CCS 811 digital gas sensor

Drivers (group__drivers.html) »
Sensor Device Drivers (group__drivers__sensors.html) |
Drivers (group__drivers.html) »
[S]ensor [A]ctuator [U]ber [L]ayer (group__drivers__saul.html)

Device Driver for AMS CCS 811 digital gas sensor for monitoring Indoor Air Quality (IAQ) More...

Detailed Description

Device Driver for AMS CCS 811 digital gas sensor for monitoring Indoor Air Quality (IAQ)

CCS811 digital gas sensor for monitoring indoor air quality

The driver is for the usage with RIOT-OS (https://github.com/RIOT-OS/RIOT).

Table of contents

Overview

About the sensor

The CCS811 is an ultra-low power digital sensor which detects **Volatile Organic Compounds (VOC)** for **Indoor Air Quality (IAQ)** monitoring that. The sensor allows to

- convert raw sensor data to Total Volatile Organic Compound (TVOC) and equivalent CO2 (eCO2),
- compensate gas readings due to temperature and humidity using an external sensor,
- trigger interrupts when new measurement results are available or eCO2 value exceeds thresholds,
- correct baseline automatically or manually
- connect a NTC thermistor to provide means of calculating the local ambient temperature,
- power-save using a sleep mode and wakeup feature.

> **Note**
>
> The sensor is connected to I2C interface and uses clock stretching. The I2C implementation of the MCU has to support clock stretching to get CCS811 working.

Back to table of contents

Supported Features

> **Note**
>
> There are two driver module versions, the `ccs811` module which provides only basic functionality and the `ccs811_full` module with additional functionality.

The `ccs811_full` module includes the `ccs811` module automatically. If code size is critical, the `ccs811` module can be used, otherwise using the `ccs811_full` module is recommended.

The driver supports the following features when modules `ccs811` and `ccs811_full` are used.

Feature	Module

read raw and converted gas sensor data (eCO2, TVOC)	`ccs811`
poling for new sensor gas data	`ccs811`
power saving using sleep mode with wakeup	`ccs811`
data ready and threshold interrupt handling	`ccs811_full`
ambient temperature calculation with NTC	`ccs811_full`
compensate gas readings using an external sensor	`ccs811_full`
manual baseline handling	`ccs811_full`

Back to table of contents

Measurement Process

Sensor modes

After power up, the sensor starts automatically in *Idle, Low Current Mode* (**CCS811_MODE_IDLE (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a86b44b3000d550d67eeb4668597fca97)**). To start periodic measurements, the mode of the sensor has to be changed to any measurement mode. Measurement modes with different output data rates are available:

Mode	Driver symbol	Period	RAW data	IAQ values
Idle, Low Current Mode	**CCS811_MODE_IDLE (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a86b44b3000d550d67eeb4668597fca97)**	-	-	-
Constant Power Mode	**CCS811_MODE_1S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a7031457a9e0df5968ccdf8eb42900947)**	1 s	X	X
Pulse Heating Mode	**CCS811_MODE_10S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6abc602af5c4495f493ad285341596e036)**	10 s	X	X
Low Power Pulse Heating Mode	**CCS811_MODE_60S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6acb7e93193b7d952313a14c7e29ff4240)**	60 s	X	X
Constant Power Mode	**CCS811_MODE_250MS (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a4c1a55eb39d8f0c812fa01605e377599)**	250 ms	X	-

In *Constant Power Mode* with measurements every 250 ms (**CCS811_MODE_250MS (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a4c1a55eb39d8f0c812fa01605e377599)**) only raw data are available. In all other measurement modes, the Indoor Air Quality (IAQ) values are available additionally. The *Constant Power Mode* with measurements every 250 ms (**CCS811_MODE_250MS (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a4c1a55eb39d8f0c812fa01605e377599)**) is only intended for systems where an external host system wants to run an algorithm with raw data.

