Quick-start guide for XENSIV™ PAS CO2 Shield2Go

About this document

Scope and purpose

This application note serves as a starting guide and will focus on the setup and communication of the XENSIV™ PAS CO2 sensor driven by a microcontroller. Main focus will be the I²C functionality along with a quick example of how to set up communication and start basic measurement using the Cypress PSoC® 6 WiFi-BT Pioneer Kit and the Arduino Due.

Intended audience

Application engineers, system engineers and makers.

Table of contents

1 Introduction

The XENSIV™ PAS CO2 sensor is a real carbon dioxide (CO_2) sensor in an unprecedentedly small form factor. Designed on the basis of a unique photoacoustic spectroscopy (PAS) concept, the sensor saves more than 75 percent space compared to existing commercial real CO_2 sensors. Its direct ppm readings, SMD capability and simple design allow for quicker and easier integration into customers' systems in low- and high-volume applications alike.

The photoacoustic principle can be traced back to over 100 years ago, first discovered by Alexander Graham Bell in 1880. The photoacoustic effect involves the formation of sound waves (pressure changes) following light absorption in a material sample. The sound signal is quantified by detectors such as microphones. In order to obtain this effect, the light intensity must vary. A PAS gas sensor is based on the principle that gases absorb light in a specific wavelength of the infrared spectrum. CO_2 molecules, for example, have strong absorption in the λ = 4.2 µm wavelength.

The XENSIV™ PAS CO2 sensor module integrates, on the same PCB, the PAS transducer, a microcontroller for signal processing, algorithms and a MOSFET. As depicted in the block diagram, the PAS transducer includes: i) a proprietary infrared emitter with blackbody radiation, which is periodically chopped by the MOSFET; ii) a narrow-band optical filter passing the CO_2 specific wavelength λ = 4.2 µm, significantly improving the sensor selectivity compared to other gases, including humidity; and iii) Infineon's high-SNR (signal-to-noise ratio) MEMS microphone XENSIV™ IM69D130, detecting the pressure changes generated by the CO_2 molecules. All the components are developed and designed in-house in accordance with Infineon's high-quality guidelines. The sensor therefore benefits from Infineon's illustrious record of accomplishments in MEMS design and acoustic capabilities, resulting in it being best-in-class for price/performance.

The XENSIV™ PAS CO2 sensor is ideal for smart-home and building automation as well as various indoor air quality IoT devices such as air purifiers, thermostats, weather stations and personal assistants. The sensor enables end users to track, understand and improve the air quality surrounding them in a timely and highly energy-efficient manner.

