Erriez MH-Z19B CO2 sensor library for Arduino

1.0.1

Generated by Doxygen 1.9.8

Chapter 1

Erriez MH-Z19B CO2 sensor library for Arduino

Erriez MH-Z19B/C CO2 sensor library for Arduino

This is a MH-Z19B / MH-Z19C CO2 sensor library for Arduino. It has been built from scratch to support hardware and software serial with a small footprint.

The MH-Z19 is a NDIR (Non-Dispersive Infrared) type gas sensor with built-in temperature compensation to measure CO2 concentration in air.

Library features

- Small code/memory footprint
- Hardware and software serial interface at 9600 baud 8N1
- Read CO2 concentration 400..5000 ppm +/-50ppm+3% minimum 5 seconds interval
- Chip detection
- Smart warming-up detection
- Read firmware version
- Set/get range 2000 or 5000 ppm
- Set/get auto calibration (Automatic Baseline Correction 24h interval)
- Manual 400ppm calibration command
- CRC checks on communication protocol and timeout handling
- Interface for sending undocumented commands

Pins

WARNING: The pins between MH-Z19B and MH-Z19C are different. See tables below:

```
 ++
//      __     _______
// +------------------+
// |                  |
// |   . . . . . .    |
// |   1 2 3 4 5 6 7  |
// +------------------+
//
// MH-Z19B front connector:
//   Pin 1: Yellow  None
//   Pin 2: Green   UART (TXD) TTL Level Data Output  -> TO RXD
//   Pin 3: Blue    UART (RXD) TTL Level Data Input   -> TO TXD
//   Pin 4: Red     Positive Power Supply (Vin +5V)
//   Pin 5: Black   Negative Power Supply (GND)
//   Pin 6: White   None
//   Pin 7: Brown   Analog Output Vo (Not used)
//
// MH-Z19C front connector:
//   Pin 1: PWM
//   Pin 2: UART (TXD) TTL Level data output  -> TO RXD
//   Pin 3: UART (RXD) TTL Level data input   -> TO TXD
//   Pin 4: Positive Power Supply (Vin +5V)
//   Pin 5: Negative Power Supply (GND)
//   Pin 6: Analog Output Vo
//   Pin 7: HD (Hand-operated calibration)
//
// The following ESP8266 pins are reserved:
//    TX/RX:  Serial (in use)
//    A0:     Analog (cannot be used)
//    D0-RST: Wake (cannot be used)
//    D1/D2:  I2C (can be used when I2C not used)
//    D3:     Output data flash (corrupts MH-Z19B on boot)
//    D4:     Boot (in use by boot pin / LED)
//    D5..D8: SPI <- Can be used when SPI not used
```

Tested Hardware

The following targets are supported and tested:

- AVR: UNO, MINI, Pro Mini 8/16 MHz, ATMega2560, Leonardo
- ARM: DUE
- ESP8266: Mini D1 & D2, NodeMCU
- ESP32: Lolin D32

Examples

- ErriezMHZ19BGettingStarted
- ErriezMHZ19BSerialPlottter
- ErriezMHZ19B7SegmentDisplay

Documentation

- Online HTML
- Doxygen PDF
- MH-Z19B datasheet PDF
- MH-Z19C datasheet PDF

CO2 Concentrations

The table below displays the human impact of CO2:

CO2 ppm	Description
0..399	Incorrect values. Minimum value starts at 400ppm outdoor fresh air.
400..1000	Concentrations typical of occupied indoor spaces with good air exchange.
1000..2000	Complaints of drowsiness and poor air quality. Ventilation is required.
2000..5000	Headaches, sleepiness and stagnant, stale, stuffy air. Poor concentration, loss of attention, increased heart rate and slight nausea may also be present.
$>$5000	Higher values are extremely dangerous and cannot be measured by this sensor.

Usage

- Operating voltage is between 4.5 and 5VDC, 150mA peak current (average $<$ 60mA).
- UART pins are compatible with processors running at 3.3V without level converters.
- Keep sensor outside direct sunlight.

Calibration

The sensor requires an internal calibration regularly. Without it, the minimum value drifts away which is noticeable after a few weeks of operation. With my experiments, the minimum value was drifted to 800ppm after 3 months continues operation without a calibration.

There are two calibration options:

1. Automatic calibration, performed every 24 hours (default).
2. Manual calibration.

1. Automatic Calibration

Automatic calibration is recommended when the sensor cannot be moved outdoor with fresh air. This calibration method requires a regularly ventilated room at 400ppm, at least once in 1..3 weeks. Additionally, it requires continues power-up without interruptions, otherwise the calibration data will not be updated correctly.

Automatic calibration configuration:

- Set auto calibration on: `setAutoCalibration(true)` (Default from manufacture).
- Set auto calibration off: `setAutoCalibration(false)`.

The status can be read with function `getAutoCalibration()`.

Note:
For simplicity, this library uses the terminology `Automatic Calibration` which is identical to the `ABC (Automatic Baseline Correction) logic on/off` mentioned in the datasheet.

2. Manual ZERO Calibration (400ppm)

Procedure for manual calibration at 400ppm:

- Turn automatic calibration off.
- Power the sensor up outdoor in fresh air for at least 20 minutes. (Not in a forest or a farm which produces background CO2)
- Call `startZeroCalibration()` once. This will send command `0x87 Zero Point Calibration`, but is not a zero calibration as stated in the datasheet. There is no nitrogen needed as this calibration is performed at 400ppm.