Note

- After setting the mode, the sensor is in conditioning period that needs up to 20 minutes, before accurate readings are generated, see the data sheet (https://ams.com/documents/20143/36005/CCS811_DS000459_7-00.pdf/3cfdaea5-b602-fe28-1a14-18776b61a35a) for more details.
- During the early-live (burn-in) period, the CCS811 sensor should run for 48 hours in the selected mode of operation to ensure sensor performance is stable, see the data sheet for more details.
- When the sensor operating mode is changed to a new mode with a lower sample rate, e.g., from *Pulse Heating Mode* (**CCS811_MODE_10S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6abc602af5c4495f493ad285341596e036)**) to *Low Power Pulse Heating Mode* (**CCS811_MODE_60S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6acb7e93193b7d952313a14c7e29ff4240)**), it should be placed in *Idle, Low Current Mode* (**CCS811_MODE_IDLE (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a86b44b3000d550d67eeb4668597fca97)**) for at least 10 minutes before enabling the new mode. When the sensor operating mode is changed to a new mode with a higher sample rate, e.g., from Low Power Pulse Heating Mode* (**CCS811_MODE_60S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6acb7e93193b7d952313a14c7e29ff4240)**) to *Pulse Heating Mode* (**CCS811_MODE_10S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6abc602af5c4495f493ad285341596e036)**), there is no requirement to wait before enabling the new mode.

When default configuration parameters from **ccs811_params.h (ccs811__params_8h.html)** are used, the **CCS811_MODE_1S (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a7031457a9e0df5968ccdf8eb42900947)** measurement mode is used automatically. The application can change the measurement mode either

- by using the **ccs811_set_mode (ccs811_8h.html#ad9d9c8ccaec99388620c90c6d8341acd)** function, or
- by defining **CCS811_PARAM_MODE** before **ccs811_params.h (ccs811__params_8h.html)** is included.

Back to table of contents

Measurement results

Once the measurement mode is set, the user task can use function **ccs811_read_iaq (ccs811_8h.html#a44b640a9ded3e8d625bfbeb65105fb4f)** to fetch the results. The function returns **raw data** as well as **Indoor Air Quality (IAQ)** values. If some of the results are not needed, the corresponding parameters can be set to `NULL` .

While raw data represents simply the current through the sensor and the voltage across the sensor with the selected current, IAQ values are the results of the processing of these raw data by the sensor. IAQ values consist of the **equivalent CO2 (eCO2)** with a range from 400 ppm to 8192 ppm and **Total Volatile Organic Compound (TVOC)** with a range from 0 ppb to 1187 ppb.

```
uint16_t iaq_tvoc;
uint16_t iaq_eco2;
uint16_t raw_i;
uint16_t raw_v;
...
/* get the results and do something with them */
if (ccs811_read_iaq (ccs811_8h.html#a44b640a9ded3e8d625bfbeb65105fb4f) (&sensor, &tvoc, &eco2, &raw_i, &raw_v) == CCS811_OK (ccs811_8h.html#a681
        f09a12b30f0b0d73aa7a5e94a0abeab7e402a1c832c3ce907dfeec6cbd9518)) {
    ...
}
else {
    ... /* error handling */
}
...
```

If the **ccs811_read_iaq (ccs811_8h.html#a44b640a9ded3e8d625bfbeb65105fb4f)** function is called and no new data are available, the function returns the results of the last valid measurement and error code **CCS811_ERROR_NO_NEW_DATA (ccs811_8h.html#a681f09a12b30f0b0d73aa7a5e94a0abea6718f6b4c0fd99d05c3b4a01a883ea5c)**.

There are two approaches to wait until new data are available:

- data-ready interrupt (**CCS811_INT_DATA_READY (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1cb2385afadf7c9391354a15389d2e7617aa70ebe5)**)
- data-ready status function (**ccs811_data_ready (ccs811_8h.html#a70ccdeb006d85d74195c8e0f4ad6f452)**)

```
uint16_t iaq_tvoc;
uint16_t iaq_eco2;
uint16_t raw_i;
uint16_t raw_v;
...
/* check whether new data are available, get the data and do something with them */
if (ccs811_data_ready (ccs811_8h.html#a70ccdeb006d85d74195c8e0f4ad6f452) (&sensor) == CCS811_OK (ccs811_8h.html#a681f09a12b30f0b0d73aa7a5e94a0ab
        eab7e402a1c832c3ce907dfeec6cbd9518) &&
    ccs811_read_iaq (ccs811_8h.html#a44b640a9ded3e8d625bfbeb65105fb4f) (&sensor, &tvoc, &eco2, &raw_i, &raw_v) == CCS811_OK (ccs811_8h.html#a681
        f09a12b30f0b0d73aa7a5e94a0abeab7e402a1c832c3ce907dfeec6cbd9518))
{
    ...
}
...
```

