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

I²C Specification for PVC4000 MEMS Pirani Vacuum Transducer

1 Interface Connection

The PVC4000 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.

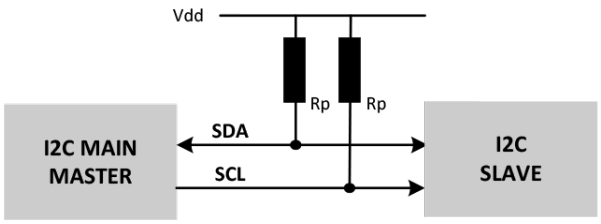


Figure 1: I²C master-slave configuration

The recommended pull-up resistor (Rp) values depend on the system implementation, but a value between 2.2 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 PVC4000 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).

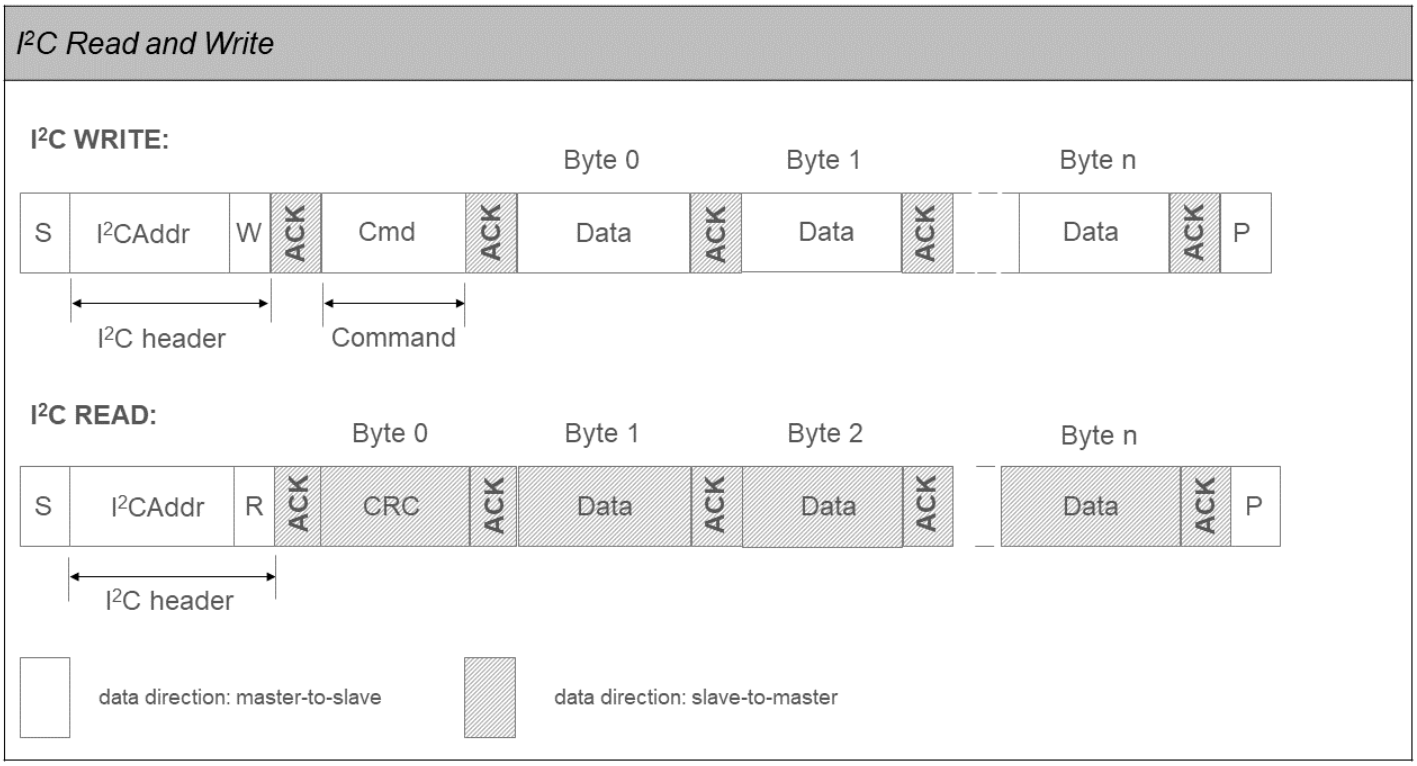
The value of the R/_W bit in the header determines the data direction for the entire rest of the data transfer sequence: if R/_W = 0 (WRITE) the direction remains master-to-slave; if R/_W = 1 (READ) the direction changes to slave-to-master after the header byte.

