

|  |  |
| --- | --- |
| **COINES – documentation and API description** | |
| Document revision | 1.6 |
| Release date | September 2018 |
| Document number | BST-COINES-SD001-06 |
|  |  |
| Notes | Content in this document is subject to change without notice.  Product photos and pictures are for illustration purposes only and may differ from the real product’s appearance. |

**Contents**

[1. Introduction 4](#_Toc520808412)

[2. Installation (Windows) 4](#_Toc520808413)

[2.1. System requirements 4](#_Toc520808414)

[2.2. Installation of COINES 4](#_Toc520808415)

[2.3. Installation of compiler environment 4](#_Toc520808416)

[2.4. Installation of Python environment 5](#_Toc520808417)

[3. Installation (Linux) 5](#_Toc520808418)

[3.1. System requirements 5](#_Toc520808419)

[3.2. Installation of COINES 5](#_Toc520808420)

[3.3. Installation of compiler environment 6](#_Toc520808421)

[3.4. Installation of Python environment 6](#_Toc520808422)

[4. Quick start guide 6](#_Toc520808423)

[4.1. Compiling and testing samples 6](#_Toc520808424)

[5. coinesAPI description 7](#_Toc520808425)

[5.1. Overview of the architecture 7](#_Toc520808426)

[5.2. Folder structure 8](#_Toc520808427)

[5.3. coinesAPI calls: Interface and board information 9](#_Toc520808428)

[coines\_open\_comm\_intf 9](#_Toc520808429)

[coines\_close\_comm\_intf 9](#_Toc520808430)

[coines\_get\_board\_info 9](#_Toc520808431)

[5.4. coinesAPI calls: GPIO oriented calls 9](#_Toc520808432)

[coines\_set\_pin\_config 9](#_Toc520808433)

[coines\_get\_pin\_config 10](#_Toc520808434)

[coines\_set\_shuttleboard\_vdd\_vddio\_config 10](#_Toc520808435)

[5.5. coinesAPI calls: Sensor communication 10](#_Toc520808436)

[coines\_config\_i2c\_bus 10](#_Toc520808437)

[coines\_config\_spi\_bus 10](#_Toc520808438)

[coines\_write\_i2c 10](#_Toc520808439)

[coines\_read\_i2c 11](#_Toc520808440)

[coines\_write\_spi 11](#_Toc520808441)

[coines\_read\_spi 11](#_Toc520808442)

[coines\_delay\_msec 11](#_Toc520808443)

[5.6. coinesAPI calls: Streaming feature 12](#_Toc520808444)

[coines\_config\_streaming 12](#_Toc520808445)

[coines\_start\_stop\_streaming 13](#_Toc520808446)

[coines\_read\_stream\_sensor\_data 13](#_Toc520808447)

[coines\_trigger\_timer 14](#_Toc520808448)

[6. Hardware description (APP2.0) 14](#_Toc520808449)

[7. Protocol description 15](#_Toc520808450)

[8. FAQ 15](#_Toc520808451)

[9. Legal disclaimer 16](#_Toc520808452)

[9.1. Engineering samples 16](#_Toc520808453)

[9.2. Product usage 16](#_Toc520808454)

[9.3. Application examples and hints 17](#_Toc520808455)

[10. Document history and modification 18](#_Toc520808456)

# Introduction

COINES ("**CO**mmunication with **IN**ertial and Environmental **S**ensors") provides a low-level interface to Bosch Sensortec's Application Board APP2.0. The user can access Bosch Sensortec's MEMS sensors through a C or a Python interface. COINES can be used along with SensorAPI of the sensor.

SensorAPI is available at <https://github.com/BoschSensortec> . Source code of sample applications and SensorAPI are provided with COINES library as a package. User can modify, compile and run the sample applications.

COINES can be used to see how to use the SensorAPI in an embedded environment and allows convenient data logging.