Now the sensor is calibrated. Repeat the sequence more often for higher accuracy.

3. MH-Z19B only: Manual SPAN Calibration

The SPAN point calibration procedure is not implemented in this library as it requires special calibration equipment. This functionality is not available in MH-Z19C.

4. MH-Z19C only: Hand-operated calibration

Procedure according to the MH-Z19C datasheet:

- Connect module's HD pin to low level(0V), lasting for 7 seconds at least.
- Before calibrating the zero point, please ensure that the sensor is stable for more than 20 minutes at 400ppm ambient environment.
- The application is responsible to control the external MH-Z19C HD pin and is not available on the MH-Z19B.

MH-Z19B/C API

Initialization Software Serial

Use a Software Serial when no hardware serial is available. Sometimes a 3rd party library is required, for example for ESP32 targets by installing `ESPSoftwareSerial`. It must be installed into `.arduino15/packages/esp32/hardware/esp32/<version>/libraries/EspSoftware↩Serial`, because the library contains a naming conflict with existing `SoftwareSerial.h` built-in libraries.

```
 ++
#include <ErriezMHZ19B.h>
#include <SoftwareSerial.h>

// Pin defines
#define MHZ19B_TX_PIN        4
#define MHZ19B_RX_PIN        5

// Create software serial object
SoftwareSerial mhzSerial(MHZ19B_TX_PIN, MHZ19B_RX_PIN);

// Create MHZ19B object with software serial
ErriezMHZ19B mhz19b(&mhzSerial);
```

Initialization Hardware Serial

Any hardware serial like `Serial`, `Serial1`, `Serial2` etc can be used when supported by the CPU. Multiple hardware serial ports are only available on targets like ATMEGA2560, Leonardo and SAM DUE boards:

```
 ++
#include <ErriezMHZ19B.h>

// Create MHZ19B object with hardware serial
ErriezMHZ19B mhz19b(&Serial1);
```

General initialization

The optional items of the initialization sequence can be omitted.

```
 ++
void setup()
{
    // Initialize serial
    Serial.begin(115200);
    Serial.println(F("\nErriez MH-Z19B CO2 Sensor example"));

    // Initialize software serial at fixed baudrate
    mhzSerial.begin(9600);

    // Optional: Detect MH-Z19B sensor (check wiring / power)
    while ( !mhz19b.detect() ) {
        Serial.println(F("Detecting MH-Z19B sensor..."));
        delay(2000);
    };

    // Sensor requires 3 minutes warming-up after power-on
    while (mhz19b.isWarmingUp()) {
        Serial.println(F("Warming up..."));
        delay(2000);
    };
}
```

Read CO2 loop

Read CO2 with minimum interval asynchronous function `isReady()`. A good practice is to check error returns $<$ 0.

```
 ++
void loop()
{
    int16_t result;

    // Minimum interval between CO2 reads
    if (mhz19b.isReady()) {
        // Read CO2 from sensor
        result = mhz19b.readCO2();

        // Print result
        if (result < 0) {
            // Print error code
            switch (result) {
                case MHZ19B_RESULT_ERR_CRC:
                    Serial.println(F("CRC error"));
                    break;
                case MHZ19B_RESULT_ERR_TIMEOUT:
                    Serial.println(F("RX timeout"));
                    break;
                default:
                    Serial.print(F("Error: "));
                    Serial.println(result);
                    break;
            }
        } else {
            // Print CO2 concentration in ppm
            Serial.print(result);
            Serial.println(F(" ppm"));
        }
    }
}
```

Print internal settings

All tests are performed with sensor version string `"0443"`.

```
 ++
char firmwareVersion[5];

// Optional: Print firmware version
Serial.print(F("  Firmware: "));
mhz19b.getVersion(firmwareVersion, sizeof(firmwareVersion));
```

```
Serial.println(firmwareVersion);

// Optional: Set CO2 range 2000ppm or 5000ppm (default) once
// Serial.print(F("Set range..."));
// mhz19b.setRange2000ppm();
// mhz19b.setRange5000ppm();

// Optional: Print operating range
Serial.print(F("  Range: "));
Serial.print(mhz19b.getRange());
Serial.println(F("ppm"));

// Optional: Print Automatic Baseline Calibration status
Serial.print(F("  Auto calibrate: "));
Serial.println(mhz19b.getAutoCalibration() ? F("On") : F("Off"));
```

Set automatic calibration

Turn automatic calibration on or off once at startup:

```
 ++
// Optional: Set automatic calibration on (true) or off (false) once
mhz19b.setAutoCalibration(true);
```

Documented commands

The following commands are documented, used and tested by the library:

Command	Description
0x79	Set auto calibration on/off
0x86	Read CO2 concentration
0x87	Calibration zero point at 400ppm (not 0 ppm)
0x88	Calibrate span point (NOT IMPLEMENTED)
0x99	Set detection range

Not documented commands (tested)

The following commands are **not documented**, are used and tested by the library:

Command	Description
0x7D	Get auto calibration status (NOT DOCUMENTED)
0x9B	Get range detection (NOT DOCUMENTED)
0xA0	Get firmware version (NOT DOCUMENTED)

More information about undocumented commands: https://revspace.nl/MH-Z19B.