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 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 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.

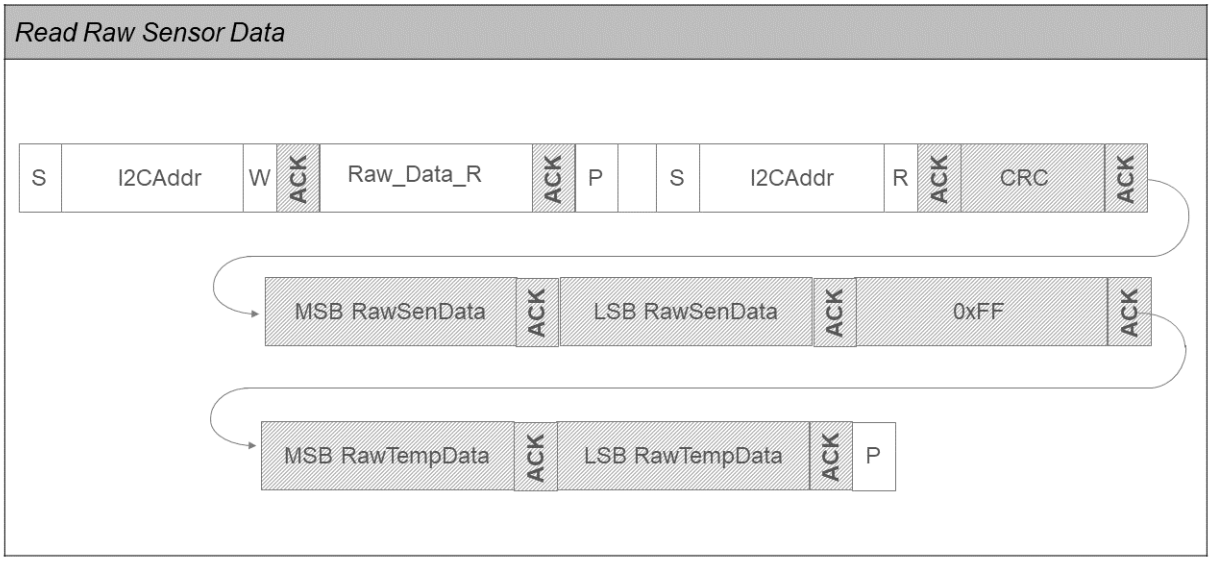
Read Command	Value	Representation	Read Command Description
Cal_Data_R	nill	Big endian	Read Calibrated Sensor Data
Raw_Data_R	0xD0	Big endian	Read Raw Sensor Data
Reg1_R	0xD3	Little endian	Read Register 1
Reg2_R	0xD4	Little endian	Read Register 2
Cal_Tbl_X_R	0xD1	Little endian	Read Calibration Table X Column
Cal_Tbl_Y_R	0xD2	Little endian	Read Calibration Table Y Column

Write Command	Value	Representation	Write Command Description
In_W	0xF0	Little endian	Enter Write Mode
Out_W	0xF1	Little endian	Exit Write Mode
Reg1_W	0xE2	Little endian	Write to Register 1
Reg2_W	0xE3	Little endian	Write to Register 2
Cal_Tbl_X_W	0xE0	Little endian	Write to Calibration Table X Column
Cal_Tbl_Y_W	0xE1	Little endian	Write to Calibration Table Y Column

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. Please refererence section 5.3 regarding reading and writing the baseline temperature from and to the sensor.