The full working environment consists of:

* A Bosch Sensortec MEMS sensor on a shuttle board mounted on the socket of:
* Bosch Sensortec's application board APP2.0
* PC (Windows 7/10, Linux) to which the APP2.0 is connected via USB
* COINES software release as found here: <http://www.bosch-sensortec.com>
* C compiler and/or a Python interpreter is also required (details see below)

# Installation (Windows)

## System requirements

COINES should be usable on any recent PC or laptop system which has at least a performance as an “office PC”. The hardware should provide a USB 2.0 interface.

COINES is developed and tested on Windows 7, 8 and 10 systems.

## Installation of COINES

* Run the Installer
* Accept the **End User License Agreement** and click **Next**
* Click **Install** to start Installation
* Click **Start --> All programs --> COINES --> Examples** to view examples

## Installation of compiler environment

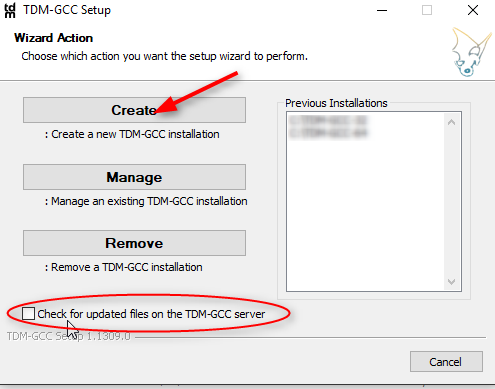
COINES C examples can be built using GNU C compiler (GCC). There are various distributionsof GCC**.** TDM-GCC is easy to install and hence preferred for COINES. TDM GCC is based on MinGW GCC.

**1. Download the TDM32/TDM64 bundle (**[**link**](http://tdm-gcc.tdragon.net/)**)**

Use TDM32 bundle if your Windows OS is 32-bit and TDM64 bundle if 64-bit

**2. Start the Installer**

Ensure that the option ***Check for updated files on the TDM GCC*** server is unchecked. Click Create and proceed with the installation



## Installation of Python environment

The Python environment used for COINES under Windows will be Anaconda [[link](https://www.anaconda.com/)]. However, any other Python 3.x environment can also be used. Installation instructions will follow once Python support is added to COINES.

# Installation (Linux)

## System requirements

COINES should be usable on any recent PC or laptop system, which has at least a performance as an “office PC”. The hardware should provide a USB 2.0 interface.

COINES was developed and tested on Ubuntu 16.04.3 LTS distribution. The following instructions apply to such an environment. COINES was also tested in Ubuntu 14.04, CentOS 7 ,Fedora 27 and Debian Jessie/Stretch (Raspberry Pi 3 hardware)

## Installation of COINES

1. Download the installer.
2. Open a terminal/console, change directory to the directory where the installer is (and where you want to have COINES installed) and make the installer executable:

$ chmod +x coines\_v0.7-beta.sh

1. Ensure that you are connected to the Internet before running the installer.

$ ./coines\_v0.7-beta.sh

1. Accept the End User License agreement
2. The installer will prompt you if the required dependencies/packages are not installed. ( This step requires root rights )
3. Install the optional COINES Code Editor if required.

## Installation of compiler environment

In the standard Linux distribution, the compiler environment is almost fully available. If the installer was not able to download the packages, the user should manually install **libusb-dev** and **libudev-dev** in case of Debian. In case of Redhat based distros like Fedora or CentOS install **libusbx-devel** and **systemd-devel**.

## Installation of Python environment

The python environment required to run COINES is already available in the standard Ubuntu distribution.

