BMV080
Ultra-mini Particulate Matter Sensor

BMV080 Ultra-mini Particulate Matter Sensor – Datasheet

Document revision	1.2
Document release date	January 2025
Document number	BST-BMV080-DS000-10
Sales part number(s)	0273.017.054-1NV
Notes	Data and descriptions in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product appearance.

1 Basic description

1.1 Introduction

The BMV080 particulate matter sensor is an ultra-mini opto-electronic sensor capable of measuring particulate matter mass concentration. It delivers real-time measurements for particulate matter with diameters equal to or smaller than 2.5 µm (PM2.5). The sensor also offers mass concentration of particulate matter with diameters equal to or smaller than 1 µm (PM1) and equal to or smaller than 10 µm (PM10), enabling a comprehensive understanding of air quality across various particulate size ranges. The BMV080 measures naturally and freely moving particulate matter by using the ambient airflow close to the sensor. Its novel measurement principle, revolutionary small size, and low power consumption enable its integration into ultra-compact Internet of Things (IoT) devices such as air quality monitors, smart thermostats, smart air purifiers, and wearables.

The BMV080 has the following features:

- Ultra-compact form factor
- Precise particulate matter concentration measurement
- Innovative fanless design
  - Noiseless device operation
  - No inlets or channels needed – minimum industrial design impact on the host system
  - Maintenance free
  - Novel principle, measuring in free space
  - Enables dust-proof or waterproof integration

1.2 Particulate matter and health

Air pollution is a major environmental risk to health. Studies and research on airborne particulate matter and its impact on public health consistently show evidence of adverse health effects at specific exposure levels. While the range of health effects is broad, the respiratory and cardiovascular systems are the most affected by exposure to particulate matter. Adverse health effects have been demonstrated even for relatively small increases in particulate concentration compared to clean air conditions.

While particulate matter comes in a vast range of particle sizes, the biggest impact on human health is from particulates in the PM2.5 range, which comprises all particles smaller than 2.5 µm in diameter. Due to its small size, PM2.5 particulates can easily enter deep into the human respiratory system and provoke serious health problems.

The World Health Organization (WHO) set different air quality guidelines to assess the health effects of air pollution and thresholds for health-harmful pollution levels. As exposure to PM2.5 can cause both short-term and long-term effects, the latest WHO recommended guidelines as of 2021 (WHO global air quality guidelines, Table 0.1) provide two threshold levels related to the annual and the 24-hour mean:

- 5 µg/m^3 annual mean
- 15 µg/m^3 24-hour mean

Different countries worldwide use different air quality standards, also known as Air Quality Indices (AQI), to communicate current and future air pollution levels to the public. PM2.5 concentration is one of the pollutants considered in the calculation of each AQI. For example, the United States Environmental Protection Agency (EPA) defined a standard to correlate exposure to PM2.5 to air quality.

Table 1 shows PM2.5-specific AQI sub-indices and the relative cautionary statements.

Table 1: PM2.5 specific AQI and cautionary statements defined by the EPA

PM2.5 breakpoints (µg/m³, 24-hour average)	**Air quality index (AQI) category**	**Air quality index description**
0.0 – 12.0	Good	Air quality is satisfactory, and air pollution poses little or no risk.
12.1 – 35.4	Moderate	Air quality is acceptable. However, there may be a risk for some people, particularly those who are unusually sensitive to air pollution.
35.5 – 55.4	Unhealthy for sensitive groups	Members of sensitive groups may experience health effects. The general public is less likely to be affected.
55.5 – 150.4	Unhealthy	Some members of the general public may experience health effects; members of sensitive groups may experience more serious health effects.
150.5 – 250.4	Very unhealthy	Health alert: the risk of health effects is increased for everyone.
> 250.4	Hazardous	Health warnings of emergency conditions: everyone is more likely to be affected.

1.3 Operating principle

The BMV080 sensor uses a laser-based optical technology to measure particulate matter mass concentration based on particle counts and relative particle velocities in free space, as illustrated in Figure 1. The natural ambient airflow in the proximity of the sensor is utilized in the measurement. Figure 1 shows how light is scattered after colliding with particles in different directions. The light continues at the top of the particle to indicate the light path when there is no collision.

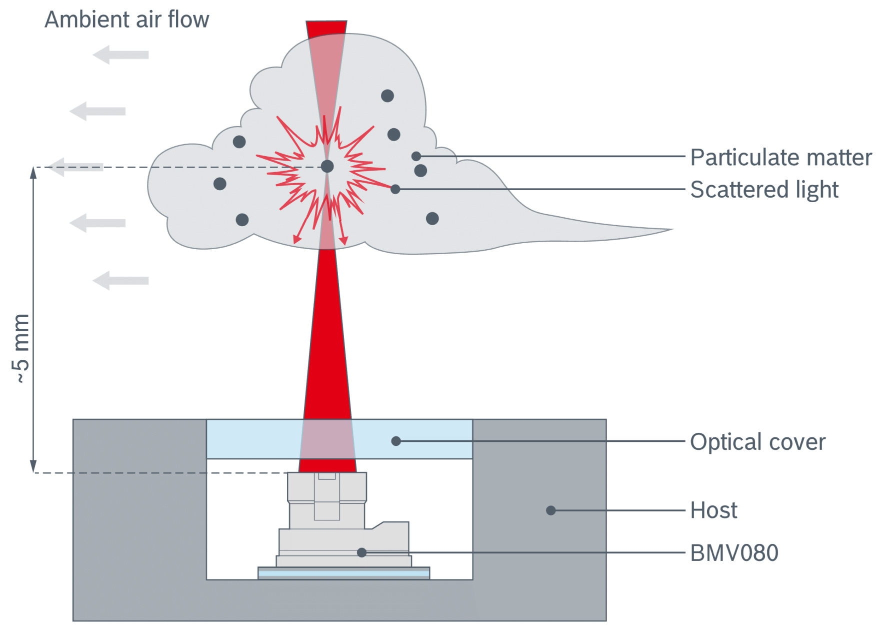


Figure 1: BMV080 sensor operating principle

The measurement procedure is as follows:

- Laser light is emitted from the sensor and focused by the sensor lens at approximately 5 mm from the top of the sensor's lens surface.
- Particles traveling in free space due to the natural ambient airflow are detected when passing through the laser focal (sensitive) region.

- Due to the interaction between particles and light, the light scatters in different directions; a fraction is back-scattered towards the sensor, where the integrated photo-detectors detect it.
- The back-scattered signal is processed by unique algorithms (based on particle counts, particle relative velocity, probed air volume during measurement) to derive the particulate matter mass concentration.

The BMV080 consists of hardware and software components.

Figure 2 shows a BMV080 sensor integrated into a host system, where the BMV080 sensor driver (software) runs on the host processing unit (e.g., MCU).

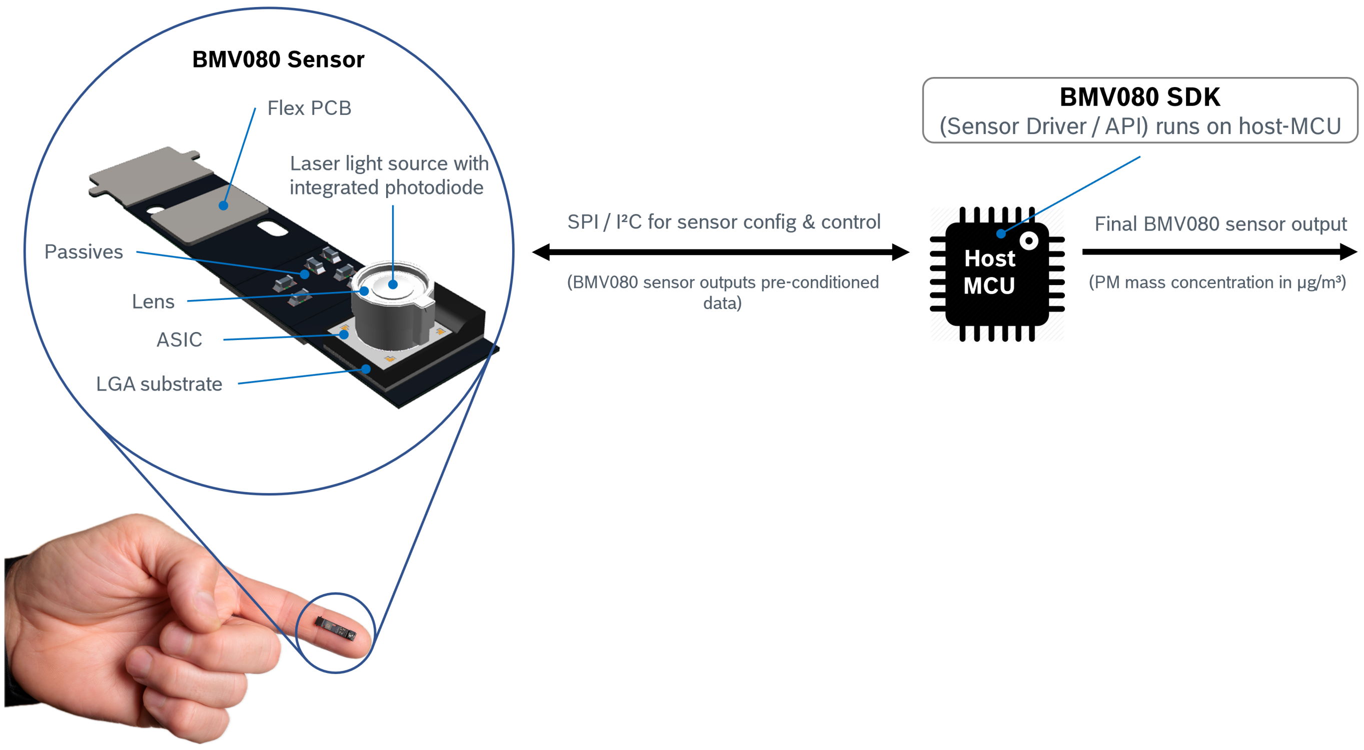

BMV080 Sensor
Flex PCB
Laser light source with integrated photodiode
Passives
Lens
ASIC
LGA substrate
SPI / I²C for sensor config & control
(BMV080 sensor outputs pre-conditioned data)
BMV080 SDK
(Sensor Driver / API) runs on host-MCU
Host MCU
Final BMV080 sensor output
(PM mass concentration in µg/m³)

Figure 2: BMV080 with hardware and software components

Table of contents

List of figures

List of tables

2 Technical specifications

2.1 Standard test conditions

Table 2 specifies the BMV080 sensor under the laboratory standard test conditions. Deviation from the standard test conditions may impact the sensor performance.

Table 2: Standard test conditions

Parameter	**Value**	**Unit**
Ambient temperature	25 ± 2	°C
Relative ambient humidity	50 ± 10	%rH
Relative particle velocity	0.1 – 1.5	m/s
Relative particle flow	Laminar, plane parallel	–
Reference instrument	Aerosol particle size spectrometer LAP 322	–
Particle source	Arizona Road Dust (ARD) Ultrafine A1, ISO 12103-1	–
Integration time	10	s
Vibration suppression	Disabled	–
Measurement algorithm	High precision [1]	–

To precisely estimate the particle mass concentration, the BMV080 must gather sufficient statistical data. To obtain best results in low PM2.5 concentration situations (i.e., < 12 µg/m³) when few particles are present in ambient air, an integration time (IT) of 10 s has been used for standard test conditions.

The sensor performance is specified using the following parameters:

- Precision [ε]: linear regression residual (how much does PM2.5 fluctuate under standard test conditions when test is repeated)
- Linearity [R]: Pearson correlation coefficient (linearity of response to proportional changes)

Figure 3 shows the correlation between a single BMV080 and the reference device under standard test conditions.

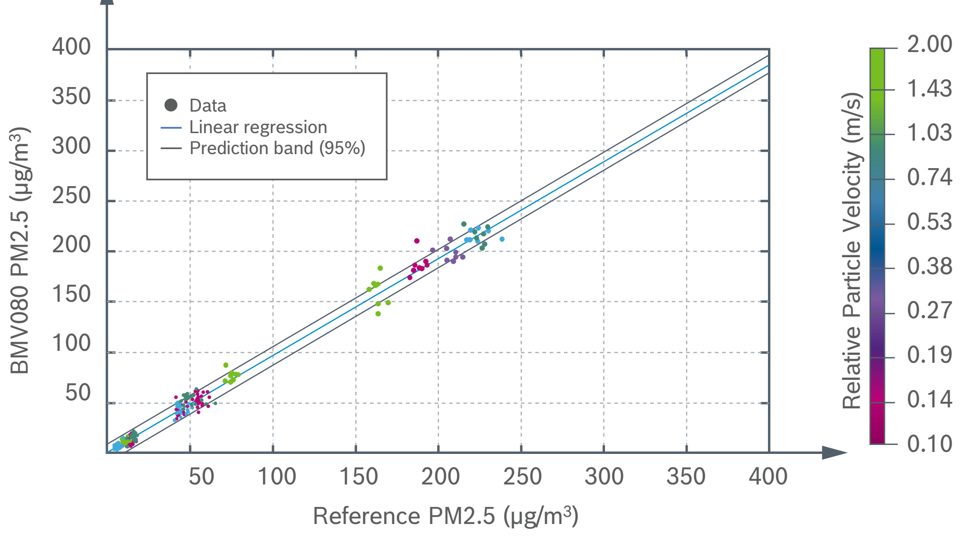


Figure 3: BMV080 characteristics - correlation to the reference device

[1] Refer to Section 5.2.1.2.

2.2 Sensor technical specification

This section describes the technical specification of the BMV080 sensor.

Table 3: BMV080 technical specification

Parameter	**Symbol**	**Typical Value**
PM2.5 measurement range [2]	–	0 – 1000 µg/m³
PM2.5 output resolution [2]	–	1 µg/m³
Minimum detectable particle size [2]	–	0.5 µm
Relative particle velocity [2]	–	0.02 – 1.5 m/s
Precision [2, 3]	ε	$\pm$ 10 µg/m³ @ 0 – 100 µg/m³ $\pm$ 10 % @ 101 – 1000 µg/m³
Linearity [2]	R	$\geq$ 0.98 @ 0 – 400 µg/m³ $\geq$ 0.95 @ 401 – 1000 µg/m³
Measurement modes	–	Continuous measurement mode Duty cycling measurement mode
Measurement algorithm	–	High precision Balanced Fast response
Maximum output data rate [4]	ODR	0.97 Hz
Interface	–	SPI, I²C
Average total current [5]	–	$<$ 68 mA @ 0.97 Hz ODR
Sleep current	–	$<$ 30 µA
Start-up time [6]	$t_{\text{start-up}}$	1.9 s – trigger mode: software polling 2.9 s – trigger mode: IRQ, i.e. hardware interrupt
Operating lifetime (MTTF) [7]	–	10 years
Sensor dimensions [8]	–	4.4 mm x 3.0 mm x 20.0 mm
Sensor weight	–	0.092 g
Laser class	–	Class 1, according to IEC 60825-1

Table 4 shows the absolute minimum and maximum ratings of the sensor BMV080.