Figure 1 XENSIV™ PAS CO2 sensor

Introduction

1.1 XENSIV™ PAS CO2 Shield2Go

- 12 V generated on board
- UART and I²C compatible
- Breakable header

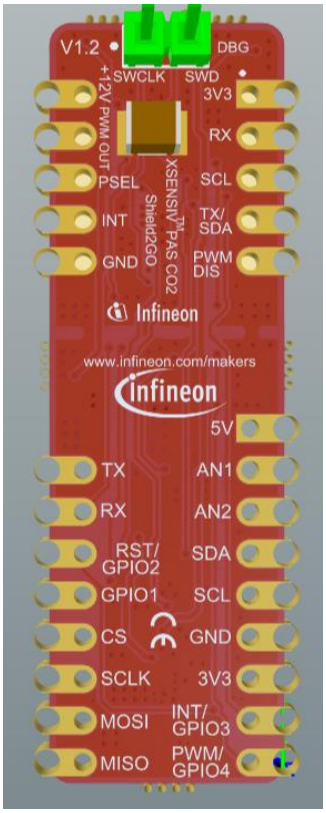

Figure 2 Pinout XENSIV™ PAS CO2 Shield2Go

Figure 3 Pin descriptions

Pin	Symbol	Type	Function
1	VDD	Power supply (3.3 V)	3.3 V digital power supply
2	RX	Input	UART receiver pin (3.3 V domain)
3	SCL	Input/Output	I²C clock pin (3.3 V domain)
4	TX_SDA	Input/Output	UART transmitter pin/I²C data pin (3.3 V domain)
5	PWM_DIS	Input	PWM disable input pin (3.3 V domain)[2)]
6	GND	Ground	Ground
7	INT	Output	Interrupt output pin (3.3 V domain)
8	PSEL	Input	Communication interface select input pin (3.3 V domain)[1)]
9	PWM	Output	PWM output pin (3.3 V domain)
10	VDD12	Power supply (12 V)	12 V power supply for the IR emitter

