SGPC3 Driver Integration (for Dedicated I²C Hardware)

A Step-by-Step Guide

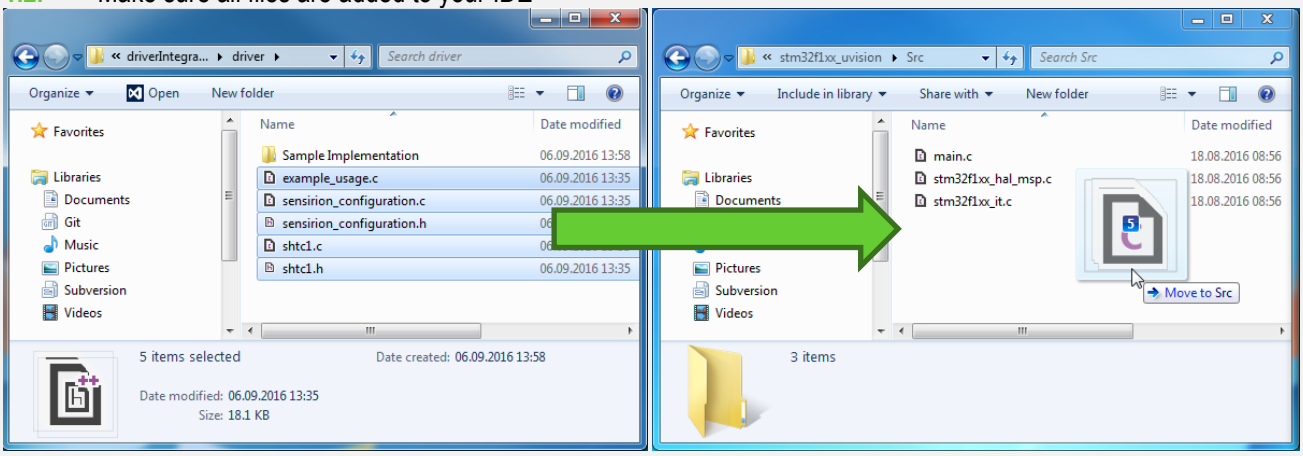
Preface

The easiest way to integrate the SGPC3 sensor into a device is Sensirion's SGPC3 driver. This document explains how to implement the hardware abstraction layer of the SGP driver and describes the provided API

COPY FILES TO YOUR PROJECT — STEP 1 — 2 — 3

1.1. Copy all SGP driver files (.c and .h) into your software project folder.

1.2. Make sure all files are added to your IDE



IMPLEMENT sensirion_configuration.c — 1 — STEP 2 — 3

To use your I²C hardware the file **sensirion_configuration.c** needs to be completed. All parts marked with "`// IMPLEMENT`" have to be replaced with code performing the necessary setup.

2.1. Implement the I²C initialization for your specific hardware.

```
void sensirion_i2c_init()
{
    // IMPLEMENT
}
```

2.2. Implement `sensirion_i2c_read()`, which executes a read command on the I²C bus, reading the given number of bytes. The specified `address` is the address of the SGP sensor. Write the number of read bytes (`count`) into the given `data` buffer.

```
int8_t sensirion_i2c_read(uint8_t address, uint8_t* data, uint16_t count)
{
    // IMPLEMENT
    return 0;
}
```

Return: 0 if read command is executed successfully, else an error code.

2.3. Implement `sensirion_i2c_write()`, which executes a write command on the I²C bus. The specified `address` is the address of the SGP sensor. Write the given number of bytes (`count`) from the buffer `data` to the I²C bus.

```
int8_t sensirion_i2c_write(uint8_t address, const uint8_t* data, uint16_t count)
{
    // IMPLEMENT
    return 0;
}
```

Note: Some implementations of I²C write/read expect an 8 bit sensor address (instead of 7 bit). In this case use `(address<<1)` instead of `address` in your implementation.

Return: 0 if the write command is executed successfully, else an error code.

2.4. Implement `sensirion_sleep_usec()`, which delays the execution for the given time in microseconds.

```
void sensirion_sleep_usec(uint32_t useconds) {
    // IMPLEMENT
}
```

MEASURE IAQ (tVOC) AND SIGNAL VALUES

1 2 STEP 3

The SGP driver provides functions to probe the sensor, to get the serial ID, and to measure/read tVOC.

3.1. Call `sgp_probe()` to initialize the I²C bus and test if the sensor is available.

```
int16_t sgp_probe(void);
```

Return: 0 if the sensor is detected, else an error code.

3.2. Call `sgp_get_serial_id ()` to readout the serial id of the SGP sensor.

```
int16_t sgp_get_serial_id (u64 *serial_id);
```

Return: 0 if the command is successful, else an error code.

