CO Detector Digital Sensor

BM22S3221-1
Arduino Library Description

Revision: V1.00 Date: August 18, 2023

www.bestmodulescorp.com

Contents

Introduction

The BM22S3221-1 is a CO (Carbon Monoxide) Detector Digital Sensor from Best Modules, which uses the UART communication method. This document provides the description of the BM22S3221-1 Arduino Lib functions and how to install the Arduino Lib. The example uses the BMA56M322 module to demonstrate the functions of alarm signal output and CO concentration read.

Applicable types:

Part No.	Description
BM22S3221-1	CO Detector Digital Sensor
BMA56M322	On-board BM22S3221-1 Sensor

Arduino Lib Functions

Arduino Lib Name: BM22S3221-1		Lib Version: V1.0.1
Constructors & Initialisation		
1	BM22S3221_1(uint8_t statusPin, HardwareSerial *theSerial = &Serial)	
	Description	Constructor (Hardware UART)
	Parameter	statusPin: STATUS pin, connects to the STATUS pin of the BM22S3221-1 or the BMA56M322 *theSerial: Hardware Serial interface
	Return Value	—
	Note	—
2	BM22S3221_1(uint8_t statusPin, uint8_t rxPin, uint8_t txPin)	
	Description	Constructor (Software UART)
	Parameter	statusPin: STATUS pin, connects to the STATUS pin of the BM22S3221-1 or the STA pin of the BMA56M322 rxPin: RX pin, connects to the TX pin of the BM22S3221-1 or the BMA56M322 txPin: TX pin, connects to the RX pin of the BM22S3221-1 or the BMA56M322
	Return Value	—
	Note	—
3	void begin()	
	Description	Module initialisation
	Parameter	—
	Return Value	void
	Note	—
4	void preheatCountdown()	
	Description	Wait for the module preheating to complete
	Parameter	—
	Return Value	void
	Note	1. The end of this function indicates that the preheating is completed 2. When the module is powered on, this function will determine whether its automatic output function is enabled a. Enable: Keep reading the remaining preheating time that is automatically output from information packet, exit the function when the time is zero; b. Disable: Exit the function after a delay time of about 120s

Performance Functions		
5	uint8_t getSTATUS()	
	Description	Get the STATUS pin level
	Parameter	—
	Return Value	STATUS pin level: 0: Low level 1: High level
	Note	—
6	uint8_t getWorkStatus()	
	Description	Get the module cuttent operating status
	Parameter	—
	Return Value	Device status (1 byte): Bit 0 is 1: This indicates that the calibration is in progress Bit1~2: Reserved Bit 3 is 1: This it indicates that the calibration is has completed Bit 4 is 1: This it indicates that the zero point calibration has is completed Bit 5 is 1: This it indicates that the gas calibration has is completed Bit 6 is 1: This it indicates that the sensor is in a fault state condition Bit 7 is 1: This it indicates an alarm status
	Note	—
7	uint16_t readCOValue()	
	Description	Read the CO concentration value
	Parameter	—
	Return Value	CO concentration value, unit: ppm
	Note	Maximum acceptable value 10000ppm
8	uint16_t readADValue()	
	Description	Get the A/D value corresponding to the current CO concentration
	Parameter	—
	Return Value	12-bit A/D value
	Note	—
9	uint16_t readRefValue()	
	Description	Read the power-on reference value
	Parameter	—
	Return Value	12-bit A/D value
	Note	—
10	uint8_t requestInfoPackage(uint8_t array[])	
	Description	Get all the information of the module
	Parameter	array[]: Used for storing the module information (34 bytes)
	Return Value	Execution result(1)
	Note	1. It is recommended that this function should be used after the serial interface automatic output function is disabled 2. Refer to the datasheet for the definition of each byte
11	bool isInfoAvailable()	
	Description	Query if the module automatic output information has been received
	Parameter	—
	Return Value	Receiving status of the module automatic output information true: Received false: Not received
	Note	Used after the serial automatic output function is enabled