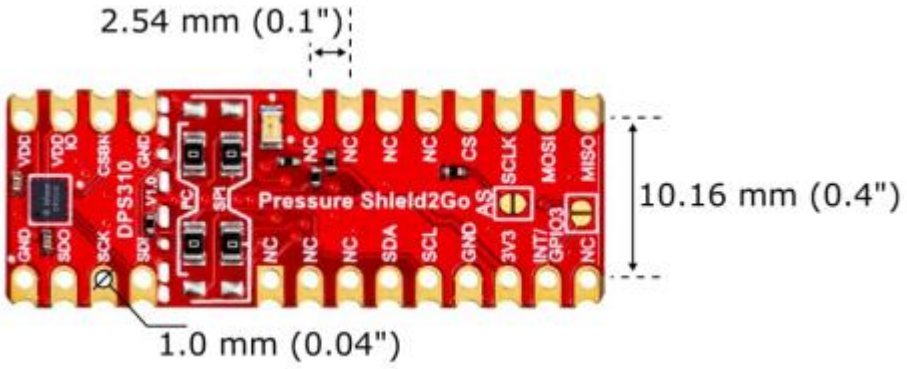

Figure 4 Example of an I²C bus configuration

1.2 Shield2Go platform

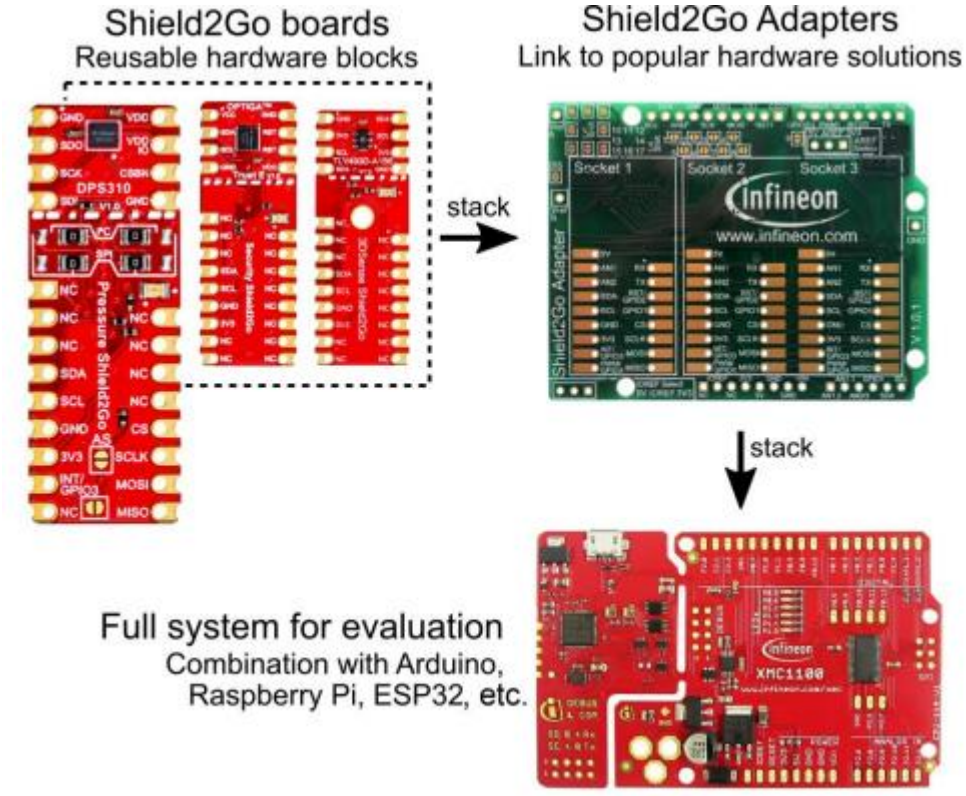

Figure 5 Combining Shield2Go board and My IoT adapter to a system

2 Available libraries

There are two libraries available, one for the PSoC environment and one for the Arduino environment. More details and full library structure can be found in their dedicated websites.

https://github.com/Infineon/arduino-pas-co2-sensor

https://github.com/Infineon/sensor-xensiv-pasco2#latest-v0.X#$$ASSET_REPO$$/sensor-xensiv-pasco2/latest-v0.X

2.1 Software installation

1. Install Arduino IDE. If you are new to Arduino, please download the program and install it first.

2. Install XMC Board. The official Arduino boards are already available in the Arduino software, but other third-party boards as the Infineon XMC MCU based need to be explicitly included. Follow the instructions in the link to add the XMC board family to Arduino. Do not forget to install as well the JLink software.

3. Install the library. In the Arduino IDE, go to the menu Sketch > Include library > Library Manager. Type pas-co2-sensor and install the library.