3.3. Call `sgp_get_feature_set_version()` to readout the feature set version and product type of the SGP sensor. If the product type is 0 it is a SGP30 gas sensor, if it is 1 it is an SGPC3 gas sensor.

```
int16_t sgp_get_feature_set_version (u16 *feature_set_version, u8 *product_type);
```

Return: 0 if the command is successful, else an error code.

3.4. Call `sgp_measure_iaq_blocking_read()` to start a tVOC measurement and to readout the value.

```
int16_t sgp_measure_iaq_blocking_read(uint16_t *tvoc_ppb);
```

Note: This function blocks the processor while the measurement is in progress.
Return: 0 if the command is successful, else an error code.

3.5. For non-blocking measurement and readout of tVOC use the two functions `sgp_measure_iaq()` and `sgp_read_iaq()`.

```
int16_t sgp_measure_iaq(void);
```

Return: 0 if the command is successful, else an error code.

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement result using `sgp_read_iaq()`.

```
int16_t sgp_read_iaq(uint16_t *tvoc_ppb);
```

Note: If the measurement is still in progress, this function returns an error code.
Return: 0 if the command is successful, else an error code.

3.6. For best performance and faster startup times, the current baseline needs to be persistently stored on the device before shutting it down and set again accordingly after boot up.

Use `sgp_get_iaq_baseline()` to get the baseline.

```
int16_t sgp_get_iaq_baseline (uint16_t *baseline);
```

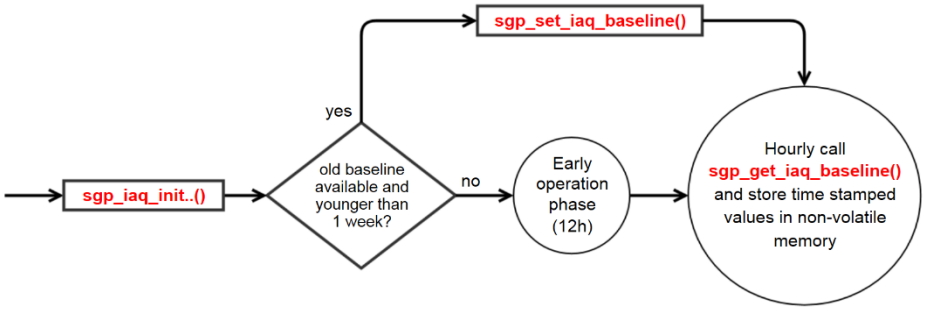
Return: 0 if the command is successful, else an error code.
Note: If the call is not successful, the `baseline` value must be discarded. Approximately in the first 15 seconds of operation after `sgp_probe` or `sgp_iaq_init` the call will fail unless a previous baseline was restored.

3.7. Use `sgp_set_iaq_baseline()` to set the baseline.

```
int16_t sgp_set_iaq_baseline (uint16_t baseline);
```

Return: 0 if the command is successful, else an error code.
Note: The baseline value must be *exactly* as returned by `sgp_get_iaq_baseline()` and should only be set if it's less than one week old.

3.8. SGP baseline states



Call `sgp_iaq_init...()` to reset all SGP baselines. After the accelerated warm-up phase (see 3.9. for durations), the initialization takes 20 seconds, during which `sgp_iaq_measure()` output will not change.

If no stored baseline is available after initializing the baseline algorithm, the sensor has to run for 12 hours until the baseline can be stored. This will ensure an optimal behavior for subsequent startups. Reading out the baseline prior should be avoided unless a valid baseline is restored first. Once the baseline is properly initialized or restored, the current baseline value should be stored approximately once per hour. While the sensor is off, baseline values are valid for a maximum of seven days.

3.9. Call one of the initialization functions `sgp_iaq_init...()` to initialize or re-initialize the indoor air quality algorithm. There are four versions corresponding to different accelerated warm-up durations. They accelerate the sensor switch-on behavior after the sensor has been switched off (down-time).

Sensor down-time	Recommended initialization	Accelerated warm-up
5 min or less	`sgp_iaq_init0()` followed by `sgp_set_iaq_baseline()`	-
5 min to 1 h	`sgp_iaq_init16()` followed by `sgp_set_iaq_baseline()`	16 s
1 h to 24 h	`sgp_iaq_init64()` followed by `sgp_set_iaq_baseline()`	64 s
24 h to 1 week	`sgp_iaq_init184()` followed by `sgp_set_iaq_baseline()`	184 s
1 week or more / initial switch-on	`sgp_iaq_init184()`	184 s

The above initialization logic ensures fast switch-on behavior for typical IAQ applications. It can be modified and optimized according to the specific limitations and requirements of individual devices. During the accelerated warm-up phase, `sgp_iaq_measure()` output will not change.

```
int16_t sgp_iaq_init0(void);
int16_t sgp_iaq_init16(void);
int16_t sgp_iaq_init64(void);
int16_t sgp_iaq_init184(void);
```

Return: 0 if the command is successful, else an error code.
Note: `sgp_iaq_init64()` is already called as part of `sgp_probe()`.