12	void readInfoPackage(uint8_t array[])	
	Description	Read the module automatic output information
	Parameter	array[]:Used for storing the module information (34 bytes)
	Return Value	—
	Note	1. This function should be used after “isInfoAvailable()==true” 2. Refer to the datasheet for the definition of each byte
13	uint8_t resetModule()	
	Description	Reset the module
	Parameter	—
	Return Value	Execution result(1)
	Note	After executing this function, the module enters a preheating status
14	uint8_t restoreDefault()	
	Description	Restore the module parameters to their factory settings
	Parameter	—
	Return Value	Execution result(1)
	Note	After executing this function, the module enters a preheating status
Parameter Query		
15	uint16_t getFWVer()	
	Description	Get the firmware version
	Parameter	—
	Return Value	Firmware version
	Note	0x0105 represents V1.5
16	uint8_t getProDate(uint8_t array[])	
	Description	Get the production date
	Parameter	array[]: Used for storing the production date array[0]/array[1]/array[2]: year/month/day
	Return Value	0: Execution success 1: Data verification error 2: Communication timeout
	Note	For example: 0x21,0x06,0x2 represents 2021/6/2
17	bool isAutoTx()	
	Description	Query if the serial interface automatic output function is enabled
	Parameter	—
	Return Value	Serial interface automatic output function status: true: Automatic output enabled false: Automatic output disabled
	Note	—
18	uint8_t getStatusPinActiveMode()	
	Description	Query the STATUS pin output high or low when the module alarms
	Parameter	—
	Return Value	Alarm level: 0x08: Output high 0x00: Output low
	Note	—
19	uint16_t getAlarmThreshold()	
	Description	Get the alarm CO concentration threshold
	Parameter	—
	Return Value	CO alarm threshold, unit: ppm
	Note	—

20	uint16_t getExitAlarmThreshold()	
	Description	Get the exit alarm CO concentration
	Parameter	—
	Return Value	Exit alarm CO concentration threshold, unit: ppm
	Note	—
Parameter Configuration		
21	uint8_t setAutoTx(uint8_t autoTx= AUTO)	
	Description	Set whether the module automatically outputs all the module data through TX pin
	Parameter	autoTx: Automatic output stasu 0x08(AUTO): Automatic output (default) 0x00(PASSIVE): No automatic output
	Return Value	Execution result(1)
	Note	1. The information output period is about 1s 2. The module is unable to receive any instructions during the information transmission period (about 40ms)
22	uint8_t setStatusPinActiveMode(uint8_t statusMode = HIGH_LEVEL)	
	Description	Set the STSTUS pin output level when the module alarms
	Parameter	statusMode: STSTUS pin output level when the module alarms 0x08(HIGH_LEVEL): Output high (default) 0x00(LOW_LEVEL): Output low
	Return Value	Execution result(1)
	Note	—
23	uint8_t setAlarmThreshold(uint16_t alarmThreshold = 180)	
	Description	Set the CO alarm concentration
	Parameter	alarmThreshold: CO alarm concentration (defaults to 180ppm), unit: ppm
	Return Value	Execution result(1)
	Note	—
24	uint8_t setExitAlarmThreshold(uint8_t exitAlarmThreshold = 55)	
	Description	Set the exit alarm CO concentration value
	Parameter	exitAlarmThreshold: Exit alarm CO concentration value (defaults to 55), unit: ppm
	Return Value	Execution result(1)
	Note	This value needs to be less than the CO alarm concentration value
25	uint8_t calibrateModule()	
	Description	Start the gas calibration function
	Parameter	—
	Return Value	Execution result(1)
	Note	1. Do not send commands to the module during calibration 2. The calibration should be implemented in standard concentration gas environment, refer to the datasheet for details 3. Wait for the module calibration to be completed after start-up, then the calibrateCountdown() is usable
26	void calibrateCoundown()	
	Description	Wait for the module calibrating to complete
	Parameter	—
	Return Value	—
	Note	1. The end of this function indicates that the calibrating is completed 2. When the module is powered on, this function will determine whether its automatic output function is enabled a. Enable: Keep reading the remaining preheating time that is automatically output from information packet, exit the function when the time is zero; b. Disable: Exit the function after delaying the set calibration time of about 120s