Stress above limits, which are stated as "absolute maximum ratings" in Table 4, may cause permanent damage to the device. These are stress ratings only and functional operation of the device under those conditions or any conditions beyond those indicated as "recommended operating conditions" is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

[2] Specified under the standard test conditions defined in Section 2.1. Deviation from standard test conditions may impact sensor performance. Tolerance intervals cover 99 % of the population (95 % confidence interval).
[3] Precision is defined as variation around average PM2.5 value during stable conditions within a 90 s window after a 30 s stabilization time.
[4] ODR is configurable, i.e., using duty cycling measurement mode.
[5] During active measurement.
[6] For more details refer to the timing sequence information given in Sections 5.2.5.1.1 and 5.2.5.1.2.
[7] This applies for a sensor in integrated condition as defined in section 4.1.3 of this datasheet and assuming a 24 h/day operation under standard test conditions (refer to Table 2) for an indoor air quality mission profile. Please note that the lifetime might vary depending on different operating conditions.
[8] Refer to Table 6.

Table 4: Absolute minimum and maximum ratings

Parameter		**Minimum**	**Maximum**
Supply voltages [9]	VDDIO	1.2 V (-5 %)	3.3 V (+5 %)
	VDDD	2.5 V (-5 %)	3.3 V (+5 %)
	VDDL	3.3 V (-5 %)	3.3 V (+5 %)
	VDDA	2.5 V (-5 %)	3.3 V (+5 %)
ESD	Human body model (HMB)	-2 kV	2 kV
	Charged device model (CDM)	-500 V	500 V
Storage [10] temperature range		-40 °C	+70 °C
Sensor operating temperature range [11]		+15 °C	+65 °C
Operating humidity and condensation range [12]		0 %rH	95 %rH

2.3 Measurement modes

It is possible to implement different measurement modes using the BMV080 sensor driver presented in Section 5. Once turned ON, the BMV080 sensor can provide a PM2.5 reading every second. Taking this into consideration, the following measurement modes can be implemented to save power consumption and increase operating lifetime.

2.3.1 Continuous measurement mode

In this mode, BMV080 delivers particle concentrations with the maximum defined Output Data Rate of 0.97 Hz.

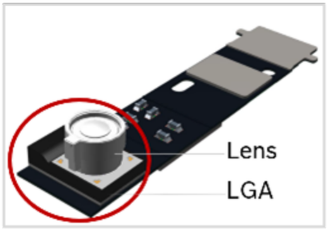
Figure 4 shows the continuous measurement mode integration timing.

[9] Supply pins are described in Table 10.

[10] Applicable when BMV080 sensors are stored in Bosch standard sealed drypack for up to 2 years. For more details, refer to section 3.6.5 Storage Condition in BMV080 integration guideline (BST-BMV080-AN000-02).

[11] Given self heating during operation resulting in sensor internal temperature increase of ~15 K in continuous measurement mode, BMV080 is capable to operate at ambient temperatures <15 °C depending on thermal integration design. For more details, refer to section 3.3 on thermal integration best practices in BMV080 integration guideline (BST-BMV080-AN000-02).

[12] No condensation allowed on the sensor, especially on the lens and LGA areas circled below.



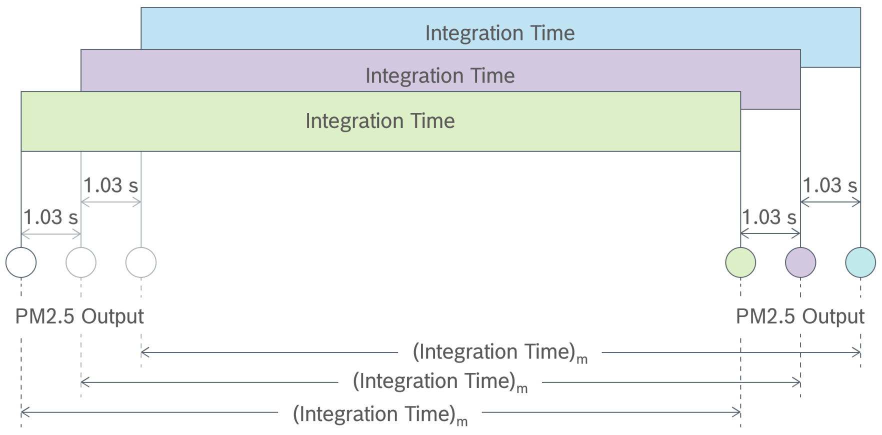


Figure 4: Continuous measurement mode integration timing

2.3.2 Duty cycling measurement mode

In this mode, BMV080 reports a value periodically based on a configured ‘duty cycling period’.

Figure 5 shows the duty cycling measurement mode integration timing.

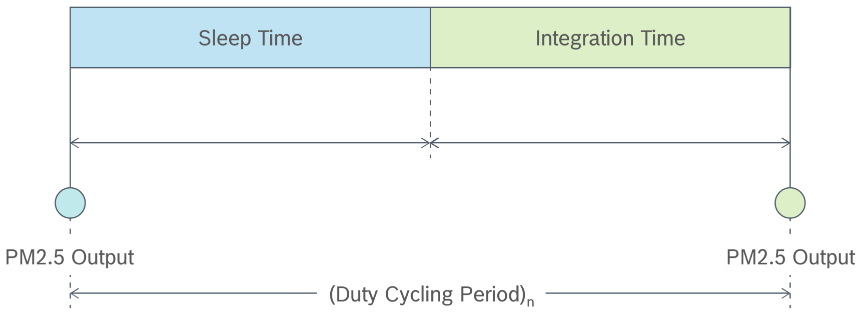


Figure 5: Duty cycling measurement mode integration timing

Note: Precision target as defined in Table 3 does not apply in case of Duty Cycling Measurement mode.

2.4 Power consumption

Table 5 shows the typical power consumption for the different measurement modes.

Table 5: Power Consumption

Measurement mode	**Duty cycling period**	**Power consumption (mW)**[13]
Duty cycling mode	1 min 1 measurement in 1 min	30.4
	5 min 1 measurement in 5 min	6.2
	10 min 1 measurement in 10 min	3.1
	60 min 1 measurement in 60 min	0.6
Continuous mode 1 measurement every 1 s	Not applicable	181.9

[13] Power consumption estimation is based on electrical integration of the BMV080 based on Power Optimized Configuration as per 4.4.1.3.2.

3 Dimensional drawings

3.1 Schematic

Figure 6 shows the BMV080 package. The sensor consists of the following components:

- LGA package
- Lens
- Flex-PCB

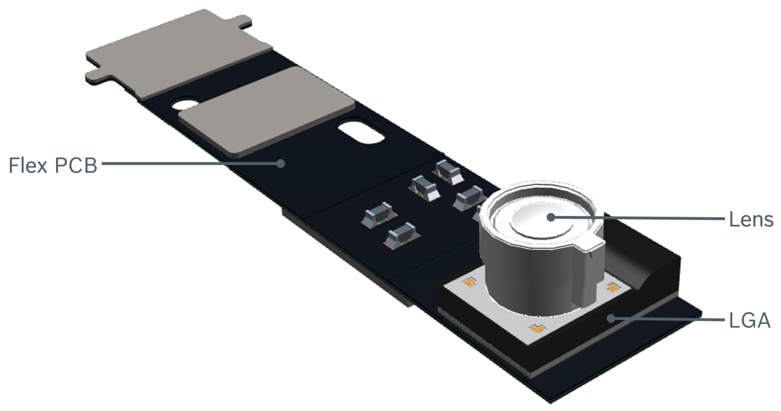


Figure 6: BMV080 package overview

Table 6 provides the BMV080 package dimensions and is further detailed in Figure 7. A 3D CAD model of the BMV080 design to support the integration in a host system is available on request, see Section 8.

Table 6: BMV080 package dimensions

Parameter		Nominal dimension [mm]
Flex PCB size	Width	5.5 (with ears of ZIF connector) 4.4 (without ears)
	Length	20.0
Total height		3.005
Lens height		1.83
LGA substrate thickness		0.2
FPC thickness		0.38

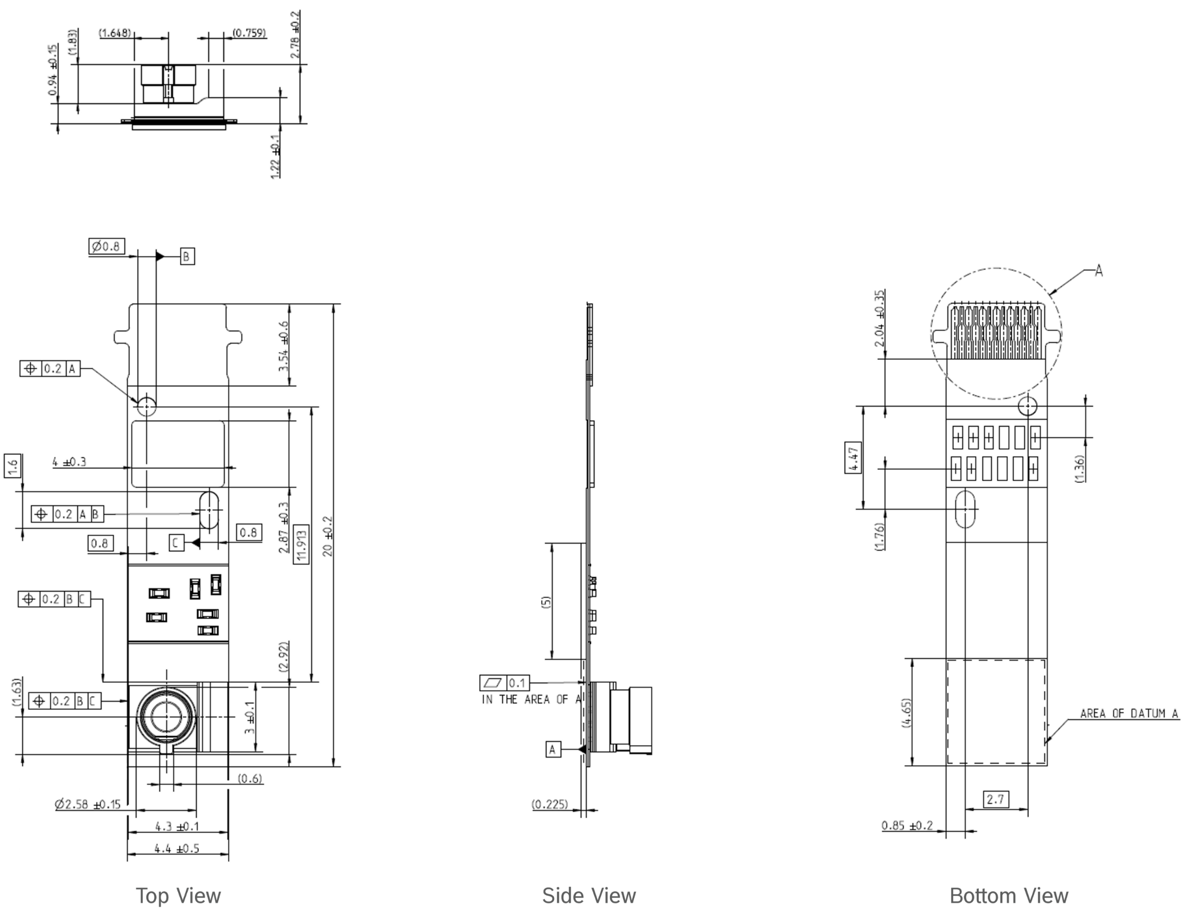


Figure 7: BMV080 dimensional drawings (dimensions in mm)

3.2 Flex PCB

Bending of the Flex-PCB is only allowed in the bending area highlighted with red rectangle shown in Figure 8. Bending radii and maximum bending angle for flex PCB are currently under investigation for the allowed bending area.

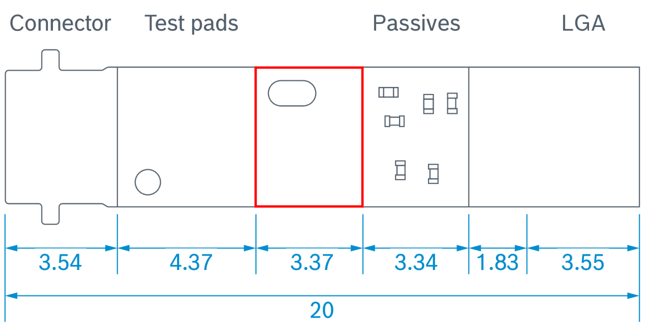


Figure 8: BMV080 flex PCB footprint

The connector area of the BMV080's flex PCB is compatible with the following ZIF connectors (used in the host system):

- Molex 503566-1302, Easy-On FPC connector, 0.30 mm pitch, 13 circuits, mated height 0.95 mm
- KYOCERA AVX, Series 6844, Part number: 046844713002846+
- Greenconn CFTD104-1302A001C2AD

Details of the connector area of the flex PCB are shown in Figure 9.

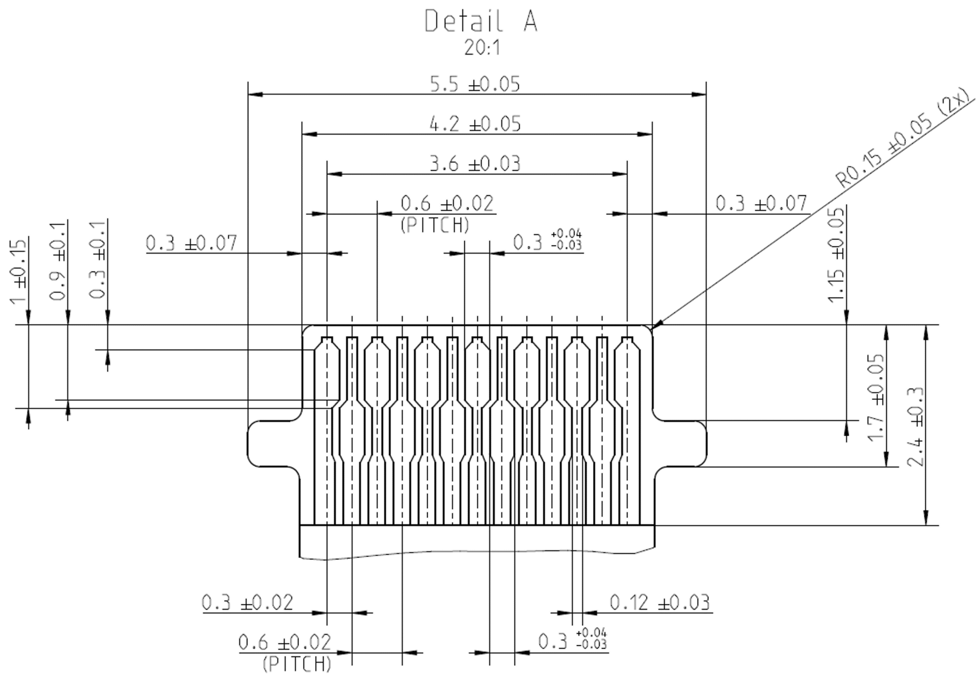


Figure 9: ZIF connector in detail (dimensions in mm)

Note:

Please be aware that the pin numbering scheme used for the BMV080 sensor (Figure 23) might differ from the pin numbering schemes of the ZIF connectors which are used in the host system (see list above). For a comparison please refer to Figure 10.

