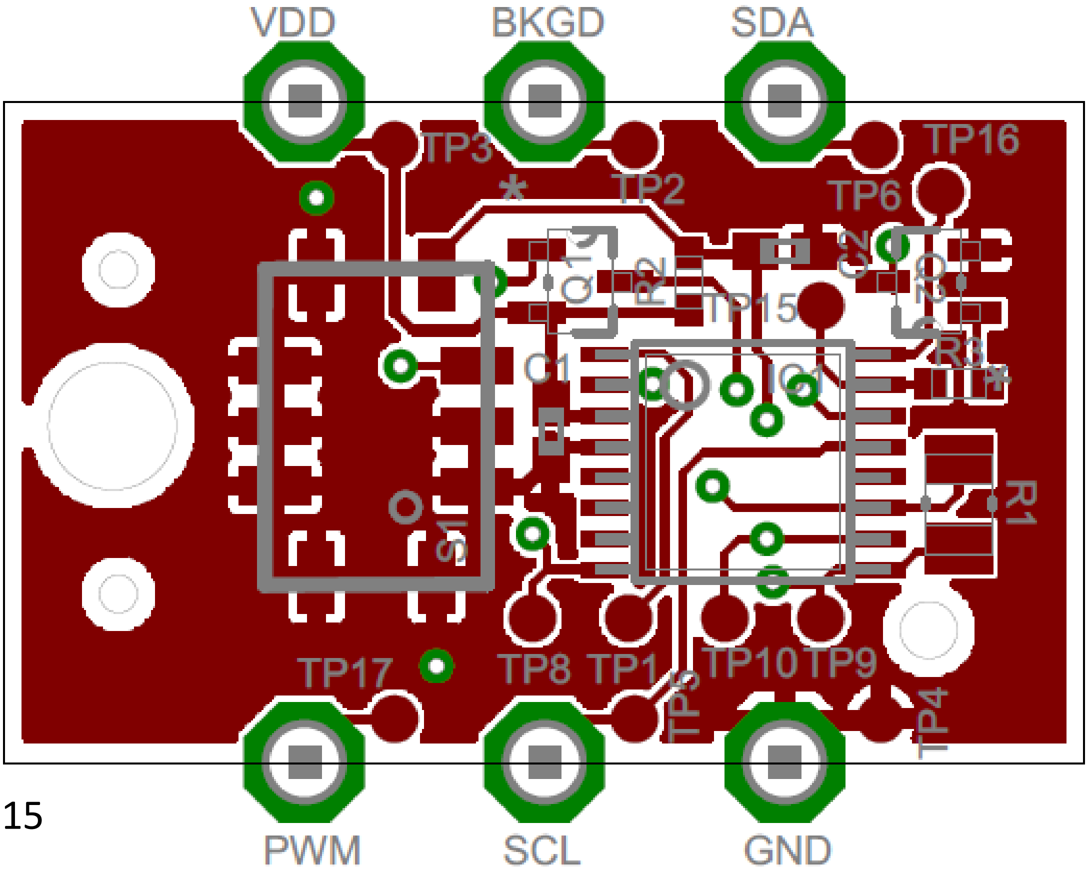
SGX SENSORTECH

VZ8(6)9 rev B I2C communication quick manual

1. VZ PCBA considerations

External pull-up resistors (4k7) are required on SDA And SCL (they are not implemented on VZ PCBA)

VDD for VZ8(6)9T = 3V3
VDD for VZ8(6)9F = 5V0



HW level : rev D
SW level : VZ869_PA8_B_151015

2. Theory of operation

When the device is connected to the I2C bus line, the device is working as a slave device. The master can write/read the IAQS using the I2C interface command.
The IAQS device address contains seven fixed bits.
The IAQS device speed is set in "standard Mode": bit rates up to 100 kbit/s.

Device addressing:

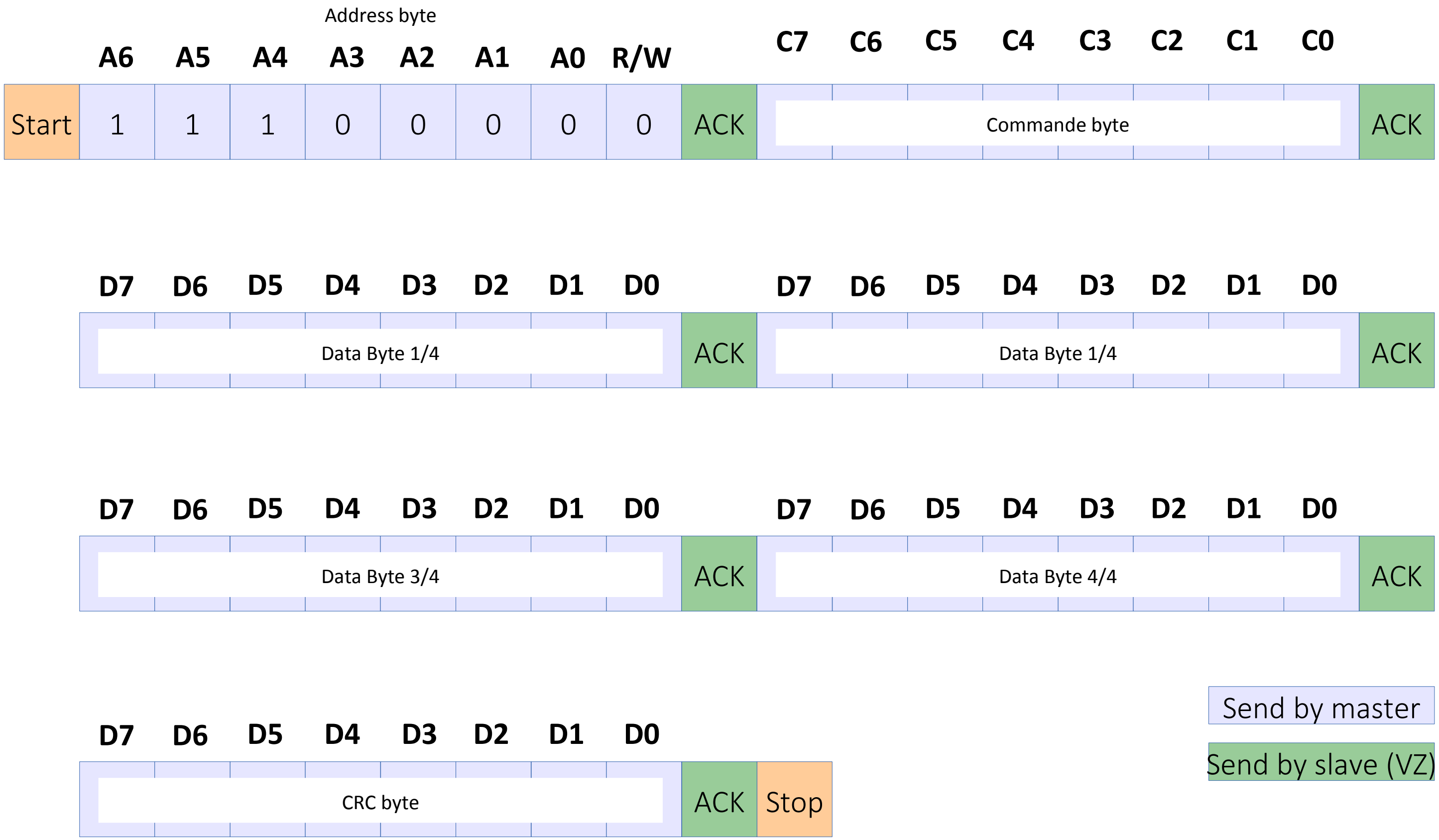
The address byte is the first byte received following the START condition from the master device. The first part of the address byte consists of a 4-bit device code which is set to 1110 for the IAQS. The device code is followed by three address bits (A2, A1, A0) which are programmed at 0:

IAQS address (7 bits) = 0b1110000

3.Sending data to VZ module

In all case, the data frame must be composed like this:

- 1x commande byte
- 4x data bytes
- 1X CRC byte (refert to §5)



Address byte

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

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data Byte 1/4								ACK	Data Byte 1/4								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data Byte 3/4								ACK	Data Byte 4/4								ACK