NOTE: Sending untested commands may damage the sensor permanently! Use at your own risk.

```
 ++
int16_t result;

result = mhz19b.sendCommand(MHZ19B_CMD_NOT_DOCUMENTED, 0x00, 0x00, 0x00, 0x00, 0x00);

// 9 Bytes response is located in mhz19b.rxBuffer[9]
```

Library configuration

Unfortunately, the sensor has no possibility to read warming-up status, so the library must wait at least 3 minutes after reset or power-on. To speedup the boot process, macro `MHZ19B_SMART_WARMING_UP` can be enabled in `ErriezMHZ19B.h` to enable smart warming-up when the MCU is reset and MH-Z19B powered $>$ 3 minutes.

Response timing

The screenshot below displays the response timing of a synchronous readCO2() call which takes 22.1ms on an Arduino UNO:

- 9.4ms: Transmit 9 Bytes at 9600 baud
- 3.2ms: MH-Z19B to process command
- 9.3ms: Return response 9 Bytes at 9600 baud
- 183us: Arduino UNO to process response with Software Serial.

Library installation

Please refer to the `Wiki` page.

Other Arduino Libraries and Sketches from Erriez

`Erriez Libraries and Sketches`

MIT License

This project is published under `MIT license` with an additional end user agreement (next section).

End User Agreement :ukraine:

End users shall accept the `End User Agreement` holding export restrictions to Russia to stop the WAR before using this project.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 ErriezMHZ19B Class Reference

Class ErriezMHZ19B.

`#include <ErriezMHZ19B.h>`

Public Member Functions

- ErriezMHZ19B (Stream *serial)
  Constructor with serial Stream.
- ~ErriezMHZ19B ()
  Destructor.
- bool detect ()
  Detect MHZ19B sensor.
- bool isWarmingUp ()
  Check if sensor is warming-up after power-on.
- bool isReady ()
  Check minimum interval between CO2 reads.
- int16_t readCO2 ()
  Read CO2 from sensor.
- int8_t getVersion (char *version, uint8_t versionLen)
  Get firmware version (NOT DOCUMENTED)
- int8_t setRange2000ppm ()
  Set CO2 range 2000 ppm.
- int8_t setRange5000ppm ()
  Set CO2 range 5000 ppm.
- int16_t getRange ()
  Get CO2 range in PPM (NOT DOCUMENTED)
- int8_t setAutoCalibration (bool calibrationOn)
  Enable or disable automatic calibration.
- int8_t getAutoCalibration ()
  Get status automatic calibration (NOT DOCUMENTED)
- int8_t startZeroCalibration ()
  Start Zero Point Calibration manually at 400ppm.
- int8_t sendCommand (uint8_t cmd, byte b3=0, byte b4=0, byte b5=0, byte b6=0, byte b7=0)
  Send serial command to sensor and read response.

Public Attributes

- uint8_t rxBuffer [MHZ19B_SERIAL_RX_BYTES]

4.1.1 Detailed Description

Class ErriezMHZ19B.

Definition at line 91 of file ErriezMHZ19B.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ErriezMHZ19B()

```
ErriezMHZ19B::ErriezMHZ19B (
            Stream * serial )
```

Constructor with serial Stream.

Parameters

serial	Serial Stream pointer.

Definition at line 49 of file ErriezMHZ19B.cpp.

4.1.2.2 ~ErriezMHZ19B()

```
ErriezMHZ19B::~ErriezMHZ19B ( )
```

Destructor.

The serial Stream pointer is cleared and requires a new constructor to reuse it again.

Definition at line 58 of file ErriezMHZ19B.cpp.

4.1.3 Member Function Documentation

4.1.3.1 detect()

```
bool ErriezMHZ19B::detect ( )
```

Detect MHZ19B sensor.

Return values

true	Sensor detected.
false	Sensor not detected, check wiring/power.

Definition at line 70 of file ErriezMHZ19B.cpp.

4.1.3.2 getAutoCalibration()

```
int8_t ErriezMHZ19B::getAutoCalibration ( )
```

Get status automatic calibration (NOT DOCUMENTED)

Return values

<0	MH-Z19B response error codes.
1	Automatic calibration on.
0	Automatic calibration off.

Definition at line 296 of file ErriezMHZ19B.cpp.

4.1.3.3 getRange()

```
int16_t ErriezMHZ19B::getRange ( )
```

Get CO2 range in PPM (NOT DOCUMENTED)

This function verifies valid read ranges of 2000 or 5000 ppm.
Note: Other ranges may be returned, but are undocumented and marked as invalid.

Return values

<0	MH-Z19B response error codes.
MHZ19B_RANGE_2000	Range 2000 ppm.
MHZ19B_RANGE_5000	Range 5000 ppm (default).

Definition at line 252 of file ErriezMHZ19B.cpp.

4.1.3.4 getVersion()

```
int8_t ErriezMHZ19B::getVersion (
            char * version,
            uint8_t versionLen )
```

Get firmware version (NOT DOCUMENTED)

This is an undocumented command, but most sensors returns ASCII "0430 or "0443".

Parameters

version	Returned character pointer to version (must be at least 5 Bytes) Only valid when return is set to MHZ19B_RESULT_OK.
versionLen	Number of characters including NULL of version buffer.

Returns

MH-Z19B response error codes.

Definition at line 189 of file ErriezMHZ19B.cpp.