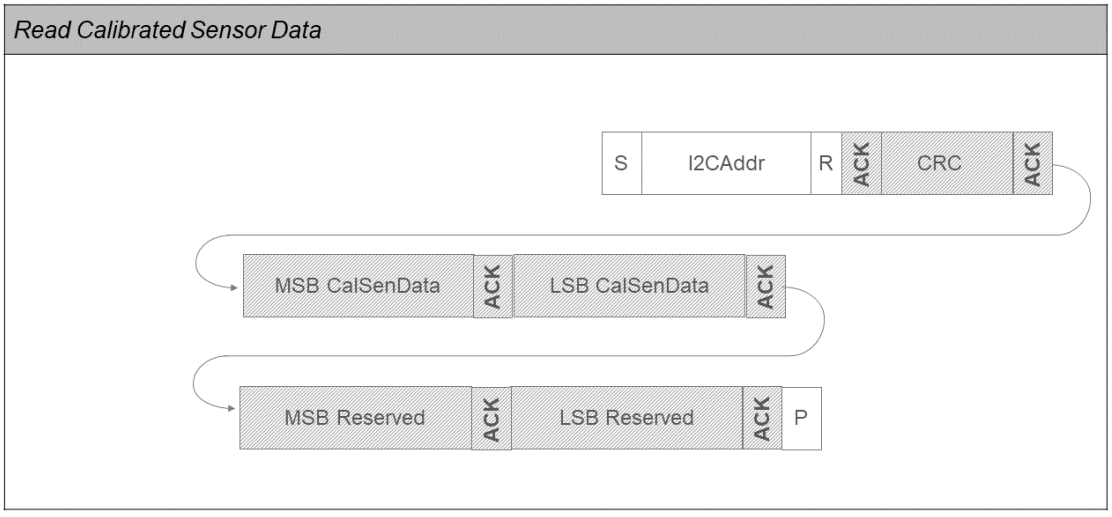
The raw temperature data comes from the sensor MCU's internal temperature sensor, and is uncalibrated.



5.2 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 refer to section 5.4 regarding reading and writing the calibration table (i.e. the lookup table).

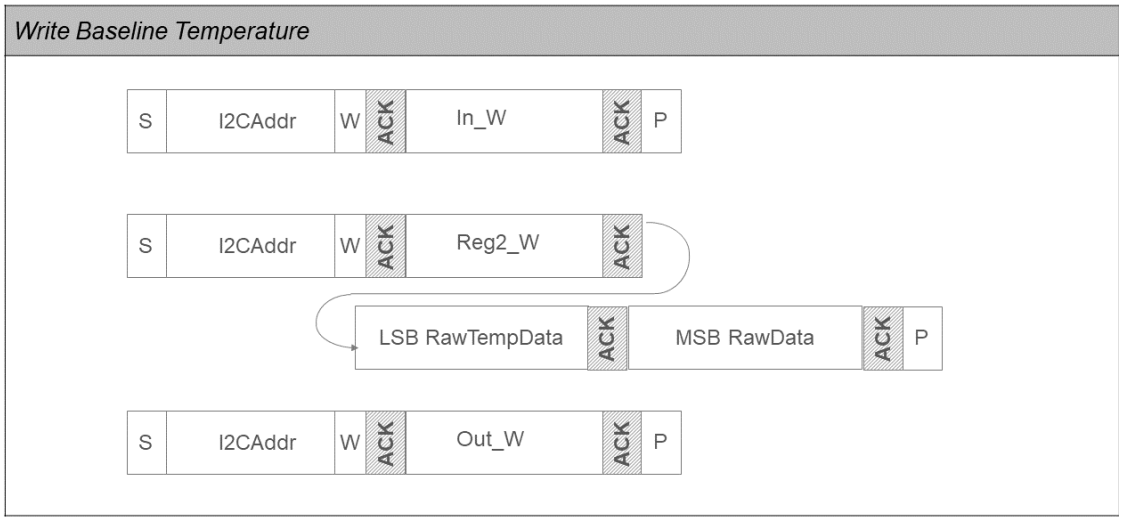
The calibrated value is user-defined.

