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

I²C Specification for PAV1000 and PAV3000 Series Air Velocity Sensors

1 Interface Connection

Each Posifa Technologies sensor 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 5 V supply voltage (PAV1000) or the 3.3 V power supply (PAV3000) via two pull-up resistors (Rp). In a system with a master-slave configuration, the Posifa Air Velocity Sensor Module is the slave.

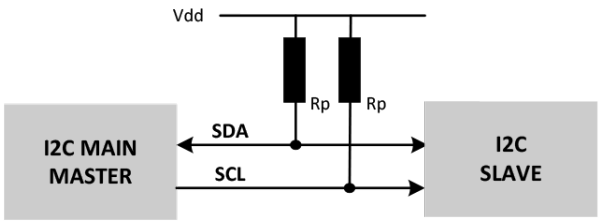

Figure 1: I²C master-slave configuration

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.

2 I²C Address

Each Posifa sensor 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 for PAV1000 and 0x28 for PAV3000.

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

The I²C protocol transfers data in byte packages. Each byte is followed by an ACK from the slave. The most significant bit (MSB) is transmitted first.

The master initiates the communication by sending a START condition, followed by the 7-bit address and a R/W bit. The R/W determines the direction of the transfer; a write bit is from master to slave and a read bit is from slave to master.

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

4 Command Set and Data Transfer Sequences

4.1 Read Sensor Data

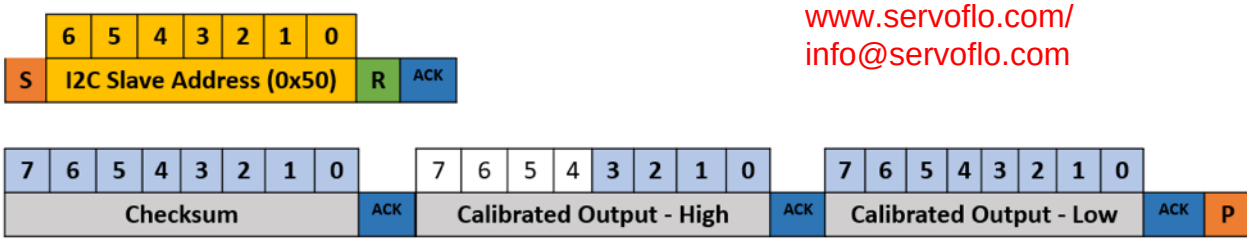

Figure 2: Read sensor data

Reading sensor data is initiated by sending the I²C address followed by a read bit. The slave will then transmit three bytes per Figure 2. The checksum is to ensure data integrity and is described in a following section. The calibrated output is a 12-bit integer. Only the least significant four bits in the high byte are valid.

4.2 Read Sensor Raw Data

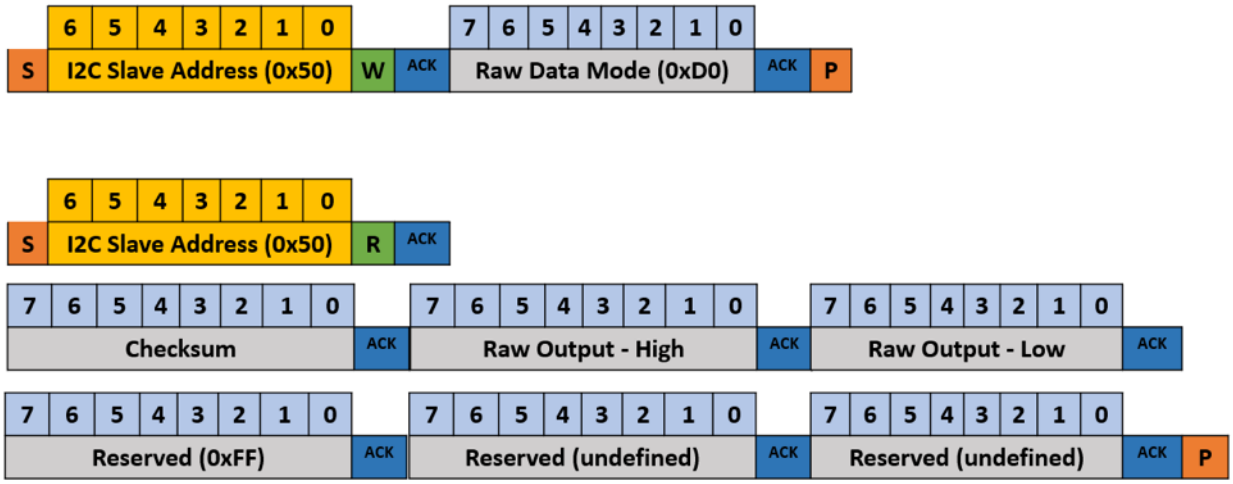

Figure 3: Read sensor raw data

To read raw output, the slave needs to be put into Raw Data Mode by writing the command 0xD0 to the I²C slave address. After the slave is in Raw Data Mode, reading data will return six bytes, per Figure 3. The two bytes following the checksum byte represent the raw output, which is a 16-bit integer. The byte after the raw output bytes has a fixed value of 0xFF. The last two bytes are undefined.

To exit the Raw Data Mode, simply power cycle the slave (i.e. turn off the slave power supply and then turn it on).

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

5 Checksum

The checksum used for data integrity 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:

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

6 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

7 Revision history

Date	Author	Version	Changes
September 2019		0.1	Pre-release