

# Programming guide for XENSIV™ PAS CO2

## Target application: demand control ventilation

### About this document

#### Scope and purpose

This application note serves as a programming 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 I<sup>2</sup>C 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 system architects of HVAC systems.

### Table of contents

<b>About this document</b> .....	<b>1</b>
<b>Table of contents</b> .....	<b>1</b>
<b>1 Introduction</b> .....	<b>2</b>
1.1 Introduction to the I <sup>2</sup> C bus.....	3
1.2 I <sup>2</sup> C communication protocol.....	3
<b>2 I<sup>2</sup>C application circuit</b> .....	<b>5</b>
<b>3 Initialization sequence</b> .....	<b>6</b>
<b>4 Quick start with the PSoC® 6 WiFi-BT Pioneer Kit</b> .....	<b>7</b>
4.1 Bridge Control Panel .....	8
4.2 Basic code for starting measurement .....	10
4.3 Additional functionality .....	11
<b>5 Quick start with the Arduino Due</b> .....	<b>13</b>
5.1 Arduino IDE.....	13
5.2 Basic Arduino code for starting measurement .....	15
<b>6 UART interface</b> .....	<b>18</b>
6.1 Write transactions .....	19
6.2 Read transactions.....	20
<b>7 PWM interface</b> .....	<b>22</b>
<b>Revision history</b> .....	<b>26</b>

## Introduction

### 1 Introduction

The XENSIV™ PAS CO2 sensor is a real carbon dioxide (CO<sub>2</sub>) 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<sub>2</sub> 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<sub>2</sub> molecules, for example, have strong absorption in the  $\lambda = 4.2 \mu\text{m}$  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<sub>2</sub> specific wavelength  $\lambda = 4.2 \mu\text{m}$ , 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<sub>2</sub> 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.

In the following the I<sup>2</sup>C interface will be explained in detail and at the end of the application note the UART and PWM interface will be covered.



**Figure 1** XENSIV™ PAS CO2 sensor

Introduction

1.1 Introduction to the I<sup>2</sup>C bus

The I<sup>2</sup>C bus is a bidirectional two-line bus, enabling communication between any kind of integrated circuit that supports this protocol, either by hardware or software. Examples of such ICs are LCD controllers, EEPROMs, RAMs, data converters or general-purpose microcontrollers. The main advantage of this protocol is its two-line interface, as shown in Figure 2.

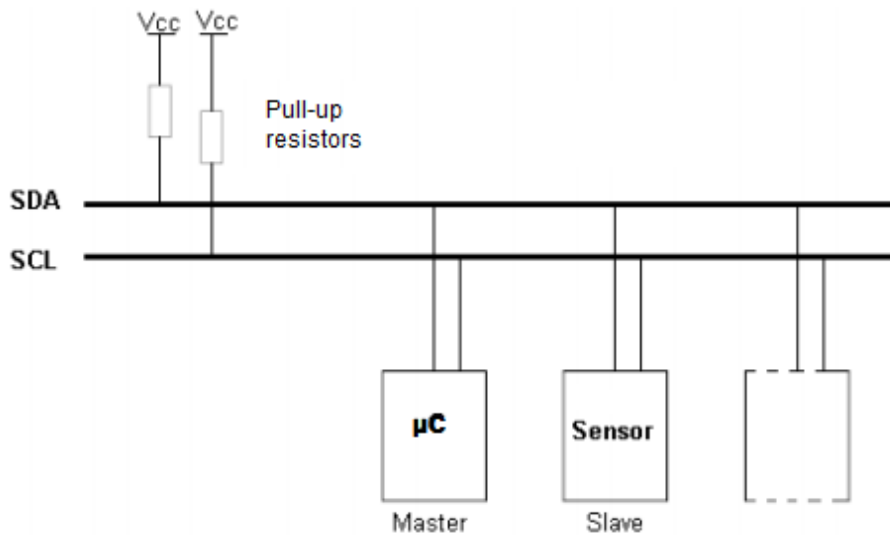


Figure 2 Example of an I<sup>2</sup>C bus configuration

1.2 I<sup>2</sup>C communication protocol

The word “master” refers to a device that initiates and terminates a transfer and also provides the clock signals on line SCL. Master devices operate as transmitters or receivers.

At the start of each transfer, a slave is addressed by its own unique address. A transfer consists of a start condition, the data bits, an acknowledge bit and possibly a stop condition. This concept is shown in Figure 3.

A start condition is defined by a falling edge at SDA while the SCL is high. A stop condition is defined by a rising edge at SDA while the SCL line is high. When transmitting data, no changes at the SDA line while the clock is high are permitted, otherwise this will result in a stop or a start condition! In order to avoid this, the master should change the data at SDA only when the SCL line is low.

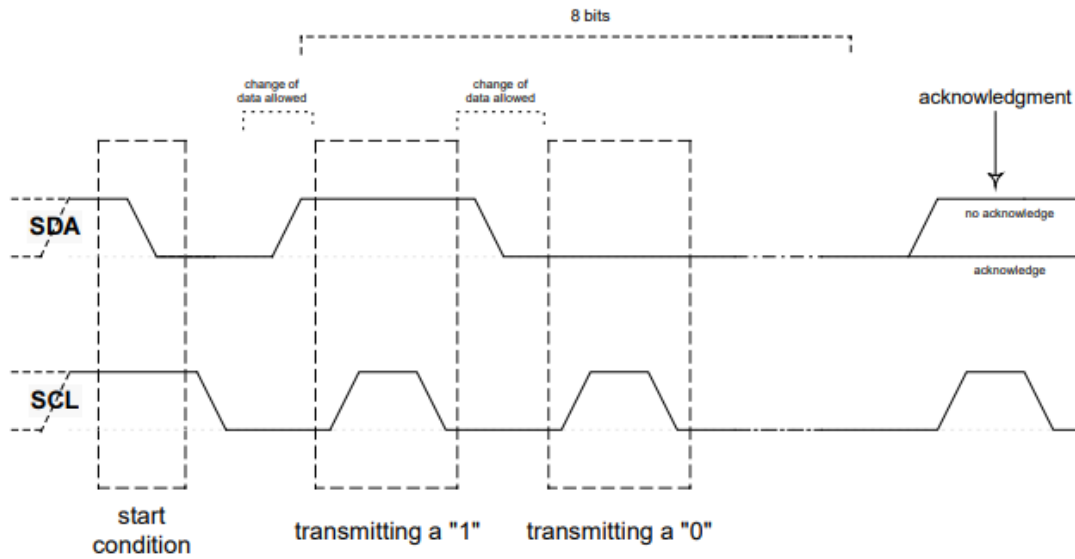
After the transmission of the 8 data bits, the master sets the SDA line to high and the slave acknowledges the transfer by pulling the SDA line to ground. This indicates a successful transfer.

Master-read mode is not exactly the same. The master still provides the clock but the slave now submits the data (requested by the master) at the SDA line. At the end of the transmission, the master does not acknowledge (SDA is set and remains high).

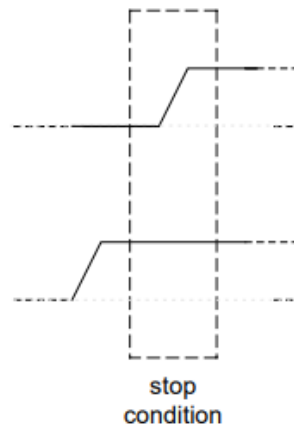
For further information the interested reader is referred to the original I<sup>2</sup>C-bus specification (UM10204 Rev 6, NXP Semiconductors).

An example for reading the status of the XENSIV™ PAS CO2 sensor is shown in Figure 4. An introduction to the basic registers will be given in the latter half of this application note. A more detailed description of all available registers and functions of the XENSIV™ PAS CO2 sensor can be found in a separate application note.