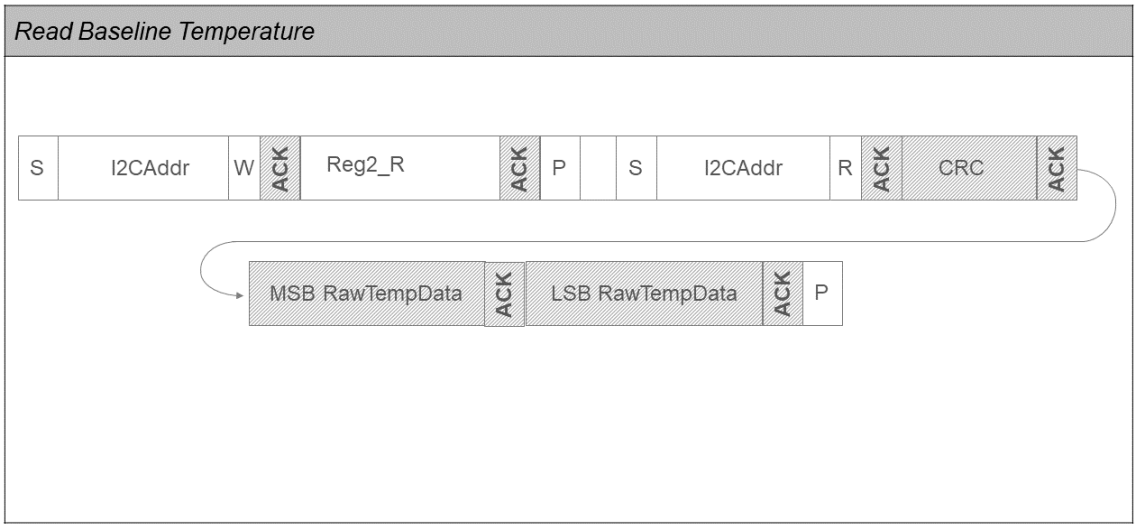


5.3 Read and Write Baseline Temperature

The raw sensor data is compensated to its value at the baseline temperature. The baseline temperature is stored in Register 2. The user can reset this baseline temperaure to the ambient temperature at which sensor calibration is performed. The user obtains the ambient temperature through the Raw_Data_R command. We recommend that users perform calibration at the room temperature, i.e. 22 °C ± 3 °C).

We recommend that Register 2 is read after write to ensure that the desired value was written correctly.





5.4 Read and Write Calibration Table

The sensors can provide calibrated output if the user stores the calibration data in the sensor. The internal calibration table consists of a two-dimensional array with 15 elements, of which 10 are for storing the user-provided calibration data.

Index	X - Raw Data	Y - Calibrated Data
0	Raw data at ATM	Pressure value at ATM
1	Raw data at pressure setpoint 1	Pressure value at setpoint 1
2	Raw data at pressure setpoint 2	Pressure value at setpoint 2
...	...	...
10	Raw data at pressure setpoint 9	Pressure value at setpoint 9
11	Reserved, do not change	Reserved, do not change
12	Reserved, do not change	Reserved, do not change
13	Reserved, do not change	Reserved, do not change
14	Reserved, do not change	Reserved, do not change