2.2 Library integration

3 Quick start with XMC2Go

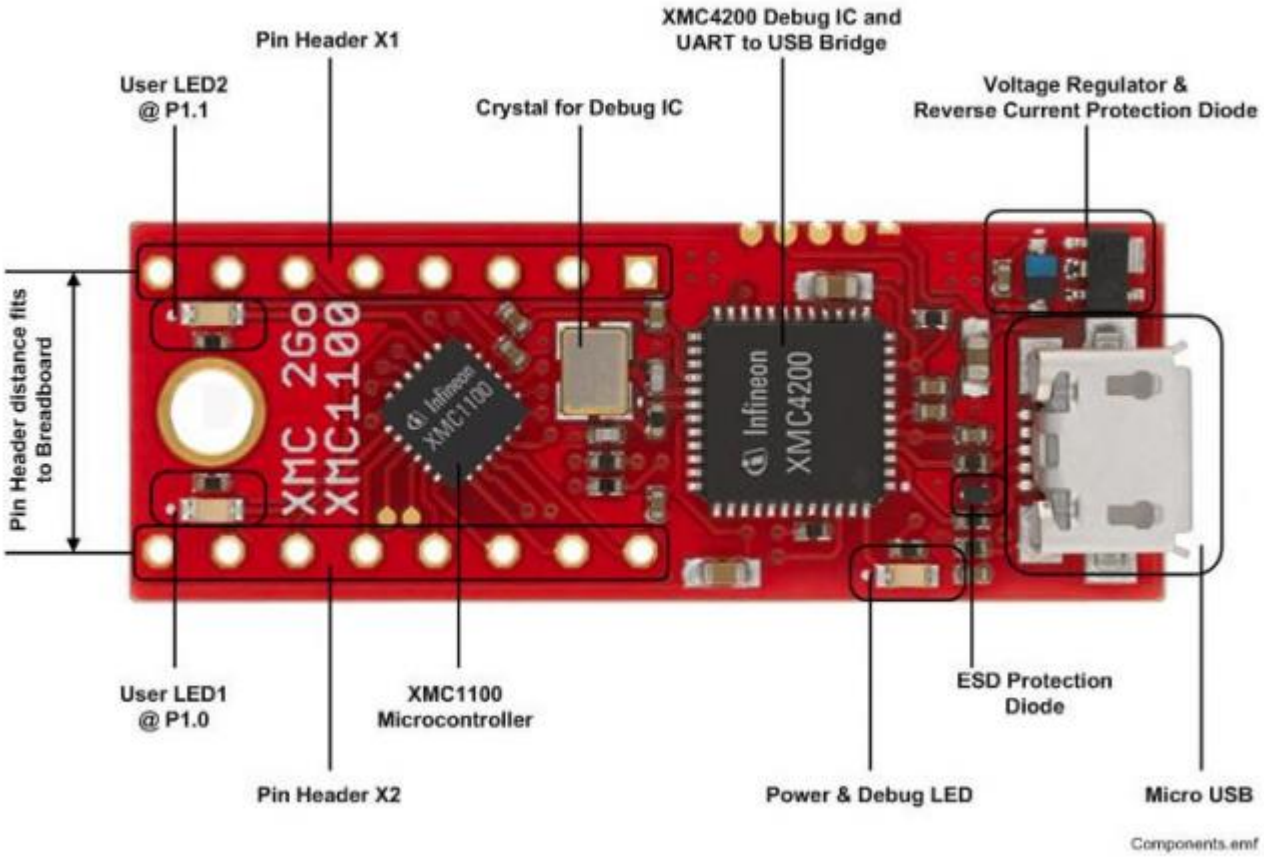

Figure 6 XENSIV™ PAS CO2 Sensor2Go Kit I²C interface connection to the Arduino Due[4)]

The pin connection of the device to the Arduino Due is equivalent to the connection to the PSoC® 6 WiFi-BT Pioneer Kit. When using another Arduino or other digital pins, make sure that the respective pull-up resistors are available.

[4)] Pinout of Arduino Due from https://content.arduino.cc/assets/Pinout-Due_latest.pdf

3.1 Hardware connection

After installing the Arduino IDE, make sure to install the right package “Arduino SAM Boards” (32-bit ARM® Cortex-M3) including the Arduino Due with the board manager (see Figure 11). Make sure to select the respective board and COM port in the Tools dropdown menu (see Figure 12). Use the programming port for uploading sketches and communicating with the Arduino Due. It is recommended to first check with the “Blink” example if communication with the Arduino Due is present and responsive. After the communication with the Arduino Due is set and confirmed, implementation of the code can begin.