4.1.3.5 isReady()

```
bool ErriezMHZ19B::isReady ( )
```

Check minimum interval between CO2 reads.

Not described in the datasheet, but it is the same frequency as the built-in LED blink.

Return values

true	Ready to call readCO2().
false	Conversion not completed.

Definition at line 133 of file ErriezMHZ19B.cpp.

4.1.3.6 isWarmingUp()

```
bool ErriezMHZ19B::isWarmingUp ( )
```

Check if sensor is warming-up after power-on.

The datasheet mentions a startup delay of 3 minutes before reading CO2.
Experimentally discovered, the sensor may return CO2 data earlier. To speed-up the boot process, it is possible to check if the CO2 value changes to abort the warming-up, for example when the MCU is reset and keep the sensor powered.
Recommended to disable this option for deployment by disabling macro MHZ19B_SMART_WARMING_UP in header file.

Return values

true	Sensor is warming-up.
false	Sensor is ready to use.

Definition at line 96 of file ErriezMHZ19B.cpp.

4.1.3.7 readCO2()

```
int16_t ErriezMHZ19B::readCO2 ( )
```

Read CO2 from sensor.

Return values

<0	MH-Z19B response error codes.

Return values

0..399	ppm Incorrect values. Minimum value starts at 400ppm outdoor fresh air.
400..1000	ppm Concentrations typical of occupied indoor spaces with good air exchange.
1000..2000	ppm Complaints of drowsiness and poor air quality. Ventilation is required.
2000..5000	ppm Headaches, sleepiness and stagnant, stale, stuffy air. Poor concentration, loss of attention, increased heart rate and slight nausea may also be present. Higher values are extremely dangerous and cannot be measured.

Definition at line 158 of file ErriezMHZ19B.cpp.

4.1.3.8 sendCommand()

```
int8_t ErriezMHZ19B::sendCommand (
            uint8_t cmd,
            byte b3 = 0,
            byte b4 = 0,
            byte b5 = 0,
            byte b6 = 0,
            byte b7 = 0 )
```

Send serial command to sensor and read response.

Send command to sensor and read response, protected with a receive timeout.
Result is available in public rxBuffer[9].

Parameters

cmd	Command Byte
b3	Byte 3 (default 0)
b4	Byte 4 (default 0)
b5	Byte 5 (default 0)
b6	Byte 6 (default 0)
b7	Byte 7 (default 0)

Definition at line 345 of file ErriezMHZ19B.cpp.

4.1.3.9 setAutoCalibration()

```
int8_t ErriezMHZ19B::setAutoCalibration (
            bool calibrationOn )
```

Enable or disable automatic calibration.

Parameters

calibrationOn	true: Automatic calibration on. false: Automatic calibration off.

Returns

MH-Z19B response error codes.

Definition at line 281 of file ErriezMHZ19B.cpp.

4.1.3.10 setRange2000ppm()

```
int8_t ErriezMHZ19B::setRange2000ppm ( )
```

Set CO2 range 2000 ppm.

Returns

MH-Z19B response error codes.

Definition at line 221 of file ErriezMHZ19B.cpp.

4.1.3.11 setRange5000ppm()

```
int8_t ErriezMHZ19B::setRange5000ppm ( )
```

Set CO2 range 5000 ppm.

Returns

MH-Z19B response error codes.

Definition at line 233 of file ErriezMHZ19B.cpp.

4.1.3.12 startZeroCalibration()

```
int8_t ErriezMHZ19B::startZeroCalibration ( )
```

Start Zero Point Calibration manually at 400ppm.

The sensor must be powered-up for at least 20 minutes in fresh air at 400ppm room temperature. Then call this function once to execute self calibration.
Recommended to use this function when auto calibrate turned off.

Returns

MH-Z19B response error codes.

Definition at line 321 of file ErriezMHZ19B.cpp.

4.1.4 Member Data Documentation

4.1.4.1 rxBuffer

```
uint8_t ErriezMHZ19B::rxBuffer[MHZ19B_SERIAL_RX_BYTES]
```

Definition at line 128 of file ErriezMHZ19B.h.

The documentation for this class was generated from the following files:

- src/ErriezMHZ19B.h
- src/ErriezMHZ19B.cpp

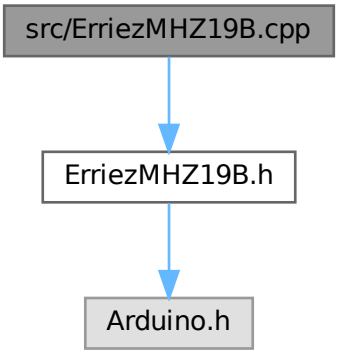
Chapter 5

File Documentation

5.1 src/ErriezMHZ19B.cpp File Reference

MH-Z19B CO2 sensor library for Arduino.

`#include "ErriezMHZ19B.h"`
Include dependency graph for ErriezMHZ19B.cpp:



5.1.1 Detailed Description

MH-Z19B CO2 sensor library for Arduino.

This sensor library is re-build from scratch.

Design choices:

- Keep code and memory size as small as possible.
- Use documented functions as much as possible for reliability and to prevent bricking the sensor.
- PWM not implemented in this library, because it is not accurate and reduces code size.

Source: https://github.com/Erriez/ErriezMHZ19B Documentation: https://erriez.↩
github.io/ErriezMHZ19B

Definition in file ErriezMHZ19B.cpp.