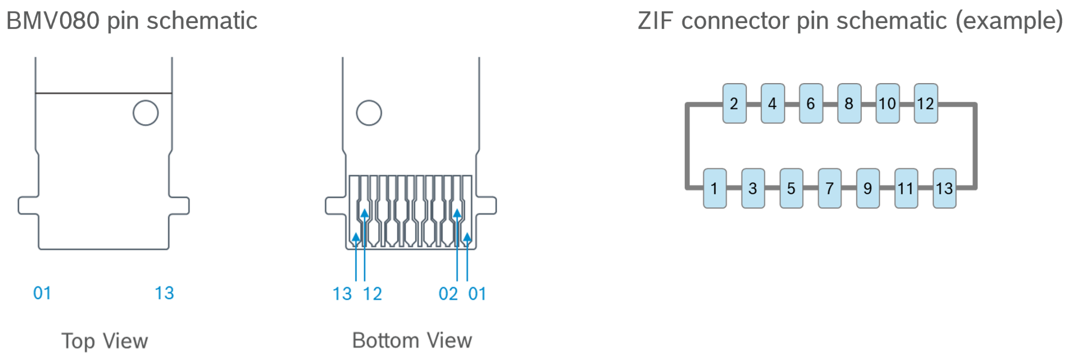


Figure 10: Comparison of pin numbering schemes for BMV080 and ZIF connectors

4 Hardware integration guidelines

The BMV080 has 4 hardware design interfaces, shown in Figure 11. Detailed requirements for each interface are specified in the following sections:

- Mechanical: contains environmental considerations as well as information about footprint, handling and mounting of the sensor into the host. In addition, important advice with respect to contamination is given.
- Thermal: thermal connection to the host for dissipating heat generated by the sensor.
- Optical: optical components added by the host to the sensor optical path, e.g., optical cover.
- Electrical & Communication: electrical connections for power and data communication between the sensor and host.

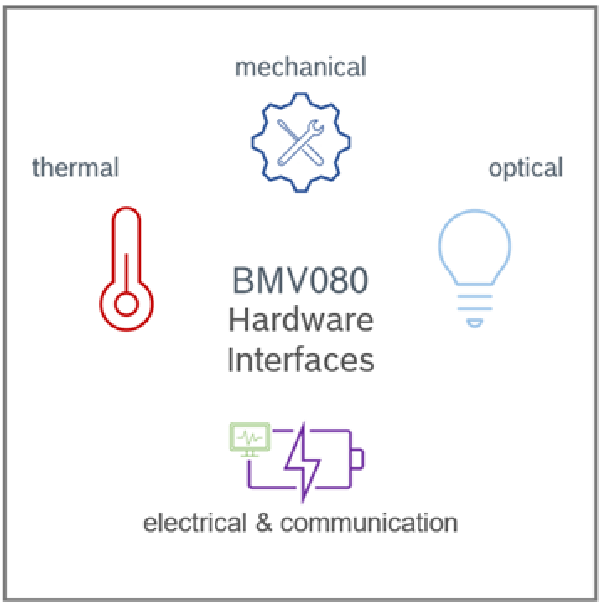


Figure 11: BMV080 hardware design interfaces

4.1 Mechanical interface

4.1.1 Environmental considerations

The BMV080 measures particulate matter mass concentration in a probing region at approximately 5 mm above the sensor lens surface. To ensure the sensor's functionality and performance, it is recommended to follow these requirements for sensor placement:

- Sensor location in the host system: place the BMV080 minimum 17 mm away from sharp edges to avoid turbulent air flows that could affect the sensor performance, see Figure 12.
- Objects in front of the sensor (obstruction): reflections caused by objects in the optical path can influence the sensor functionality. The BMV080 software detects these events as obstructions. While occasional obstruction (e.g., by waving hands) is filtered out in the BMV080 software and does not influence the sensor performance, static obstruction (i.e., a fixed object in the obstruction-sensitive region, see Figure 13) will influence the sensor functionality. When the BMV080 software detects a static obstruction, the PM2.5 calculation is not available, and an obstruction flag is returned. Therefore, static obstructions caused by the integration into the host shall be avoided.

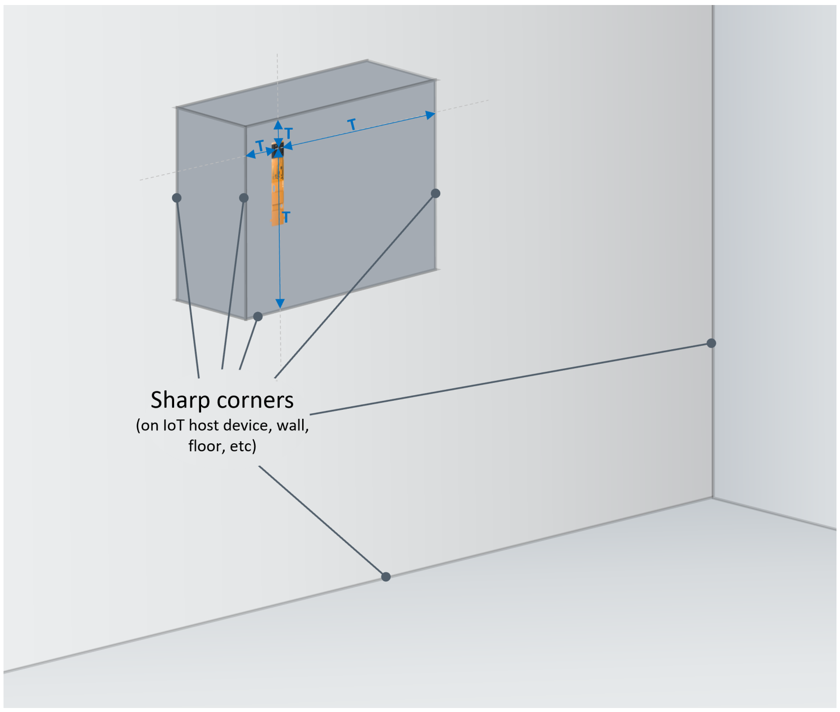


Figure 12: Placement of BMV080 away from sharp edges

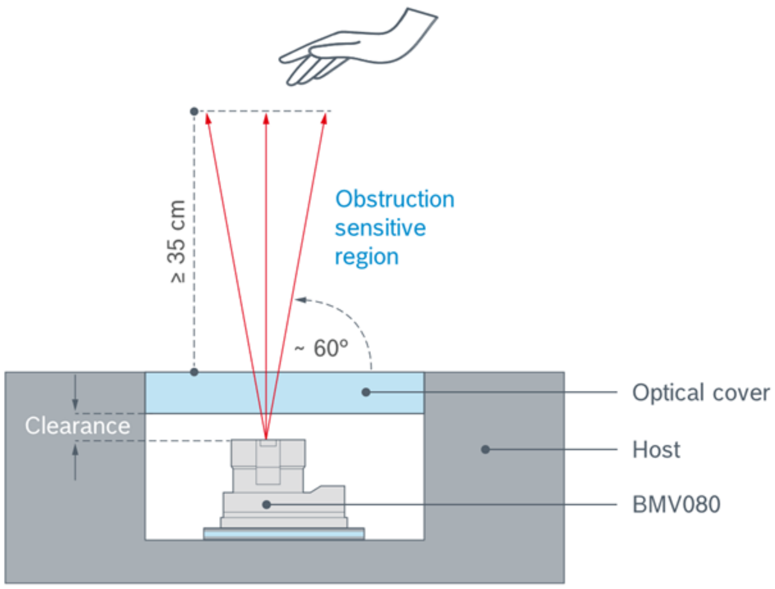


Figure 13: BMV080 integrated in a host system

The optical properties and the geometry of the object(s) in the optical path of the BMV080 determine the size of the obstruction-sensitive region. For a white reflecting surface perpendicular to the laser light emitted by the BMV080, this distance is $\geq$ 350 mm from the host surface. This distance reduces for less reflecting objects (e.g., skin).

4.1.2 BMV080 footprint

A detailed description of the BMV080 footprint, dimensions, used connectors, and drawings can be found in Chapter 3.

4.1.3 Mounting and assembly

The mounting and assembly concept must fulfill the following requirements to ensure functional integration of the BMV080 into a host system:

- Mechanically fix the BMV080 in the host system
- Protect the BMV080 (e.g., lens) from damage
- Correct positioning to ensure a clear optical path for the sensor optics
- Enable electrical connection to the host system
- Enable thermal connection to the host system
- The mounting position within host has to ensure that
  - no humidity (for example coming from any condensation) covers the lens, optical cover inner surface or any other part of the integrated BMV080,
  - no contamination is applied on the lens and the inner surface of the optical cover of the integrated BMV080 (this could be achieved, e.g., by using a clean room environment for the assembly process. Manufacturing and test of the sensor at Bosch Sensortec is performed in an ISO 7 clean room environment).

The following sections describe two mounting and assembly concept examples.

The BMV080 is placed inside a cavity formed into the housing structure of the host system that includes the optical cover, as shown in Figure 14. The cavity's shape and dimensions define the sensor's position inside the housing structure and ensure the required clearance between the BMV080 and the optical cover (Section 4.3). The BMV080 needs to be fixed in the cavity from the backside, and a thermal connection needs to be enabled (Section 4.2). This can be done, for example, with a plastic component (e.g., molded polymer) that includes a metal spring and is attached to the housing structure with a clip structure.

This concept has the following options:

- Precise control of the clearance between the BMV080 lens and the optical cover
- Flexibility to place the PCB for the sensor position (side by side, stacked, etc. – see Figure 15).

4.1.3.1 Concept 1: Case mounting

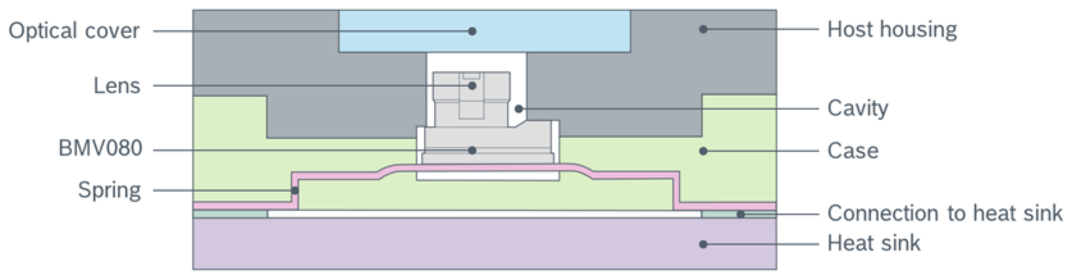


Figure 14: Concept 1: Case mounting

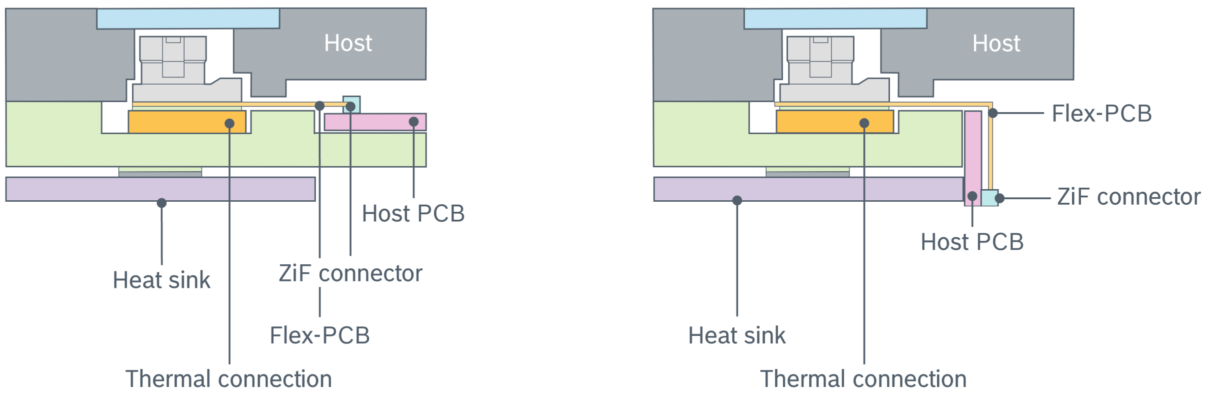


Figure 15: Placement of PCB for the sensor position

Figure 16 shows an example assembly process for Concept 1 – Case mounting.

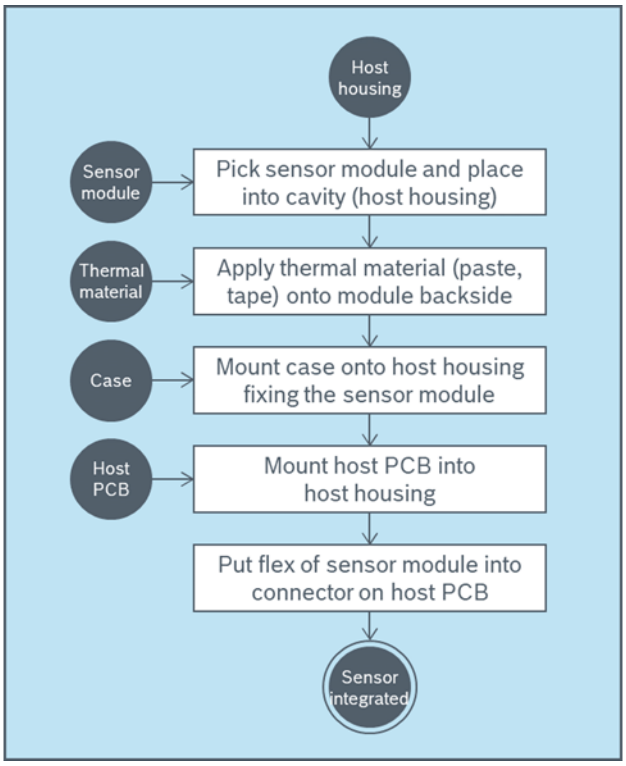

Host housing
Sensor module
Pick sensor module and place into cavity (host housing)
Thermal material
Apply thermal material (paste, tape) onto module backside
Case
Mount case onto host housing fixing the sensor module
Host PCB
Mount host PCB into host housing
Put flex of sensor module into connector on host PCB
Sensor integrated

Figure 16: Process flow for concept 1 - Case mounting

4.1.3.2 Concept 2: PCB mounting

The BMV080 is mounted directly on the host PCB, working simultaneously as a mechanical fixation and thermal connection. For example, BMV080 can be fixed to the host PCB using a cage system (Figure 17). Different options can be used to fix the cage system to the host PCB, for example, screws. Clearance between the BMV080 lens and cover glass has to be ensured by the fixation of the PCB in the host housing (for info about clearance, refer to section 4.3.1).

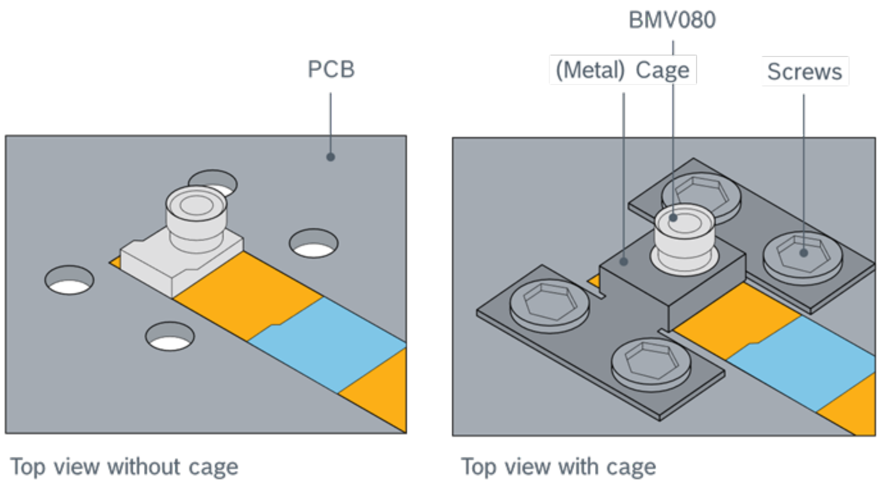


Figure 17: Concept 2 – PCB mounting

This concept allows for the following options:

- Use PCB as a heat sink
- Direct mounting on host PCB

Figure 18 shows an assembly process flow example recommended for Concept 2 – PCB mounting.