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

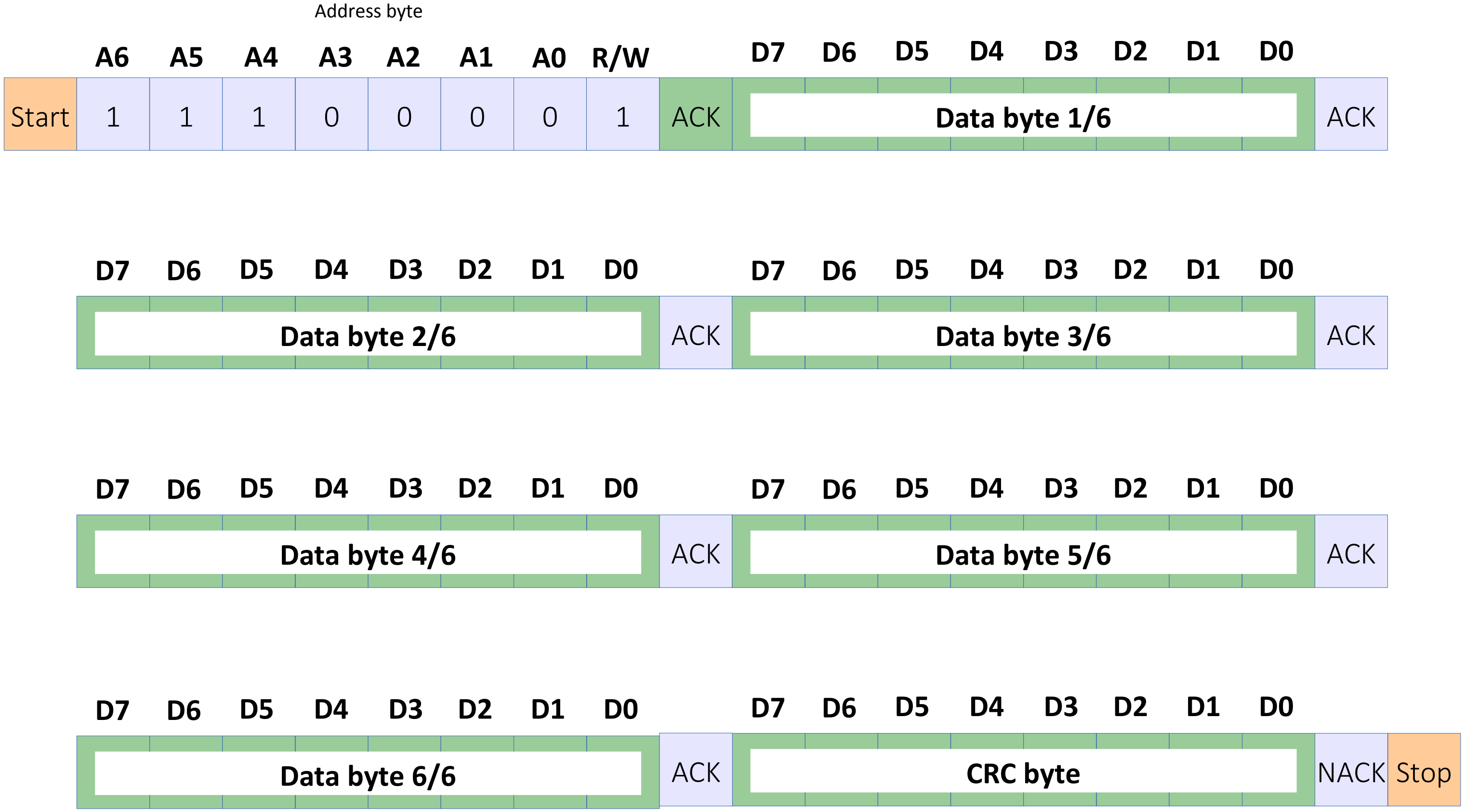
Send by master

Send by slave (VZ)

4. VZ response frame

In all case, the VZ response frame is composed like this:

- 6x data bytes
- 1x CRC byte (refert to §5)



Send by master

Send by slave (VZ)

Address byte

	A6	A5	A4	A3	A2	A1	A0	R/W		D7	D6	D5	D4	D3	D2	D1	D0	
Start	1	1	1	0	0	0	0	1	ACK	Data byte 1/6								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data byte 2/6								ACK	Data byte 3/6								ACK