5.2 ErriezMHZ19B.cpp

Go to the documentation of this file.

```
00001 /*
00002  * MIT License
00003  *
00004  * Copyright (c) 2020-2026 Erriez
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining a copy
00007  * of this software and associated documentation files (the "Software"), to deal
00008  * in the Software without restriction, including without limitation the rights
00009  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010  * copies of the Software, and to permit persons to whom the Software is
00011  * furnished to do so, subject to the following conditions:
00012  *
00013  * The above copyright notice and this permission notice shall be included in all
00014  * copies or substantial portions of the Software.
00015  *
00016  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022  * SOFTWARE.
00023  */
00024
00041 #include "ErriezMHZ19B.h"
00042
00043
00049 ErriezMHZ19B::ErriezMHZ19B(Stream *serial) : _serial(serial), _tLastReadCO2(0)
00050 {
00051 }
00052
00058 ErriezMHZ19B::~ErriezMHZ19B()
00059 {
00060     _serial = nullptr;
00061 }
00062
00070 bool ErriezMHZ19B::detect()
00071 {
00072     // Check valid PPM range
00073     if (getRange() > 0) {
00074         return true;
00075     }
00076
00077     // Sensor not detected, or invalid range returned
00078     // Try recover by calling setRange(MHZ19B_RANGE_5000);
00079     return false;
00080 }
00081
00096 bool ErriezMHZ19B::isWarmingUp()
00097 {
00098     // Wait at least 3 minutes after power-on
00099     if (millis() < MHZ19B_WARMING_UP_TIME_MS) {
00100 #ifdef MHZ19B_SMART_WARMING_UP
00101         static int16_t _lastCO2 = -1;
00102         int16_t co2;
00103
00104         // Sensor returns valid data after CPU reset and keep sensor powered
00105         co2 = readCO2();
00106         if (_lastCO2 == -1) {
00107             _lastCO2 = co2;
00108         } else {
00109             if (_lastCO2 != co2) {
00110                 // CO2 value changed since last read, no longer warming-up
00111                 _tLastReadCO2 = 0;
00112                 return false;
00113             }
00114         }
00115 #endif
00116         // Warming-up
00117         return true;
00118     }
00119
00120     // Not warming-up
00121     return false;
00122 }
00123
00133 bool ErriezMHZ19B::isReady()
00134 {
00135     // Minimum CO2 read interval (Built-in LED flashes)
00136     if ((millis() - _tLastReadCO2) > MHZ19B_READ_INTERVAL_MS) {
00137         return true;
00138     }
```

```
00139
00140     return false;
00141 }
00142
00158 int16_t ErriezMHZ19B::readCO2()
00159 {
00160     int16_t result;
00161
00162     // Set timestamp
00163     _tLastReadCO2 = millis();
00164
00165     // Send command "Read CO2 concentration"
00166     result = sendCommand(MHZ19B_CMD_READ_CO2);
00167
00168     // Check result
00169     if (result == MHZ19B_RESULT_OK) {
00170         // 16-bit CO2 value in response Bytes 2 and 3
00171         result = (rxBuffer[2] << 8) | rxBuffer[3];
00172     }
00173
00174     return result;
00175 }
00176
00189 int8_t ErriezMHZ19B::getVersion(char *version, uint8_t versionLen)
00190 {
00191     int8_t result;
00192
00193     // Argument check
00194     if (versionLen < 5) {
00195         return MHZ19B_RESULT_ARGUMENT_ERROR;
00196     }
00197
00198     // Clear version
00199     memset(version, 0, 5);
00200
00201     // Send command "Read firmware version" (NOT DOCUMENTED)
00202     result = sendCommand(MHZ19B_CMD_GET_VERSION);
00203
00204     // Check result
00205     if (result == MHZ19B_RESULT_OK) {
00206         // Copy 4 ASCII characters to version array like "0443"
00207         for (uint8_t i = 0; i < 4; i++) {
00208             // Version in response Bytes 2..5
00209             version[i] = rxBuffer[i + 2];
00210         }
00211     }
00212
00213     return result;
00214 }
00215
00221 int8_t ErriezMHZ19B::setRange2000ppm()
00222 {
00223     // Send "Set range" command
00224     return sendCommand(MHZ19B_CMD_SET_RANGE,
00225                        0x00, 0x00, 0x00, (MHZ19B_RANGE_2000 >> 8), (MHZ19B_RANGE_2000 & 0xff));
00226 }
00227
00233 int8_t ErriezMHZ19B::setRange5000ppm()
00234 {
00235     // Send "Set range" command
00236     return sendCommand(MHZ19B_CMD_SET_RANGE,
00237                        0x00, 0x00, 0x00, (MHZ19B_RANGE_5000 >> 8), (MHZ19B_RANGE_5000 & 0xff));
00238 }
00239
00252 int16_t ErriezMHZ19B::getRange()
00253 {
00254     int16_t result;
00255
00256     // Send command "Read range" (NOT DOCUMENTED)
00257     result = sendCommand(MHZ19B_CMD_GET_RANGE);
00258
00259     // Check result
00260     if (result == MHZ19B_RESULT_OK) {
00261         // Range is in Bytes 4 and 5
00262         result = (rxBuffer[4] << 8) | rxBuffer[5];
00263
00264         // Check range according to documented specification
00265         if ((result != MHZ19B_RANGE_2000) && (result != MHZ19B_RANGE_5000)) {
00266             result = MHZ19B_RESULT_ERROR;
00267         }
00268     }
00269
00270     return result;
00271 }
00272
00281 int8_t ErriezMHZ19B::setAutoCalibration(bool calibrationOn)
00282 {
```