# Quick start guide

After installation, connect the APP 2.0 with shuttle board mounted example see the output

## Compiling and testing samples

Open command prompt in Windows or terminal in Linux, change directory into one of the sample files. Build the COINES examples using **mingw32-make** (in Windows) **and make** (in Linux)

bosch@bosch-vm:~/COINES/v0.7/examples/c/bmi08x/read\_chip\_id $ make  
Platform: Linux  
[ MKDIR ] build  
[ CC ] bmi08x\_chipid.c  
[ CC ] /home/bosch/COINES/v0.7/sensorAPI/bmi08x/bmi08a.c  
[ CC ] /home/bosch/COINES/v0.7/sensorAPI/bmi08x/bmi08g.c  
[ CC ] /home/bosch/COINES/v0.7/sensorAPI/bmi08x/bmi085.c  
[ CC ] /home/bosch/COINES/v0.7/sensorAPI/bmi08x/bmi088.c  
[ LD ] bmi08x\_chip\_id

Use **mingw32-make** from Windows command prompt

C:\COINES\v0.7\examples\c\bmi08x\read\_chip\_id> mingw32-make

Platform: Windows

gcc: "C:\TDM-GCC-64\bin\gcc.exe".

[ MKDIR ] build

[ CC ] bmi08x\_read\_chip\_id.c

[ CC ] c:/COINES/v0.7/sensorAPI/bmi08x/bmi08a.c

[ CC ] c:/COINES/v0.7/sensorAPI/bmi08x/bmi08g.c

[ CC ] c:/COINES/v0.7/sensorAPI/bmi08x/bmi085.c

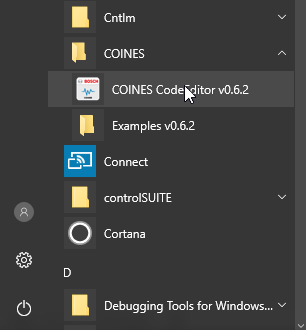
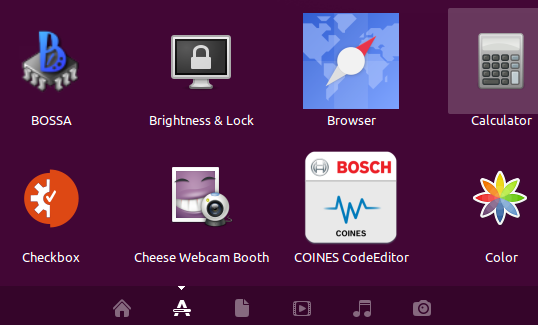
[ CC ] c:/COINES/v0.7/sensorAPI/bmi08x/bmi088.c

[ LD ] bmi08x\_read\_chip\_id.exe

Connect the **BST APP2.0 board** via USB within this example – the BMI08x shuttle board mounted.

bosch@bosch-vm:~/COINES/v0.7/examples/c/bmi08x/read\_chip\_id$./bmi08x\_chip\_id  
Accel chip ID - 1f  
Gyro chip ID - f

You can also use the **COINES Code Editor**  from start menu to edit, build and run the example programs. Check **Help -> Code Editor Quick Start Guide** in **COINES Code Editor** to learn more