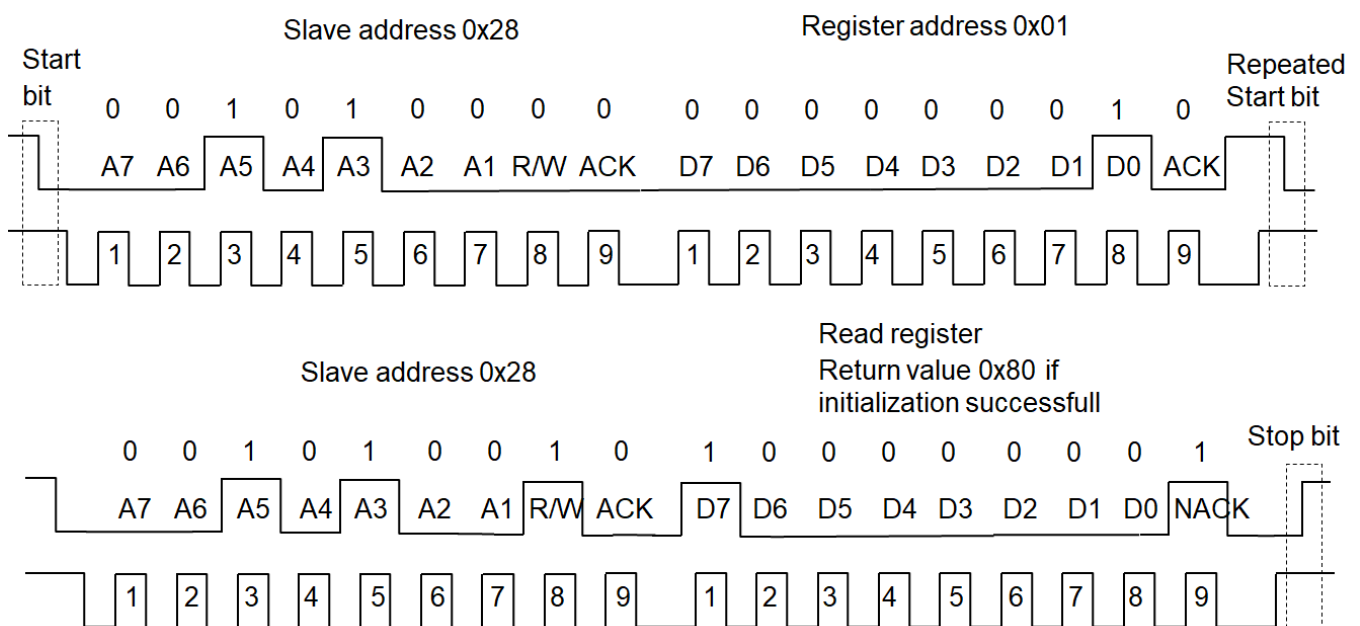
Introduction



**Acknowledgment:**  
 SDA line is pulled low by the slave during the SCL pulse while in slave write mode.  
 In slave read mode, SDA line remains HIGH during the SCL pulse (no acknowledge by the master).



**Figure 3 Example of an I<sup>2</sup>C transaction**



**Figure 4 I<sup>2</sup>C transaction for reading the status of the sensor**

I2C application circuit

## 2 I<sup>2</sup>C application circuit

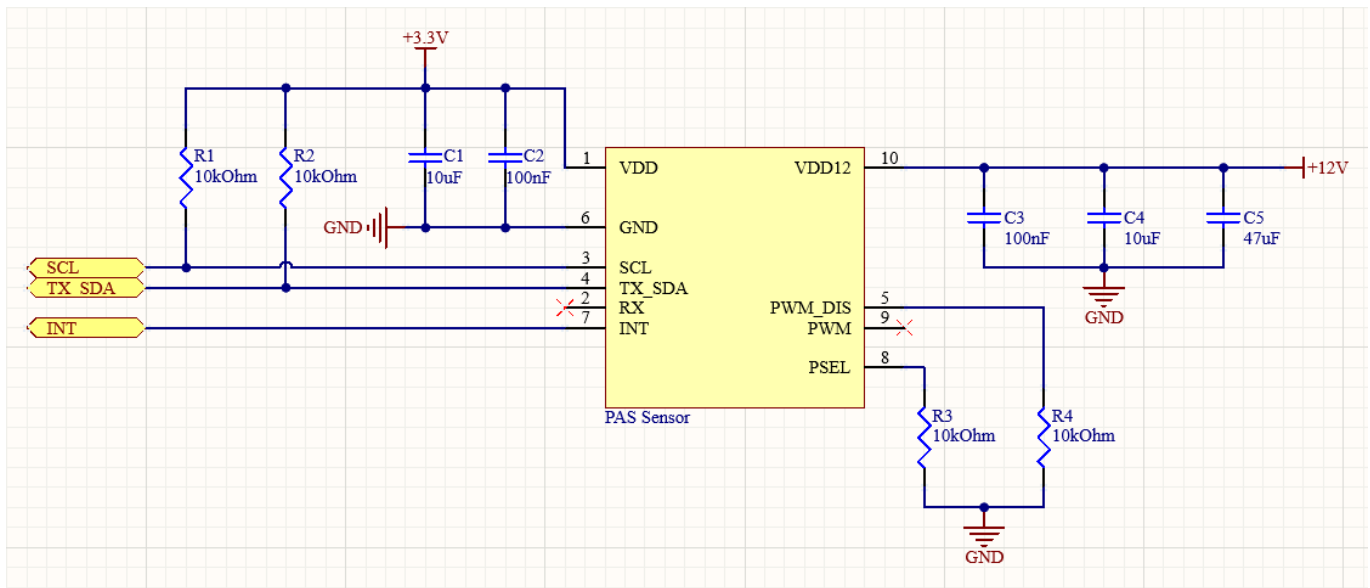


Figure 5 Application circuit example

Table 1 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	I <sup>2</sup> C clock pin (3.3 V domain)
4	TX_SDA	Input/Output	UART transmitter pin/I <sup>2</sup> C data pin (3.3 V domain)
5	PWM_DIS	Input	PWM disable input pin (3.3 V domain) <sup>2)</sup>
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) <sup>1)</sup>
9	PWM	Output	PWM output pin (3.3 V domain)
10	VDD12	Power supply (12 V)	12 V power supply for the IR emitter

<sup>1)</sup> High level selects UART and low level selects I<sup>2</sup>C. It is recommended to the user to hard wire the pin to VDD or GND, depending on the wanted interface.

<sup>2)</sup> If a communication interface is used by the application, it is recommended to hard-wire pin PWM\_DIS to GND. Note: In this case the device will start in continuous mode and not idle mode which needs to be considered when changing the measurement period.

## Initialization sequence

### 3 Initialization sequence

In order not to damage the sensor or other components, a certain initialization sequence must be followed:

- Connect all the necessary pins as seen in Figure 5 of the XENSIV™ PAS CO2 Sensor2Go Kit to the microcontroller (power off).
- Power on the microcontroller (3.3 V supply for the communication).
- Supply 12 V externally for the heater.
- Run script/code (configure settings, start measurement, etc.).



**Figure 6 Initialization sequence**

When powering off the sensor, the sequence must be reversed.

Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

## 4 Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

PSoC® 6 bridges the gap between expensive, power-hungry application processors and low-performance microcontrollers. The ultra-low-power PSoC 6 microcontroller architecture offers the processing performance needed by IoT devices, eliminating the tradeoffs between power and performance. The PSoC 6 microcontroller contains a dual-CPU architecture, with both CPUs on a single chip. It has an ARM® Cortex-M4 for high-performance tasks, and an ARM® Cortex-M0+ for low-power tasks. With security built in, your IoT system is protected.

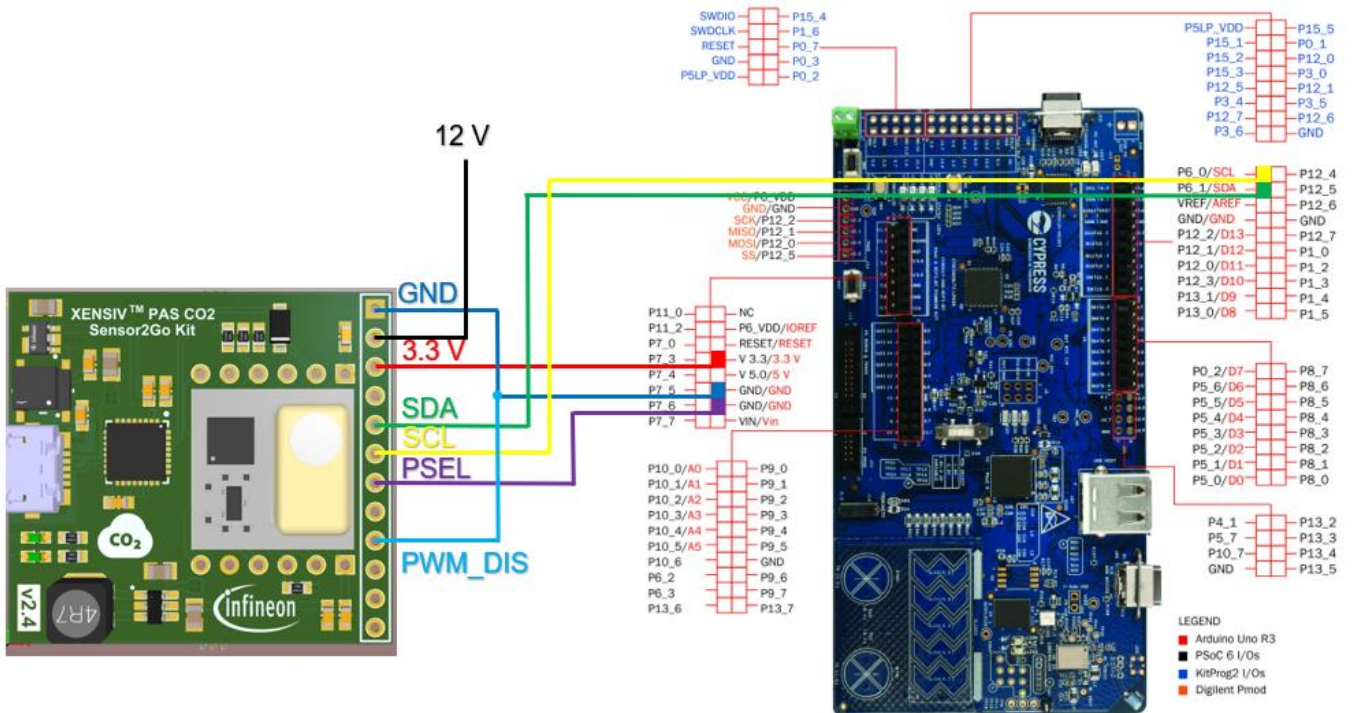


Figure 7 XENSIV™ PAS CO2 Sensor2Go Kit I²C interface connection to the PSoC® 6 WiFi-BT Pioneer Kit