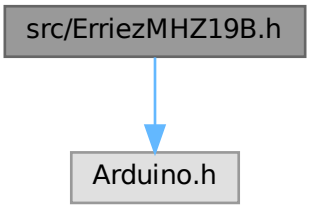
```
00283     // Send command "Set Automatic Baseline Correction (ABC logic function)"
00284     return sendCommand(MHZ19B_CMD_SET_AUTO_CAL, (calibrationOn ? 0xA0 : 0x00));
00285 }
00286
00296 int8_t ErriezMHZ19B::getAutoCalibration()
00297 {
00298     int8_t result;
00299
00300     // Send command "Get Automatic Baseline Correction (ABC logic function)" (NOT DOCUMENTED)
00301     result = sendCommand(MHZ19B_CMD_GET_AUTO_CAL);
00302
00303     // Check result
00304     if (result == MHZ19B_RESULT_OK) {
00305         // Response is located in Byte 7: 0 = off, 1 = on
00306         result = rxBuffer[7] & 0x01;
00307     }
00308
00309     return result;
00310 }
00311
00321 int8_t ErriezMHZ19B::startZeroCalibration()
00322 {
00323     // Send command "Zero Point Calibration"
00324     return sendCommand(MHZ19B_CMD_CAL_ZERO_POINT);
00325 }
00326
00345 int8_t ErriezMHZ19B::sendCommand(uint8_t cmd, byte b3, byte b4, byte b5, byte b6, byte b7)
00346 {
00347     uint8_t txBuffer[MHZ19B_SERIAL_RX_BYTES] = { 0xFF, 0x01, cmd, b3, b4, b5, b6, b7, 0x00 };
00348     int8_t result = MHZ19B_RESULT_OK;
00349     unsigned long tStart;
00350
00351     // Check serial initialized
00352     if (_serial == nullptr) {
00353         return MHZ19B_RESULT_ERROR;
00354     }
00355
00356     // Add CRC Byte
00357     txBuffer[8] = calcCRC(txBuffer);
00358
00359     // Clear receive buffer
00360     while (_serial->available()) {
00361         _serial->read();
00362     }
00363
00364     // Write serial data
00365     _serial->write(txBuffer, sizeof(txBuffer));
00366
00367     // Flush serial data
00368     _serial->flush();
00369
00370     // Clear receive buffer
00371     memset(rxBuffer, 0, sizeof(rxBuffer));
00372
00373     // Wait until all data received from sensor with receive timeout protection
00374     tStart = millis();
00375     while (_serial->available() < MHZ19B_SERIAL_RX_BYTES) {
00376         if ((millis() - tStart) >= MHZ19B_SERIAL_RX_TIMEOUT_MS) {
00377             return MHZ19B_RESULT_ERR_TIMEOUT;
00378         }
00379     }
00380
00381     // Read response from serial buffer
00382     _serial->readBytes(rxBuffer, MHZ19B_SERIAL_RX_BYTES);
00383
00384     // Check received Byte[0] == 0xFF and Byte[1] == transmit command
00385     if ((rxBuffer[0] != 0xFF) || (rxBuffer[1] != cmd)) {
00386         result = MHZ19B_RESULT_ERROR;
00387     }
00388
00389     // Check received Byte[8] CRC
00390     if (rxBuffer[8] != calcCRC(rxBuffer)) {
00391         result = MHZ19B_RESULT_ERR_CRC;
00392     }
00393
00394     // Return result
00395     return result;
00396 }
00397
00398 // ----------------------------------------------------------------------------
00399 // Private functions
00400 // ----------------------------------------------------------------------------
00401
00409 uint8_t ErriezMHZ19B::calcCRC(uint8_t *data)
00410 {
00411     byte crc = 0;
00412
```

```
00413     // Calculate CRC on 8 data Bytes
00414     for (uint8_t i = 1; i < 8; i++) {
00415         crc += data[i];
00416     }
00417     crc = 0xFF - crc;
00418     crc++;
00419
00420     // Return calculated CRC
00421     return crc;
00422 }
```

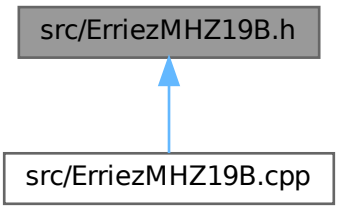
5.3 src/ErriezMHZ19B.h File Reference

MH-Z19B CO2 sensor library for Arduino.

`#include <Arduino.h>`
Include dependency graph for ErriezMHZ19B.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ErriezMHZ19B

  Class ErriezMHZ19B.