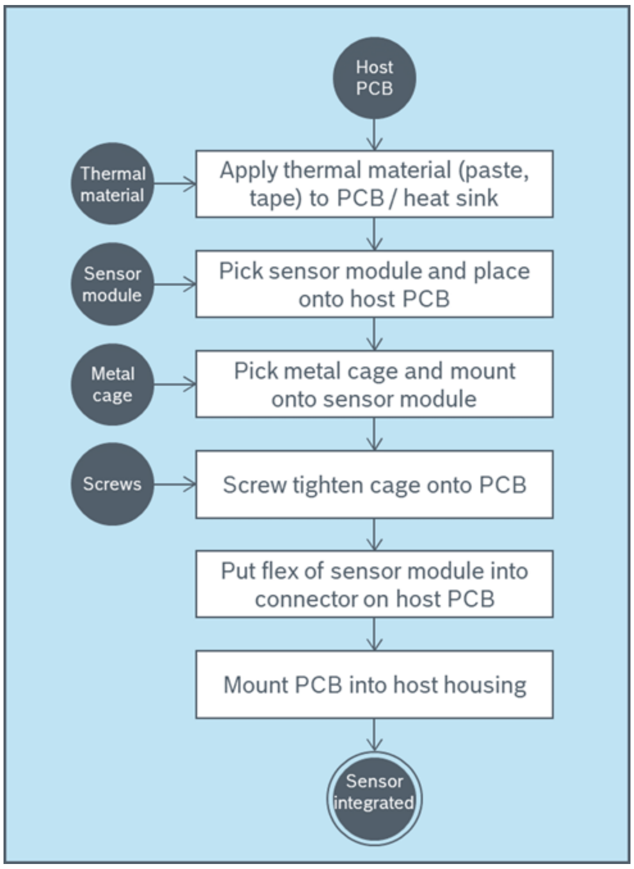

Host PCB
Thermal material
Apply thermal material (paste, tape) to PCB / heat sink
Sensor module
Pick sensor module and place onto host PCB
Metal cage
Pick metal cage and mount onto sensor module
Screws
Screw tighten cage onto PCB
Put flex of sensor module into connector on host PCB
Mount PCB into host housing
Sensor integrated

Figure 18: Process flow for concept 2 – PCB mounting

4.1.4 Handling and mounting guidelines

During the assembly process, the following handling guidelines must be considered to avoid any damage to the sensor:

- Maximum forces applied to the LGA package (Figure 19)
  - $F_{\text{max},\,x} = 5\,\text{N}$
  - $F_{\text{max},\,y} = 5\,\text{N}$
  - $F_{\text{max},\,z} = 3\,\text{N}$
- Avoid any force on the lens
- Avoid any contamination on the lens surface (e.g., fingerprints, dust)

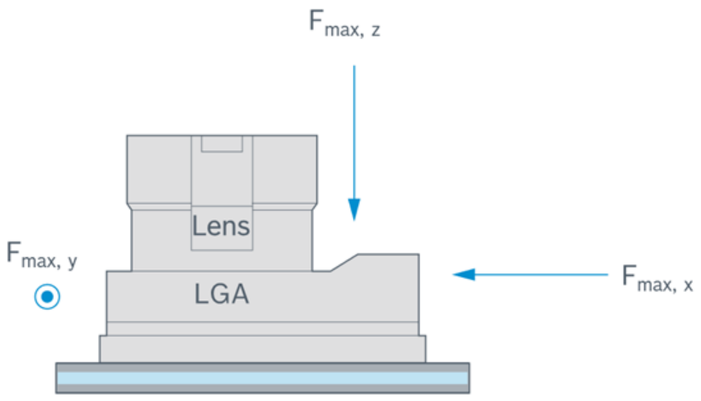


Figure 19: Maximum forces applied to the LGA package

4.2 Thermal interface

The BMV080 sensor needs to be connected to a cooling system to dissipate the heat generated by the sensor.

Figure 20 shows a simplified overview of a thermal setup. The lower surface of the BMV080 Flex-PCB is attached to a cooling element (e.g., a heat sink) via a thermal contact (in this case a tape). The heat sink itself is part of the host system.

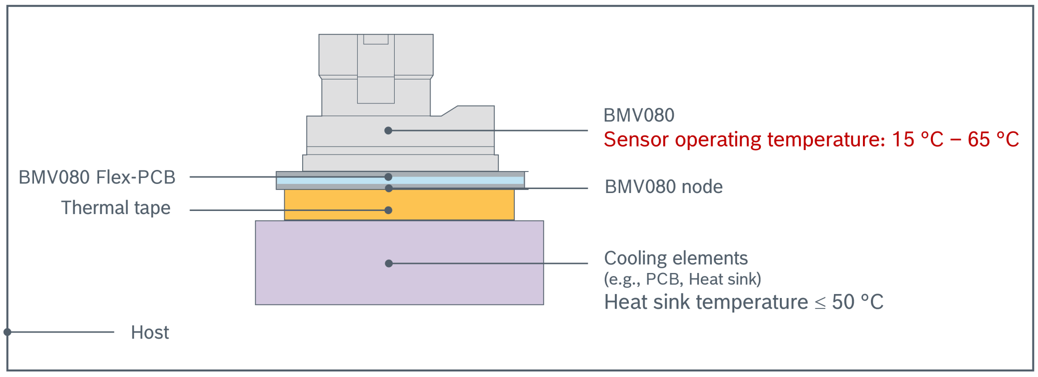


Figure 20: Sketch of the thermal integration situation

When designing the thermal interface from the BMV080 Flex-PCB to the host, the following parameters must be considered, see Table 7.

Table 7: Given parameters for the thermal setup

Parameter	**Value**
Maximum sensor operating temperature	65°C
Maximum power (continuous measurement) (P_{max})	181.9 mW

The maximum sensor operating temperature is 65 °C, the sensor internal temperature increase is 15 K due to power dissipation in the sensor (This value is based on continuous measurement mode).

Please refer to section 3.3 in BMV080 integration guideline for more details on thermal integration best practices.

4.3 Optical interface

In the host system, the BMV080 has to be integrated behind an optical cover to protect the lens from contamination (e.g., fingerprints, dust) and mechanical damage in the application. This section covers the mechanical and optical properties of the optical cover.

4.3.1 Mechanical properties

- Optical cover thickness (d) and clearance (σ) between the lens and optical cover: the sensor-sensitive region is located approx. 5 mm above the lens surface. It must be ensured that the sensitive region is above the optical cover – outside the housing in free air.
- Optical window width (Φ): the optical cover has to be wide enough to transmit the laser beams.
- Maximum tilt angle (α) between the optical cover and the BMV080: the lens surface and the optical cover have to be almost parallel to avoid distortion of the optical signal.

Figure 21 and Table 8 show the mechanical requirements of the optical interface.

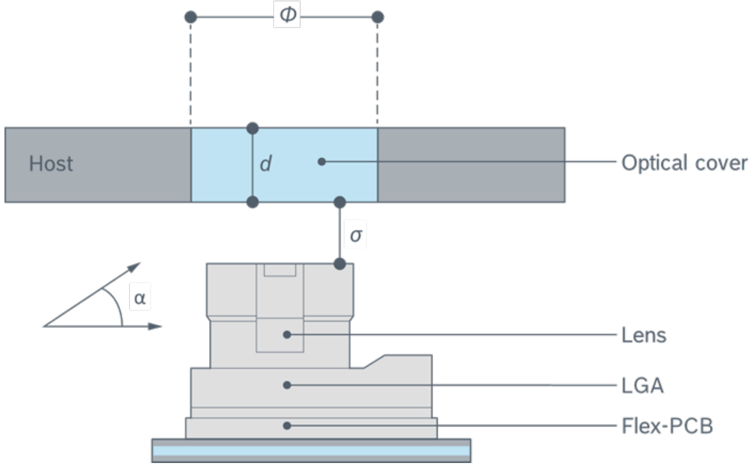


Figure 21: Sketch of the optical interface

Table 8: Optical interface mechanical properties

Parameter	**Symbol**	**Value**	**Unit**
Optical cover thickness	d	$0.3 \leq d \leq 0.8$	mm
Optical window width	Φ	≥ 1.4	mm
Tilt angle	α	≤ 4	°
Clearance between the top of the BMV080 lens and the bottom of the optical cover	σ	0.2 ± 0.1	mm

Scratches and damage on the optical cover can affect the sensor performance. A hard material like glass is recommended (e.g., Corning® Gorilla® Glass 6).

The optical cover must be kept free from fingerprints, dust, and other impurities and contamination.

4.3.2 Optical properties

- Refractive index: this parameter influences the laser beam shape, and therefore it needs to be specified to avoid affecting the sensor performance
- Transmissivity: the cover glass must allow the transmission of the laser beams. A transmissivity below specifications will decrease the optical signal intensity, thus affecting the sensor functionality.

Optical properties are specified in Table 9.

Table 9: Properties of optical cover

Parameter	**Typical value**
Refractive index	1.45 – 1.77 for λ = 850 nm
Transmissivity	$\geq$ 90 % for an angle of incidence of 30°, P-polarization, λ = 850 nm

Additional requirements to the optical cover:

- Optical surface roughness needed to avoid any beam distortion
- Avoid any polarization filter

4.4 Electrical and communication interface

The BMV080 is connected to the host system by the electrical interface for data communication and power supply.

4.4.1 Electrical interface

4.4.1.1 Pin configuration and function

Figure 22 shows a pin schematic of the BMV080. Figure 23 shows the BMV080 flex-PCB. Table 10 shows the functional description of each pin.

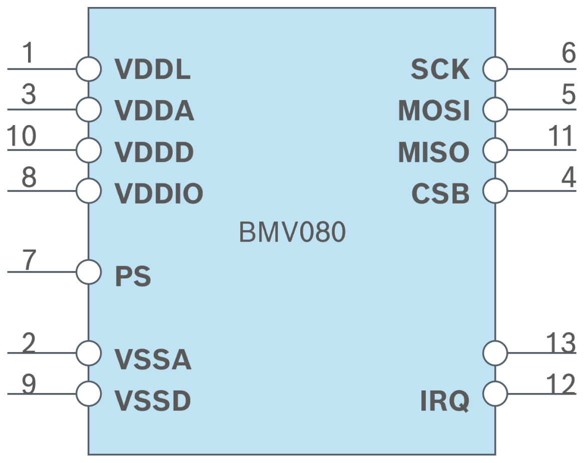


Figure 22: BMV080 pin schematic

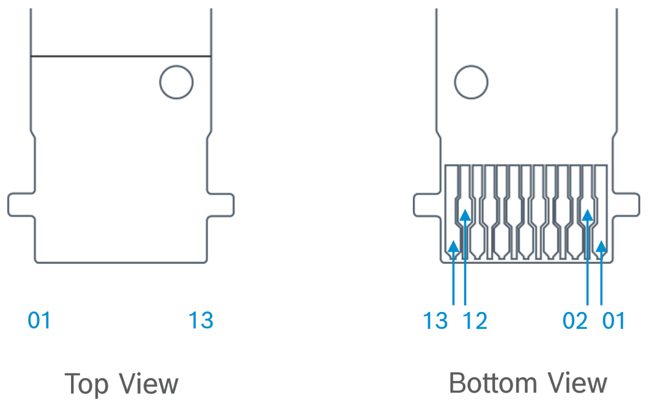


Figure 23: Top view of the bottom metal of the flex PCB showing the connector pins numbering

The protocol select (PS) pin is a logic input to configure the type serial interface, SPI, or I²C. Figure 24 shows how the PS pin can be connected to select the SPI or the I²C protocol. The SPI protocol is selected when this pin is connected to a logic low (VSSD). If this pin is connected to a logic high (VDDIO), the I²C protocol is selected. If this pin is not connected (not recommended), the I²C protocol is selected. The pin state is latched during power-up by the digital core.

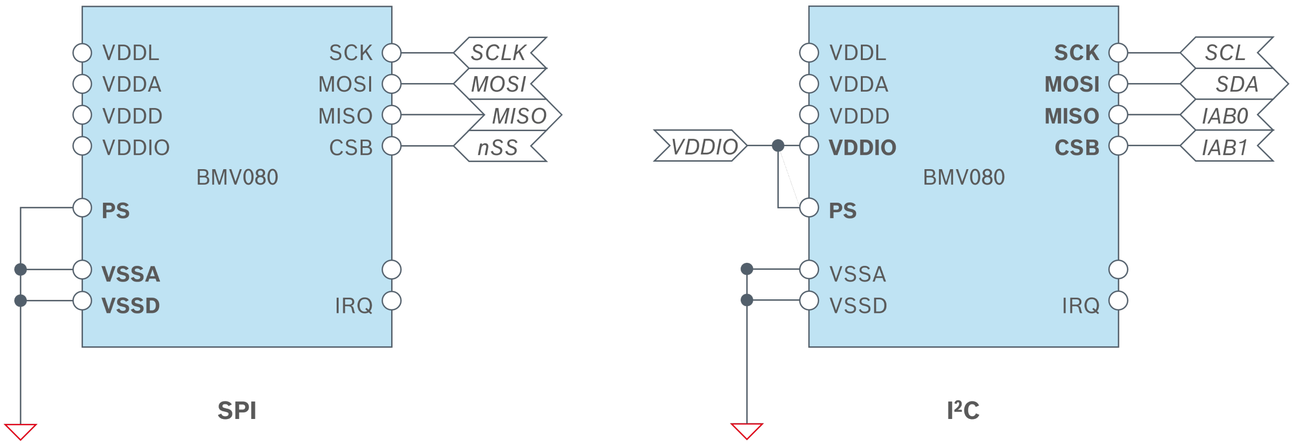


Figure 24: Different configurations for protocol selection (SPI / I²C)

Table 10: BMV080 pin description

Label	**Pin no.**	**Type** [14]	**Description**
VDDL	1	P	Laser supply voltage. The laser supply voltage is 3.3 V and the supply pin should be decoupled from VSSA by >1 µF.
VDDA	3	P	ADC supply voltage. The ADC supply voltage is 2.5 V– 3.3 V which should be decoupled from VSSA by >1 µF.
VDDD	10	P	Digital supply voltage. The digital supply voltage is 2.5 V to 3.3 V which should be decoupled from VSSD by >1 µF.
VDDIO	8	P	Interface power supply. The interface power supply is from 1.2 V (min) to 3.3 V (max), with 1.8 V being the typical value. This pin is typically connected to the same supply of the host interface (e.g., application processor, microcontroller, or FPGA).
PS	7	DI	Protocol select. Logic input. This pin is used to configure the type of serial interface, SPI or I²C. ▪ The SPI protocol is selected if this pin is connected to a logic low (VSSD). ▪ If this pin is connected to a logic high (VDDIO), the I²C protocol is selected. Note: ▪ If this pin is not connected (not recommended), the I²C protocol is selected. ▪ The pin state is latched during power-up by the digital core.
VSSA	2	GND	Analog ground. This pin is the ground reference for all analog domains, namely VDDL and VDDA. Connecting VSSA and VSSD as close as possible to the BMV080 pin header is recommended.
VSSD	9	GND	Digital ground. This pin is the ground reference for all digital domains, namely VDDD and VDDIO. Connecting VSSA and VSSD as close as possible to the BMV080 pin header is recommended.
SCK	6	DI	Serial clock. Digital input. This pin functions as serial input for both serial interface protocols (SPI and I²C).
MOSI	5	DI/DO	SPI: This pin functions as Master Out Slave In (MOSI). I²C: This pin functions as a Serial Data line (SDA).
MISO	11	DI/DO	SPI: This pin functions as Master In Slave Out (MISO). I²C: This pin functions as I²C Address Bit 0 (IAB0). IAB0 allows adjusting the I²C Address Bit 0 of the slave by applying a logic low (VSSD) or logic high (VDDIO).
CSB	4	DI	SPI: This pin functions as not Slave Select (nSS). I²C: This pin functions as I²C Address Bit 1 (IAB1). IAB1 allows adjusting the I²C Address Bit 1 of the slave by applying a logic low (VSSD) or logic high (VDDIO).
IRQ	12	DO	Interrupt line. Digital out. Active low. Internal pull-up is enabled by default for IRQ pin.
Do not connect	13	DO	Keep pin floating. Do not connect to ground or apply voltage.