Table 2 Pin connections

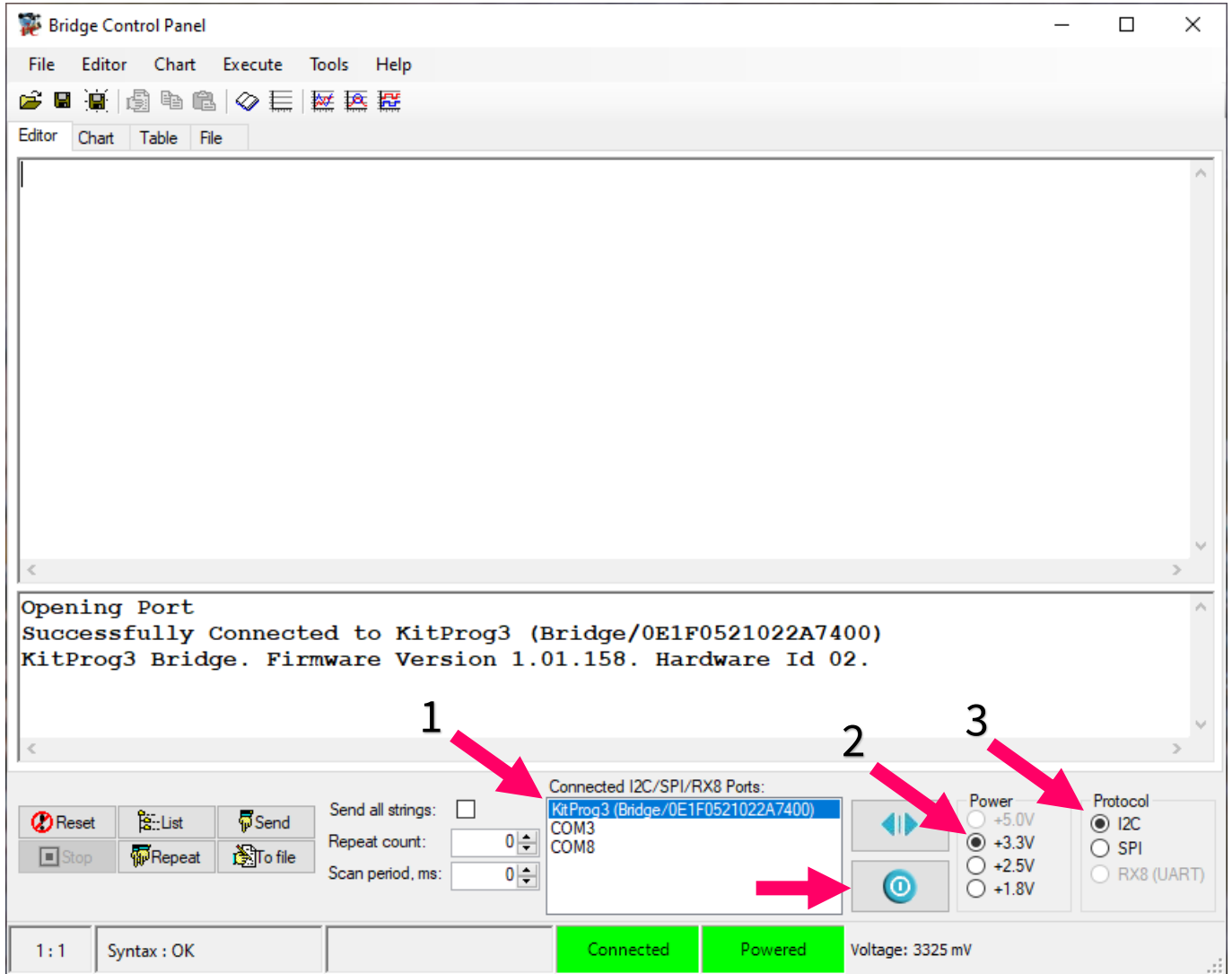
Position	Symbol	Connection to the PSoC® 6 WiFi-BT Pioneer Kit
1	GND	Ground
2	VDD12	12 V power supply (externally) <sup>3)</sup>
3	VDD3.3	3.3 V digital power supply <sup>3)</sup>
4	RX	Not connected
5	TX/SDA	I²C data pin (3.3 V domain)
6	SCL	I²C clock pin (3.3 V domain)
7	PSEL	Ground
8	INT	Not connected (in this case)
9	PWM_DIS	Ground
10	PWM	Not connected (in this case)
11	SWD	Not connected (in this case)
12	SWCLK	Not connected (in this case)

<sup>3)</sup> Power supply tolerance ±10 percent

Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

### 4.1 Bridge Control Panel

Bridge Control Panel is a simple debugging tool that comes with PSoC Programmer. It is used to communicate with target devices over I<sup>2</sup>C/UART/SPI serial communication interfaces.



**Figure 8 Initialization of the device**

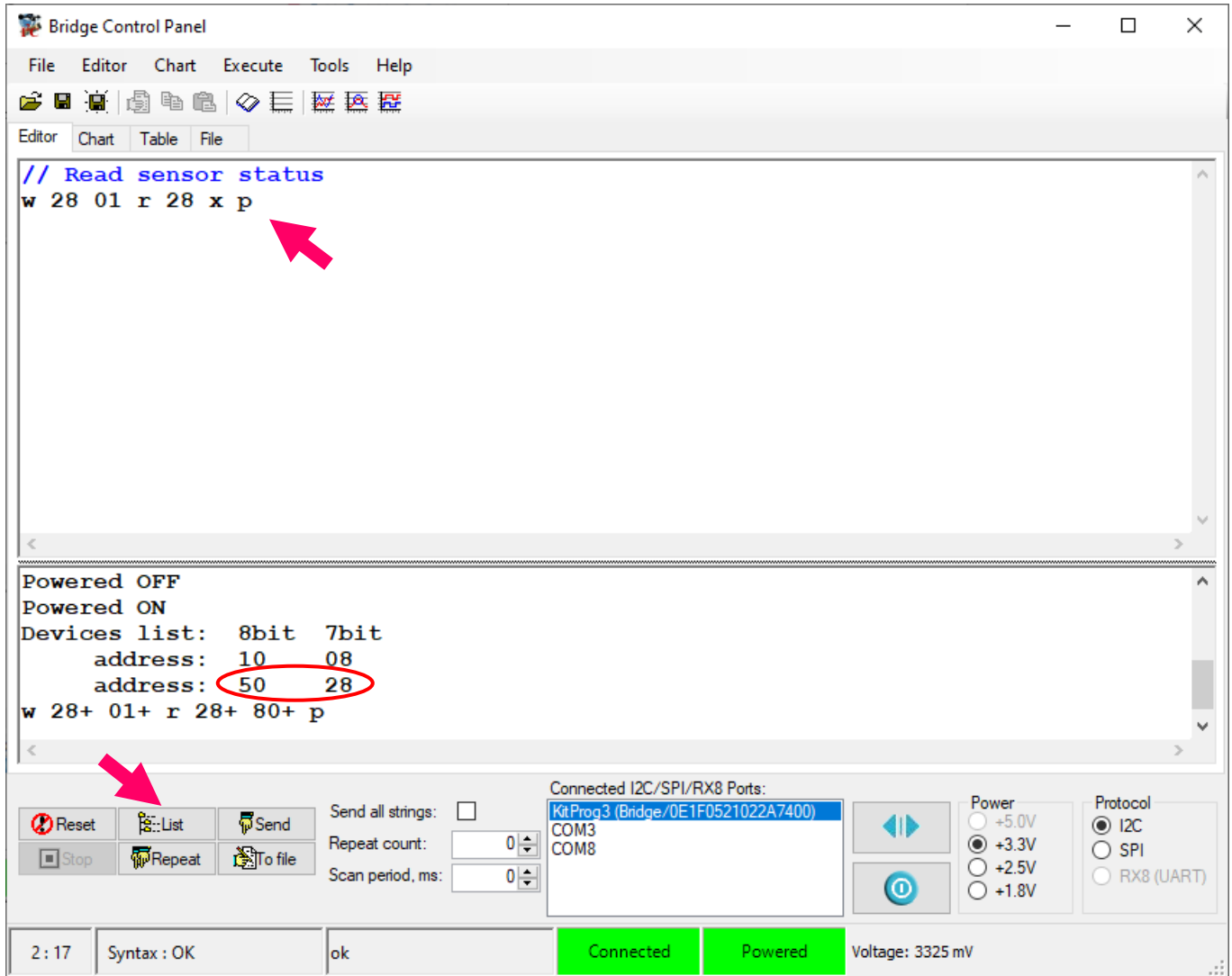
After wiring up the XENSIV™ PAS CO2 Sensor2Go Kit and the PSoC® 6 WiFi-BT Pioneer Kit as shown in Figure 7, the following steps cover initialization and communication with the sensor:

1. Select COM port accordingly (KitProg3).
2. Select 3.3 V in the Power menu.
3. Select I<sup>2</sup>C protocol.

With the “Toggle power” button you can switch the 3.3 V power supply on and off. Only after the 3.3 V is supplied can the 12 V be supplied externally safely. Make sure when powering off the 3.3 V supply with the “Toggle power” button to first power off the external 12 V supply and then the 3.3 V supply to avoid damaging the device.