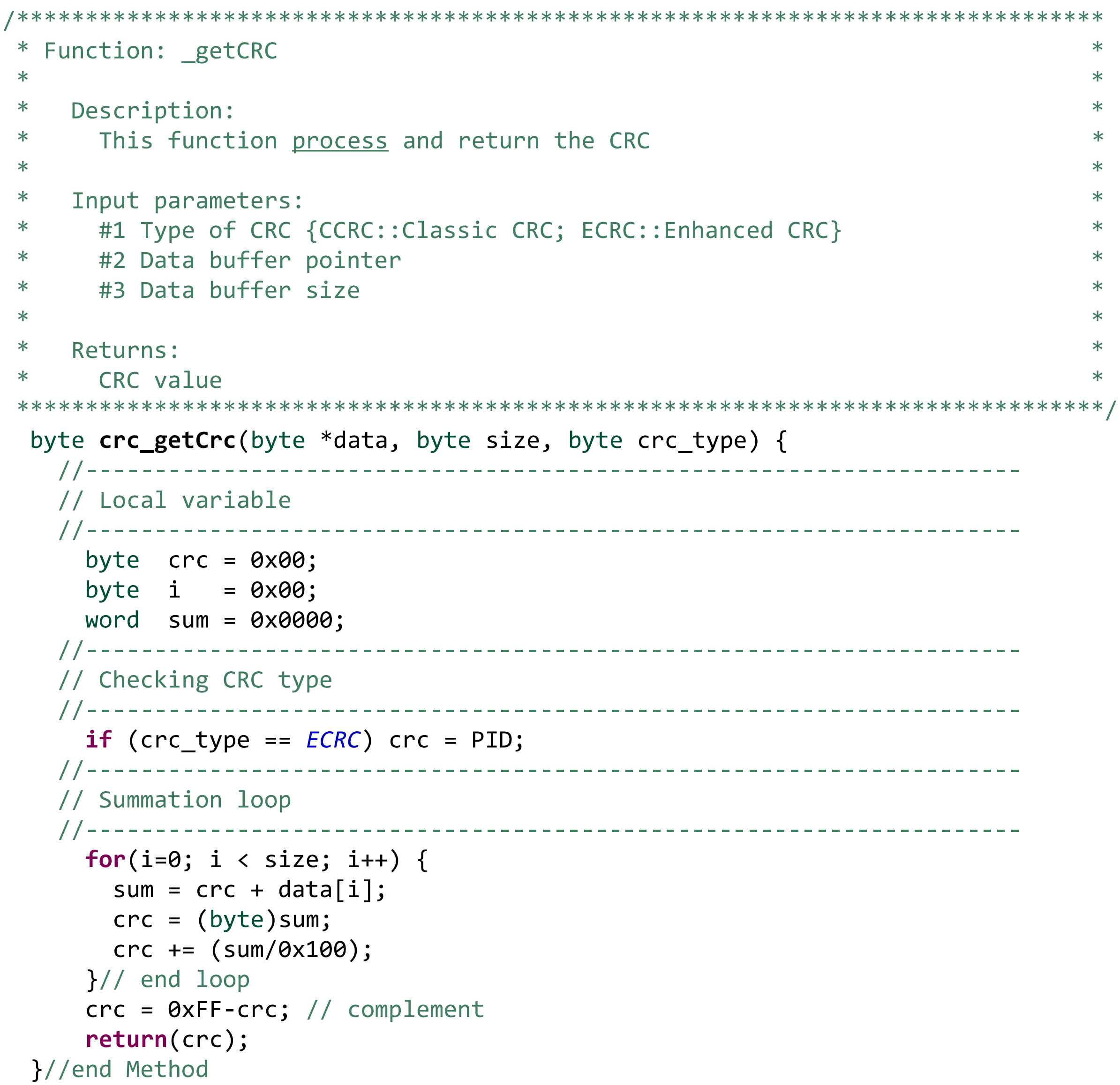
D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0	
Data byte 4/6								ACK	Data byte 5/6								ACK

D7	D6	D5	D4	D3	D2	D1	D0		D7	D6	D5	D4	D3	D2	D1	D0		
Data byte 6/6								ACK	CRC byte								NACK	Stop

5. CRC method

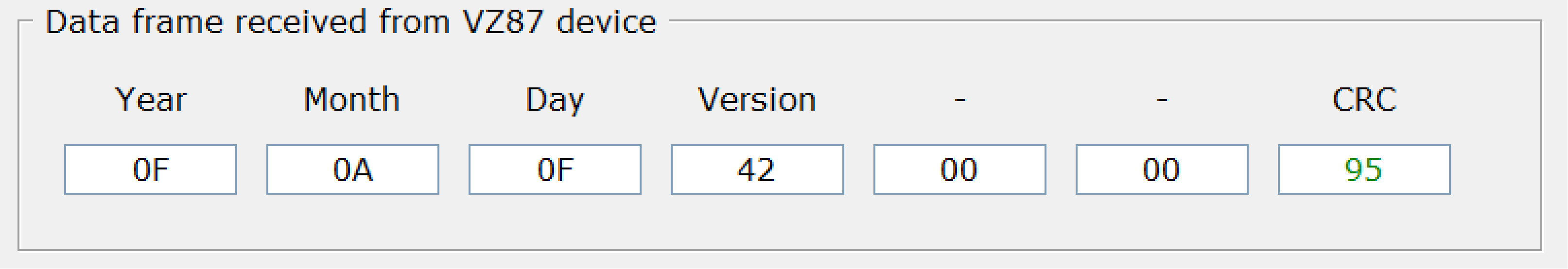
Note :

- In VZ module the CRC type is allways set to CCRC (0x00)
- ADDRESS byte is NOT taken in account for CRC processing



```
/*******************************************************************************
 * Function: _getCRC                                                           *
 *                                                                             *
 *   Description:                                                              *
 *     This function process and return the CRC                                *
 *                                                                             *
 *   Input parameters:                                                         *
 *     #1 Type of CRC {CCRC::Classic CRC; ECRC::Enhanced CRC}                  *
 *     #2 Data buffer pointer                                                  *
 *     #3 Data buffer size                                                     *
 *                                                                             *
 *   Returns:                                                                  *
 *     CRC value                                                               *
 *******************************************************************************/
  byte crc_getCrc(byte *data, byte size, byte crc_type) {
    //--------------------------------------------------------------------
    // Local variable
    //--------------------------------------------------------------------
      byte  crc = 0x00;
      byte  i   = 0x00;
      word  sum = 0x0000;
    //--------------------------------------------------------------------
    // Checking CRC type
    //--------------------------------------------------------------------
      if (crc_type == ECRC) crc = PID;
    //--------------------------------------------------------------------
    // Summation loop
    //--------------------------------------------------------------------
      for(i=0; i < size; i++) {
        sum = crc + data[i];
        crc = (byte)sum;
        crc += (sum/0x100);
      }// end loop
      crc = 0xFF-crc; // complement
      return(crc);
  }//end Method
```