4.4.1.2 Proposal for filtering signal errors

Strong disturbing signals on the SCK pin may influence the measurement results of the BMV080. In an environment where strong disturbance signals are present, the SCK pin could be protected with a suitable low pass filter, which filters out the disturbance but allows normal communication.

[14] P = power supply, DI = digital in, DO = digital out, GND = ground.

4.4.1.3 Power domains

The BMV080 has four power domains, listed in Table 11. The passive components specific to the four BMV080 power domains are already included on the flex-PCB.

Table 11: Power domains specification and current consumption

Power domain	**Electrical specification**		**Power supply rejection ration**	**Absolute maximum rating**	**Current consumption**	
	Min	**Max**			**Sleep mode**	**Measurement mode**
VDDIO[15]	1.2 V - 5 %	3.3 V + 5 %	Ensure dynamically the listed min. and max. supply values	3.6 V	< 3 µA	0.8 mA
VDDD	2.5 V - 5 %	3.3 V + 5 %	Ensure dynamically the listed min. and max. supply values	3.6 V	< 15 µA	21.6 mA
VDDL	3.3 V - 5 %	3.3 V + 5 %	100 mV VPP at any frequency	3.6 V	< 3 µA	18.29 mA
VDDA	2.5 V - 5 %	3.3 V + 5 %	100 mV VPP at any frequency	3.6 V	< 3 µA	27.0 mA

4.4.1.4 Connection diagrams

The following connection diagrams are examples of how to electrically connect the BMV080 to a host system. Other configurations are possible. Common acronyms used in every connection diagram are Voltage Battery (VBAT) and Power Management Integrated Circuit (PMIC).

4.4.1.4.1 Single supply rail

Figure 25 illustrates the simplest connection diagram possible. All four power domains of the BMV080 sensor, i.e., VDDL, VDDA, VDDD, and VDDIO, are powered by the same rail (3.3 V). The power consumption of the BMV080 is the highest since VDDA and VDDD are supplied at the highest voltage possible.

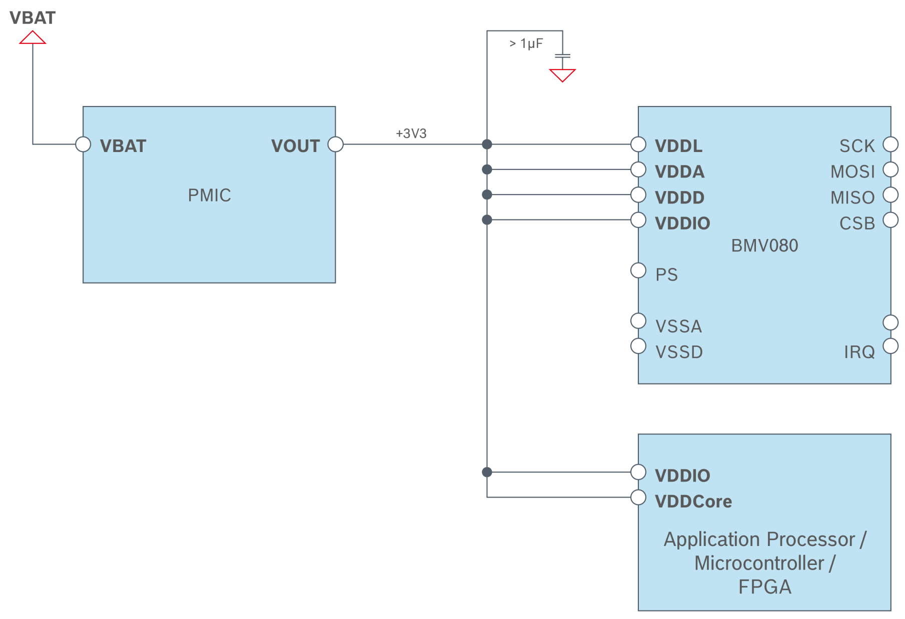


Figure 25: Power-supply configuration with a single supply rail

[15] Reduced SPI speed for VDDIO < 1.8 V.

4.4.1.4.2 Power optimized configuration

Figure 26 shows the connection diagram targeting lowest power consumption for each BMV080 power domain while keeping the full SPI speed. Therefore, each power domain is supplied at its lowest allowed voltage (See Section 4.4.1.3):

- VDDL is supplied at 3.3 V
- VDDA and VDDD are supplied at 2.5 V
- VDDIO is supplied by a second PMIC that also supplies the host (e.g., Application Processor, Microcontroller, or FPGA).

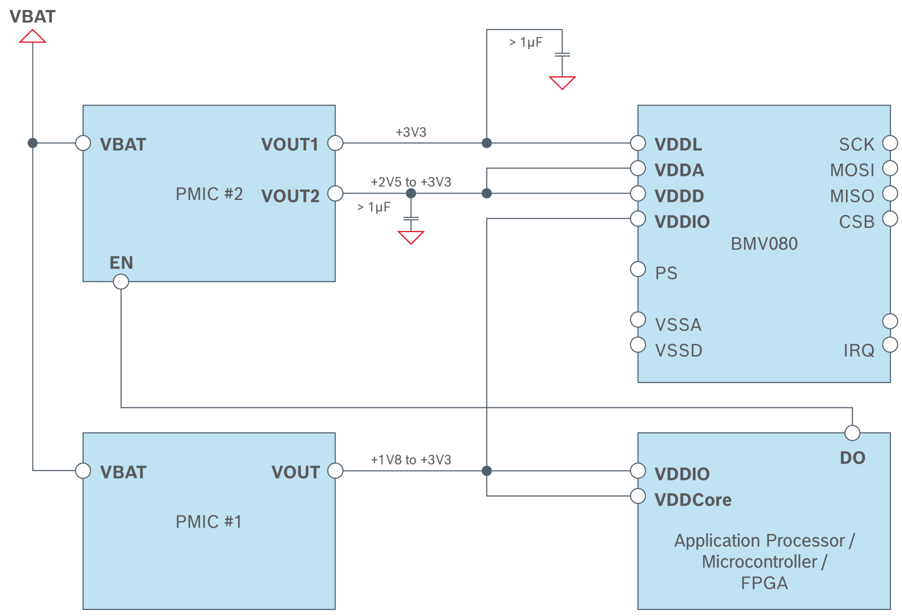


Figure 26: Power-supply configuration with the lowest voltage level

Since VDDD and VDDIO do not share the same power rail, a power-up sequence has to be implemented to ensure the latching of the right information during start-up (see Figure 27). For this purpose, it is possible to use a dedicated PMIC (PMIC #2) controlled by the host.

The sequence is as follows:

1. PMIC #1 supplies the host and also provides VDDIO to BMV080
2. The host boots up. After the required period, a digital output (DO) of the host is used to enable the PMIC #2
3. PMIC #2 supplies the remaining domains of BMV080, i.e., VDDL, VDDA, and VDDD

Whenever the BMV080 is not actively measuring, further energy consumption can be saved by turning completely off the BMV080 sensor instead of putting it into sleep mode; a possible realization of this strategy would consist of disabling PMIC #2 through an enable command generated by the host application processor/microcontroller/FPGA.

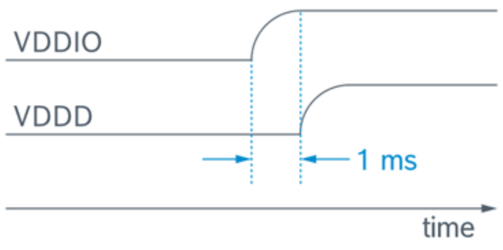


Figure 27: Power-up sequence diagram

4.4.1.4.3 Separated power supply for analog and digital domains

The separation of analog and digital power domains allows the selection of dedicated PMICs for noise and efficiency. While the low noise PMIC #2 supplies the analog domains VDDL and VDDA, the highly efficient PMIC #1 supplies the digital domains, including the host (See Figure 28). Also, the coupling of digital noise from the digital to the analog domain can be efficiently suppressed.

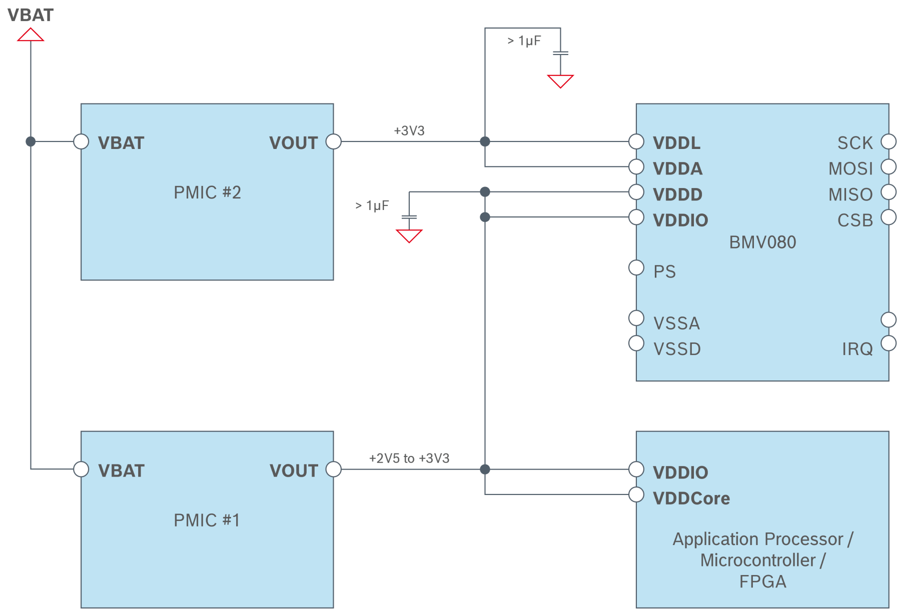


Figure 28: Power-supply configuration with separated analog and digital domains

4.4.2 Communication interface

The sensor supports the I²C and SPI digital interfaces, where it acts as a slave for both protocols. The I²C interface supports the Standard, Fast and High Speed modes. The SPI interface supports both SPI mode 0 (CPOL = CPHA = ‘0’) and mode 3 (CPOL = CPHA = ‘1’) in 4-wire configuration.

4.4.2.1 Interface selection

Interface selection is done automatically based on PS (protocol select) status. If PS is connected to VDDIO, the I²C interface is active. If PS is pulled down, the SPI interface is activated. After power-up, the level on PS is latched. Hence there cannot be any reselection of the protocol afterwards. For more details, please refer to section 4.4.1.1

Table 12: Protocol selection by setting PS pin level

PS pin level (pin no. 7 BMV080 ZIF connector)	Selected protocol
1'b0	SPI
1'b1	I²C

4.4.2.2 Pin assignment

Table 13: Pin functions depending on selected protocol

Pin no. BMV080 ZIF connector	SPI	I²C
6	Serial Clock (SCK)	SCL
5	Master Out Slave In (MOSI)	SDA (external pull-up only)
4	Chip Select (CSB), chip-select, low-active	Selection LSB of device address (see 4.4.2.4.1 I²C device address)
11	Master In Slave Out (MISO)	Selection LSB of device address (see 4.4.2.4.1 I²C device address)
7	Protocol Selection (PS) high level = I²C low level = SPI	

Attention: The MOSI pin has specific usage during serial data communication, depending on the protocol:

- SPI, 4-wire: uni-directional master-out data SPI,
- I²C: bi-directional serial data (external pull-up only)

4.4.2.3 SPI protocol

The SPI interface is compatible with SPI mode 0 (CPOL = CPHA = '0') and mode '3' (CPOL = CPHA = '1'). The automatic selection between mode 0 and 3 is determined by the value of SCK after the CSB falling edge.

- Frequency 1 MHz – 10 MHz [16]
- 4 wire connection

4.4.2.3.1 SPI 4-wire read mode 0

A read transfer consists of 16-bit header information clocked out on the MOSI line followed by reading an arbitrary number of 16-bit words from the MISO line. The Chip Select Signal (CSB) stays low during the whole transfer.

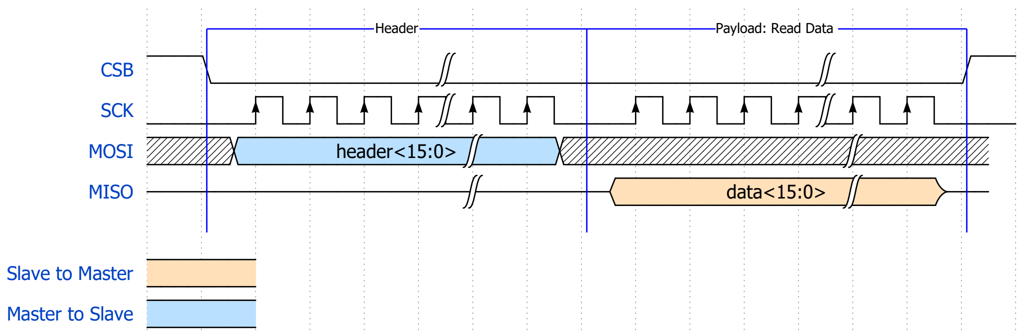


Figure 29: SPI 4-wire read mode 0

[16]reduced SPI speed for VDDIO < 1.8 V.

4.4.2.3.2 SPI 4-wire write mode 0

A write transfer consists of 16-bit header information followed by an arbitrary number of 16-bit words of payload. Header and payload are clocked out on the MOSI line. The Chip Select Signal (CSB) stays low during the whole transfer.

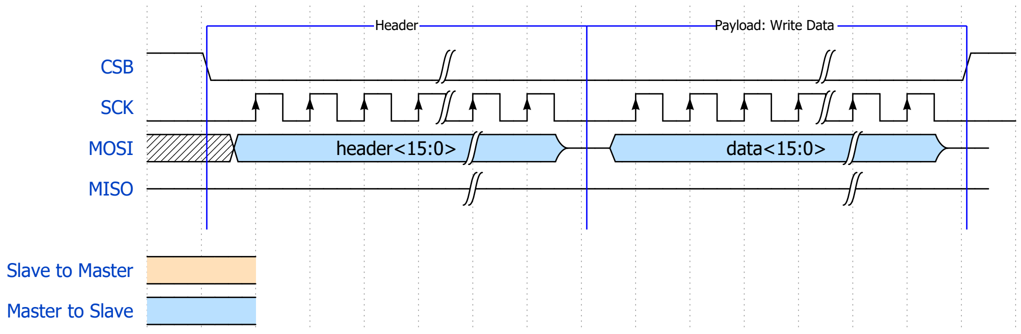


Figure 30: SPI 4-wire write mode 0

4.4.2.4 I²C protocol

- Standard-mode (100 kHz)
- Fast-mode (400 kHz)
- Fast-mode Plus (Fm+, 1 MHz, recommended)

The protocol is compliant to the I2C specification UM10204 Rev. 3, with the following restrictions:

- 7-bit device addressing only
- No clock stretching

4.4.2.4.1 I²C address selection

The I²C device address can be selected depending on the unused CSB and MISO pin. These pins cannot be left floating; if left floating the I²C address will be undefined! This selection is latched after power-up and cannot be altered afterwards.