**Windows 10 Start Menu Ubuntu 16.04 Application Menu**

# coinesAPI description

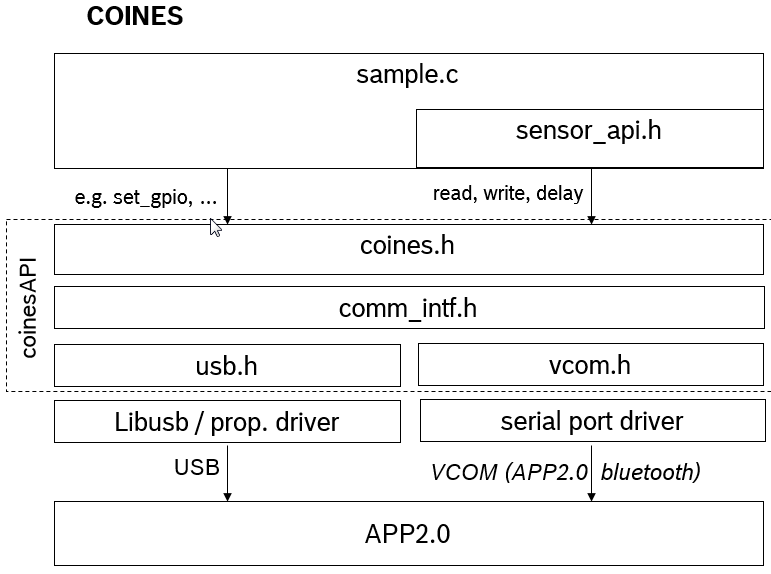
## Overview of the architecture

Bosch Sensortec recommends using the SensorAPI in order to communicate with the sensors [[link](https://github.com/BoschSensortec) ]. The SensorAPI is written in C and is an abstraction layer, which makes it much more convenient for the user to access the register map of the sensor in order to configure certain functionality and obtain certain information from it.

For making use of the API, 3 function pointers must be set to the appropriate read/write functions of the selected bus on the system (either I2C or SPI), as well as one function pointer to a system's function causing delays in milliseconds.

In order to execute C code using SensorAPI on a PC, the coinesAPI provides the mentioned read/write/delay functions. These functions are wrapper functions, embedding the actual SensorAPI payloads into a transport package, sending this via USB/VCOM to the APP2.0, where the payload is translated into corresponding SPI or I2C messages and sent to the sensor on the shuttle board.

Using this method, the full functionality of the SensorAPI can be used on PC side, sample code can be modified and tested, and data can be logged in a convenient way.

This setup has the challenge of lacking the real-time capabilities known from a pure microcontroller environment. To overcome this, the coinesAPI offers streaming functions, which allow the user to schedule data readout directly on the microcontroller, either based on a data interrupt coming from the sensors or based on the timer of the microcontroller. The scheduler waits for the configured interrupt (sensor interrupt or timer interrupt) and reads out areas of the register map, which can be configured by the user.

As an example, the user could choose to read out the 6 bytes from the register map of a certain inertial sensor, containing the sensor data of three axis (2 bytes per axis). If the user would configure for example a readout once per milliseconds, the result would be a data stream of three-axis sensor data at a rate of 1 kHz.

## Folder structure

COINES core library can be found in **<COINES\_INSTALL\_PATH>\coinesAPI** folder

Examples files can be found in **<COINES\_INSTALL\_PATH>\examples**.

The example files are also available in a compressed archive. If the user has modified the example code and needs to revert it, he can take the code from the archives.

## coinesAPI calls: Interface and board information

### coines\_open\_comm\_intf

Open the communication interface, either COINES\_COMM\_INTF\_USB (USB connection) or COINES\_COMM\_INTF\_VCOM port (serial port or virtual COM port).

int16\_t coines\_open\_comm\_intf(enum coines\_comm\_intf intf\_type);

### coines\_close\_comm\_intf

Close the communication interface.

int16\_t coines\_close\_comm\_intf(enum coines\_comm\_intf intf\_type);

### coines\_get\_board\_info

With this call, the user can obtain some board information.

int16\_t coines\_get\_board\_info(struct coines\_board\_info \*data);

The data structure contains the following items

struct coines\_board\_info {

/\*!Board hardware ID \*/

uint16\_t hardware\_id;

/\*!Board software ID \*/

uint16\_t software\_id;

/\*!Type of the board like APP2.0, Arduino Due\*/

uint8\_t board;

/\*!Shuttle ID of the sensor connected\*/

uint16\_t shuttle\_id;

};

## coinesAPI calls: GPIO oriented calls

### coines\_set\_pin\_config

Set the pin direction (PIN\_DIRECTION\_IN or PIN\_DIRECTION\_OUT) and the state (COINES\_PIN\_VALUE\_HIGH or COINES\_PIN\_VALUE\_LOW) for the pin with the pin number pin\_number (for details on pin numbering, see [Hardware description-APP2.0](#_Hardware_description_(APP2.0)) ) .