Quick start with the PSoC® 6 WiFi-BT Pioneer Kit



**Figure 9 Testing I<sup>2</sup>C communication**

After powering up both supplies in the respective order it is recommended to first check the basic I<sup>2</sup>C communication by pressing the “List” button. The bridge control panel will now list all the available I<sup>2</sup>C slave devices available on the bus. Check the terminal for slave address 0x28 (0x50 8 bit), which is the address of XENSIV™ PAS CO2.

Known issues and mistakes are:

- No slave devices listed when checking for slave response with the “List” button (error message in the terminal: “No device found”)
  - Make sure all wires are properly connected and not loose.
- Bridge control panel is crashing
  - Make sure there is no shortage as a result of wrongly connected supply pins.
- Device doesn’t receive commands (e.g. w 28- 01- r 28- FF- p)
  - Make sure I<sup>2</sup>C wires (SDA/SCL) are properly connected and not loose.

The commands are written in the “Editor” window and are sent and executed by pressing “Enter”.

## Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

### 4.2 Basic code for starting measurement

In this section the basic operation for starting a measurement with the XENSIV™ PAS CO2 will be described.

After initializing the device according to the sequence in chapter 3 and section 4.1, the first thing recommended to do is check if the initialization went correctly without any errors. This can be done by checking the **sensor status register**. After that, and before starting a measurement, it is recommended to check and set the pressure compensation. This can be done with the two **pressure compensation registers**. The default pressure set in these registers is 1015 hPa. Now everything is set to start a measurement. There are two different measurement modes available: single-shot measurement and continuous measurement. The mode can be configured in the **measurement mode configuration register**. When using the continuous measurement mode, the measurement period can be defined in the two **measurement period configuration registers** beforehand. Either way, after configuring the measurement mode, a measurement sequence is triggered. By reading the **measurement status register** it is possible to check if the measurement sequence is completed and thus if a new CO<sub>2</sub> concentration value is available in the **CO<sub>2</sub> concentration result register**.

#### Sensor status register (SENS\_STS, address: 0x01)

```
1 w 28 01 r 28 x p
```

If the sensor is initialized correctly the return value is 0xC0.

#### Pressure compensation registers (PRES\_REF\_H and PRES\_REF\_L, address: 0x0B and 0x0C)

```
1 w 28 0B r 28 x p
2 w 28 0C r 28 x p
3 w 28 0B 03 p
4 w 28 0C F5 p
```

Registers PRES\_REF\_L and PRES\_REF\_H are used to store the ambient atmospheric pressure. The concatenation of PRES\_REF\_H (MSB) and PRES\_REF\_L (LSB) defines the pressure value that shall be considered by the device. The concatenated pressure value is coded as an unsigned short integer (1 bit = 1 hPa). In this example the pressure is set to 1013 hPa. For correct operation, the user shall ensure that the pressure value programmed is within the specified pressure operating range of the device. This valid range of operation is 750 hPa to 1150 hPa.

#### Measurement period configuration registers (MEAS\_RATE\_H and MEAS\_RATE\_L, address: 0x02 and 0x03)

```
1 w 28 02 00 p
2 w 28 03 0A p
```

Registers MEAS\_RATE\_H and MEAS\_RATE\_L define the measurement period used in continuous mode. The concatenation of MEAS\_RATE\_H (MSB) and MEAS\_RATE\_L (LSB) defines the period. The concatenated value is coded as a two's complement signed short integer (1 bit = 1 s). In this example the measuring rate is set to 10 s. The configurable range is from 0005<sub>H</sub> (5 s) to 0FFF<sub>H</sub> (4095 s). When writing to MEAS\_RATE\_H and MEAS\_RATE\_L, the new value is not immediately considered by the device. It is internally latched at the next transition from idle mode to continuous mode.

#### Measurement mode configuration register (MEAS\_CFG, address: 0x04)

```
1 w 28 04 02 p
2 w 28 04 01 p
```

This register defines the operation settings of the device. With code in line 1 a single-shot measurement is triggered and with code in line 2 the continuous measurement mode is configured. Note that after one

## Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

measurement sequence the emitter needs at least 10 s to cool down. Measurement rate values in continuous mode below 5 s are treated as being equal to 5 s. For single-shot measurement make sure to delay the following measurement sequence by at least 10 s for accurate readings.

### Measurement status register (MEAS\_STS, address: 0x07)

1	w	28	07	r	28	x	p
---	---	----	----	---	----	---	---

This register displays status information of the sensor. Once a measurement sequence is completed and the new CO<sub>2</sub> concentration value is available the return value of this register is 0x10.

### CO<sub>2</sub> concentration result register (CO2PPM\_H and CO2PPM\_L, address: 0x05 and 0x06)

1	w	28	05	r	28	x	p
2	w	28	06	r	28	x	p

Registers CO2PPM\_H and CO2PPM\_L are used to store the result of the last CO<sub>2</sub> concentration measurement. The concatenation of CO2PPM\_H (MSB) and CO2PPM\_L (LSB) defines the CO<sub>2</sub> concentration value. The concatenated CO<sub>2</sub> concentration value is coded as a two's complement signed short integer (1 bit = 1 ppm). This field is updated at the end of each measurement sequence. When reading the CO<sub>2</sub> concentration value, the user shall first read registers CO2PPM\_H and then CO2PPM\_L.

### Summary

1	w	28	01	r	28	x	p	// Read sensor status
1	w	28	0B	r	28	x	p	// Read pressure (MSB)
2	w	28	0C	r	28	x	p	// Read pressure (LSB)
3	w	28	0B	03	p			// Set pressure (MSB)
4	w	28	0C	F5	p			// Set pressure (LSB)
5	w	28	02	00	p			// Set measurement period (MSB)
6	w	28	03	0A	p			// Set measurement period (LSB)
7	w	28	04	02	p			// Trigger continuous measurement
8	(w	28	04	01	p)			// Trigger single shot measurement
9	w	28	07	r	28	x	p	// Read measurement status
10	w	28	05	r	28	x	p	// Read CO2 concentration (MSB)
11	w	28	06	r	28	x	p	// Read CO2 concentration (LSB)

**Attention:** Full detailed register map has been covered in a separate application note (see product page)

**Note:** The pressure compensation register should be updated regularly to compensate for barometric pressure variation.

## 4.3 Additional functionality

In this section additional possible functions of the XENSIV™ PAS CO2 will be introduced. The full description of the available functionality will be covered in a separate application note.

### Interrupt pin configuration register (INT\_CFG, address: 0x08)

1	w	28	08	18	p
---	---	----	----	----	---

This register defines the configuration of pin INT. Pin INT is a multi-purpose output pin that can be configured to perform several functions. The electrical configuration can be either set as push pull and low active or push pull and high active.

## Quick start with the PSoC® 6 WiFi-BT Pioneer Kit

The following functions can be configured. Alarm threshold violation notification pin, Data Ready notification pin, sensor busy notification pin and early measurement start notification pin. In this example the interrupt is configured as push pull and high active early measurement start notification pin. The indication if an interrupt did occur can be read in the measurement status register as well as the clearing of the sticky bits.

### Alarm threshold register (ALARM\_TH\_H and ALARM\_TH\_L, address: 0x09 and 0x0A)

1	w	28	09	03	p
2	w	28	0A	E8	p

Registers ALARM\_TH\_H and ALARM\_TH\_L define the value used as a threshold for the alarm violation. The concatenation of ALARM\_TH\_H (MSB) and ALARM\_TH\_L (LSB) define the threshold value that shall be considered by the device. The concatenated value is coded as a two's complement signed short integer (1 bit = 1 ppm). In this example the alarm threshold is set to 1000 ppm. The indication if a threshold violation did occur can be read in the measurement status register.

### Automatic baseline offset compensation reference (CALIB\_REF\_H and CALIB\_REF\_L, address: 0x0D and 0x0E)

1	w	28	0D	01	p
2	w	28	0E	90	p

Registers CALIB\_REF\_H and CALIB\_REF\_L define the reference value used for the automatic baseline offset compensation or forced compensation. The concatenation of CALIB\_REF\_H (MSB) and CALIB\_REF\_L (LSB) define the reference value. The concatenated offset value is coded as a 2's complement signed short integer (1 bit = 1 ppm). In this example the automatic baseline offset compensation reference is set to 400 ppm. For correct operation, the user shall ensure that the compensation value programmed is within the specified operating range of the device. This valid range of operation is 350 ppm to 900 ppm. The automatic baseline offset compensation or forced compensation can be enabled/disabled in the measurement mode configuration register and be reset in the soft reset register. More details regarding the automatic baseline offset compensation and forced compensation are covered in a separate application note (see product page).