When using data-ready interrupts, the default configuration parameter for the interrupt pin can be overridden by defining **CCS811_PARAM_INT_PIN** before **ccs811_params.h (ccs811__params_8h.html)** is included.

Compensation

If information about the environment like temperature and humidity are available from another sensor, they can be used by CCS811 to compensate gas readings due to temperature and humidity changes.

> **Note**
>
> This feature can only be used with the `ccs811_full` module.

Function **ccs811_set_environmental_data (ccs811_8h.html#a76d4999e645bf4c5828c8c95f29bc1e8)** can be used to set these environmental data. In the following example, the Sensirion SHT3x humidity and temperature sensor is used to fetch environmental data.

```
int16_t  temperature;   /* in hundredths of a degree Celsius */
int16_t  humidity;      /* in hundredths of a percent */
...
if (sht3x_get_results (sht3x, &temperature, &humidity))
    /* set CCS811 environmental data with values fetched from SHT3x */
    ccs811_set_environmental_data (ccs811_8h.html#a76d4999e645bf4c5828c8c95f29bc1e8) (ccs811, temperature, humidity);
...
```

Back to table of contents

Negative Thermal Coefficient Thermistor (NTC)

CCS811 supports an external interface for connecting a negative thermal coefficient thermistor (R_NTC) to provide a cost effective and power efficient means of calculating the local ambient temperature.

> **Note**
>
> This feature can only be used with the `ccs811_full` module.

The sensor measures the voltage V_NTC across R_NTC as well as the voltage V_REF across a connected reference resistor (R_REF). Function **ccs811_read_ntc (ccs811_8h.html#aee00eaec33e486c62b1ab9601b6a7f78)** can be used at any time to fetch the current resistance of R_NTC. It uses the resistance of R_REF and measured voltages V_REF and V_NTV with the following equation to determine R_NTC:

```
R_NTC = R_REF / V_REF * V_NTC
```

Using the data sheet of the NTC, the ambient temperature can be calculated. See application note AMS AN000372 for more details. For example, with Adafruit CCS811 Air Quality Sensor Breakout (https://www.adafruit.com/product/3566) the ambient temperature can be determined as following:

```
...
#define CCS811_R_REF       100000 /* resistance of the reference resistor */
#define CCS811_R_NTC       10000  /* resistance of NTC at a reference temperature */
#define CCS811_R_NTC_TEMP  25     /* reference temperature for NTC */
#define CCS811_BCONSTANT   3380   /* B constant */

/* get NTC resistance */
uint32_t r_ntc;
ccs811_read_ntc (ccs811_8h.html#aee00eaec33e486c62b1ab9601b6a7f78) (&sensor, CCS811_R_REF, &r_ntc);

/* calculation of temperature from application note ams AN000372 */
double ntc_temp;
ntc_temp  = log (structlog.html)((double)r_ntc / CCS811_R_NTC);      /* 1 */
ntc_temp /= CCS811_BCONSTANT;                       /* 2 */
ntc_temp += 1.0 / (CCS811_R_NTC_TEMP + 273.15);     /* 3 */
ntc_temp  = 1.0 / ntc_temp;                         /* 4 */
ntc_temp -= 273.15;                                 /* 5 */
....
```

Back to table of contents

Interrupts

CCS811 supports two types of interrupts that can be used to fetch data:

- data ready interrupt (**CCS811_INT_DATA_READY (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1cb2385afadf7c9391354a15389d2e7617aa70ebe5)**)
- threshold interrupt (**CCS811_INT_THRESHOLD (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1cb2385afac3ac04f6d85738e1c4416ebbbc71910f)**)

Note

- Interrupts can only be used with the `ccs811_full` module.
- It is not possible to use both interrupts at the same time.

