SENSORTECH
An Amphenol Company

SGX VZ89TE I2C communication

The SGX VZ-89TE combines state-of-the-art MOS sensor technology with intelligent detection algorithms to monitor tVOCs and CO2 equivalent variations in confined spaces, e.g. meeting rooms or vehicle cabins.
The dual signal output can be read through a multiplexed PWM output or through an I2C bus.
This datasheet will describe the I2C communication

1. Theory of the operations :

When the device is connected to the I2C bus line, the device is working as a slave device. The master can write/read data to/from VZ-module using the I2C interface commands.

- The VZ89 device address contains seven fixed bits.
- The VZ89 communication must be set in "standard Mode": bit rates up to 100 kbit/s.
- The pull-up resistors (4k7) on SDA and SCL line must be implemented on the master board (There are no pull-up resistor on VZ-89TE PCBA)
- A delay (~100ms) between the command (Write) frame and the status-request (Read) frame should be implemented.

Scope picture of the delay (100ms) between
The command (Write) frame and the
status-request frame (Read)

For more information please contact:

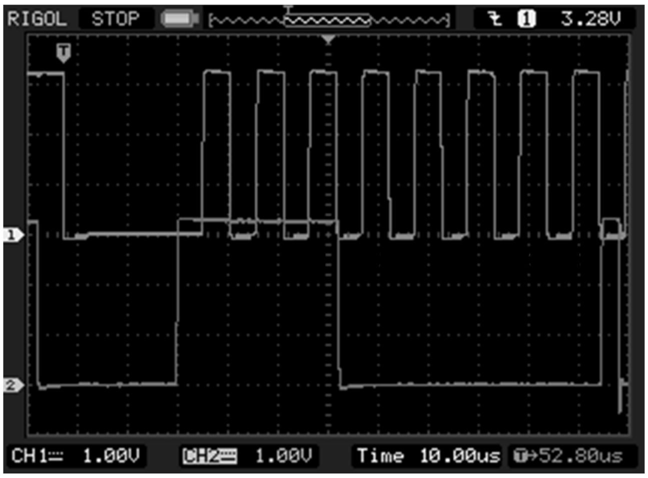
info.em@sgxsensortech.com

SGX Sensortech, Courtils 1
CH-2035 Corcelles-Cormondrèche
Switzerland

www.sgxsensortech.com

2. Device addressing

The address byte is the first byte received following the START condition from the master device. The first part of the address byte consists of a 4-bit device code which is set to 1110 for the IAQS. The device code is followed by three address bits (A2, A1, A0) which are programmed at 0.



Scope picture of the the **address**–byte
for a **Write** condition

3. Sending the command frame :

The master sends a command byte (8 bits) followed by 4 data bytes and a CRC (8bits) in order to set parameters to the VZ-89TE or to request its status, as follow:

Command (8 bits)	Data 0 (8 bits)	Data 1 (8 bits)	Data 2 (8 bits)	Data 3 (8 bits)	CRC (8 bits)

Command list

#	Command value	Command Name	Description
3.1	0b00001000	setPPMCO2	This command is used to send the ppmCO2 value given by an external analyser to the VZ89TE in order to recalibrate its outputs.
3.2	0b00001001		Reserved for a future implementation.
3.3	0b00001010		Reserved for a future implementation.
3.4	0b00001011		Reserved for a future implementation.
3.5	0b000001100	getStatus	This command is used to read the VZ89TE status coded on 6x bytes + 1 CRC byte as follow: Byte-1 = VOC signal level value. Byte-2 = CO2-equivalent signal level value. Byte-3 = Raw sensor value MSB. Byte-4 = Raw sensor value. Byte-5 = Raw sensor value LSB. Byte-6 = Error status byte. Byte-7 = CRC
3.6	0b000001101	getRevision	This command will return the revision code of the module as follow: Byte-1 = Year. Byte-2 = Month. Byte-3 = Day . Byte-4 = ASCII code for a charter. Byte-5 = 0. Byte-6 = 0. Byte-7 = CRC.

Command list (continued)

#	Command value	Command Name	Description
3.7	0b000001110	Reserved For Engineering	Please don't use it because this command will overwrite the manufacturing calibration values.
3.8	0b000001111	Reserved For Engineering	Please don't use it because this command will overwrite the manufacturing calibration values.
3.9	0b000010000	getR0	This command is used to read the R0 (calibration) value in [kOhms] coded on 6x bytes + 1 CRC byte as follow: Byte-1 = LSB. Byte-2 = MSB. Byte-3 = 0. Byte-4 = 0. Byte-5 = 0. Byte-6 = 0. Byte-7 = CRC

4. Writing data to VZ module (I2C_SendBlock)

Send by master | Send by slave

	A6	A5	A4	A3	A2	A1	A0	R/W		C7	C6	C5	C4	C3	C2	C1	C0	
	Address byte									Commande byte								
Start	1	1	1	0	0	0	0	0	ACK									ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data 0								ACK	Data 1								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data 2								ACK	Data 3								ACK

CRC7	CRC6	CRC5	CRC4	CRC3	CRC2	CRC1	CRC0		
CRC								ACK	Stop

5. Reading VZ status data (I2C_RecvBlock)

Send by master | Send by slave

	A6	A5	A4	A3	A2	A1	A0	R/W		D7	D6	D5	D4	D3	D2	D1	D0	
	Address byte with R/W bit set									**VOC** value (8bits)								
Start	1	1	1	0	0	0	0	1	ACK									ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
CO2-equivalent level value (8bits)								ACK	**Raw sensor** value [MSB]								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Raw sensor value								ACK	**Raw sensor** value [LSB]								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0		
Error byte								ACK	CRC								NACK	Stop

6. The CRC byte:

The CHECKSUM byte contains the inverted modulo-256 sum over all data bytes
The sum is calculated by ADD with carry where the carry bit of each addition is added to the LSB of its resulting sum. This guarantees security also for the MSBs of the data bytes.
The CRC byte sent by the MASTER is calculated with all data byte including command byte.
The CRC byte sent by the SLAVE (VZ89TE) is calculated with al data byte.

```
/**********************************************************
 * getCRC
 *
 *        Description :
 *        This function calcul, then return the CRC value
 *        of a data buffer.
 *
 *        Input parameters :
 *                  1. A pointer on the data to process.
 *                  2. The size of the data buffer
 *
 *        Return :
 *                  the CRC value
 **********************************************************/

 byte getCRC(byte *buffer, byte size) {
   /* Local variable */
     byte crc = 0x00;
     byte i   = 0x00;
     word sum = 0x0000;
   /* Summation with carry */
     for(i=0; i < size; i++) {
         sum += buffer[i];
     }//end loop
     crc =  (byte)sum;
     crc += (sum / 0x0100); // Add with carry
     crc = 0xFF-crc;        // Complement
   /* Returning results*/
          return(crc);
 //end function
```