int16\_t coines\_set\_pin\_config(enum coines\_multi\_io\_pin pin\_number, enum coines\_pin\_direction direction, enum coines\_pin\_value pin\_value);

### coines\_get\_pin\_config

Gets the configuration of the pin with the number pin\_number.

int16\_t coines\_get\_pin\_config(enum coines\_multi\_io\_pin pin\_number, enum coines\_pin\_direction \*pin\_direction, enum coines\_pin\_value \*pin\_value);

### coines\_set\_shuttleboard\_vdd\_vddio\_config

This API is used to configure the VDD and VDDIO of the sensor. For APP2.0, a voltage level of 0 or 3300 mV is supported. Any values above 0 will default to 3300 mV.

int16\_t coines\_set\_shuttleboard\_vdd\_vddio\_config(uint16\_t vdd\_millivolt, uint16\_t vddio\_millivolt);

## coinesAPI calls: Sensor communication

### coines\_config\_i2c\_bus

Configures the I2C bus.

int16\_t coines\_config\_i2c\_bus(enum coines\_i2c\_bus bus, enum coines\_i2c\_mode i2c\_mode);

The first argument refers to the bus on the board. Currently, on APP2.0, there is only one bus available, so the argument is always COINES\_I2C\_BUS\_0.

The following I2C modes are available:

* COINES\_I2C\_STANDARD\_MODE
* COINES\_I2C\_FAST\_MODE
* COINES\_I2C\_SPEED\_3\_4\_MHZ
* COINES\_I2C\_SPEED\_1\_7\_MHZ

### coines\_config\_spi\_bus

Configures the SPI bus of the board. The argument coines\_spi\_bus refers to the bus on the board. On APP2.0, there is only one bus available, so the user should only use COINES\_SPI\_BUS\_0. The SPI speed can be chosen in various discrete steps, as defined in enum coines\_spi\_speed in coines.h. (For example, COINES\_SPI\_SPEED\_2\_MHZ sets the SPI speed to 2 MHz.)

int16\_t coines\_config\_spi\_bus(enum coines\_spi\_bus bus, uint32\_t spi\_speed, enum coines\_spi\_mode spi\_mode);

### coines\_write\_i2c

This API is used to write the data in I2C (bus COINES\_I2C\_BUS\_0) communication.

int8\_t coines\_write\_i2c(uint8\_t dev\_addr, uint8\_t reg\_addr, uint8\_t \*reg\_data, uint16\_t count);

Arguments:

* dev\_addr : Device address for I2C write.
* reg\_addr : Starting address for writing the data.
* reg\_data : Data to be written.
* count : Number of bytes to write.

### coines\_read\_i2c

This API is used to read the data in I2C (bus COINES\_I2C\_BUS\_0) communication.

int8\_t coines\_read\_i2c(uint8\_t dev\_addr, uint8\_t reg\_addr, uint8\_t \*reg\_data, uint16\_t count);

Arguments:

* dev\_addr : Device address for I2C read.
* reg\_addr : Starting address for reading the data.
* reg\_data : Buffer to take up the read data.
* count : Number of bytes to read.

### coines\_write\_spi

This API is used to write the data in SPI communication (bus COINES\_SPI\_BUS\_0).

int8\_t coines\_write\_spi(uint8\_t dev\_addr, uint8\_t reg\_addr, uint8\_t \*reg\_data, uint16\_t count);

Arguments:

* dev\_addr : Chip select pin number for SPI write.
* reg\_addr : Starting address for writing the data.
* reg\_data : Data to be written.
* count : Number of bytes to write.

### coines\_read\_spi

This API is used to read the data in SPI communication (bus COINES\_SPI\_BUS\_0).

int8\_t coines\_read\_spi(uint8\_t dev\_addr, uint8\_t reg\_addr, uint8\_t \*reg\_data, uint16\_t count);

Arguments:

* dev\_addr : Chip select pin number for SPI read.
* reg\_addr : Starting address for reading the data.
* reg\_data : Buffer to take up the read data.
* count : Number of bytes to read.

