:

checksum + sum = 0xC9 + 0x37 = 0x00

4 Command Set and Data Transfer Sequences

4.1 Sensor Commands

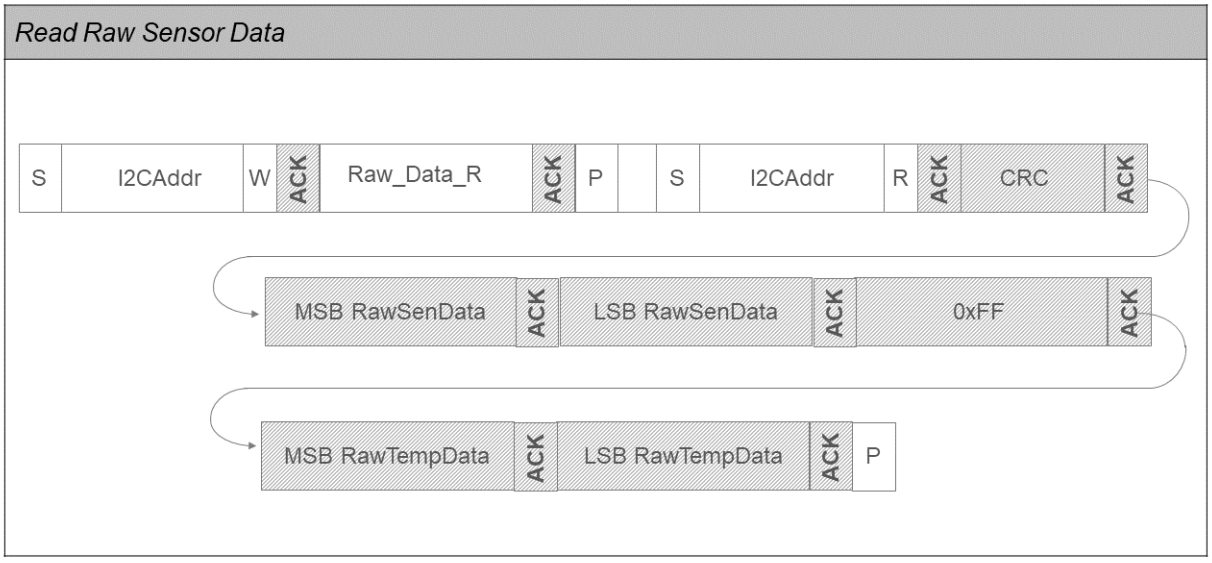
The following commands are supported. The usage details are provided in the remainder of Section 4.

Read Command	Value	Representation	Read Command Description
Cal_Data_R	Nill	Big-endian	Read calibrated sensor data
Raw_Data_R	0xD0	Big-endian	Read raw sensor data

4.2 Read Raw Sensor Data

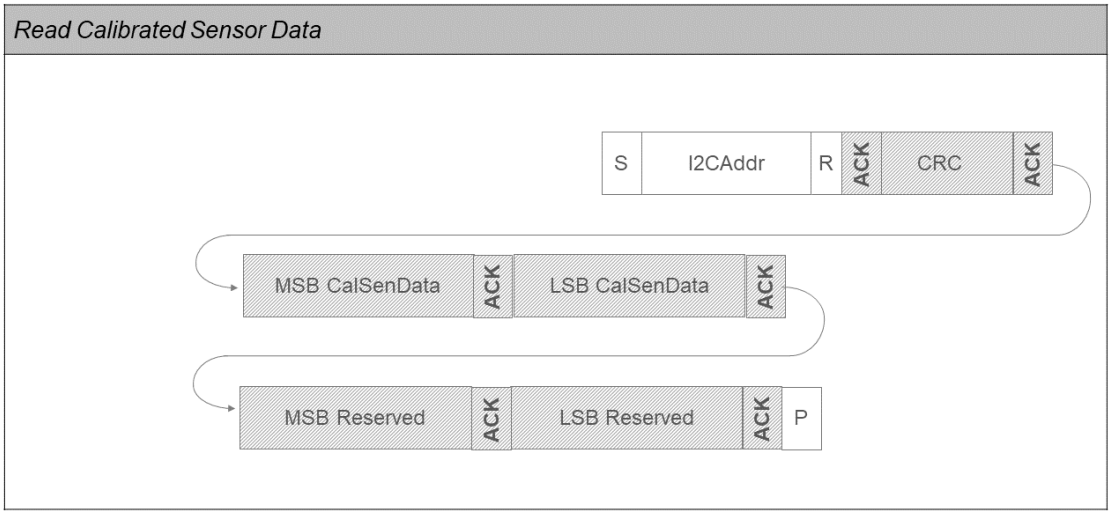
The command Raw_Data_R returns the raw sensor data and the raw temperature data. The raw sensor data comes from the sensing element and is uncalibrated, but temperature-compensated to the baseline temperature.

The raw temperature data comes from the sensor MCU's internal temperature sensor, and is uncalibrated.



4.3 Read Calibrated Sensor Data

The command Cal_Data_R returns the calibrated sensor data. A lookup-table-based piecewise linearization function is used to convert the raw sensor data to the corresponding calibrated value. Please reference the PGS1000 data sheet regarding the interpretation of calibrated value.



5 Limitations

The I²C bus is susceptible to noise and can lock up, especially if there are glitches on the SCL or the master does not acknowledge the first byte sent from the slave.

The following guidelines are best practices for the I²C bus and to avoid lock-ups:

- Minimize the signal length between the sensor and microcontroller (< 30 cm). Signal lengths over 10 cm should be shielded
- Every data read from a slave should be acknowledged by an ACK from the master
- It should be possible to hard-reset the sensor should the I²C bus lock up

6 Revision history

Date	Author	Version	Changes
March 2022		1.0	Release