Quick start with XMC2Go

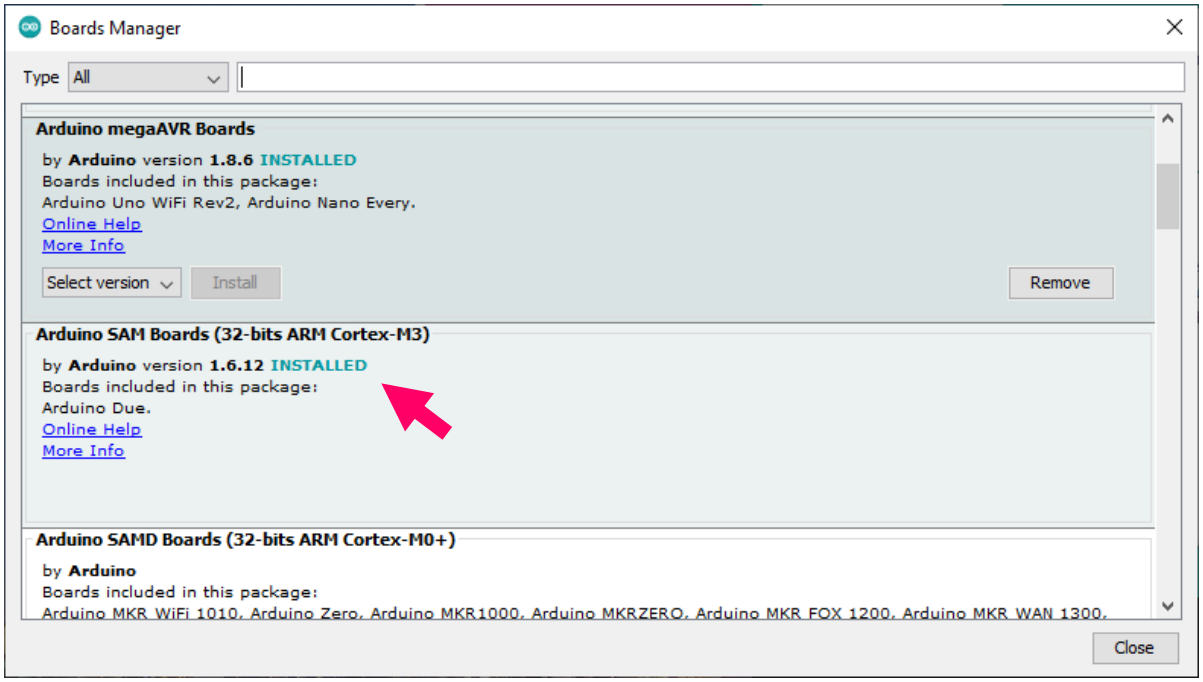

Figure 7 Arduino IDE settings: board manager

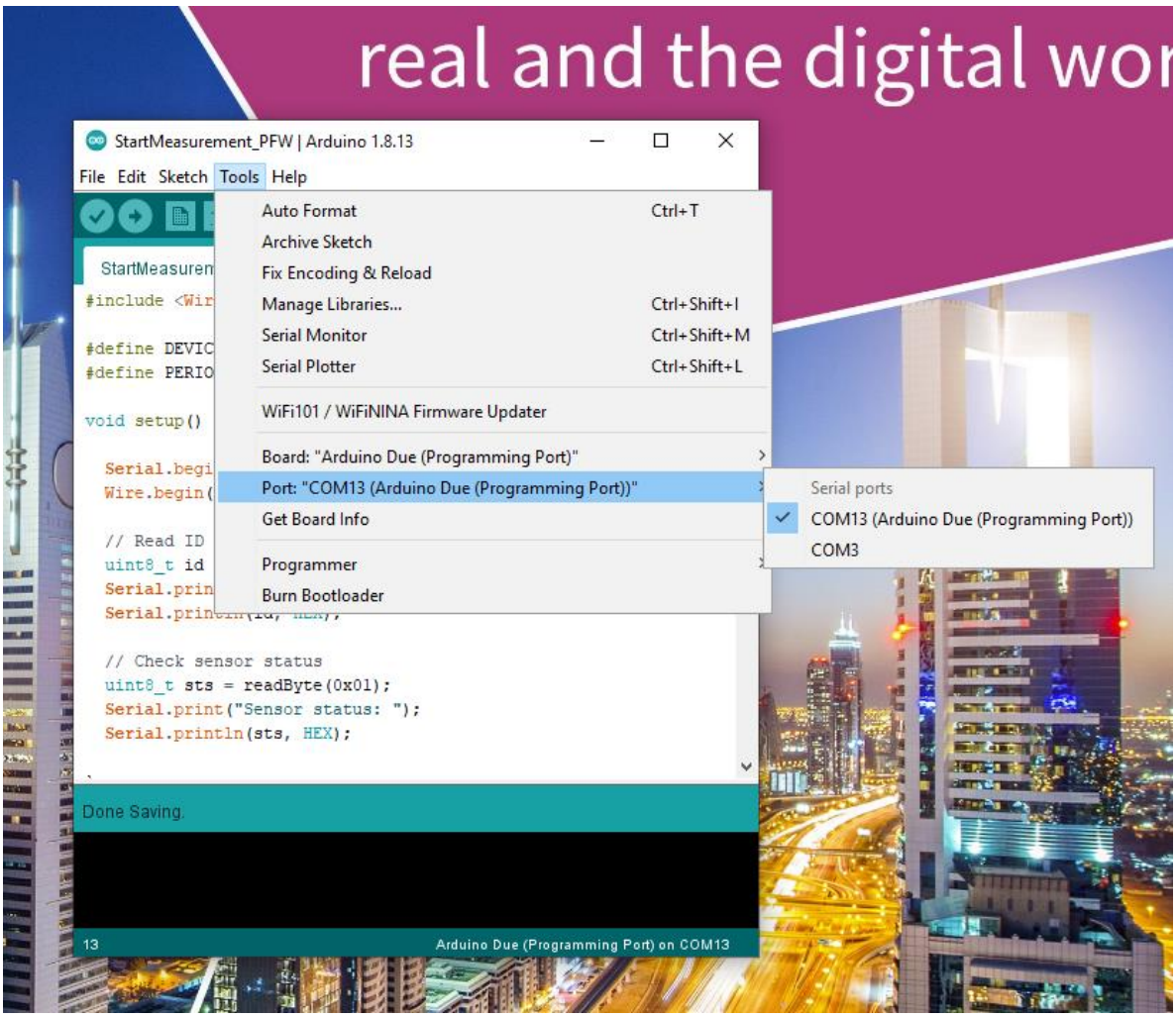

Figure 8 Arduino IDE settings: selecting the port

3.2 Step by step CO2 concentration readout example

It is recommended to implement functions for read and write commands. After that the device can be initialized, checked and operated. A measurement can be started just like with the PSoC® 6 WiFi-BT Pioneer Kit by writing and reading the responding registers. One thing to note is that it is important to set sufficient delays so that no command packages get lost or skipped. Following are four Arduino code examples: one for reading the register, one for writing into the register, and one script each for utilizing these functions to start a single-shot measurement or continuous mode measurement.

readByte

```
uint8_t readByte(uint8_t regAddress)
{
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.endTransmission(false);
  //request 1 byte from slave
  if (Wire.requestFrom(deviceAddress, 1U, 1U) > 0)
  {
    return Wire.read();
  }
  else
  {
    return 0x0;
  }
}
```

writeByte

```
bool writeByte(uint8_t regAddress, uint8_t data)
{

  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.write(data);
  if (Wire.endTransmission() != 0)
  {
    return false;
  }
  else
  {
    return true;
  }
}
```

Code for starting a single-shot measurement

```
#include <Wire.h>

#define deviceAddress 0x28
#define PERIOD 10000

void setup() {

  Serial.begin(115200);
  Wire.begin();

  // Check sensor status