### Scratch pad register (SCRATCH\_PAD, address: 0x0F)

1	w	28	0F	01	p
2	w	28	0F	r 28	x p

This register provides a readable and writable address space for data integrity test during runtime. This register is not associated with a specific hardware functionality.

### Soft reset register (SENS\_RST, address: 0x10)

1	w	28	10	A3	p
2	w	28	10	DF	p
3	w	28	10	FE	p
4	w	28	10	BC	p
5	w	28	10	FC	p

This register is used to trigger a soft reset. It also covers the settings of the filter and resetting the automatic baseline offset compensation or forced compensation. By default, the filter is enabled and can be disabled by writing 0xDF to this register. With writing 0xFE the filter is enabled again. The context of the automatic baseline offset compensation can be reset with writing 0xBC and for the reset of the forced compensation it is 0xFC.

Quick start with the Arduino Due

## 5 Quick start with the Arduino Due

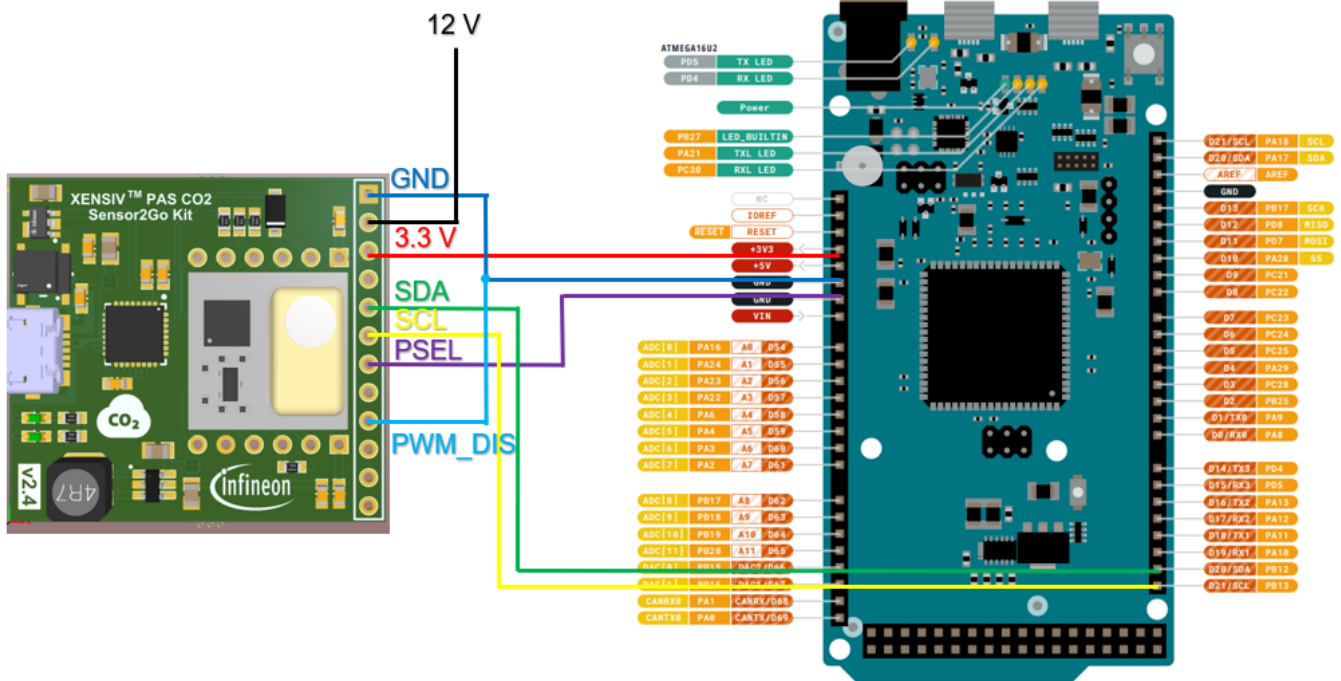


Figure 10 XENSIV™ PAS CO2 Sensor2Go Kit I²C interface connection to the Arduino Due<sup>4)</sup>

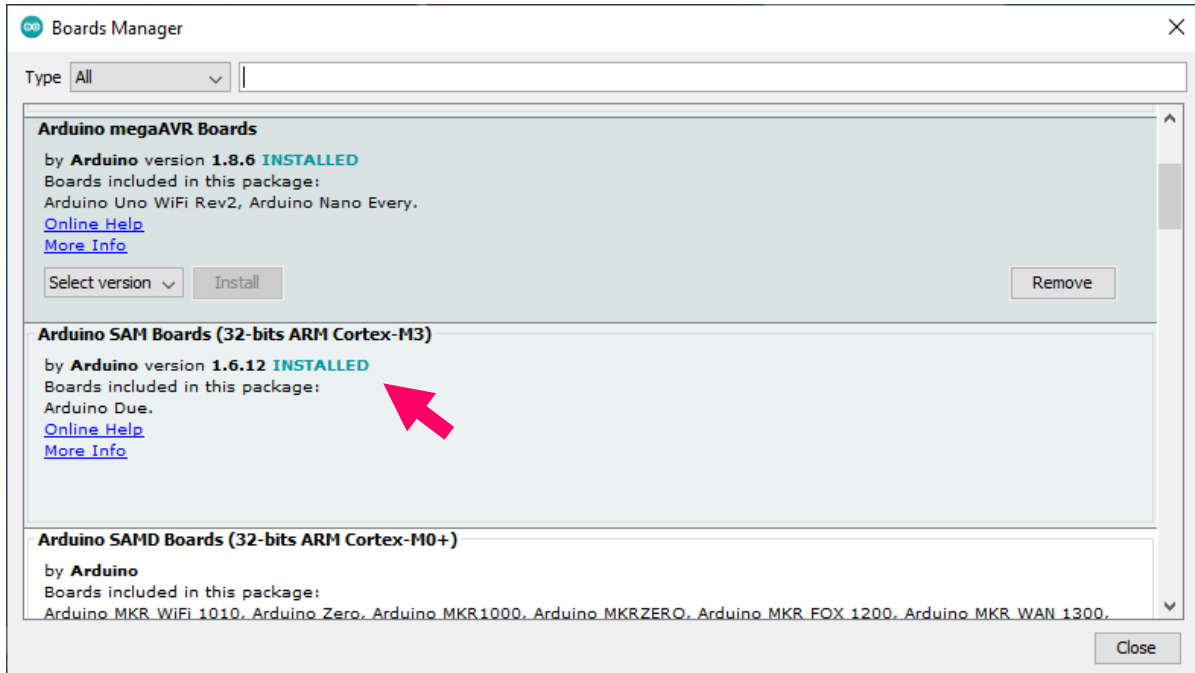
The pin connection of the device to the Arduino Due is equivalent to the connection to the PSoC® 6 WiFi-BT Pioneer Kit. When using another Arduino or other digital pins, make sure that the respective pull-up resistors are available.

<sup>4)</sup> Pinout of Arduino Due from [https://content.arduino.cc/assets/Pinout-Due\\_latest.pdf](https://content.arduino.cc/assets/Pinout-Due_latest.pdf)