### coines\_delay\_msec

This API is used introduce delay in terms of millisceconds.

void coines\_delay\_msec(uint32\_t delay\_ms);

## coinesAPI calls: Streaming feature

### coines\_config\_streaming

This API is used to send the streaming settings to the board.

int16\_t coines\_config\_streaming(uint8\_t channel\_id, struct coines\_streaming\_config \*stream\_config, struct coines\_streaming\_blocks \*data\_blocks);

The channel\_id is an identifier for the stream, it must be an integer number, **starting from 1** for the first stream and be incremented stream by stream.

The stream\_config structure and the coines\_streaming\_blocks structure contain the streaming configuration. However, based on what the user is going to do, not all parameters need to be set.

* Always mandatory to set:
  + data\_block.no\_of\_blocks : number of blocks to stream (must at least be one)
  + For each block b:
    - data\_block.reg\_start\_addr[b] : start address of the block in the register map
    - stream\_block.no\_of\_data\_bytes[b] : number of addresses to read, starting from the start address
* Mandatory to set if sensor data is read from i2c bus:
  + stream\_config.intf = COINES\_SENSOR\_INTF\_I2C;
  + stream\_config.i2c\_bus : I2C bus (in case of APP2.0, this is always COINES\_I2C\_BUS\_0)
  + stream\_config.dev\_addr : I2C address of the sensor
* Mandatory to set if sensor data is read from SPI bus:
  + stream\_config.intf = COINES\_SENSOR\_INTF\_SPI;
  + stream\_config.spi\_bus : SPI bus (in case of APP2.0, this is always COINES\_SPI\_BUS\_0)
  + stream\_config.cs\_pin : CS pin of the sensor, information can be obtained from the shuttle board documentation for the sensor.
* Mandatory to set if polling mode is requested:
  + stream\_config.sampling\_units : either milliseconds (COINES\_SAMPLING\_TIME\_IN\_MILLI\_SEC) or microseconds (COINES\_SAMPLING\_TIME\_IN\_MICRO\_SEC)
  + stream\_config.sampling\_time : sampling period in the unit as defined in stream\_config.sampling\_units
* Mandatory to set if interrupt mode is requested:
  + stream\_config.int\_pin : pin of the interrupt which shall trigger the sensor read-out. If the interrupt output of the sensor is used, the required information about the pin number can be obtained from the shuttle board documentation for the sensor.
  + stream\_config.int\_timestamp : it can be configured if the sensor data is tagged with a timestamp (COINES\_TIMESTAMP\_ENABLE) or not (COINES\_TIMESTAMP\_DISABLE).

### coines\_start\_stop\_streaming

This API is used to start or stop the streaming.

int16\_t coines\_start\_stop\_streaming(enum coines\_streaming\_mode stream\_mode, enum coines\_stream\_samples samples, uint8\_t start\_stop);

Arguments:

* stream\_mode: streaming mode (either COINES\_STREAMING\_MODE\_POLLING or COINES\_STREAMING\_MODE\_INTERRUPT)
* samples : number of samples to receive, can be one (COINES\_STREAM\_ONE\_SAMPLE), two (COINES\_STREAM\_TWO\_SAMPLES) or infinite (COINES\_STREAM\_INFINITE\_SAMPLES) number of samples
* start\_stop : flag to either start (COINES\_STREAMING\_START) or stop (COINES\_STREAMING\_STOP) the streaming

### coines\_read\_stream\_sensor\_data

This API is used to read the data being streamed from the sensor.

int16\_t coines\_read\_stream\_sensor\_data(uint8\_t sensor\_id, uint32\_t number\_of\_samples, uint8\_t \*data, uint32\_t \*valid\_samples\_count);