Table 14: Host interface - I²C device address selection

Bit no.	<6>	<5>	<4>	<3>	<2>	<1>	<0>
	1	0	1	0	1	CSB	MISO

Below are some other I²C address selection possibilities:

- CSB = 0, MISO = 0, address = 0x54
- CSB = 0, MISO = 1, address = 0x55
- CSB = 1, MISO = 0, address = 0x56
- CSB = 1, MISO = 1, address = 0x57

4.4.2.4.2 I²C read

A read sequence consists of two consecutive I²C transfers: 16 bits of header followed by an arbitrary number of payload data. The payload is always a multiple of 16 bits. For details on the header transfer, see Section 4.4.2.4.4.

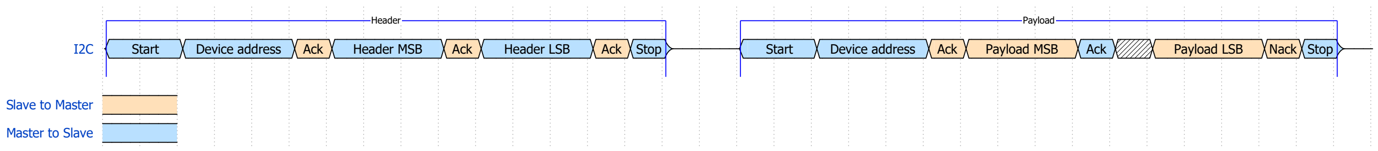


Figure 31: I²C read

4.4.2.4.3 Complete I²C write

A write sequence consists of two consecutive I²C transfers: 16 bits of header followed by an arbitrary number of payload data. The payload is always a multiple of 16 bits. For details on the header transfer, see Section 4.4.2.4.4.

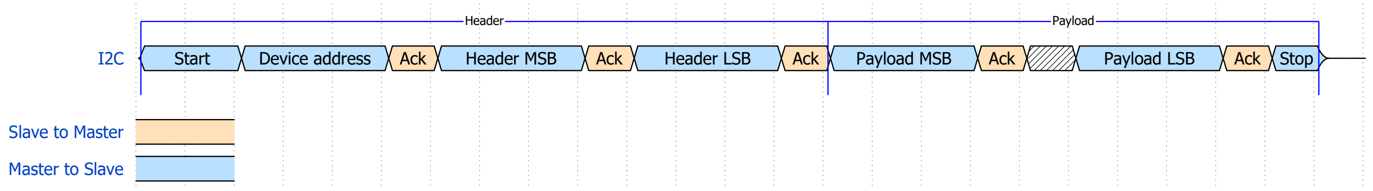


Figure 32: Complete I²C write

4.4.2.4.4 I²C header transfer

The header is transferred by sending the slave address in write mode (R/$\overline{W}$= ’0’), resulting in slave address 10101XX0 (‘X’ is determined by the state of CSB and MISO pin). Then the master sends the 16 bits of header information MSB first. The reception of a byte is acknowledged by the slave. The transaction is ended by a stop condition.

Figure 34 illustrates the header transmission. The device address is transferred first, followed by the R=W flag set to 0. Depending on the access type, the transfer is continued or followed by another one:

- Write transfer: with R/$\overline{W}$ flag set to 0, is the header transmission plus following payload data.
- Read transfer: with R/$\overline{W}$ flag set to 1, assumes a preceding single header transmission without payload data. It is up to the user to either follow-up with a new header transmission or another read access.

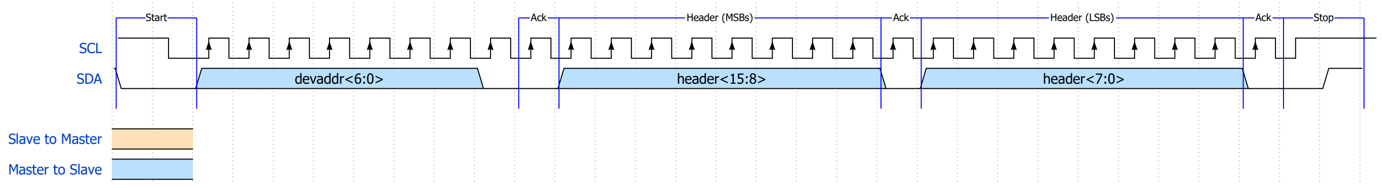


Figure 33: I²C header transfer

4.4.2.4.5 I²C data read

After the header is transferred, a read transfer can be issued by sending the slave address in read mode (R/$\overline{W}$ = ‘1’), resulting in slave address 10101XX1 (‘X’ is determined by state of CSB and MISO pin). Then the master can read an arbitrary number of 16-bit words, MSB first. The master acknowledges the reception of every byte. Data is read until a NACK (issued by the slave) followed by a stop condition occurs.

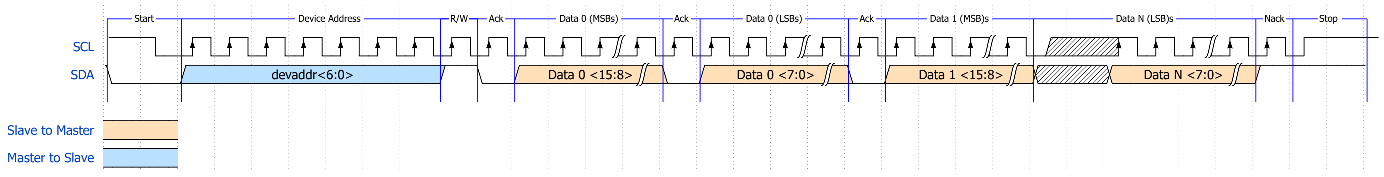


Figure 34: I²C data read

4.4.2.4.6 I²C data write

After the header is transferred, a write transfer can be issued by sending the slave address in write mode (R/$\overline{W}$ = ‘0’), resulting in slave address 10101XX0 (‘X’ is determined by state of CSB and MISO pin). Then the master can write an arbitrary number of 16-bit words, MSB first. The reception of a byte is acknowledged by the slave. The transaction is ended by a stop condition.

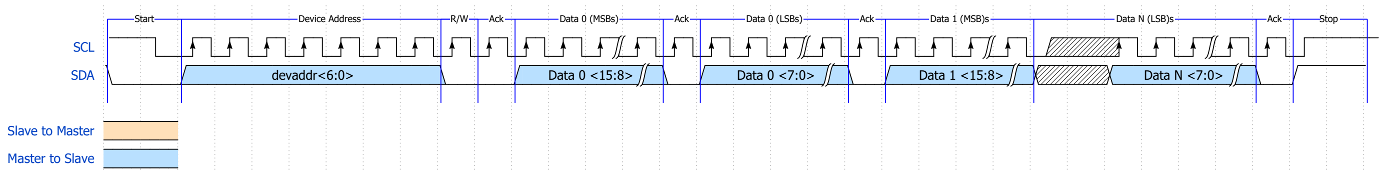


Figure 35: I²C data write

4.5 Maintenance and service

Due to its unique measurement principle, the BMV080 particulate matter sensor does not comprise an air inlet or any moving or rotating parts (like e.g., a fan), i.e., the sensor is designed to be maintenance free. Therefore, there is no need for the user to perform regular maintenance or service tasks.

However, it is possible that dust, fingerprints, or other contamination accumulates on the outer surface of the optical cover of the host system. This might lead to a slight impact on the sensor performance. Hence, it can be beneficial to clean the optical cover surface using a wiping tissue from time to time. A suitable tissue shall be used to avoid scratches on the optical cover.

5 Software integration guidelines (Sensor driver)

The BMV080 sensor driver is the interface between the sensor and the user application on the host system.

Bosch Sensortec provides sensor drivers to run on the host application processor.

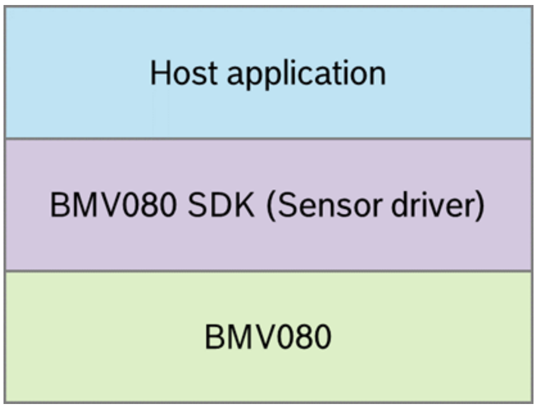


Figure 36: The sensor driver is the communication channel between the BMV080 and the user application

The sensor driver includes a complete set of ready-to-use Application Programming Interfaces (APIs) to simplify the development of a host application. Section 5.2 describes these APIs in detail. These functions can be easily used to develop a user application specific to a determined use case.

5.1 Host requirements

Table 15 shows the technical requirements for the current version of the BMV080 sensor driver.

Table 15: BMV080 sensor driver technical requirements for embedded systems

Supported platforms	**Compiler**	**ROM ***	**RAM**	
		.text + .rodata	**.bss + .data**	**.stack**
ARM Cortex-M4F	arm-none-eabi-gcc	63 kB	17 kB	11 kB
ARM Cortex-M4	arm-none-eabi-gcc	64 kB	17 kB	11 kB
ARM Cortex-M33F	arm-none-eabi-gcc	63 kB	17 kB	11 kB
ARM Cortex-M0+	arm-none-eabi-gcc	65 kB	17 kB	11 kB
ESP32	xtensa-esp32-elf-gcc	74 kB	17 kB	12 kB
ESP32-S2	xtensa-esp32s2-elf-gcc	80 kB	17 kB	12 kB
ESP32-S3	xtensa-esp32s3-elf-gcc	74 kB	17 kB	12 kB
Raspberry Pi ARMv6 32Bit	arm-linux-gnueabihf-gcc	89 kB	18 kB	18 kB
Raspberry Pi ARMv8-A 32Bit	arm-linux-gnueabihf-gcc	89 kB	18 kB	18 kB
Raspberry Pi ARMv8-A 64Bit	aarch64-linux-gnu-gcc	93 kB	18 kB	19 kB

*excluding 20 kB (approx.) for the standard library dependencies for different platforms.

Heap memory allocation is not used.

The compilation has been done in release mode with an -Os optimization flag.

The support for other platforms is being considered case-by-case. To request support for a particular embedded platform, please contact your Bosch Sensortec representative.

5.2 Application programming interface

5.2.1 Typedefs

5.2.1.1 Handles

5.2.1.1.1 bmv080_handle_t

Typedef	typedef void* **bmv080_handle_t**;
Summary	Unique handle for a sensor unit

5.2.1.1.2 bmv080_sercom_handle_t

Typedef	typedef void* **bmv080_sercom_handle_t**;
Summary	Unique handle for serial communication, i.e., the hardware connection to the sensor unit

5.2.1.2 Enumerations

5.2.1.2.1 bmv080_status_code_t

Typedef	typedef enum
	{
	    E_BMV080_OK = 0,
	    [...]
	    E_BMV080_WARNING_FIFO_FULL = 215,
	    [...]
	    E_BMV080_ERROR_NULLPTR = 100,
	    [...]
	} **bmv080_status_code_t**;

Summary	Status codes of BMV080 sensor driver

5.2.1.2.2 bmv080_measurement_algorithm_t

Typedef	typedef enum
	{
	E_BMV080_MEASUREMENT_ALGORITHM_FAST_RESPONSE = 1,
	E_BMV080_MEASUREMENT_ALGORITHM_BALANCED = 2,
	E_BMV080_MEASUREMENT_ALGORITHM_HIGH_PRECISION = 3
	} **bmv080_measurement_algorithm_t**;

Summary	Possible measurement algorithms for BMV080 sensor driver seen as follows **1.** Fast response, recommended for use-cases which require BMV080 measurements with best response time (low latency). **2.** Balanced, recommended for use-cases which require BMV080 measurements with a good balance between precision & fast response time. **3.** High precision, recommended for use-cases which require BMV080 measurements with optimum precision performance.

5.2.1.3 Structure definitions

5.2.1.3.1 bmv080_output_t

Typedef	typedef struct
	{
	    float runtime_in_sec;
	    float pm2_5_mass_concentration;
	    float pm1_mass_concentration;
	    float pm10_mass_concentration;
	    float pm2_5_number_concentration;
	    float pm1_number_concentration;
	    float pm10_number_concentration;
	    bool is_obstructed;
	    bool is_outside_measurement_range;
	    float reserved_0;
	    float reserved_1;
	    float reserved_2;
	    struct bmv080_extended_info_s *extended_info;
	} **bmv080_output_t;**

Summary	The output structure is updated by *bmv080_serve_interrupt* when sensor output is available.

name	**unit**	**description**
runtime_in_sec	s	Time passed since the start of the measurement cycle
pm2_5_mass_concentration	μg/m^3	PM2.5 mass concentration
pm1_mass_concentration	μg/m^3	PM1 mass concentration
pm10_mass_concentration	μg/m^3	PM10 mass concentration
pm2_5_number_concentration	particles/m^3	PM2.5 number concentration
pm1_number_concentration	particles/m^3	PM1 number concentration
pm10_number_concentration	particles/m^3	PM10 number concentration
is_obstructed	N/A	Flag to indicate whether the sensor is obstructed and cannot perform a valid measurement.
is_outside_measurement_range	N/A	Flag to indicate whether the PM2.5 concentration is outside the specified measurement range (0 – 1000 ug/m^3)
reserved_0	N/A	For internal use only
reserved_1	N/A	For internal use only
reserved_2	N/A	For internal use only
bmv080_extended_info_s	N/A	For internal use only

5.2.1.4 Callbacks

5.2.1.4.1 bmv080_callback_read_t

Callback	typedef int8_t(***bmv080_callback_read_t**)
	(
	  bmv080_sercom_handle_t sercom_handle,
	  uint16_t header,
	  uint16_t* payload,
	  uint16_t payload_length
	);	

Summary	Function pointer for reading an array of *payload_length* words of 16-bit *payload* All data, *header*, and *payload* are transferred as MSB first	
Precondition	Both *header* and *payload* words are 16-bit and combined. A *payload* is only transferred on a complete transmission of 16 bits. Burst transfers, i.e., reading a *header* followed by several *payload* elements, must be supported.	
Postcondition	N/A	
Arguments	sercom_handle	Handle for a serial communication interface to a specific sensor unit
	header	Header information for the following payload
	* payload	Payload to be read consisting of 16-bit words
	payload_length	payload_length Number of payload elements to be read
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.	

5.2.1.4.2 bmv080_callback_write_t

Callback	typedef int8_t(***bmv080_callback_write_t**)
	(
	  bmv080_sercom_handle_t sercom_handle,
	  uint16_t header,
	  const uint16_t* payload,
	  uint16_t payload_length
	);	

Summary	Function pointer for writing an array of *payload_length* words of 16-bit *payload* All data, *header*, and *payload* are transferred as MSB first.	
Precondition	Both *header* and *payload* words are 16-bit and combined. A payload is only transferred on a complete transmission of 16 bits. Burst transfers, i.e., reading a *header* followed by several *payload* elements, must be supported.	
Postcondition	N/A	
Arguments	sercom_handle	Handle for a serial communication interface to a specific sensor unit
	header	Header information for the following payload
	* payload	Payload to be written consisting of 16-bit words
	payload_length	Number of payload elements to be read
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.	

