I²C Communication with SMI Protocol-A

1. Description

This document is a reference for a possible coding method to achieve pressure, temperature, and status for SMI part readings using I²C. This SMI Protocol-A is used on but not limited to SMI parts SM9541, SM9543, SM3041, etc. Please check part data sheet for the correct SMI I²C protocol used. For more general information on how to interface using the I²C protocol please refer to http://en.wikipedia.org/wiki/I%C2%B2C. Note that the current example serves as only a "pseudo-code" meaning that it will not work by itself, needs additional work, and is only meant for guidance purposes. Therefore this example may look different depending on which coding language is used.

2. Retrieving data

1: If applicable, include any necessary libraries for I²C communication or other needed protocol with your microcontroller/device.

2: Assign variables and create any setup features.

```
i2cv.pressure    (14 bit Pressure in counts)
i2cv.temperature (11 bit Temperature in counts)
i2cv.status      (2 bit Status)
i2c_byte1
i2c_byte2
i2c_byte3
i2c_byte4
(*ETC*)
```

3: Send I²C initialize command (Initialize I²C bus to set up communication with device)

```
i2c_init();
```

4: Send I^2C start command (Send start I^2C condition to begin communication with device)

```
i2c_start_bus();
```

5: To address and read the SMI sensor, the master must write 8 bits total through I^2C. The 8 bits consist of the device address and a read command. Send the 7 bit I^2C device address command and a least significant bit (LSB) of 1 to tell the master what address to read from. This will give you an 8 bit address with 7 bit part address shifted left and a LSB of 1 added to end of byte. (Please check part data sheet for correct device address.)

```
Device_Read_Byte = (Device Address << 1) + 1;
i2c_write(Device_Read_Byte);
```

6: Read the SMI part at the device address to gather measurements. This can be done by setting the master into a receiving state. 4 bytes will have to be read to gather all measurement information. An acknowledge will have to be sent after each of the first 3 bytes and not acknowledge on the fourth byte to stop transmission.

```
i2c_byte1 = i2c_read_ack();
i2c_byte2 = i2c_read_ack();
i2c_byte3 = i2c_read_ack();
i2c_byte4 = i2c_read_nack();
```

7: Stop bus, this ends communication with bus, this can be a reset if trying to receive multiple readings.

```
stop_i2c_bus();
```

3. Converting Bytes

To collect pressure, temperature, and status, 4 bytes of data have to be read. These bytes will be converted and rearranged to be able to read temperature, pressure, and status of the device. If only 1 or 2 of the 3 device output values are needed, reading less bytes may be sufficient. For example If only pressure is needed, only 2 bytes can be read to obtain the full 14 bit pressure reading. See diagram below in section 3 for more details.

1: Converting Temperature Reading.

Temperature conversion consists of a right-shift of the fourth byte by 5 bits (last 5 bits will not contain any data). Then taking the third byte and shifting it left by 3 bits. This is done by multiplying by 8 ($8 = 2^3$). Adding both these values together achieves an 11 bit temperature reading.

```
i2c_byte4 >>= 5;
i2cv.temperature = (i2c_byte3 * 8) + i2c_byte4;
```

2: Converting Pressure Reading.
Pressure conversion consists of left-shifting the first byte by 8, this can be done by multiplying the first byte by 256 (256 = 2^8), then adding the second byte with eight lower order bits (LSBs) of the full 14 bit pressure reading. A bit-wise AND-operation with $3FFF_{hex}$ is then applied to remove the first two bits that contain part status information by setting those bits to a binary “00”. (If needed further inspect the device data sheet)

```
i2cv.pressure = 0x3fff & ((i2c_byte1 * 256) +
i2c_byte2);
```

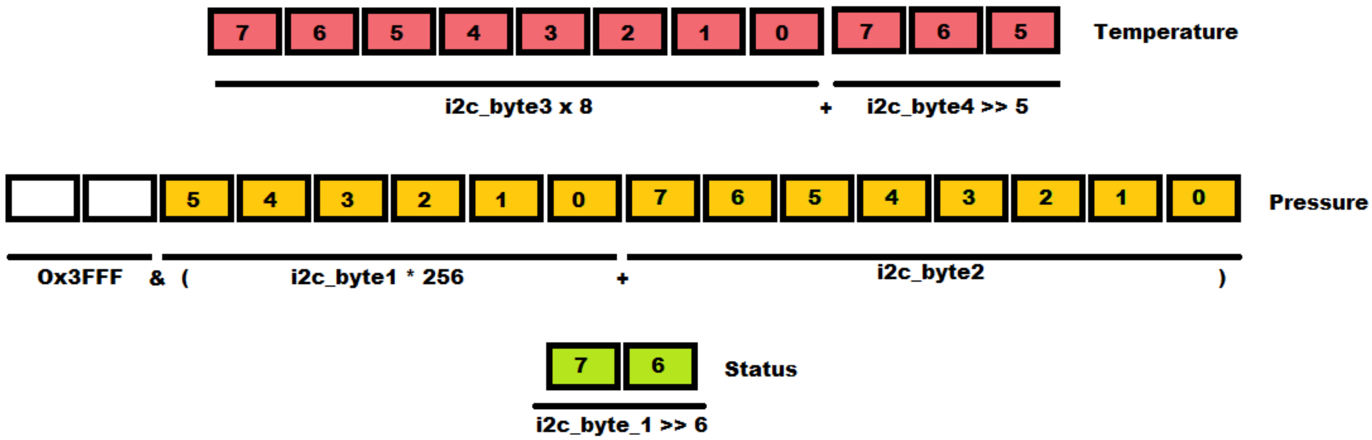
3: Converting Status Reading.
Status conversion consists of a right-shift of the first byte by 6 bits. This will remove pressure data leaving only the relevant two status bits.

```
i2cv.status = i2c_byte_1 >> 6;
```