Macros

- #define MHZ19B_WARMING_UP_TIME_MS (3UL * 60000UL)

  3 minutes warming-up time after power-on before valid data returned
- #define MHZ19B_READ_INTERVAL_MS (5UL * 1000UL)

  Minimum response time between CO2 reads (EXPERIMENTALLY DEFINED)
- #define MHZ19B_SERIAL_RX_BYTES 9

  Fixed 9 Bytes response.
- #define MHZ19B_SERIAL_RX_TIMEOUT_MS 120

  Response timeout between 15..120 ms at 9600 baud works reliable for all commands.
- #define MHZ19B_CMD_SET_AUTO_CAL 0x79

  Command set auto calibration on/off.
- #define MHZ19B_CMD_READ_CO2 0x86

  Command read CO2 concentration.
- #define MHZ19B_CMD_CAL_ZERO_POINT 0x87

  Command calibrate zero point at 400ppm.
- #define MHZ19B_CMD_CAL_SPAN_POINT 0x88

  Command calibrate span point (NOT IMPLEMENTED)
- #define MHZ19B_CMD_SET_RANGE 0x99

  Command set detection range.
- #define MHZ19B_CMD_GET_AUTO_CAL 0x7D

  Command get auto calibration status (NOT DOCUMENTED)
- #define MHZ19B_CMD_GET_RANGE 0x9B

  Command get range detection (NOT DOCUMENTED)
- #define MHZ19B_CMD_GET_VERSION 0xA0

  Command get firmware version (NOT DOCUMENTED)

Enumerations

- enum MHZ19B_Result_e {
  MHZ19B_RESULT_OK = 0 , MHZ19B_RESULT_ERROR = -1 , MHZ19B_RESULT_ERR_CRC = -2 ,
  MHZ19B_RESULT_ERR_TIMEOUT = -3 ,
  MHZ19B_RESULT_ARGUMENT_ERROR = -4 }

  Response on a command.
- enum MHZ19B_Range_e { MHZ19B_RANGE_2000 = 2000 , MHZ19B_RANGE_5000 = 5000 }

  PPM range.

5.3.1 Detailed Description

MH-Z19B CO2 sensor library for Arduino.

Source: https://github.com/Erriez/ErriezMHZ19B Documentation: https://erriez.↩
github.io/ErriezMHZ19B

Definition in file ErriezMHZ19B.h.

5.3.2 Macro Definition Documentation

5.3.2.1 MHZ19B_CMD_CAL_SPAN_POINT

```
#define MHZ19B_CMD_CAL_SPAN_POINT 0x88
```

Command calibrate span point (NOT IMPLEMENTED)

Definition at line 60 of file ErriezMHZ19B.h.

5.3.2.2 MHZ19B_CMD_CAL_ZERO_POINT

```
#define MHZ19B_CMD_CAL_ZERO_POINT 0x87
```

Command calibrate zero point at 400ppm.

Definition at line 59 of file ErriezMHZ19B.h.

5.3.2.3 MHZ19B_CMD_GET_AUTO_CAL

```
#define MHZ19B_CMD_GET_AUTO_CAL 0x7D
```

Command get auto calibration status (NOT DOCUMENTED)

Definition at line 64 of file ErriezMHZ19B.h.

5.3.2.4 MHZ19B_CMD_GET_RANGE

```
#define MHZ19B_CMD_GET_RANGE 0x9B
```

Command get range detection (NOT DOCUMENTED)

Definition at line 65 of file ErriezMHZ19B.h.

5.3.2.5 MHZ19B_CMD_GET_VERSION

```
#define MHZ19B_CMD_GET_VERSION 0xA0
```

Command get firmware version (NOT DOCUMENTED)

Definition at line 66 of file ErriezMHZ19B.h.

5.3.2.6 MHZ19B_CMD_READ_CO2

```
#define MHZ19B_CMD_READ_CO2 0x86
```

Command read CO2 concentration.

Definition at line 58 of file ErriezMHZ19B.h.

5.3.2.7 MHZ19B_CMD_SET_AUTO_CAL

```
#define MHZ19B_CMD_SET_AUTO_CAL 0x79
```

Command set auto calibration on/off.

Definition at line 57 of file ErriezMHZ19B.h.

5.3.2.8 MHZ19B_CMD_SET_RANGE

```
#define MHZ19B_CMD_SET_RANGE 0x99
```

Command set detection range.

Definition at line 61 of file ErriezMHZ19B.h.

5.3.2.9 MHZ19B_READ_INTERVAL_MS

```
#define MHZ19B_READ_INTERVAL_MS (5UL * 1000UL)
```

Minimum response time between CO2 reads (EXPERIMENTALLY DEFINED)

Definition at line 48 of file ErriezMHZ19B.h.

5.3.2.10 MHZ19B_SERIAL_RX_BYTES

```
#define MHZ19B_SERIAL_RX_BYTES 9
```

Fixed 9 Bytes response.

Definition at line 51 of file ErriezMHZ19B.h.

5.3.2.11 MHZ19B_SERIAL_RX_TIMEOUT_MS

```
#define MHZ19B_SERIAL_RX_TIMEOUT_MS 120
```

Response timeout between 15..120 ms at 9600 baud works reliable for all commands.

Definition at line 54 of file ErriezMHZ19B.h.

5.3.2.12 MHZ19B_WARMING_UP_TIME_MS

```
#define MHZ19B_WARMING_UP_TIME_MS (3UL * 60000UL)
```

3 minutes warming-up time after power-on before valid data returned

Enable smart warming-up to return false when CO2 value changes within 3 minutes pre-heating time. Can be used when MCU is reset and sensor powered-up for >3 minutes. Recommended to disable for deployment to ensure warming-up timing.