Back to table of contents

Data ready interrupt

At the end of each measurement cycle (every 250 ms, 1 second, 10 seconds, or 60 seconds), CCS811 can optionally trigger an interrupt. The signal nINT** is driven low as soon as new sensor values are ready to read. It will stop being driven low when sensor data are read with function **ccs811_read_iaq (ccs811_8h.html#a44b640a9ded3e8d625bfbeb65105fb4f)**.

The interrupt is disabled by default. It can be enabled using function **ccs811_set_int_mode (ccs811_8h.html#a014596244c461cee961f37fb84d6c314)**.

```
...
/* enable the data ready interrupt */
ccs811_set_int_mode (&sensor, CCS811_INT_DATA_READY (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1cb2385afadf7c9391354a15389d2e7617aa70ebe5));
...
```

Back to table of contents

Threshold interrupt

The user task can choose that the data ready interrupt is not generated every time when new sensor values become ready but only if the eCO2 value moves from the current range (LOW, MEDIUM, or HIGH) into another range by more than a hysteresis value. Hysteresis is used to prevent multiple interrupts close to a threshold.

The interrupt is disabled by default and can be enabled with function **ccs811_set_int_mode (ccs811_8h.html#a014596244c461cee961f37fb84d6c314)**. The ranges are defined by the **ccs811_set_eco2_thresholds (ccs811_8h.html#ae3d450b232f9b2dc94db090ec38832b7)** function and its parameters `low` and `high` as following:

Name	Range	Value	Default
LOW	below the `low` parameter	> 400	1500
MEDIUM	between the `low` and `high` parameters		
HIGH	above the value of the `high` parameter	< 8192	2500

```
...
/* set threshold parameters and enable threshold interrupt mode */
ccs811_set_eco2_thresholds (&sensor, 600, 1100, 40);
ccs811_set_int_mode (ccs811_8h.html#a014596244c461cee961f37fb84d6c314) (&sensor, CCS811_INT_THRESHOLD (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1c
       b2385afac3ac04f6d85738e1c4416ebbbc71910f));
...
```

Back to table of contents

Power Saving

The CCS811 offers a **sleep mode** with **wake-up** function. By using the active low **nWAKE** signal connected to a GPIO, power can be saved. If the nWAKE** signal is low, the CCS811 is active and can communicate over I2C. When this signal is high, the CCS811 goes into sleep mode and can't be reached via I2C. The measuring process is not affected.

The driver supports this feature when the **nWAKE** signal pin (**ccs811_params_t::wake_pin (structccs811__params__t.html#ada46200e2d3d472a8a8092ef2ee0b563)**) is configured, see the Configuration section.

Note

If the **nWAKE** signal pin is not used, it must be permanently pulled down. Sleep mode/wake-up feature can not be used in this case.

Additionally, CCS811 can be disabled with the **ccs811_power_down (ccs811_8h.html#afb9025598f9bb755e644894b5f94f7d9)** function function, when no measurements are required. For that purpose, the sensor is switched to the idle, low current mode (**CCS811_MODE_IDLE (ccs811_8h.html#add4cd91ac7b3efa8b16a12d6d00e54e6a86b44b3000d550d67eeb4668597fca97)**). To reactivate the CCS811 in the previous measurement mode, the **ccs811_power_up (ccs811_8h.html#a6dd19981227ccb1f2a523f8d72265ae7)** function has to be used.

Note

It may take several minutes before accurate readings are generated when the sensor switches back from idle mode to the previous measurement mode. Therefore, the best power-saving solution is to leave the sensor in any measurement mode and to use it with data-ready interrupt (**CCS811_INT_DATA_READY (ccs811_8h.html#a7fff180e6cfaa9cd97ac03a1cb2385afadf7c9391354a15389d2e7617aa70ebe5)**) in conjunction with the **nWAKE** signal pin.