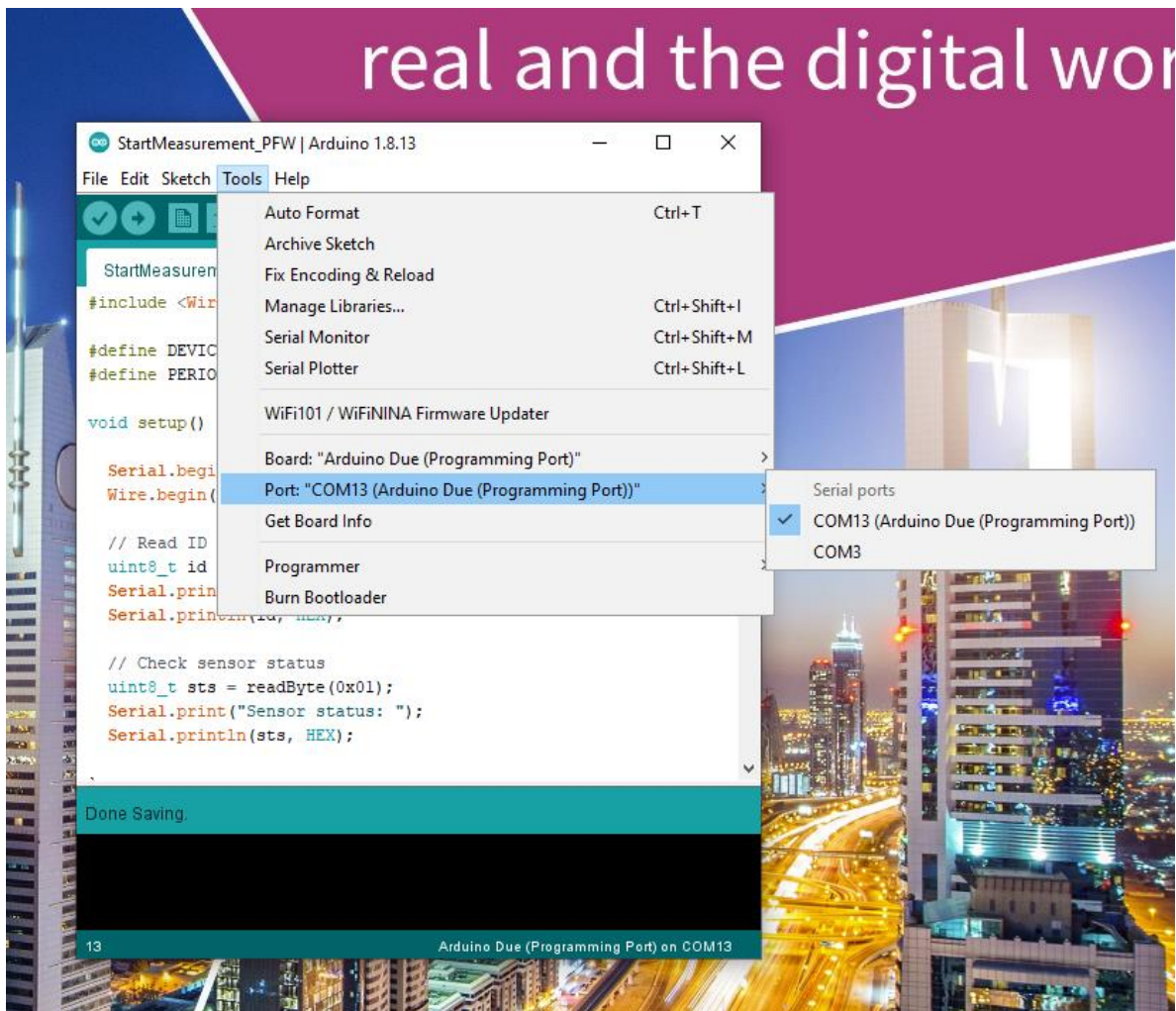
### 5.1 Arduino IDE

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 the Arduino Due



**Figure 11** Arduino IDE settings: board manager



**Figure 12** Arduino IDE settings: selecting the port

---

**Quick start with the Arduino Due****5.2 Basic Arduino code for starting measurement**

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**

```
1 uint8_t readByte(uint8_t regAddress)
2 {
3   Wire.beginTransaction(deviceAddress);
4   Wire.write(regAddress);
5   Wire.endTransmission(false);
6   //request 1 byte from slave
7   if (Wire.requestFrom(deviceAddress, 1U, 1U) > 0)
8   {
9     return Wire.read();
10  }
11  else
12  {
13    return 0x0;
14  }
15 }
```

**writeByte**

```
1 bool writeByte(uint8_t regAddress, uint8_t data)
2 {
3
4   Wire.beginTransaction(deviceAddress);
5   Wire.write(regAddress);
6   Wire.write(data);
7   if (Wire.endTransmission() != 0)
8   {
9     return false;
10  }
11  else
12  {
13    return true;
14  }
15 }
```

---

**Quick start with the Arduino Due****Code for starting a single-shot measurement**

```
1 #include <Wire.h>
2
3 #define deviceAddress 0x28
4 #define PERIOD 10000
5
6 void setup() {
7
8     Serial.begin(115200);
9     Wire.begin();
10
11     // Check sensor status
12     uint8_t sts = readByte(0x01);
13     Serial.print("Sensor status: ");
14     Serial.println(sts, HEX);
15
16     // Idle mode
17     writeByte(0x04, 0x00);
18     delay(400);
19
20     // Set pressure
21     writeByte(0x0B, 0x03);
22     writeByte(0x0C, 0xF5);
23 }
24
25 void loop()
26 {
27     // Trigger single measurement
28     writeByte(0x04, 0x01);
29     delay(400);
30
31     // Get PPM value
32     uint8_t value1 = readByte(0x05);
33     delay(5);
34     uint8_t value2 = readByte(0x06);
35     delay(5);
36
37     // Calculate ppm value
38     int16_t result = value1 << 8 | value2;
39     Serial.print("CO2: ");
40     Serial.print(result);
41     Serial.println(" ppm");
42
43     delay(PERIOD);
44
45 }
```



**Quick start with the Arduino Due****Code for starting a continuous mode measurement**

```
1 #include <Wire.h>
2 #define deviceAddress 0x28
3
4 void setup() {
5
6     Serial.begin(115200);
7     Wire.begin();
8
9     // Read ID
10    uint8_t id = readByte(0x00);
11    Serial.print("ID: ");
12    Serial.println(id, HEX);
13
14    // Check sensor status
15    uint8_t sts = readByte(0x01);
16    Serial.print("Sensor status: ");
17    Serial.println(sts, HEX);
18
19    // Idle mode
20    writeByte(0x04, 0x00);
21    delay(400);
22
23    // Set measurement rate to 10 s
24    writeByte(0x02, 0x00);
25    writeByte(0x03, 0x0A);
26
27    // Configure continous mode
28    writeByte(0x04, 0x02);
29 }
30
31 void loop() {
32
33     // Poll measurement status
34     uint8_t meas_sts = readByte(0x07);
35     delay(100);
36
37     if (meas_sts == 0x10) {
38
39         // Get PPM value
40         uint8_t value1 = readByte(0x05);
41         delay(5);
42         uint8_t value2 = readByte(0x06);
43         delay(5);
44
45         // Calculate ppm value
46         int16_t result = value1 << 8 | value2;
47         Serial.print("CO2: ");
48         Serial.print(result);
49         Serial.println(" ppm");
50     }
51     delay(1000);
52 }
```

UART interface

## 6 UART interface

When UART is selected as serial communication interface with setting pin PSEL to high, the device acts as an UART slave. As a result, it is recommended that the master uses a time out mechanism. The device operates via UART for point to point communication and therefore bus operation is not supported.

The basic format of a valid UART frame is: 1 start bit, 8 data bits, no parity bit and 1 stop bit. The baud rate is 9.6kbps. The master combines several UART frame into a message (read or write). The combination of master request and salve answer defines a transaction.

If the device detects a valid incoming message, it shall respond it with an acknowledge frame. Otherwise, it will issue a NAK notification.

It shall be noted that when a measurement sequence is initiated, the device does not respond to any incoming frame for the duration of the measurement sequence (either ACK or NAK). A message sent during a measurement sequence shall be therefore resent by the master once the measurement sequence is completed.

The device does not support bulk read and write operations. Only singly data bytes can be read or written within one transaction.

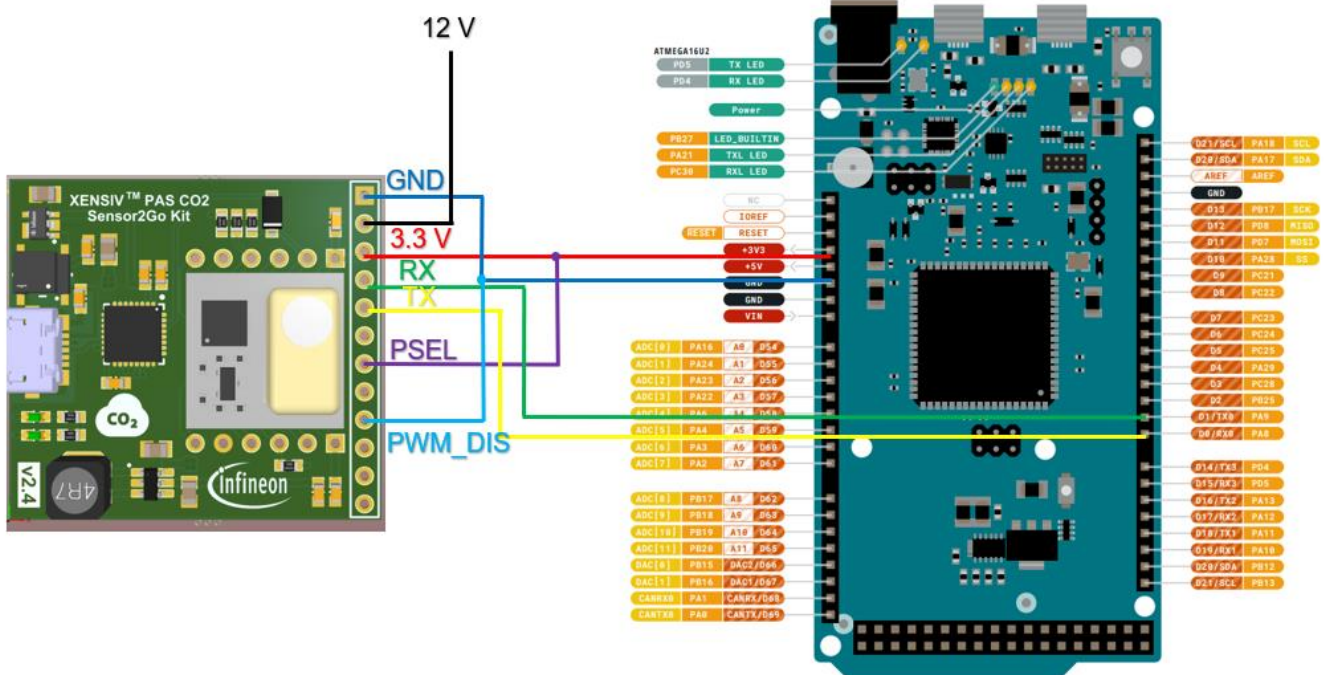


Figure 13 XENSIV™ PAS CO2 Sensor2Go Kit UART interface connection to the Arduino Due<sup>4)</sup>