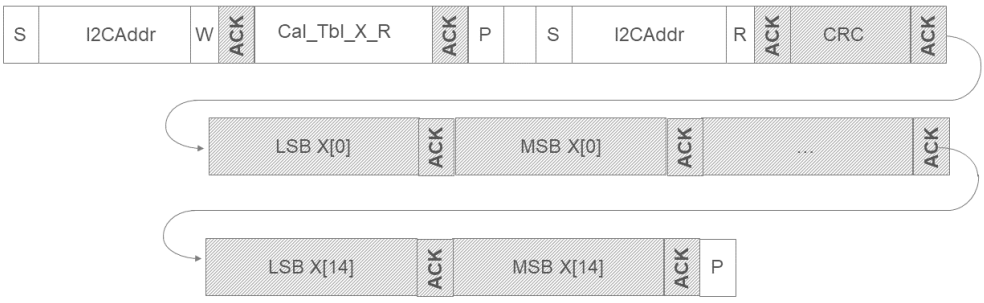
Where ATM > setpoint 1 > setpoint 2 > ... > setpoint 9

The calibration transfer function is defined as the following:

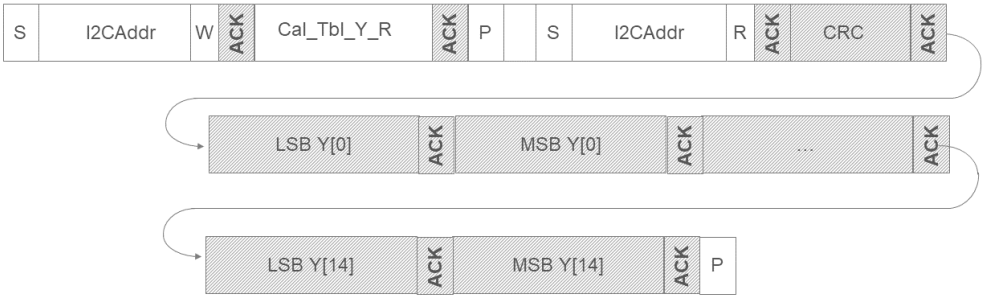
Y = (X - X[i - 1]) / (X[i] - X[i - 1]) * (Y[i] - Y[i - 1]) + Y[i - 1], where X is the raw sensor output and Y is the corresponding calibrated value.

The last four rows in the calibration table contain sensor internal data and should not be modified. The user shall read the calibration table in its entirety and rewrite the last four rows unchanged. We recommend that the calibration table is read again after write commands to make sure that the last four rows of the calibration table are unaltered.