Below is a visual diagram to aid in the process of converting bytes.

Each color represents a different type of bit. Green bits are Status, yellow bits are Pressure, and red bits are Temperature.

Rearranging these bytes produces 3 outputs, Tempature, Pressure and Status as seen below.



4. Handling of Data

Once the 4 bytes of data have been rearranged and converted, extrapolate the bytes to be able to achieve data values for the device. Use i2cv.temperature and i2cv.pressure count values from the SMI device to calculate actual temperature and pressure values. These calculated values will give temperature in degrees Celsius and pressure in counts or percent full scale (%FS) of the device will need to be converted into unit based values depending on the model number. Refer to data sheet for exact range and limitations of part.

1: Pressure Reading

To convert to pressure in the appropriate pressure unit from the byte counts, a line fit from the target count and pressure values has to be created. Creating a line fit from maximum and minimum device points allows for the extrapolation of data values from all count readings.

min_press	Minimum pressure
max_press	Maximum pressure
min_count	Minimum pressure count reading
max_count	Maximum pressure count reading

Pressure_reading = (((max_press – min_press)/(max_count – min_count))* (**i2cv.pressure** - min_count) + min_press)

For example using the SM9541-100C-S-C-3 and inspecting the corresponding data sheet, the maximum and minimum spec count values can be found. The minimum pressure count of 1638 at 10% of the output range and maximum pressure count of 14745 at 90% of the output range. This results in a FS span of 80% of the output range which is equal to 13107 counts. These minimum and maximum relate to -5 cmH_20 and 100 cmH_20 pressure readings for this part, as found in the data sheet. Assuming you receive a count of 8191 from device, the calculations in $cmH_2 0$ are as follows.

Pressure in $cmH_2 0$ = (100 – (-5))/ (13107 - 1638)* (8191– 1638) + (-5)
Pressure in $cmH_2 0$ = 55

2: Temperature Reading

To convert to temperature from byte counts, use the equation below and evaluate it using the counts of **i2cv.temperature**. The resulting temperature will be in degrees Celsius.

Temperature_reading = (**i2cv.temperature**/ (2^{11}) *200) - 50

Using a count value of 1024 received from the part, the temperature reading of the part can be achieved. The calculations for temperature are as follows.

Temperature in Celsius = (1024/ (2^{11}) *200) - 50
Temperature in Celsius = 50

3: Status Reading

SMI part status consists of 2 bits. These 2 bits gives 4 possible status readings. The 4 status readings are as follows:

00	Normal Operation
01	Command Mode
10	Stale Data
11	Diagnostic condition exists

For additional questions, please consult sales@si-micro.com.

PRESSURE SENSORS

5. Warranty and Disclaimer

Information in this document is provided solely to enable software and system implementers to use Silicon Microstructures, Inc. (SMI) products and/or services. No express or implied copyright licenses are granted hereunder to design or fabricate any silicon-based microstructures based on the information in this document.

SMI makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does SMI assume any liability arising out of the application or use of any product or silicon-based microstructure, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in SMI’s datasheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. SMI does not convey any license under its patent rights nor the rights of others. SMI makes no representation that the circuits are free of patent infringement. SMI products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SMI product could create a situation where personal injury or death may occur. Should Buyer purchase or use SMI products for any such unintended or unauthorized application, Buyer shall indemnify and hold SMI and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that SMI was negligent regarding the design or manufacture of the products.

SMI warrants goods of its manufacture as being free of defective materials and faulty workmanship. SMI standard product warranty applies unless agreed to otherwise by SMI in writing. Please refer to your order acknowledgement or contact SMI directly for specific warranty details. If warranted goods are returned to SMI during the period of coverage, SMI will repair or replace, at its option, without charge those items it finds defective. The foregoing is buyer’s sole remedy and is in lieu of all warranties, expressed or implied, including those of merchantability and fitness for a particular purpose. In no event shall SMI be liable for consequential, special, or indirect damages.

While SMI may provide application assistance to aid its customers' design process, it is up to each customer to determine the suitability of the product for its specific application. The information supplied by SMI is believed to be accurate and reliable as of this printing. However, SMI assumes no responsibility for its use. SMI assumes no responsibility for any inaccuracies and/or errors in this publication and reserves the right to make changes to any products or specifications herein without further notice.