3.10. Call `sgp_measure_tvoc_blocking_read()` to start a tVOC measurement and to readout the value in ppb.

```
int16_t sgp_measure_tvoc_blocking_read(uint16_t *tvoc_ppb);
```

Note: This function blocks the processor while the measurement is in progress.

Return: 0 if the command is successful, else an error code.

3.11. For non-blocking measurement and readout of tVOC use the two functions `sgp_measure_tvoc()` and `sgp_read_tvoc()`

```
int16_t sgp_measure_tvoc(void);
```

Return: 0 if the command is successful, else an error code.

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement result using `sgp_read_tvoc()`.

```
int16_t sgp_read_tvoc(uint16_t *tvoc_ppb);
```

Note: If the measurement is still in progress, this function returns an error code.

Return: 0 if the command is successful, else an error code.

3.12. Call `sgp_measure_signals_blocking_read()` to start signal measurements and to readout the values.

```
int16_t sgp_measure_signals_blocking_read(uint16_t *scaled_ethanol_signal);
```

Return: 0 if the command is successful, else an error code.
Note: The returned values are scaled signals. To get the resulting signals, the values must first be divided by 512.
Note: This function blocks the processor while the measurement is in progress.

```
uint16_t ret;
uint16_t scaled_ethanol_signal;

// run signals measurement
ret = sgp_measure_signals_blocking_read(&scaled_ethanol_signal);
```

3.13. For non-blocking measurement and readout of signals values use the two functions `sgp_measure_signals()` and `sgp_read_signals()`.

```
int16_t sgp_measure_signals(void);
```

Return: 0 if the command is successful, else an error code.

```
uint16_t ret;
uint16_t scaled_ethanol_signal;

// run signals measurement
ret = sgp_measure_signals();
usleep(200000);
sgp_read_signals(&scaled_ethanol_signal);
```

Be sure to wait until the SGP sensor finishes the measurement. Read the measurement results using `sgp_read_signals()`.

```
int16_t sgp_read_signals(uint16_t *scaled_ethanol_signal);
```

Note: If the measurement is still in progress, this function returns an error code.
Note: The returned values are scaled signals. To get the resulting signals, the values must first be divided by 512.
Return: 0 if the command is successful, else an error code.

REVISION HISTORY

Date	Version	Page(s)	Changes
October 2016	1.0.0	all	Initial release
January 2017	1.0.1	all	Add CO2-eq output to the driver
January 2017	1.0.2	all	Fixing layout
January 2017	1.1.0	all	Add IAQ functions
January 2017	1.1.1	3	Document how long a baseline value is valid
January 2017	1.2.0	3-5	Document baseline documentation
March 2017	1.4.0	3	Update baseline persistence documentation
April 2017	1.5.0	1, 3	SGP30
Mai 2017	2.0.0	all	Change interfaces from resistance to ethanol and h2 signals
Mai 2017	2.0.1	all	Document signal scaling
July 2017	2.1.0	all	SGPC3 Documentation
July 2017	2.2.0	4, 5	Baseline reset
September 2017	2.3.0	3, 4	Accelerated warm-up by iaq_init…()
September 2017	2.3.1	2	Document sgp_get_serial_id interface

Headquarters and Subsidiaries

Sensirion AG
Laubisruetistr. 50
CH-8712 Staefa ZH
Switzerland

Phone: +41 44 306 40 00
Fax: +41 44 306 40 30
info@sensirion.com
www.sensirion.com

Sensirion Inc., USA
Phone: +1 805 409 4900
info_us@sensirion.com
www.sensirion.com

Sensirion Japan Co. Ltd.
Phone: +81 3 3444 4940
info@sensirion.co.jp
www.sensirion.co.jp

Sensirion Korea Co. Ltd.
Phone: +82 31 345 0031 3
info@sensirion.co.kr
www.sensirion.co.kr

Sensirion China Co. Ltd.
Phone: +86 755 8252 1501
info@sensirion.com.cn
www.sensirion.com.cn

Sensirion AG (Germany)
Phone: +41 44 927 11 66
info@sensirion.com
www.sensirion.com

To find your local representative, please visit www.sensirion.com/contact