6. CRC example



Data frame received from VZ87 device

Year	Month	Day	Version	-	-	CRC
0F	0A	0F	42	00	00	95

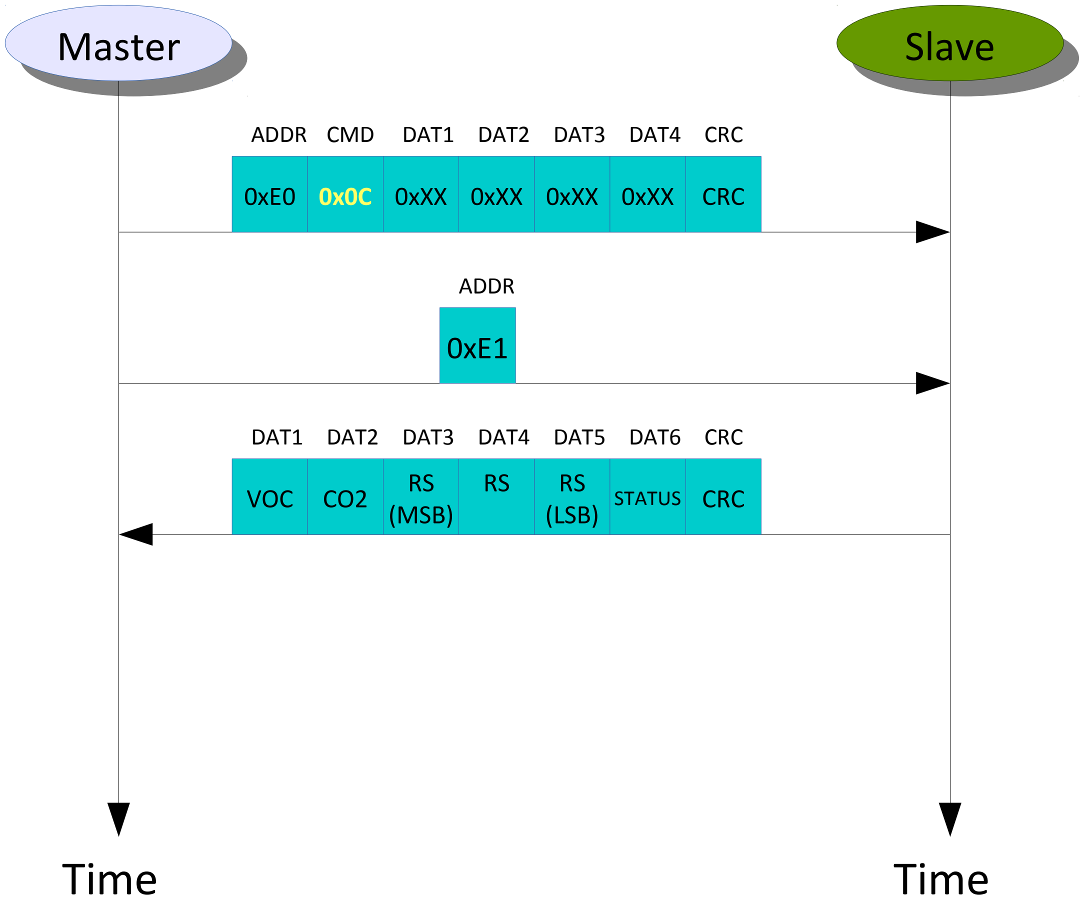
CRC processing:

0x0F + 0x0A + 0x0F + 0x42 + 0x00 + 0x00 = 0x6A

CRC = 0xFF – 0x6A = 0x95

Note :
In this example *carry* remain at 0, but this is not always the case, so *carry* as to be taken in account (as shown in the C method)