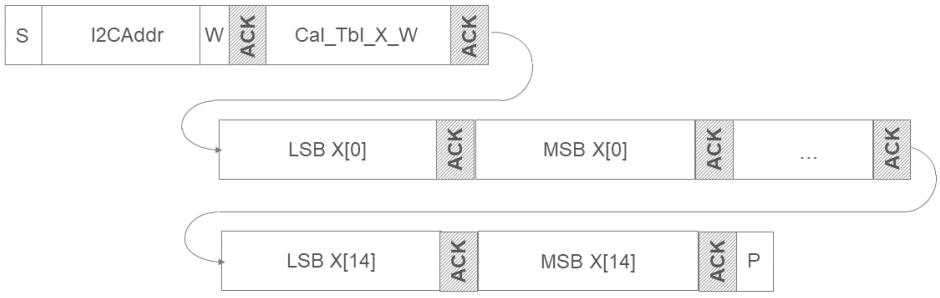
Read Calibration Table Column X



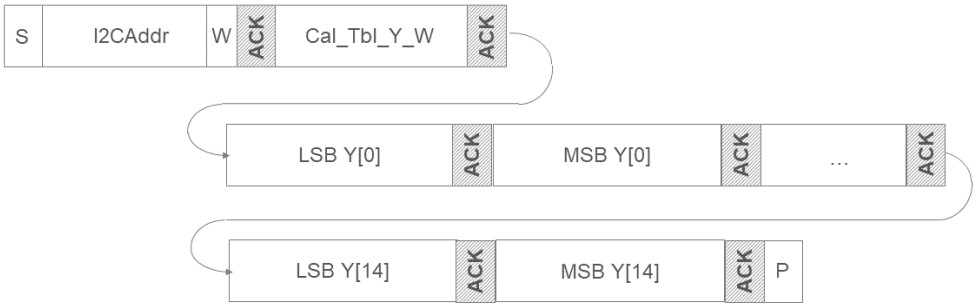
Read Calibration Table Column Y



Write Calibration Table X



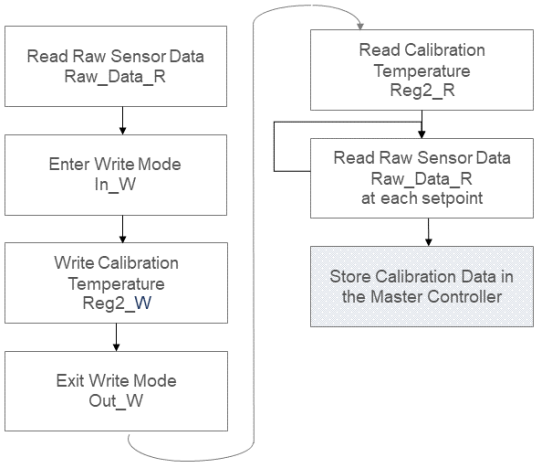
Write Calibration Table Y



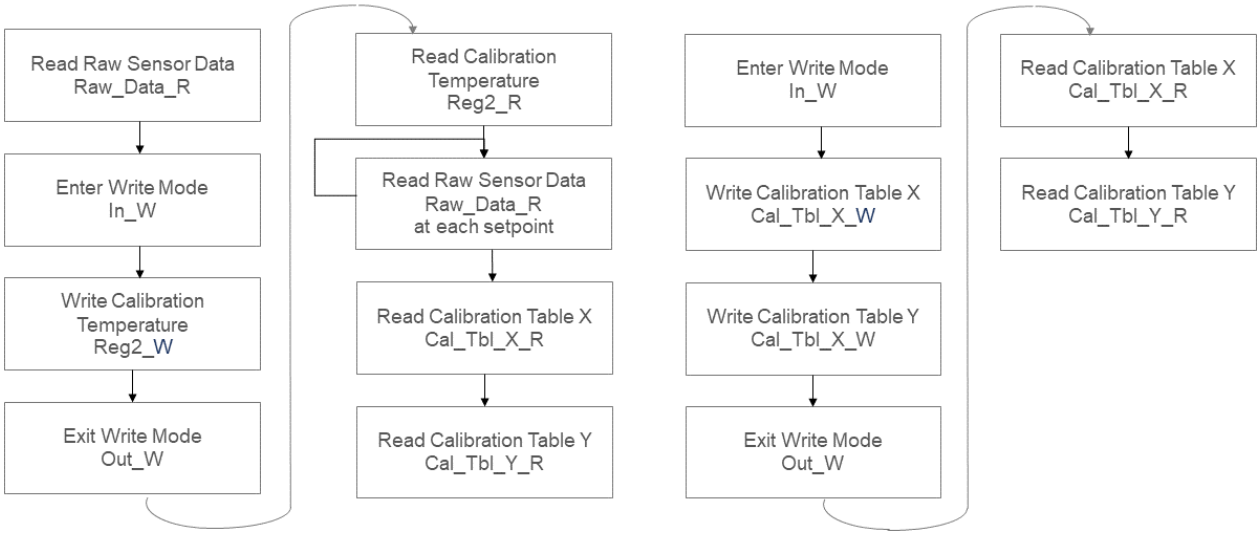
6 Communication Sequences

The user can choose to store the calibration data on or off the sensor. The following figures describe the communication sequences for both use cases.

*Communication Sequences – Calibration Data Stored **off** Sensor*



*Communication Sequences – Calibration Data Stored **in** Sensor*



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

7 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-up:

- Minimize signal length between 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

8 Revision history

Date	Author	Version	Changes
April 2021		1.0	Release