Arguments:

* sensor\_id: id of the sensor
* number\_of\_samples : number of samples the user wishes to read
* data : data buffer ( packet counter + register data + timestamp )
* valid\_samples\_count : number of samples the user has actually received (may be less than number\_of\_samples)

0 x

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| byte3\_c | byte2\_c | byte1\_c | byte0\_c | rp |  | rp+a |  |  | rq | rq+1 |  | rq+j | byte5\_t | byte4\_t | byte3\_t | byte2\_c | byte1\_t | byte0\_t |

32-bit packet counter register block-0 register block-n 48-bit timestamp (if enabled)

1-valid sample

packet\_counter = (byte3\_c << 24) | (byte2\_c << 16) | (byte1\_c << 8) | (byte0\_c)

timestamp = (byte5\_t << 40) | (byte4\_t << 32) | (byte3\_t << 24) | (byte2\_t << 16) | (byte1\_t << 8) | (byte0\_t)

48-bit timestamp is enabled by using coines\_trigger\_timer(COINES\_TIMER\_START, COINES\_TIMESTAMP\_ENABLE);

rp -> Value at register address p

a -> Size of register block – 0

rp+a -> Value at register address (p+a)

Similarly is the case for rq , j and rq+j

See coines\_streaming\_blocks structure for information regarding register blocks.

Timestamp in microseconds can be obtained using below formula,

### coines\_trigger\_timer

This API is used to trigger the timer in firmware and enable or disable the time stamp feature

int16\_t coines\_trigger\_timer(enum coines\_timer\_config tmr\_cfg,enum coines\_time\_stamp\_config ts\_cfg);

Arguments:

* tmr\_cfg : start, stop or reset the timer (COINES\_TIMER\_START, COINES\_TIMER\_STOP or COINES\_TIMER\_RESET)
* ts\_cfg : Enables/disables microcontroller timestamp (COINES\_TIMESTAMP\_ENABLE or COINES\_TIMESTAMP\_DISABLE)

# Hardware description (APP2.0)

The application board APP2.0 is built for demonstration and evaluation of Bosch Sensortec sensors. It has the following features:

* Establishment of communication between sensor and PC
* Detection which sensor is plugged in when using Bosch Sensortec's shuttle boards (each has a unique ID)
* Can be used in combination with the evaluation software Development Desktop2.0

A detailed description of the application board can be found on Bosch Sensortec's web page. The following description will only cover the most relevant aspects in combination with COINES.

**GPIO addressing**

The shuttle board has total 28 pins, of which some have a predefined functionality and some can be used as GPIO by the user.

The shuttle board connector details are given in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Pin number on shuttle board | Name / function | Pin number on shuttle board | Name / function |
| 1 | VDD (3.3V) | **28** | SHTLE\_COD #4 |
| 2 | VDDIO (3.3V) | **27** | SHTLE\_COD #3 |
| 3 | GND | **26** | SHTLE\_COD #2 |
| 4 | SPI MISO | **25** | SHTLE\_COD #1 |
| 5 | SPI: MOSI / I2C: SDA | **24** | SHTLE\_COD #0 |
| 6 | SPI: SCK / I2C: SCK | **23** | SHTLE\_COD\_GND |
| 7 | SPI: CS | **22** | IO\_4 ( GPIO #4 ) |
| 8 | IO\_5 ( GPIO #5 ) | **21** | IO\_7 ( GPIO #7 ) |
| 9 | IO\_0 ( GPIO #0 ) | **20** | IO\_6 ( GPIO #6 ) |
| 10 | SHTLE\_COD #5 | **19** | IO\_8 ( GPIO #8 ) |
| 11 | SHTLE\_COD #6 | **18** | SCL (see note) |
| 12 | SHTLE\_COD #7 | **17** | SDA (see note) |
| 13 | SHTLE\_COD #8 | **16** | IO\_3 ( GPIO #3 ) |
| 14 | IO\_1 ( GPIO #1 ) | **15** | IO\_2 ( GPIO #2 ) |

Notes:

* In coinesAPI the pins are addressed using the same numbers as on the shuttle board. For example, the GPIO #5 has the pin number 8.
* In some cases (depending on the sensor), the I2C lines are shuttle board pin 6 for the clock signal SCL and shuttle board pin 5 for the data line SDA. In such cases pins 17 and 18 may not be connected. Please carefully read the shuttle board documentation.