Back to table of contents

Baseline

CCS81 supports automatic baseline correction over a minimum time of 24 hours. Using function **ccs811_get_baseline (ccs811_8h.html#ade3f96733cfda8c29df2a2e7e9d72198)**, the current baseline value can be saved before the sensor is powered down. This baseline can then be restored with function **ccs811_set_baseline (ccs811_8h.html#afee6dd7a959938b98f4a28ca86693151)** after sensor is powered up again to continue the automatic baseline process.

Note

This feature can only be used with the `ccs811_full` module.

Back to table of contents

Error Handling

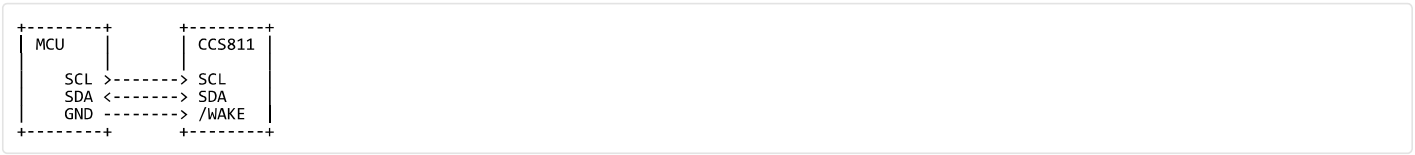
All driver functions return an error code (**ccs811_error_codes_t (ccs811_8h.html#a681f09a12b30f0b0d73aa7a5e94a0abe)**) to indicate whether its execution was successful or an error happened.

Back to table of contents

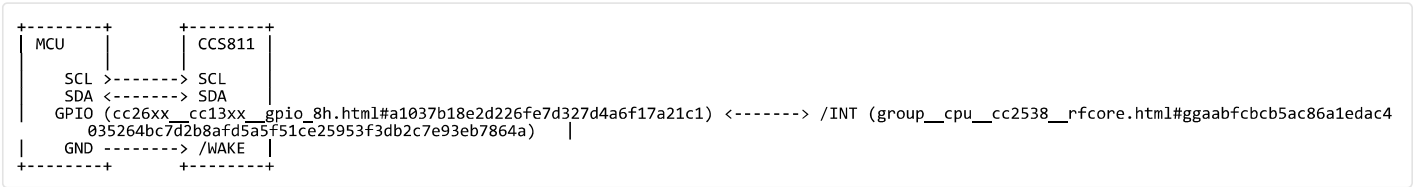
Configuration

Hardware Configurations

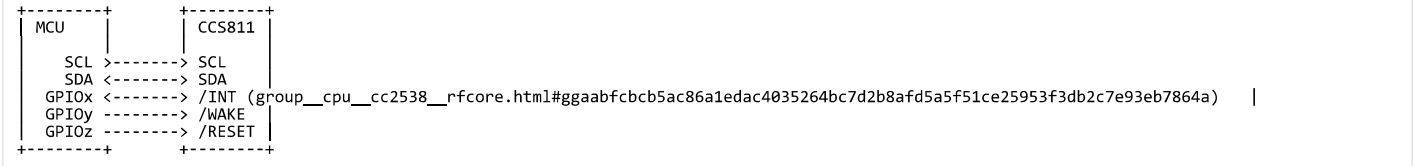
The following figure shows the most simple hardware configuration with CCS811. With this configuration interrupts, the hardware reset, and the sleep mode of the sensor with wake-up feature can't be used. The signals nINT** and **nRESET** are not connected. The **nWAKE** signal is permanently pulled low, leaving the CCS811 and I2C constantly active.



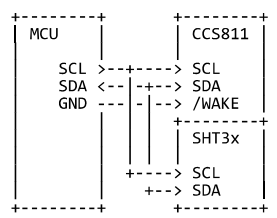
If the interrupt signal **nINT** is used to fetch new data (only with `ccs811_full` module), the interrupt pin has to be connected to a GPIO pin.



To use the hardware reset and/or the sleep mode with wake-up feature, additional GPIOs have to be used. This is the most energy-efficient hardware configuration of the sensor but requires more GPIO pins. Used GPIOs must be configured accordingly in driver configuration parameters.