7. Reading VZ8(6)9 status

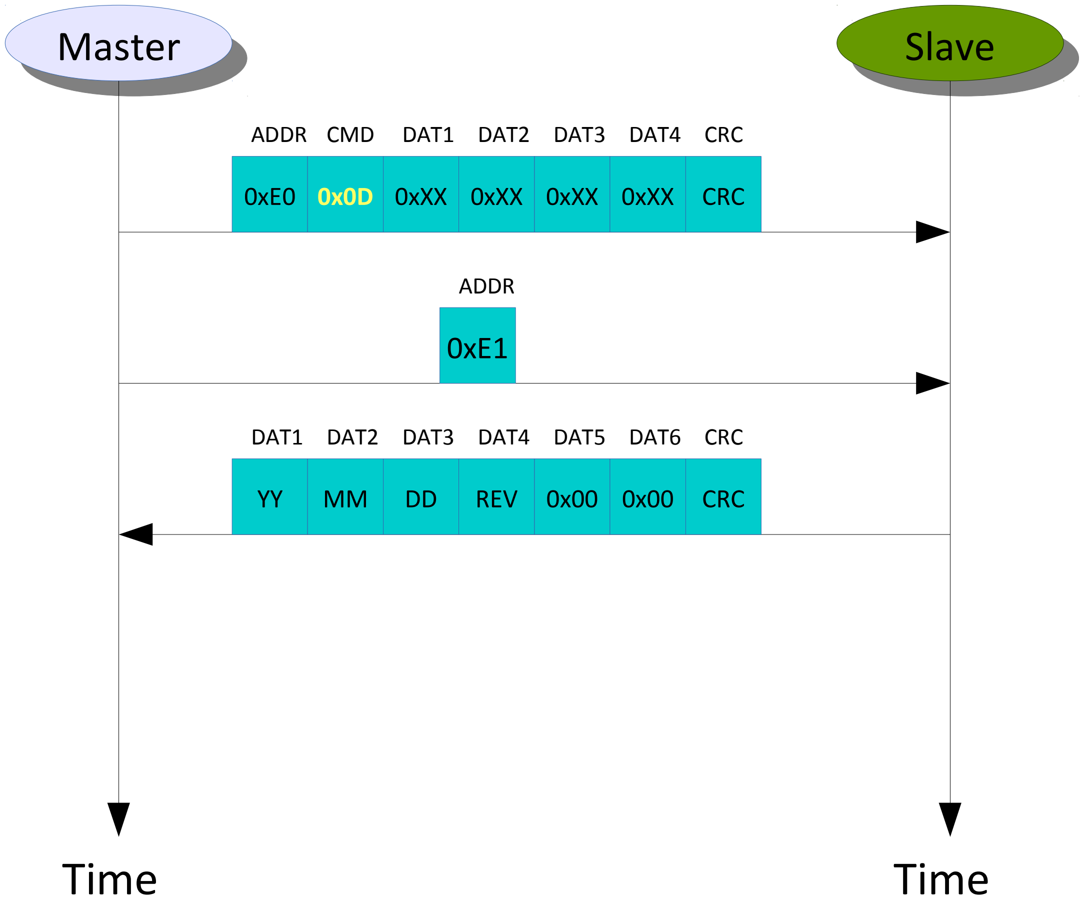


ADDR	CMD	DAT1	DAT2	DAT3	DAT4	CRC
0xE0	0x0C	0xXX	0xXX	0xXX	0xXX	CRC

ADDR
0xE1

DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	CRC
VOC	CO2	RS (MSB)	RS	RS (LSB)	STATUS	CRC