# Protocol description

To be added once VCOM implementation is available

# FAQ

1. I have Development Desktop installed on my Windows PC and COINES doesn't work

* Development Desktop uses USB driver from Thesycon GmbH and COINES uses opensource **libusb-win32** driver
* Uninstall Development Desktop USB Driver ( **USBIO Device: VID=152A PID=80C0** under **BST Board** ) from Device Manager and install driver found in **<COINES\_INSTALL\_PATH>\driver\bst\_app20\_libusb\_driver.exe**

2. I have COINES Installed and now unable to use Development Desktop

* Development Desktop uses uses USB driver from Thesycon GmbH and COINES uses opensource **libusb-win32** driver
* Uninstall libusb-win32 USB Driver ( **BST Application Board 2.0 (libusb)** under **libusb-win32  devices**) from Device Manager and install driver found in **<Development Desktop Installation folder>\USB driver\usbio\_x64.inf**

3. I want to upgrade APP2.0 firmware

* Use Development Desktop to upgrade firmware. Make sure you use Development Desktop USB Driver

4. Why are libraries ( **libusb-1.0, pthread** ) statically linked with COINES examples in Windows while dynamically linking the same in Linux ?

**pthread** and **libusb-1.0** come by default in standard Linux distributions and there is no need to link them statically in Linux . They are not native to Windows (ported from Linux) and hence they are statically linked to avoid error messages like "**DLL not found**"

5. Why **GCC** is chosen as the compiler

* To support both Windows and Linux with minimal change in code
* Supporting both **MSVC** and **GCC** has maintainability issues which becomes harder as the project grows in complexity
* For example, Thread API offered by **MSVC** is different from the one offered in **GCC**
* **MSVC** requires licensing when it comes to enterprise usage

6. Reasons for usage of **libusb**

* Cross platform support - Linux, OS X, Windows, Windows CE, Android, OpenBSD/NetBSD
* User-space driver

7. Usage of **mingw32-make** in Windows

* Can handle Windows path names better compared with MSYS make
* Comes as a part of **TDM-GCC**
* Usage of spaces in path names can be overcome using 8.3 naming format

8. Usage of **TDM-GCC**

* Complete toolchain in a single installer
* Usage of MSYS and MinGW requires users to manually download the GCC toolchain for installation.

# Legal disclaimer

## Engineering samples

Engineering Samples are marked with an asterisk (\*) 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.

## Product usage

The Purchaser shall indemnify 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 security sensitive systems. Security sensitive systems are those for which a malfunction is expected to lead to bodily harm or significant property damage. In addition, they are not fit for use in products which interact with motor vehicle systems. The resale and/or use of 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 must monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of all security relevant incidents.

## 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.

# Document history and modification

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev. No** | **Chapter** | **Description of modification/changes** | **Date** |
| 1.0 | all | Document creation | March 2018 |
| 1.1 | 2.3 | Added section 2.3.1 - TDM-GCC | April 2018 |
| 1.2 | 4.1 | Usage of COINES GUI | May 2018 |
| 1.3 | 8 | FAQ | June 2018 |
| 1.4 | 2.3 | Removed section – MSYS2 installation | July 2018 |
| 1.5 | all | Cleanup – Corrected types, formatting ,etc., | August 2018 |
| 1.6 | 5.3,5.6 | Added more info. on coines\_get\_board\_info() and coines\_read\_sensor\_data() | September 2018 |