UART interface

### 6.1 Write transactions

A Write transaction is initiated by the message made by the frame sequence below:

Frame	Description	Frame Payload	Comments
1	Initiate request	0x57 (ASCII code for “W”) or 0x77 (ASCII code for “w”)	
2	Delimiter	0x2C (ASCII code for “,”)	
3	Address 1 (4 most significant bits)	ASCII code of the hex value of the 4 most significant bits of the register address.	E.g.: in order to write register at address 03 <sub>H</sub> , the payload should be 0x30 (ASCII code for “0”).
4	Address 2 (4 least significant bits)	ASCII code of the hex value of the 4 least significant bits of the register address.	E.g.: in order to write register at address 03 <sub>H</sub> , the payload should be 0x33 (ASCII code for “3”)
5	Delimiter	0x2C (ASCII code for “,”)	
6	Data 1 (4 most significant bits)	ASCII code of the hex value of the 4 most significant bits of the written data.	E.g.: in order to write data F3 <sub>H</sub> , the payload should be 0x46 (ASCII code for “F”) or 0x66 (“f”).
7	Data 2 (4 least significant bits)	ASCII code of the hex value of the 4 least significant bits of the written data.	E.g.: in order to write data F3 <sub>H</sub> , the payload should be 0x33 (ASCII code for “3”).
8	End of message	0x0A (ASCII code for line feed “\n”)	

At the end of the reception the incoming message, the device answers with an answer message.

If the write operation is valid, the device (slave) answers it with the following message:

Frame	Description	Value	Comments
1	ACK	0x06 (ASCII code for “ACK”)	
2	End of message	0x0A (ASCII code for line feed “\n”)	

UART interface

If the write operation is not valid, the device answers with the following message:

Frame	Description	Value	Comments
1	NAK	0x15 (ASCII code for “NAK”)	
2	End of message	0x0A (ASCII code for line feed “\n”)	

## 6.2 Read transactions

A Read transaction is initiated by the message made by the frame sequence below:

Frame	Description	Frame Payload	Comments
1	Initiate request	0x52 (ASCII code for “R”) or 0x72 (ASCII code for “r”)	
2	Delimiter	0x2C (ASCII code for “,”)	
3	Address 1 (4 most significant bits)	ASCII code of the hex value of the 4 most significant bits of the register address.	E.g.: in order to write register at address 01 <sub>H</sub> , the payload should be 0x30 (ASCII code for “0”).
4	Address 2 (4 least significant bits)	ASCII code of the hex value of the 4 least significant bits of the register address.	E.g.: in order to write register at address 01 <sub>H</sub> , the payload should be 0x31 (ASCII code for “1”).
8	End of message	0x0A (ASCII code for line feed “\n”)	

At the end of the reception the incoming message, the device answers with an answer message.

If the read operation is valid, the device (slave) answers it with the following message:

Frame	Description	Value	Comments
1	Data 1 (4 most significant bits)	ASCII code of the hex value of the 4 most significant bits of the written data.	

**UART interface**

2	Data 2 (4 least significant bits)	ASCII code of the hex value of the 4 least significant bits of the written data.	
3	End of message	0x0A (ASCII code for line feed “\n”)	

If the read operation is not valid, the device answers with the following message:

Frame	Description	Value	Comments
1	NAK	0x15 (ASCII code for “NAK”)	
2	End of message	0x0A (ASCII code for line feed “\n”)	

PWM interface

## 7 PWM interface

In case no communication interface is available PWM\_DIS pin can be used to control the device. If a communication interface is used by the application, it is recommended to hard-wire pin PWM\_DIS to GND. PWM\_DIS is first asserted after the power on boot sequence (not after soft reset) and the level of the pin is checked. If a low level is detected, an internal interrupt routine configures the device into continuous mode and a measurement sequence is started. At the end of each measurement sequence, the level of pin PWM\_DIS is polled. If it is high, then the device is configured back to idle mode and output pin PWM is disabled.

Pin PWM offers the possibility to read out the CO2 concentration by delivering a PWM signal whose timing information contain the CO2 concentration value. At the end of each measurement sequence, the device updates the PWM timing with the measured CO2 concentration. To enable the PWM output two conditions must be met:

- PWM output needs to be enabled by software in the measurement mode configuration register
- PWM\_DIS pin needs to be set to GND

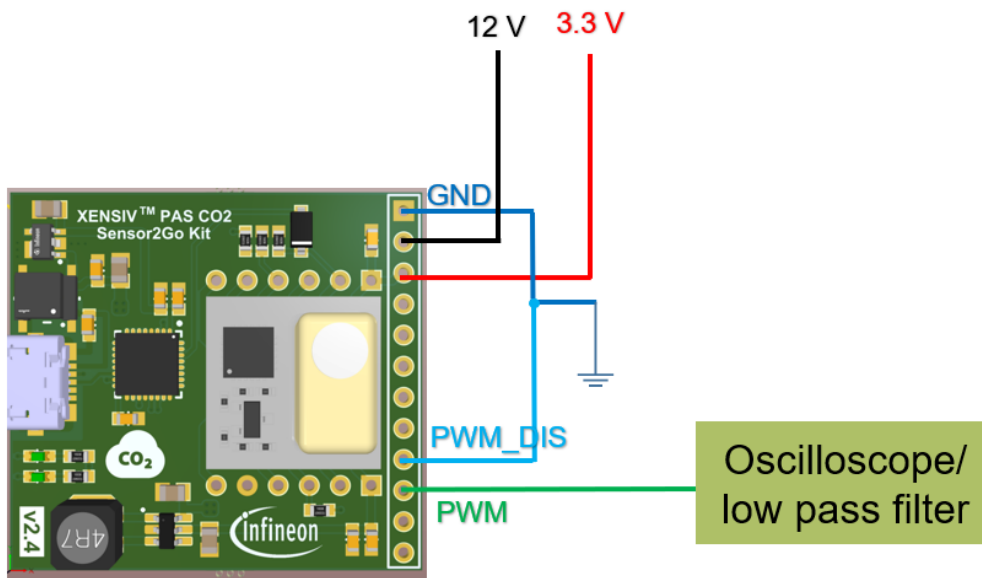


Figure 14 XENSIV™ PAS CO2 Sensor2Go Kit PWM connection example

The main specifications of the PWM signal are summarized below. The output signal can be converted by either directly measuring the pulse-duration or alternatively by employing a low-pass filter and measuring the output voltage.

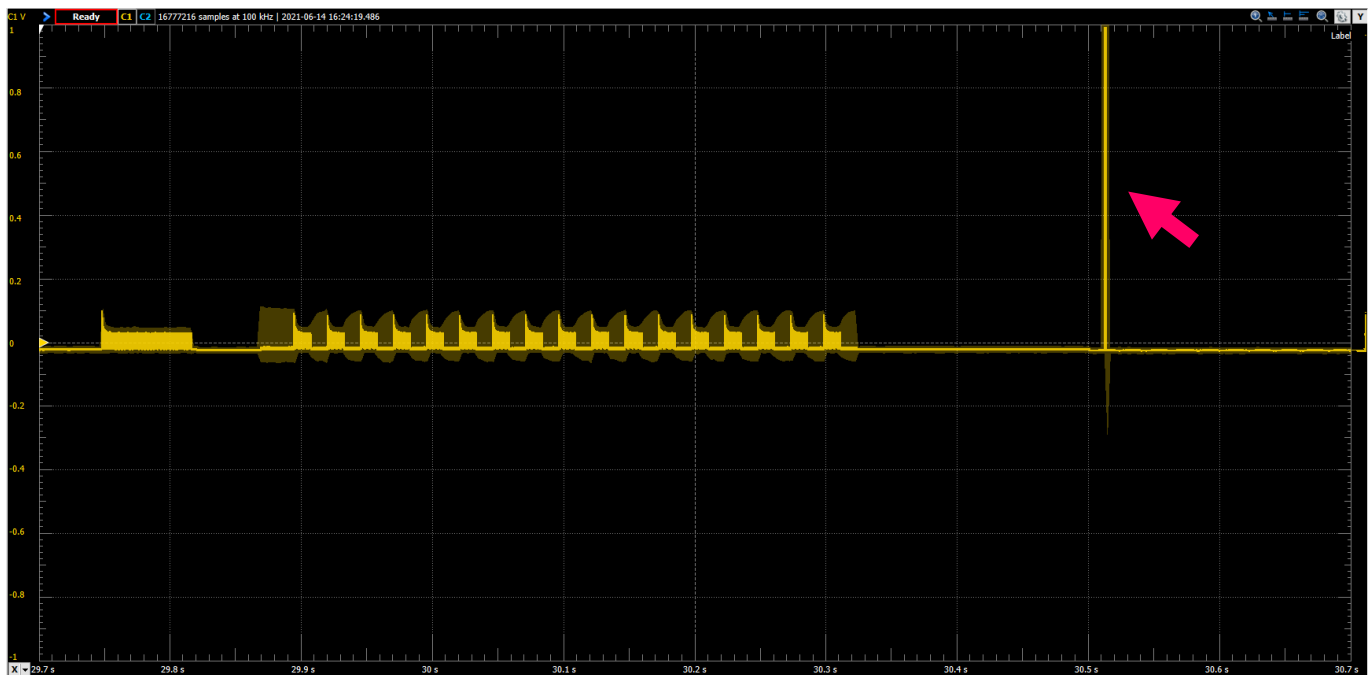
Parameter	Value
Base frequency	80Hz
Duty cycle	Linear from 0% (0ppm) to 100% (10,000ppm)
Resolution	1 ppm (1.25µs)

**PWM interface**

Typically, the PWM signal is converted to a voltage signal via a low pass filter. Since there’s an inherent trade-off between settling time, ripple and current consumption, the ideal parameterization of the low pass filter differs depending on the application. It needs to be considered that the ripple on the PWM introduces potentially an error on the CO2 concentration reading. An estimation on the introduced error is shown below.