5.2.1.4.3 bmv080_callback_delay_t

Callback	typedef int8_t(***bmv080_callback_delay_t**) (uint32_t duration_in_ms);
Summary	Function pointer for executing a software delay operation
Precondition	N/A
Postcondition	N/A
Arguments	duration_in_ms Duration of the delay in milliseconds
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.1.4.4 bmv080_callback_data_ready_t

Callback	typedef void(***bmv080_callback_data_ready_t**) (bmv080_output_t bmv080_output, void* callback_parameters);
Summary	Function pointer for handling the sensor's output information
Precondition	N/A
Postcondition	N/A
Arguments	bmv080_output Structure containing sensor output * callback_parameters are user-defined parameters to be passed to the callback function.
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.2 Driver version

5.2.2.1 bmv080_get_driver_version

Function	bmv080_status_code_t **bmv080_get_driver_version** (uint16_t* major, uint16_t* minor, uint16_t* patch, char git_hash[12], int32_t num_commits_ahead);
Summary	Get the version information of this sensor driver
Precondition	No preconditions apply, i.e., no connected sensor unit or sensor driver handle is required.
Postcondition	N/A
Arguments	* major Major version number * minor Minor version number * patch Patch version number git_hash[12] Git hash of the build num_commits_ahead Number of commits ahead from build
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.3 Sensor management

5.2.3.1 bmv080_open

Function	bmv080_status_code_t **bmv080_open** (bmv080_handle_t* handle const bmv080_sercom_handle_t sercom_handle, const bmv080_callback_read_t read, const bmv080_callback_write_t write, const bmv080_callback_delay_t delay_ms);
Summary	Open a sensor unit by initializing a new handle. The handle must be NULL initialized.
Precondition	Must be called first to create the handle required by other functions.
Postcondition	The handle must be destroyed via bmv080_close.
Arguments	* handle — Unique handle for a sensor unit sercom_handle — Unique handle for a serial communication interface read — Function pointer for reading from an endpoint write — Function pointer for writing to an endpoint delay_ms — Function pointer for a delay in milliseconds
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.3.2 bmv080_reset

Function	bmv080_status_code_t **bmv080_reset** (const bmv080_handle_t handle);
Summary	Reset a sensor unit by performing a reset of the hardware and software.
Precondition	A valid handle generated by bmv080_open is required.
Postcondition	Any parameter changed through bmv080_set_parameter is reverted to its default.
Arguments	handle — Unique handle for a sensor unit
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.3.3 bmv080_close

Function	bmv080_status_code_t **bmv080_close** (bmv080_handle_t* handle);
Summary	Close the sensor unit.
Precondition	Must be called last to destroy the handle created by *bmv080_open.*
Postcondition	N/A
Arguments	* handle — Unique handle for a sensor unit
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.4 Sensor identification

5.2.4.1 bmv080_get_sensor_id

Function	bmv080_status_code_t **bmv080_get_sensor_id** (const bmv080_handle_t handle, char id[13]);	
Summary	Get the sensor ID of a sensor unit.	
Precondition	A valid handle generated by *bmv080_open* is required. The application must have allocated the char array id with a size of 13 elements.	
Postcondition	N/A	
Arguments	handle id	Unique handle for a sensor unit Character array of 13 elements
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.	

5.2.5 Particulate matter measurement

5.2.5.1 User application flows

This section introduces two typical types of application flow by means of examples.

5.2.5.1.1 Continuous measurement

Figure 37 presents an activity diagram illustrating the process of conducting a continuous measurement, which is an unlimited duration measurement. The application is required to execute the sub-programs (highlighted in purple) from the BMV080 library.

The measurement process begins by establishing a connection with the BMV080. Following this, measurement parameters can be set using the can be set using the bmv080_set_parameter() function.

Initially, the sensor operates in sleep mode, drawing only standby current (as detailed in Table 11). A measurement is initiated by calling bmv080_start_continuous_measurement(), which activates the continuous measurement of particle density.

The service function bmv080_serve_interrupt() fetches and processes data from the BMV080. Sensor output is provided through the callback function(), which is triggered every second and is implemented on application level.

The bmv080_serve_interrupt() function can be invoked either at regular intervals (at least once per second) or event-driven, based on an external interrupt.

The measurement process can be halted by calling bmv080_stop_measurement(). This action puts the BMV080 back into sleep mode, reducing the current consumption to standby levels.

This setup allows for the implementation of various end-user applications. For instance, PM data can be logged into a database, displayed in real-time, or streamed directly to the cloud.

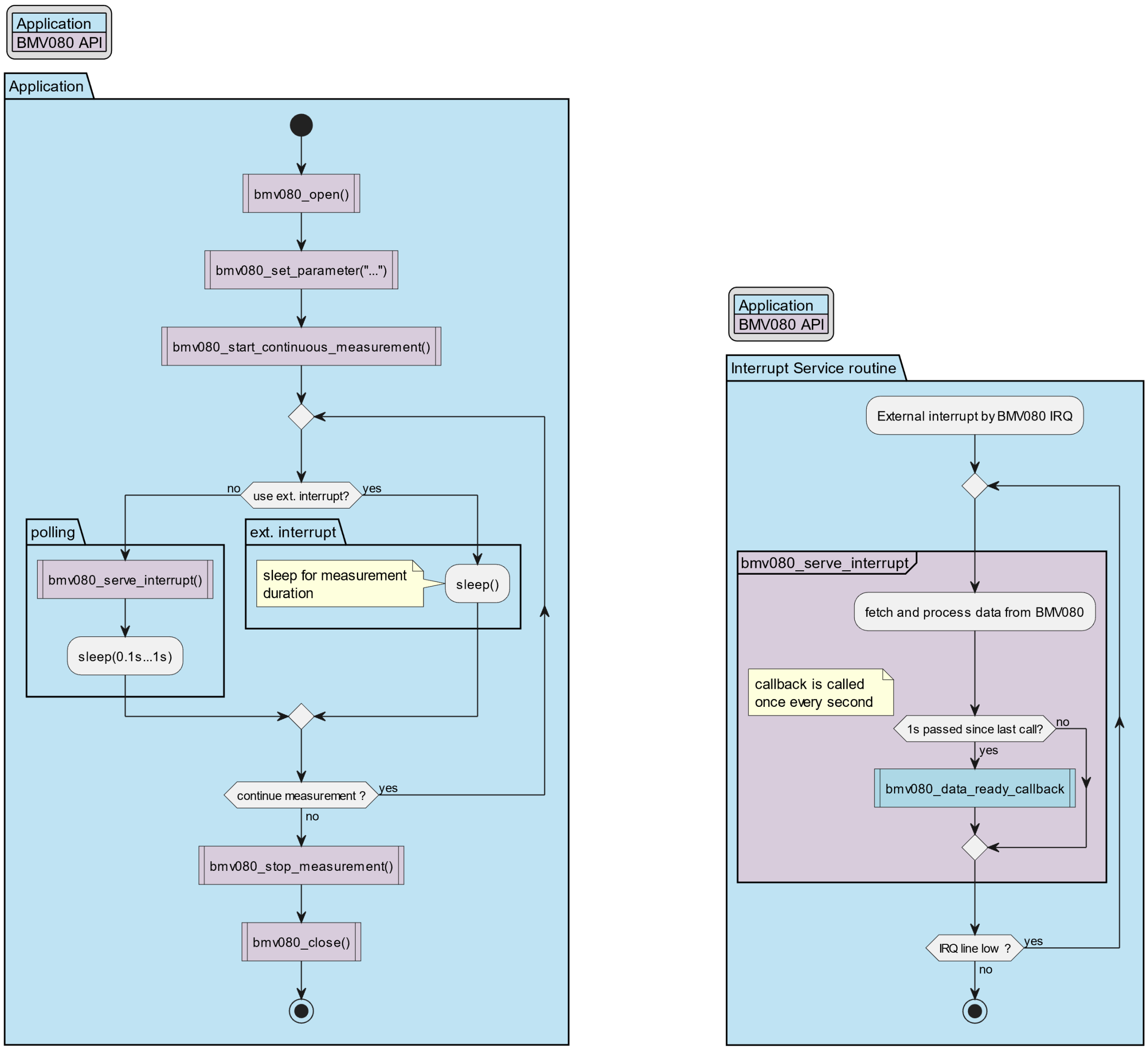

Application
BMV080 API
Application
bmv080_open()
bmv080_set_parameter("...")
bmv080_start_continuous_measurement()
use ext. interrupt?
no
yes
polling
bmv080_serve_interrupt()
sleep(0.1s...1s)
ext. interrupt
sleep for measurement duration
sleep()
continue measurement ?
yes
no
bmv080_stop_measurement()
bmv080_close()
Application
BMV080 API
Interrupt Service routine
External interrupt by BMV080 IRQ
bmv080_serve_interrupt
fetch and process data from BMV080
callback is called once every second
1s passed since last call?
no
yes
bmv080_data_ready_callback
IRQ line low ?
yes
no

Figure 37: Flow diagram of continuous measurement

A detailed explanation of each sub-program definition is in the API specification above.

Timing sequence during continuous measurement

Figure 38 depicts the timing sequence for initiating a measurement. After the measurement process is initiated by calling ‘bmv080_start_continuous_measurement()‘, the first measurement will be ready after a delay of 1.9 seconds. The sensor will then provide a new measurement every 1.03 seconds. If the IRQ line is used to trigger the ‘bmv080_serve_interrupt()‘ function, the first measurement will be ready after a longer delay of 2.9 seconds. However, subsequent measurements will continue to be available at regular intervals of 1.03 seconds.

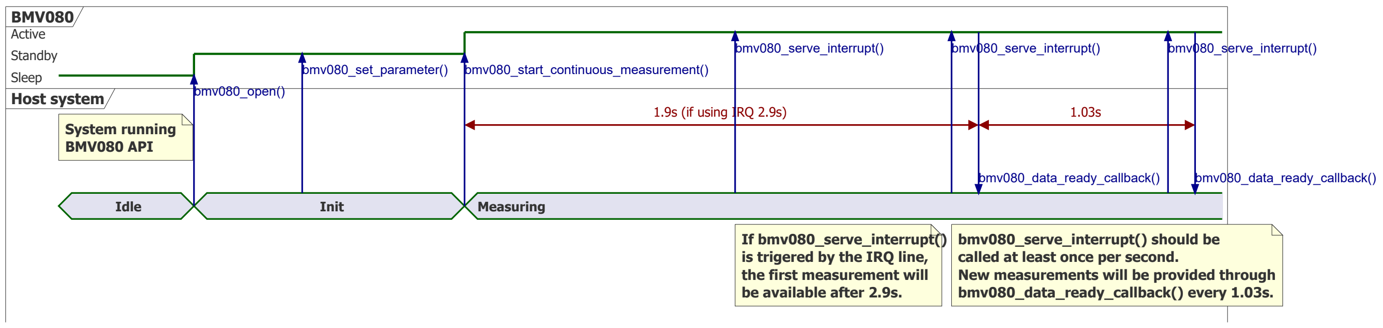

BMV080
Active
Standby
Sleep
Host system
System running BMV080 API
bmv080_open()
bmv080_set_parameter()
bmv080_start_continuous_measurement()
bmv080_serve_interrupt()
bmv080_serve_interrupt()
bmv080_serve_interrupt()
1.9s (if using IRQ 2.9s)
1.03s
bmv080_data_ready_callback()
bmv080_data_ready_callback()
Idle
Init
Measuring
If bmv080_serve_interrupt() is trigered by the IRQ line, the first measurement will be available after 2.9s.
bmv080_serve_interrupt() should be called at least once per second. New measurements will be provided through bmv080_data_ready_callback() every 1.03s.

Figure 38: Start-up sequence in case of Continuous Measurement Mode

5.2.5.1.2 Duty cycling measurement

Figure 39 is an activity diagram that shows how to perform a duty cycling measurement – repeating numerous measurements separated by a pause. The main difference from continuous measurement is the duty cycling measurement will pause before repeating the next measurement cycle.

For duty cycling measurement, the service function bmv080_serve_interrupt() must be invoked at regular intervals, at least once per second.

Note: the event-driven approach using an external interrupt is not compatible with duty cycling measurement.

The period at which new data becomes available is determined by the duty_cycling_period parameter (refer to the bmv080_set_parameter() function for more details). During the sleep time of the duty cycling period, the sensor will be in sleep mode, limiting current consumption to standby levels. For more details, see Section 2.3.2.

Data availability is signaled through the callback function bmv080_data_ready_callback(), which is triggered once every duty_cycling_period has elapsed.

This setup allows for the implementation of various end-user applications. For instance, PM data can be logged into a database, displayed in real-time, or streamed directly to the cloud.

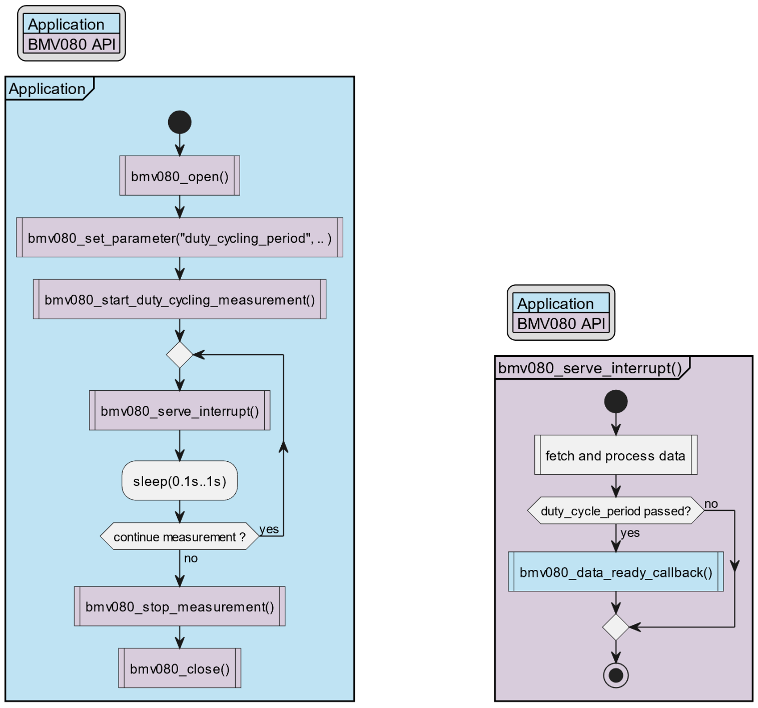


Figure 39: Flow diagram of a duty cycling measurement

Timing sequence during duty cycling measurement

Figure 40 depicts the timing sequence for initiating a measurement. After the measurement process is initiated by calling 'bmv080_start_start_duty_cycling_measurement()', the first measurement will be ready after the Integration time and an additional delay of 1.17 seconds. The sensor will then provide new measurements at the rate of the configured Duty Cycling Period.