Definition at line 45 of file ErriezMHZ19B.h.

5.3.3 Enumeration Type Documentation

5.3.3.1 MHZ19B_Range_e

```
enum MHZ19B_Range_e
```

PPM range.

Enumerator

MHZ19B_RANGE_2000	Range 2000 ppm.
MHZ19B_RANGE_5000	Range 5000 ppm (Default)

Definition at line 82 of file ErriezMHZ19B.h.

5.3.3.2 MHZ19B_Result_e

```
enum MHZ19B_Result_e
```

Response on a command.

Enumerator

MHZ19B_RESULT_OK	Response OK.
MHZ19B_RESULT_ERROR	Response error.
MHZ19B_RESULT_ERR_CRC	Response CRC error.
MHZ19B_RESULT_ERR_TIMEOUT	Response timeout.
MHZ19B_RESULT_ARGUMENT_ERROR	Response argument error.

Definition at line 71 of file ErriezMHZ19B.h.

5.4 ErriezMHZ19B.h

Go to the documentation of this file.

```
00001 /*
00002  * MIT License
00003  *
00004  * Copyright (c) 2020-2026 Erriez
00005  *
00006  * Permission is hereby granted, free of charge, to any person obtaining a copy
00007  * of this software and associated documentation files (the "Software"), to deal
00008  * in the Software without restriction, including without limitation the rights
00009  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010  * copies of the Software, and to permit persons to whom the Software is
00011  * furnished to do so, subject to the following conditions:
00012  *
00013  * The above copyright notice and this permission notice shall be included in all
00014  * copies or substantial portions of the Software.
00015  *
00016  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022  * SOFTWARE.
00023  */
00024
00033 #ifndef ERRIEZ_MHZ19B_H_
00034 #define ERRIEZ_MHZ19B_H_
00035
00036 #include <Arduino.h>
00037
00042 // #define MHZ19B_SMART_WARMING_UP
00043
00045 #define MHZ19B_WARMING_UP_TIME_MS       (3UL * 60000UL)
00046
00048 #define MHZ19B_READ_INTERVAL_MS         (5UL * 1000UL)
00049
00051 #define MHZ19B_SERIAL_RX_BYTES          9
```

```
00052
00054 #define MHZ19B_SERIAL_RX_TIMEOUT_MS     120
00055
00056 // Documented commands
00057 #define MHZ19B_CMD_SET_AUTO_CAL         0x79
00058 #define MHZ19B_CMD_READ_CO2             0x86
00059 #define MHZ19B_CMD_CAL_ZERO_POINT       0x87
00060 #define MHZ19B_CMD_CAL_SPAN_POINT       0x88
00061 #define MHZ19B_CMD_SET_RANGE            0x99
00062
00063 // Not documented commands
00064 #define MHZ19B_CMD_GET_AUTO_CAL         0x7D
00065 #define MHZ19B_CMD_GET_RANGE            0x9B
00066 #define MHZ19B_CMD_GET_VERSION          0xA0
00067
00071 typedef enum {
00072     MHZ19B_RESULT_OK = 0,
00073     MHZ19B_RESULT_ERROR = -1,
00074     MHZ19B_RESULT_ERR_CRC = -2,
00075     MHZ19B_RESULT_ERR_TIMEOUT = -3,
00076     MHZ19B_RESULT_ARGUMENT_ERROR = -4,
00077 } MHZ19B_Result_e;
00078
00082 typedef enum {
00083     MHZ19B_RANGE_2000 = 2000,
00084     MHZ19B_RANGE_5000 = 5000,
00085 } MHZ19B_Range_e;
00086
00087
00091 class ErriezMHZ19B
00092 {
00093 public:
00094     ErriezMHZ19B(Stream *serial);
00095     ~ErriezMHZ19B();
00096
00097     // Detect sensor by checking range response
00098     bool detect();
00099
00100     // Minimum wait time after power-on
00101     bool isWarmingUp();
00102
00103     // Minimum wait time between CO2 reads
00104     bool isReady();
00105
00106     // Read CO2 value
00107     int16_t readCO2();
00108
00109     // Get firmware version (NOT DOCUMENTED)
00110     int8_t getVersion(char *version, uint8_t versionLen);
00111
00112     // Set/get CO2 range, default 5000ppm)
00113     int8_t setRange2000ppm();
00114     int8_t setRange5000ppm();
00115     int16_t getRange(); // (NOT DOCUMENTED)
00116
00117     // Set and get ABC, default on (Automatic Baseline Correction)
00118     int8_t setAutoCalibration(bool calibrationOn);
00119     int8_t getAutoCalibration(); // (NOT DOCUMENTED)
00120
00121     // Start Zero Point Calibration manually at 400ppm
00122     int8_t startZeroCalibration();
00123
00124     // Serial communication
00125     int8_t sendCommand(uint8_t cmd, byte b3=0, byte b4=0, byte b5=0, byte b6=0, byte b7=0);
00126
00127     // Global receive buffer
00128     uint8_t rxBuffer[MHZ19B_SERIAL_RX_BYTES];
00129
00130 private:
00131     Stream *_serial;
00132     unsigned long _tLastReadCO2;
00133
00134     // CRC check on serial transmit/receive buffer
00135     uint8_t calcCRC(uint8_t *data);
00136 };
00137
00138 #endif // ERRIEZ_MHZ19B_H_
```

Index