8. Reading VZ8(6)9 Date code and revision



Master

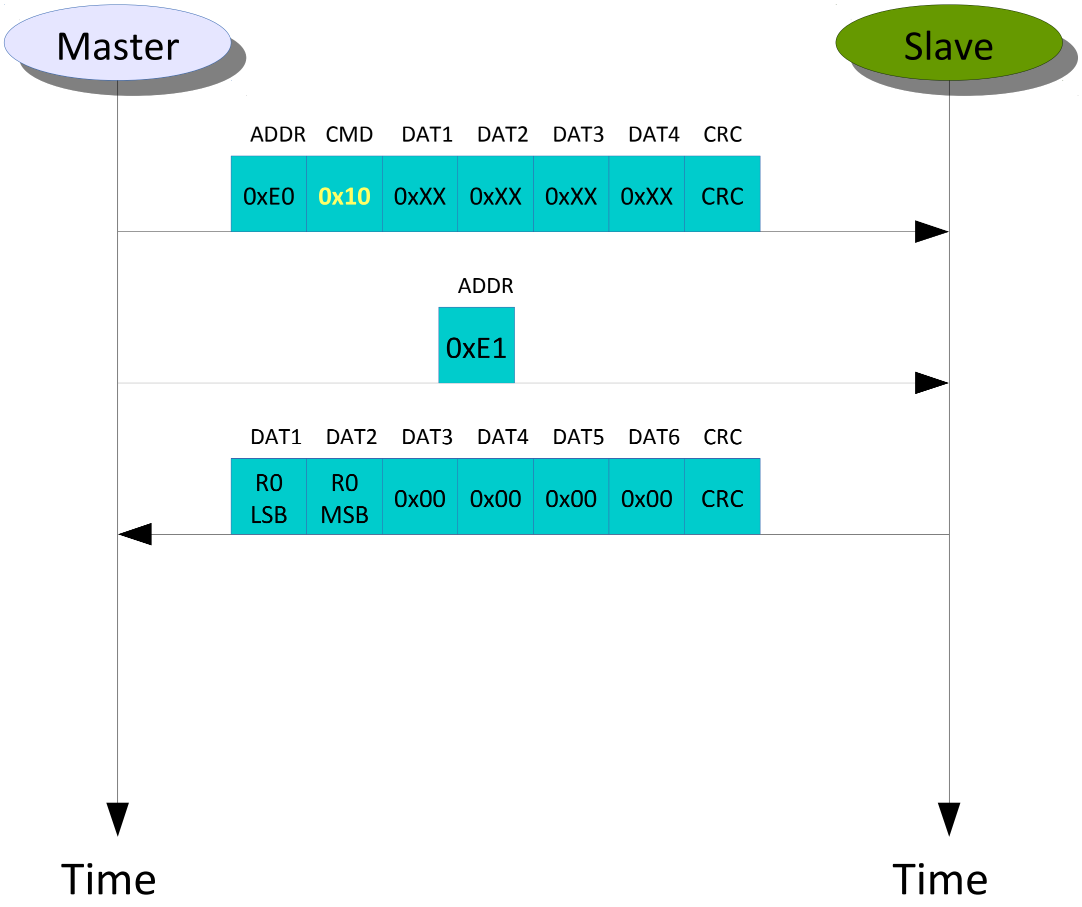
Slave

ADDR	CMD	DAT1	DAT2	DAT3	DAT4	CRC
0xE0	0x0D	0xXX	0xXX	0xXX	0xXX	CRC

ADDR
0xE1

DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	CRC
YY	MM	DD	REV	0x00	0x00	CRC

9. Reading VZ8(6)9 R0 (calibration value)



ADDR	CMD	DAT1	DAT2	DAT3	DAT4	CRC
0xE0	0x10	0xXX	0xXX	0xXX	0xXX	CRC

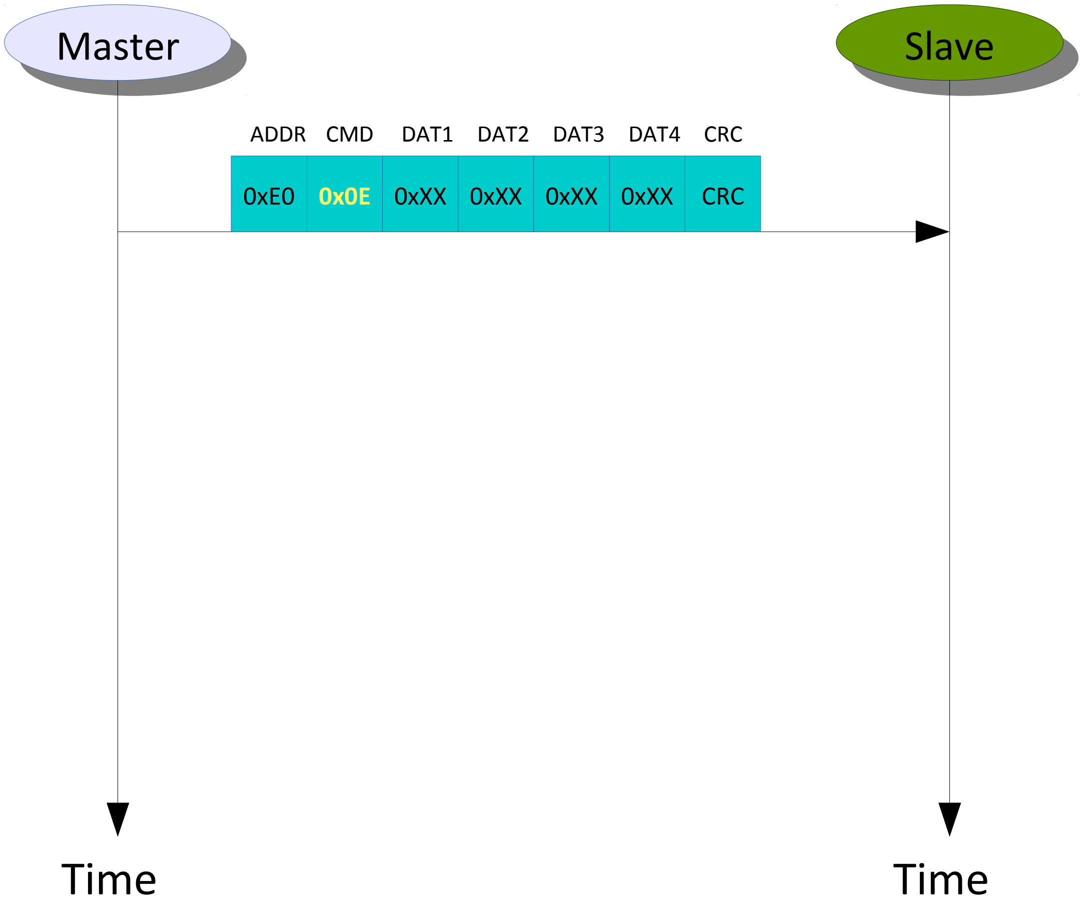
ADDR
0xE1

DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	CRC
R0 LSB	R0 MSB	0x00	0x00	0x00	0x00	CRC

Note :

- The R0 value is in[kΩ].