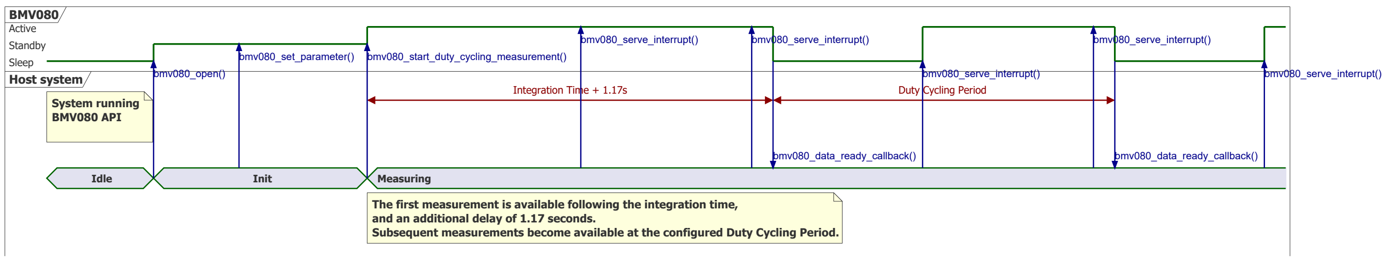

BMV080
Active
Standby
Sleep
Host system
System running BMV080 API
bmv080_open()
bmv080_set_parameter()
bmv080_start_duty_cycling_measurement()
bmv080_serve_interrupt()
Integration Time + 1.17s
Duty Cycling Period
bmv080_data_ready_callback()
Idle
Init
Measuring
The first measurement is available following the integration time, and an additional delay of 1.17 seconds. Subsequent measurements become available at the configured Duty Cycling Period.

Figure 40: Start-up sequence in case of Duty Cycling Measurement Mode

5.2.5.2 bmv080_start_continuous_measurement

Function	bmv080_status_code_t **bmv080_start_continuous_measurement** (const bmv080_handle_t handle);
Summary	Start particle measurement in continuous mode.
Precondition	A valid handle generated by *bmv080_open* is required. Optionally, parameters can be set by previous calls of *bmv080_set_parameter*.
Postcondition	The measurement mode increases energy consumption. The sensor unit stays in measurement mode until *bmv080_stop_measurement* is called. In measurement mode, *bmv080_serve_interrupt* should be called regularly.
Arguments	handle — Unique handle for a sensor unit id — Character array of 13 elements
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.5.3 bmv080_start_duty_cycling_measurement

Function	**bmv080_start_duty_cycling_measurement** (const bmv080_handle_t handle, const bmv080_callback_tick_t get_tick_ms, bmv080_duty_cycling_mode_t duty_cycling_mode);
Summary	Start particle measurement in duty cycling mode.
Precondition	A valid handle generated by *bmv080_open* is required. Optionally, duty cycling parameters (integration_time and duty_cycling_period) can be set by preceding *bmv080_set_parameter* calls.
Postcondition	The sensor unit stays in duty cycling mode until *bmv080_stop_measurement* is called. In measurement mode, *bmv080_serve_interrupt* should be called regularly.
Arguments	handle — Unique handle for a sensor unit get_tick_ms — Function pointer that provides a tick value in milliseconds (based on the host system clock) duty_cycling_mode — Mode of performing the duty cycling measurement
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.5.4 bmv080_serve_interrupt

Function	bmv080_status_code_t **bmv080_serve_interrupt** (const bmv080_handle_t handle, bmv080_callback_data_ready_t data_ready, void* callback_parameters);
Summary	Serve an interrupt using a callback function.
Precondition	A valid handle generated by bmv080_open is required with the sensor unit currently in measurement mode via *bmv080_start_continuous_measurement* or *bmv080_start_duty_cycling_measurement*. The application can call this function whenever the sensor or the application triggers an interrupt. This interrupt may be a software timeout (e.g., at least once per second) or a hardware interrupt (e.g., FIFO watermark exceeded). This function tolerates frequent, random, or rare calls to a certain extent. However, not calling bmv080_serve_interrupt over longer periods might impair the measurement mode since events could be missed. In continuous mode, new sensor output is available every second. Hence, data_ready is called once every second of the sensor unit uptime. For example, if *bmv080_serve_interrupt* was called 5 s after *bmv080_start_continuous_measurement*, the callback function data_ready would subsequently be called five times to report the collected sensor output of each period. In duty cycling mode, new sensor output is available every duty cycling period. Hence, data_ready is called at the end of the integration time, once every duty cycling period. In this case, *bmv080_serve_interrupt* must be called at least once every second. The recommendation is to call this function based on hardware interrupts. The caller application provides the callback function bmv080_callback_data_ready_t and the according callback_parameter.
Postcondition	The interrupt condition is served, e.g., FIFO is fetched, or the ASIC condition is solved.
Arguments	handle — Unique handle for a sensor unit data_ready — User-defined callback function, which is called when sensor output is available callback_parameters — User-defined parameters to be passed to the callback function
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.5.5 bmv080_stop_measurement

Function	bmv080_status_code_t **bmv080_stop_measurement** (const bmv080_handle_t handle);
Summary	Stop particle measurement.
Precondition	valid handle generated by bmv080_open is required, and the sensor unit entered measurement mode via bmv080_start_continuous_measurement or bmv080_start_duty_cycling_measurement. Must be called at the end of a data acquisition cycle to ensure that the sensor unit is ready for the next measurement cycle.
Postcondition	The sensing mode reduces energy consumption.
Arguments	handle — Unique handle for a sensor unit
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

5.2.6 Customization

5.2.6.1 bmv080_get_parameter

Function	bmv080_status_code_t **bmv080_get_parameter** (const bmv080_handle_t handle, const char* key, void* value);
Summary	Get a parameter. The table called "Parameters to get" lists the available parameters with their keys and the expected types. This function can be called multiple times and is optional.
Precondition	A valid handle generated by *bmv080_open* is required.
Postcondition	N/A
Arguments	handle — Unique handle for a sensor unit key — Key of the parameter to get value — Value of the parameter cast as void-pointer
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

Key	Type	Unit	Description
path	char*	N/A	Path to directory where log files are written. The maximum allowed length is 256 characters, which must be pre-allocated.
error_logging	bool	N/A	Enable/disable logging of error frames reported by the sensor in a CSV file. The filename format is <yyyymmddTHH-MMSS>_<SensorID>_errorFramesOutput.csv. [17]
meta_data_logging	bool	N/A	Enable/disable logging of metadata reported by the sensorin a CSV file. The filename format is <yyyymmd-dTHHMMSS>_<SensorID>_metaDataOutput.csv. [17]
pm_logging	bool	N/A	Enable/disable logging of PM data in a CSV file. The filename format is <yyyymmddTHH-MMSS>_<SensorID>_postProcessorOutput.csv. [17]
integration_time	float	s	Measurement window for computing PM value. In duty cycling mode, this also includes the sensor ON time. The default is 10 s.
duty_cycling_period	int	s	Duty cycling period (sum of integration time and sensor OFF / sleep time). Duty cycling period must be greater than integration time by at least 2 s.
do_obstruction_ detection	bool	N/A	Enable/disable obstruction notifier.
do_vibration_filtering	bool	N/A	Enable/disable vibration filtering.
measurement_ algorithm	bmv080_ measurement_ algorithm_t	N/A	Selection of measurement algorithm based on the use case. Default value is E_BMV080_MEASUREMENT_ ALGORITHM_HIGH_PRECISION. For a duty cycling measurement, this parameter is fixed to E_BMV080_ MEASUREMENT_ ALGORITHM_FAST_RESPONSE.

5.2.6.2 bmv080_set_parameter

Function	bmv080_status_code_t **bmv080_set_parameter**
	(
	   const bmv080_handle_t handle,
	   const char* key,
	   void* value
	);

Summary	Set a parameter. The table “Parameters to set” lists the available parameters with their keys and the expected types. This function can be called multiple times and is optional.
Precondition	valid handle generated by *bmv080_open* is required. This function must be called before *bmv080_start_continuous_measurement* or *bmv080_start_duty_cycling_measurement* in order to apply the parameter in the configuration of particle measurement.
Postcondition	N/A
Arguments	handle — Unique handle for a sensor unit key — Key of the parameter to set value — Value of the parameter cast as void-pointer
Return Value	E_BMV080_OK if successful. Otherwise, the return value is a BMV080 status code.

Key	**Type**	**Unit**	**Description**
path	char*	N/A	Path to directory where log files are written. The maximum allowed length is 256 characters, which must be pre-allocated.
error_logging	bool	N/A	Enable/disable logging of error frames reported by the sensor in a CSV file. The filename format is <yyyymmddTHH-MMSS>_<SensorID>_errorFramesOutput.csv. [17]
meta_data_logging	bool	N/A	Enable/disable logging of metadata reported by the sensorin a CSV file. The filename format is <yyyymmd-dTHHMMSS>_<SensorID>_metaDataOutput.csv. [17]
pm_logging	bool	N/A	Enable/disable logging of PM data in a CSV file. The filename format is <yyyymmddTHH-MMSS>_<SensorID>_postProcessorOutput.csv. [17]
integration_time	float	s	Measurement window for computing PM value. In duty cycling mode, this also includes the sensor ON time. The default is 10 seconds.
duty_cycling_period	int	s	Duty cycling period (sum of integration time and sensor OFF / sleep time). Duty cycling period must be greater than integration time by at least 2 seconds. The default is 30 seconds.
do_obstruction_ detection	bool	N/A	Enable/disable obstruction notifier.
do_vibration_filtering	bool	N/A	Enable/disable vibration filtering.
measurement_ algorithm	bmv080_ measurement_ algorithm_t	N/A	Selection of measurement algorithm based on the use case. Default value is E_BMV080_MEASUREMENT_ ALGORITHM_HIGH_PRECISION. For a duty cycling measurement, this parameter is fixed to E_BMV080_ MEASUREMENT_ ALGORITHM_FAST_RESPONSE.

[17] File logging is available only for Windows x86/x64 and Raspberry Pi, but not for embedded platforms (e.g., ARM Cortex-M, Xtensa ESP32, etc).

6 Traceability

A laser marking (DMC and OCR code) on the sensor is used to identify the single parts and to enable traceability through the production and supply chain.

Table 16: BMV080 marking convention

General			
Marking method		Laser (Direct part marking)	
Marking position		Refer to drawing	
Information about DMC			
Marking type		DMC, ECC 200 (ISO/IEC 16022)	
Module size		0.130 ± 0.01 mm	
Symbol size		14 x 14 (Row × column) nominal 1.82 mm × 1.82 mm	
Marking depth		< 15 µm	
Marking quality (Grade)		A-C (ISO/IEC TR 29158)	
Information capacity DMC		16 (Numeric)	Digit 1, digit 2, ..., digit 16
DMC sequence	Digits 1 – 7		Reserved
	Digit 8	0 = 2020 1 = 2021 ⋮ 9 = 2029	Year
	Digits 9 – 10	01 ~ 53	Work week
	Digit 11	1 = Sun 2 = Mon ⋮ 7 = Sat	Day
	Digits 12 – 16		Reserved
Information about marking text			
Font type		Arial1	
Symbol size		2.15 mm × 0.4 mm	
Information capacity marking text		6 (Numeric)	Digit 1, digit 2, ..., digit 6
Marking text sequence	Digit 1	0 = 2020 1 = 2021 ⋮ 9 = 2029	Year
	Digits 2 – 3	01 ~ 53	Work Week
	Digit 4	1 = Sun 2 = Mon ⋮ 7 = Sat	Day
	Digits 5 – 6		Reserved

7 Product compliance

7.1 Environmental safety

7.1.1 RoHS

BMV080 is in compliance with the consolidated RoHS directive 2011/65/EU of the European Parliament and Council regarding the restriction of hazardous substances in electrical and electronic equipment.

7.1.2 Halogen content

BMV080 complies with the halogen-free definition of the industry standard IEC 61249 for materials utilized in printed boards and other interconnecting structures.

7.2 Laser safety

The design of the BMV080 limits the emitted optical output power to 0.5 mW. The BMV080 is laser class 1 compliant.

7.2.1 Conformity and classification

BMV080 is

- laser class 1 compliant according to norm: “Safety of laser products –Part 1: Equipment classification and requirements”, IEC60825-1 Edition 3.0 (2014), respectively, EN 60825-1:2014/A11:2021.
- a consumer product according to norm “Safety of laser products – Particular Requirements for Consumer Laser Products”, EN 50689 (2021).
- compliant with FDA performance standards for laser products except for conformance with IEC 60825-1 Edition 3.0, as described in Laser Notice No. 56, dated May 8, 2019.

This has independently been confirmed by Seibersdorf Laboratories (Test report LE-L176/23, 2024-02-28). The Laser Class 1 classification was achieved by using C-samples on EvalKits as application setup.

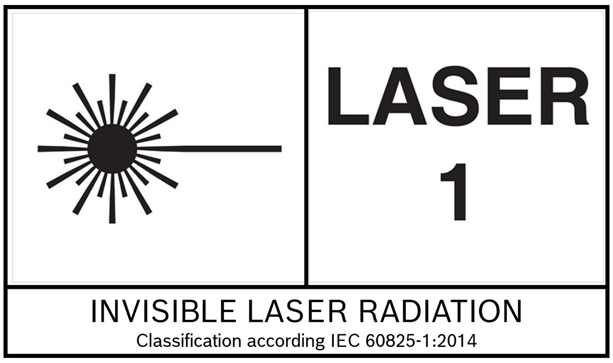

LASER
1
INVISIBLE LASER RADIATION
Classification according IEC 60825-1:2014

Figure 41: Class 1 laser product

8 Additional material

Additional material related to the BMV080 includes the following design files and software packages:

Sensor integration

3D CAD file of BMV080: BST-BMV080-CAD.stp

Sensor software

BMV080 software development kit: BST-BMV080-SDK

BMV080 EvalKit

EvalKit Software (Windows) for EvalKit: BST-BMV080-EvalKit-SW

9 Legal disclaimer

9.1 Engineering samples

Engineering Samples are marked with an asterisk (*), (E) or (e). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

9.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product data sheet. They are not fit for use in life-sustaining or safety-critical systems. Safety-critical systems are those for which a malfunction is expected to lead to bodily harm, death or severe property damage. In addition, they shall not be used directly or indirectly for military purposes (including but not limited to nuclear, chemical or biological proliferation of weapons or development of missile technology), nuclear power, deep sea or space applications (including but not limited to satellite technology).

Bosch Sensortec products are released on the basis of the legal and normative requirements relevant to the Bosch Sensortec product for use in the following geographical target market: BE, BG, DK, DE, EE, FI, FR, GR, IE, IT, HR, LV, LT, LU, MT, NL, AT, PL, PT, RO, SE, SK, SI, ES, CZ, HU, CY, US, CN, JP, KR, TW. If you need further information or have further requirements, please contact your local sales contact.

The resale and/or use of Bosch Sensortec products are at the purchaser's own risk and his own responsibility. The examination of fitness for the intended use is the sole responsibility of the purchaser.

The purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The purchaser accepts the responsibility to monitor the market for the purchased products, particularly with regard to product safety, and to inform Bosch Sensortec without delay of all safety-critical incidents.

9.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.

10 Document history and modifications

Rev. No.	Chapter	Description of modification/changes	Date
1.0	all	Initial release	Aug 2024
1.1	2.2 3.2 4.4.1.2 4.4.2.4 5.1 5.2 6 7.2	Updated technical specifications and lifetime Comparison of pin numbering schemes for BMV080 and ZIF connectors added Proposal for filtering signal errors added Corrected image for complete I²C write Host requirements updated PM1 definition added Updated content of DMC and laser marking Updated information on laser class compliance	Dec 2024
1.2	1.1 5.1 5.2.1.3 5.2.6	Update with PM10 and PM1 Updated host requirements PM10 mass concentration definition added; PM2.5, PM1, PM10 number concentration definition added Update on error logging, meta data logging and PM logging in get and set parameter definition	Jan 25

Bosch Sensortec GmbH
Gerhard-Kindler-Strasse 9
72770 Reutlingen / Germany

contact@bosch-sensortec.com
www.bosch-sensortec.com

Modifications reserved
Document number: BST-BMV080-DS000-10