Concentration	Duty cycle	Expected error	ON time	Target voltage	Ripple	Error introduced
400 ppm	4%	+/- 42 ppm	0.5 ms	0.132V	+/- 8mV	+/- 24 ppm
5000 ppm	50%	+/- 180 ppm	6.25ms	1.65V	+/- 50mV	+/-150 ppm

The number of PWM pulse issued at pin PWM depend on the device’s configuration which is covered by the measurement mode configuration register. In PWM single pulse mode only a single PWM pulse is generated before the device goes inactive. In PWM pulse train mode, a pulse train of 160 pulses (approx. 2sec) is issued before the device goes inactive. For calculating in single pulse mode, the high duty time of the pulse needs to be compared against the 80 Hz frequency considering the resolution of 1 ppm equals to 1.25 μs. For calculating in pulse train mode, the duty cycle needs to be calculated considering the values 0% (0ppm) to 100% (10,000ppm).



**Figure 15** PWM output in single pulse mode

PWM interface

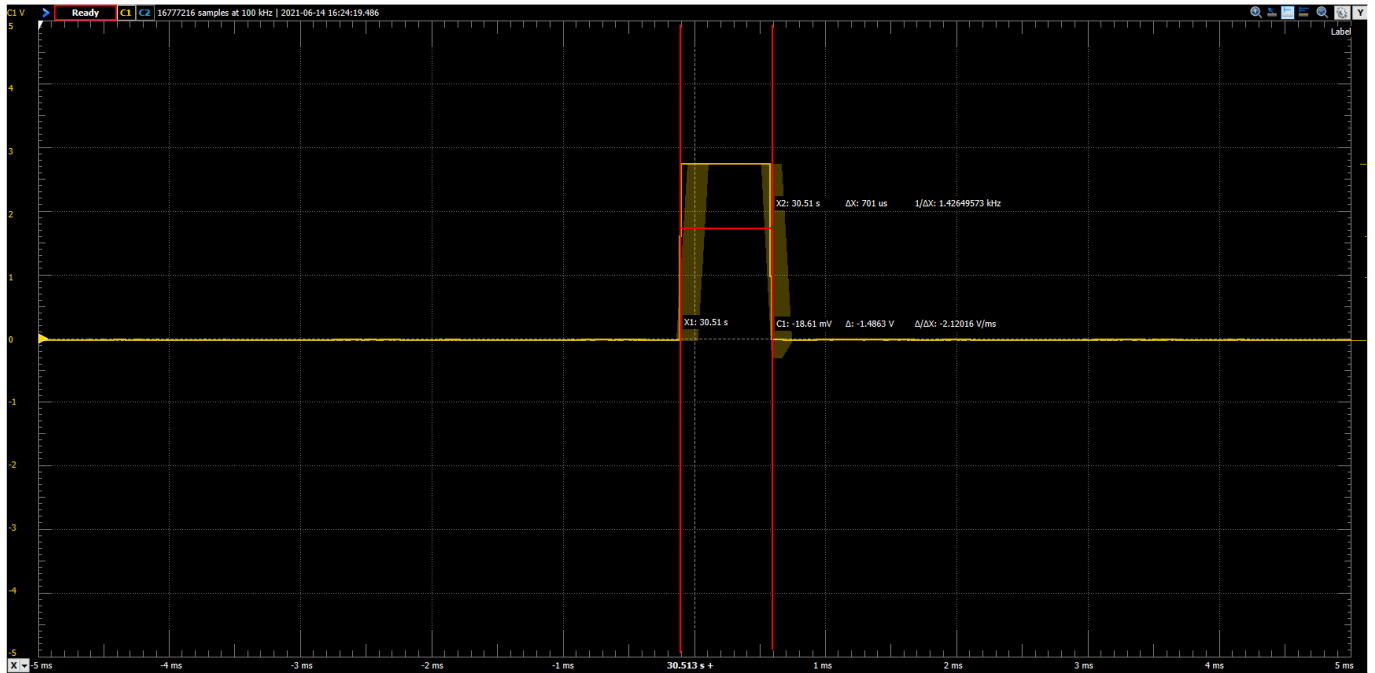


Figure 16 PWM output in single pulse mode (zoomed to pulse)

The measured high duty time of the PWM pulse in the example in Figure 14 is 701 μs. Calculating from the resolution where 1.25 μs equals to 1 ppm, the calculated CO2 concentration reading is 560 ppm.

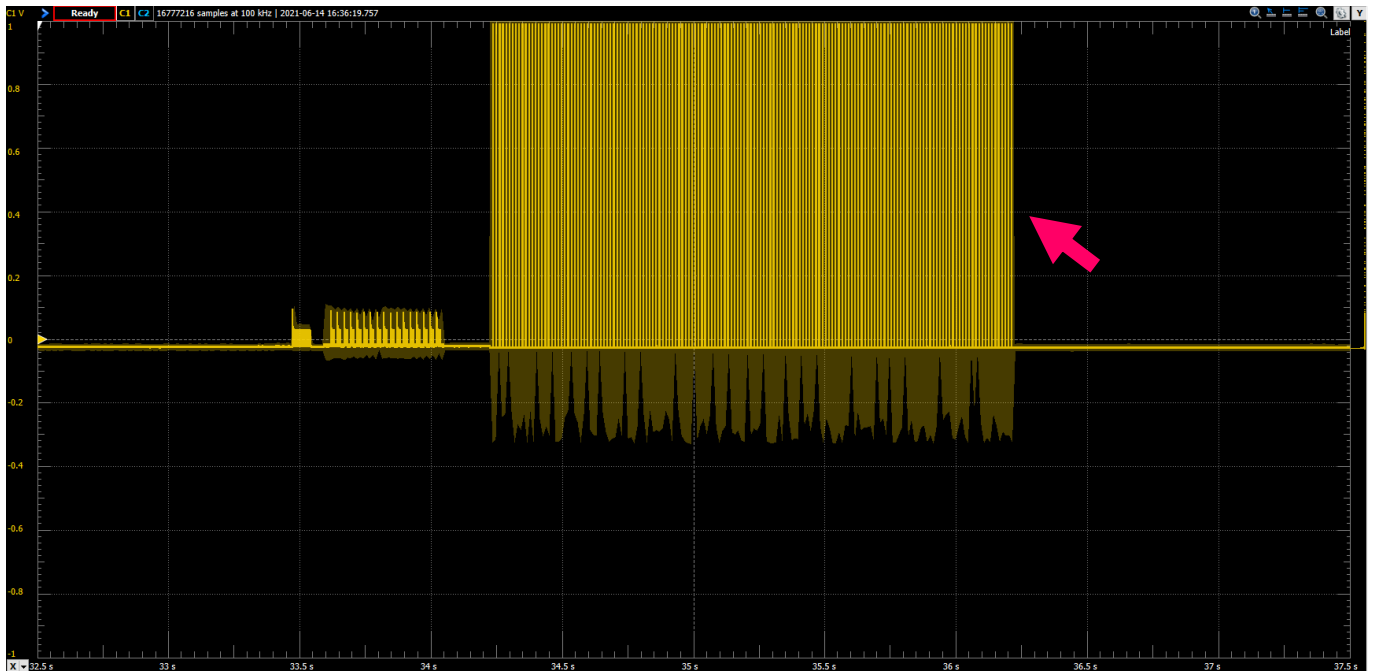


Figure 17 PWM output in train pulse mode



PWM interface



**Figure 18 PWM output in train pulse mode (zoomed to pulse)**

The measured high duty time of one PWM pulse of the PWM pulse train in the example in Figure 16 is 631.18 µs. The measured low duty time is 11.76 ms which results in a duty cycle of 5.37 % (= 537 ppm).

---

## Revision history

### Revision history

Document version	Date of release	Description of changes
V 1.0	04.11.2020	Creation
V 2.0	01.07.2021	Added description to UART, PWM and additional functionality

**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**

**© 2021 Infineon Technologies AG.**

**All Rights Reserved.**

**Do you have a question about this document?**

**Email: [erratum@infineon.com](mailto: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](http://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.