  uint8_t sts = readByte(0x01);
  Serial.print("Sensor status: ");
  Serial.println(sts, HEX);

  // Idle mode
  writeByte(0x04, 0x00);
  delay(400);

  // Set pressure
  writeByte(0x0B, 0x03);
  writeByte(0x0C, 0xF5);
}

void loop()
{
  // Trigger single measurement
  writeByte(0x04, 0x01);
  delay(400);

  // Get PPM value
  uint8_t value1 = readByte(0x05);
  delay(5);
  uint8_t value2 = readByte(0x06);
  delay(5);

  // Calculate ppm value
  int16_t result = value1 << 8 | value2;
  Serial.print("CO2: ");
  Serial.print(result);
  Serial.println(" ppm");

  delay(PERIOD);

}
```

Code for starting a continuous mode measurement

```
#include <Wire.h>
#define deviceAddress 0x28

void setup() {

  Serial.begin(115200);
  Wire.begin();

  // Read ID
  uint8_t id = readByte(0x00);
  Serial.print("ID: ");
  Serial.println(id, HEX);

  // Check sensor status
  uint8_t sts = readByte(0x01);
  Serial.print("Sensor status: ");
  Serial.println(sts, HEX);

  // Idle mode
  writeByte(0x04, 0x00);
  delay(400);

  // Set measurement rate to 10 s
  writeByte(0x02, 0x00);
  writeByte(0x03, 0x0A);

  // Configure continous mode
  writeByte(0x04, 0x02);
}

void loop() {

  // Poll measurement status
  uint8_t meas_sts = readByte(0x07);
  delay(100);

  if (meas_sts == 0x10) {

    // Get PPM value
    uint8_t value1 = readByte(0x05);
    delay(5);
    uint8_t value2 = readByte(0x06);
    delay(5);

    // Calculate ppm value
    int16_t result = value1 << 8 | value2;
    Serial.print("CO2: ");
    Serial.print(result);
    Serial.println(" ppm");
  }
  delay(1000);
}
```