If CCS811 sensor is used in conjunction with another sensor, e.g., a SHT3x sensor, the hardware configuration looks like following:



Back to table of contents

Driver Configuration Parameters

The following configuration parameters can be used to configure the sensor during its initialization (**ccs811_init (ccs811_8h.html#a8e1e27e8cd38d6684e70f0a60395d3a9)**):

Parameter	Member	Define macro	Default
I2C device	**ccs811_params_t::i2c_dev (structccs811__params__t.html#a3e2f5895e54cb4ff243010fcf5f9ba2f)**	CCS811_PARAM_I2C_DEV	**I2C_DEV(0) (group__drivers__periph_**
I2C slave address	**ccs811_params_t::i2c_addr (structccs811__params__t.html#a558456ac2f9b91980b42dbe6cfe0e2c8)**	CCS811_PARAM_I2C_ADDR	**CCS811_I2C_ADDRESS_1 (ccs811_8h**
Measurement mode	**ccs811_params_t::mode (structccs811__params__t.html#a5b6abd7b2f40b72df7417610459caf3b)**	CCS811_PARAM_MODE	**CCS811_MODE_1S (ccs811_8h.html#add4cd91ac7b3efa8**
Interrupt mode	**ccs811_params_t::int_mode (structccs811__params__t.html#afdcd31e233d660035f38b6e942c04639)**	CCS811_PARAM_INT_MODE	**CCS811_INT_NONE (ccs811_8h.html#a7fff180e6cfaa9cd97**
Interrupt pin	**ccs811_params_t::int_pin (structccs811__params__t.html#aa2c6c7b78d810e2f49d951c875132425)**	CCS811_PARAM_INT_PIN	**GPIO_PIN(0, 0) (group__drivers__peri**
Wake-up pin	**ccs811_params_t::wake_pin (structccs811__params__t.html#ada46200e2d3d472a8a8092ef2ee0b563)**	CCS811_PARAM_WAKE_PIN	**GPIO_UNDEF (group__drivers__perip**
Reset pin	**ccs811_params_t::reset_pin (structccs811__params__t.html#a322bc5bd35cea9f78c6ddc63d010bdd6)**	CCS811_PARAM_RESET_PIN	**GPIO_UNDEF (group__drivers__perip**

The default configuration of these parameters can be overridden by defining according macros before including **ccs811_params.h (ccs811__params_8h.html)**, for example:

```
#define CCS811_PARAM_I2C_DEV     (I2C_DEV(1))
#define CCS811_PARAM_I2C_ADDR    (CCS811_I2C_ADDRESS_2)
#define CCS811_PARAM_MODE        (CCS811_MODE_10S)
#define CCS811_PARAM_RESET_PIN   (GPIO_PIN(0, 0))
#define CCS811_PARAM_WAKE_PIN    (GPIO_PIN(0, 1))
#define CCS811_PARAM_INT_PIN     (GPIO_PIN(0, 2))
#define CCS811_PARAM_INT_MODE    (CCS811_INT_DATA_READY)
...
#include "ccs811.h"
#include "ccs811_params.h (ccs811__params_8h.html)"
```

Alternatively, the complete set of default configuration parameters could also be overridden by a single definition, for example:

```
#define CCS811_PARAMS    { .i2c_dev   = I2C, \
                           .i2c_addr  = CCS811_I2C_ADDRESS_2, \
                           .mode      = CCS811_MODE_10S, \
                           .reset_pin = GPIO_PIN(0, 0), \
                           .wake_pin  = GPIO_PIN(0, 1), \
                           .int_pin   = GPIO_PIN(0, 2), \
                           .int_mode  = CCS811_INT_DATA_READY, \
                         }
```

Back to table of contents

Files

file **ccs811_params.h (ccs811__params_8h.html)**
Default configuration for AMS CCS811 digital gas sensors.

file **ccs811_regs.h (ccs811__regs_8h.html)**
Register definitions for the AMS CCS811 digital gas sensor.

file **ccs811.h (ccs811_8h.html)**
Device Driver for AMS CCS811 digital gas sensor.