10. Setting VZ8(6)9 R0 with current RS value

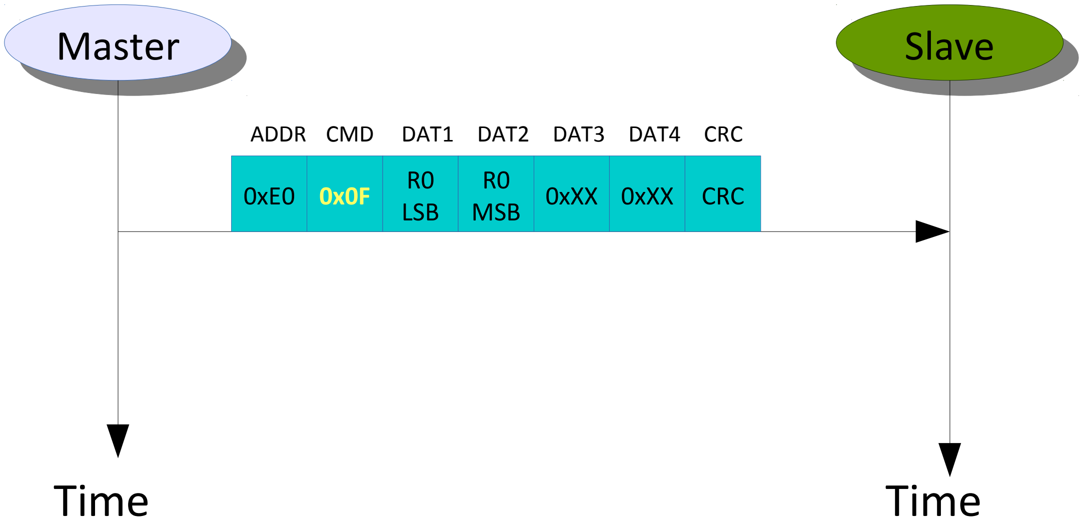


ADDR	CMD	DAT1	DAT2	DAT3	DAT4	CRC
0xE0	0x0E	0xXX	0xXX	0xXX	0xXX	CRC

Note :

- When this command is sent the VZ take the current raw sensor (RS) value as new R0 (calibration value).
- VZ module doesn't respond to this command.

11. Setting VZ8(6)9 R0 with I2C data

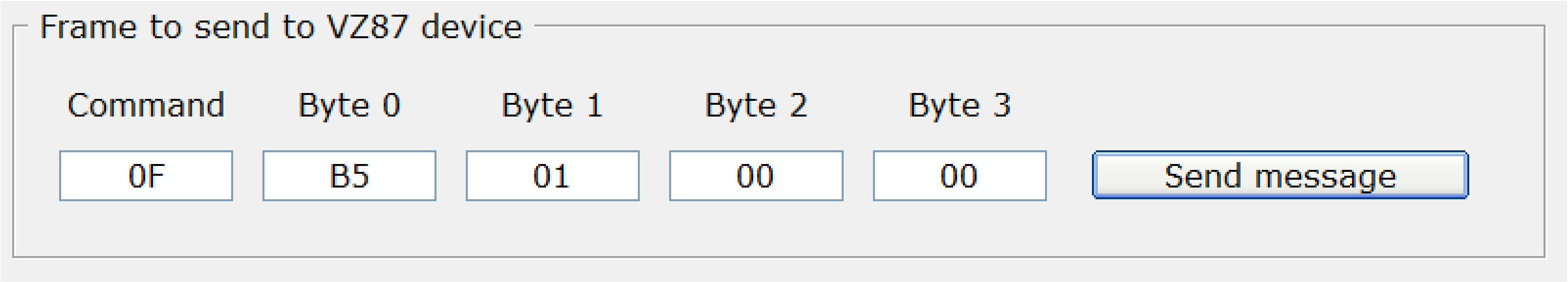


ADDR	CMD	DAT1	DAT2	DAT3	DAT4	CRC
0xE0	0x0F	R0 LSB	R0 MSB	0xXX	0xXX	CRC

Note :

- When this command is sent the VZ take the I2C {DAT1, DAT2} value as new R0 (calibration value).
- This value must be in [kΩ].
- VZ module doesn't respond to this command.

For example, setting 437k as new R0 :
(the interface automatically process the CRC, it is why it is not shown)



Frame to send to VZ87 device

Command	Byte 0	Byte 1	Byte 2	Byte 3
0F	B5	01	00	00

Contact :

Francois Carnal
+41 (0)32 732 16 86
francois.carnal@sgxsensortech.com

END