4 Appendix

4.1 Schematics for XENSIV™ PAS CO2 Shield2Go

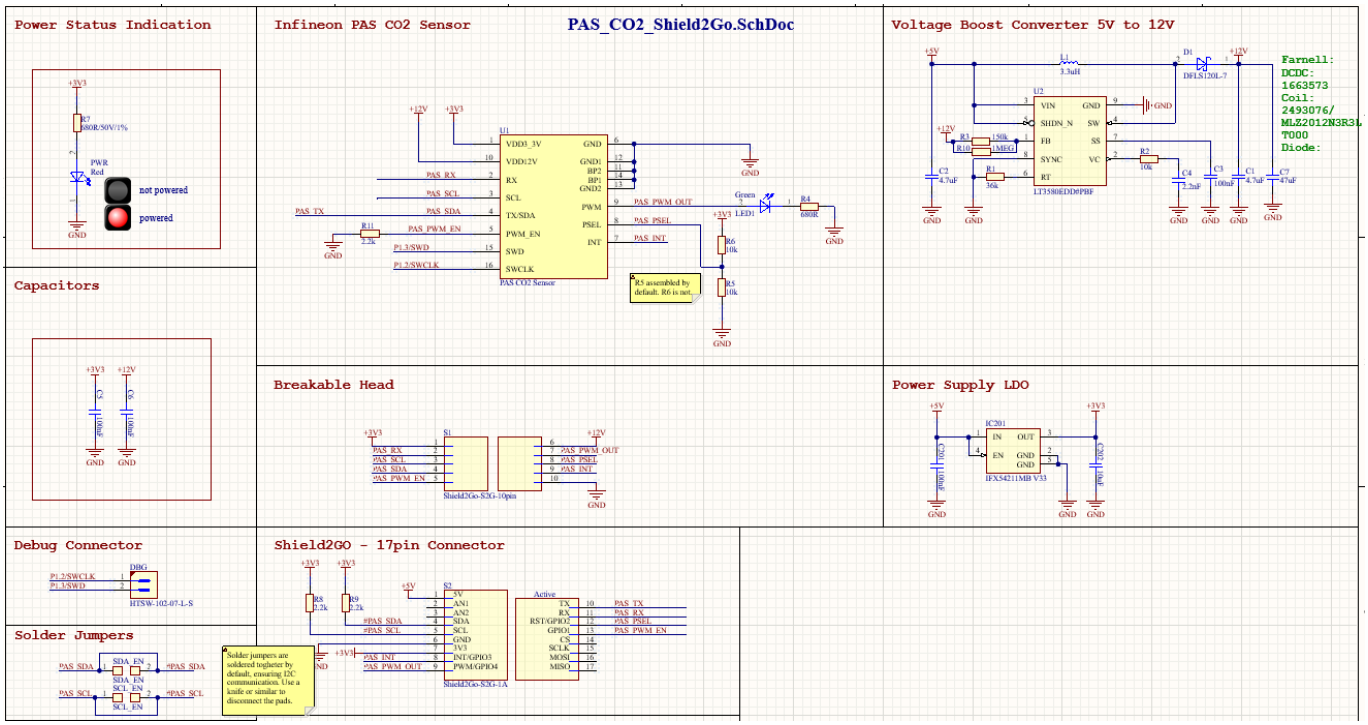

Figure 9 XENSIV™ PAS CO2 Shield2Go schematic

Revision history

Document version	Date of release	Description of changes
V 1.0	04.11.2020	Creation

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-11-04

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN_2011_PL38_2011_134235

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.