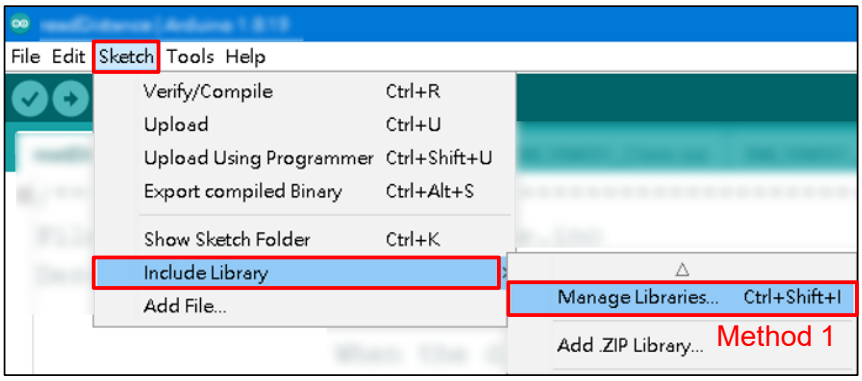
Note 1: 0 – Instruction execution success; 1 – Data verification error in module response;
2 – Communication timeout

Arduino Lib Download and Installation

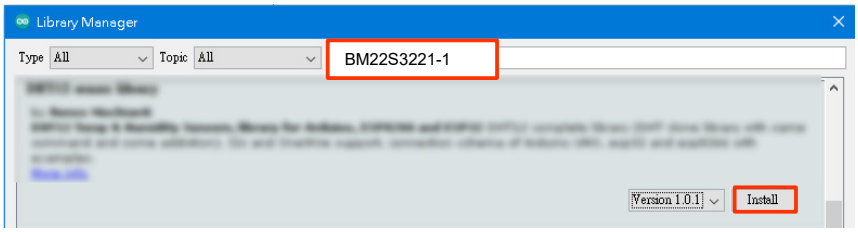
BM22S3221-1 Library: Refer to the following two methods to install the BM22S3221-1 Arduino Library.

Method 1: Search for installation

Search for installation: Arduino IDE → Sketch → Include Library → Manage Libraries... → Search BM22S3221-1 → Install



Search for Installation Step 1

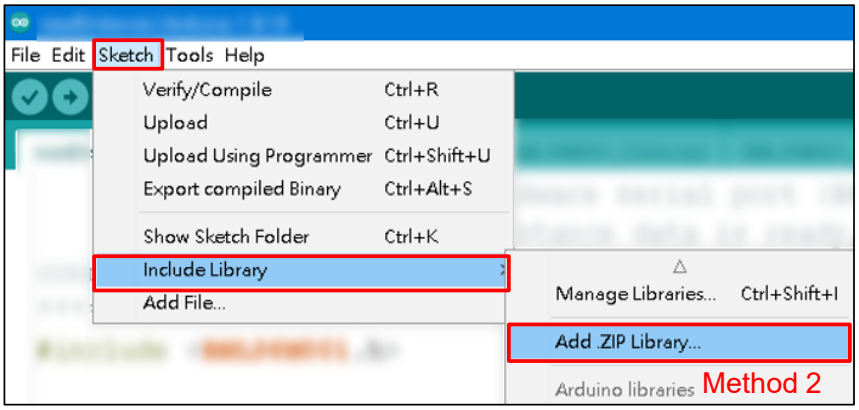


Search for Installation Step 2

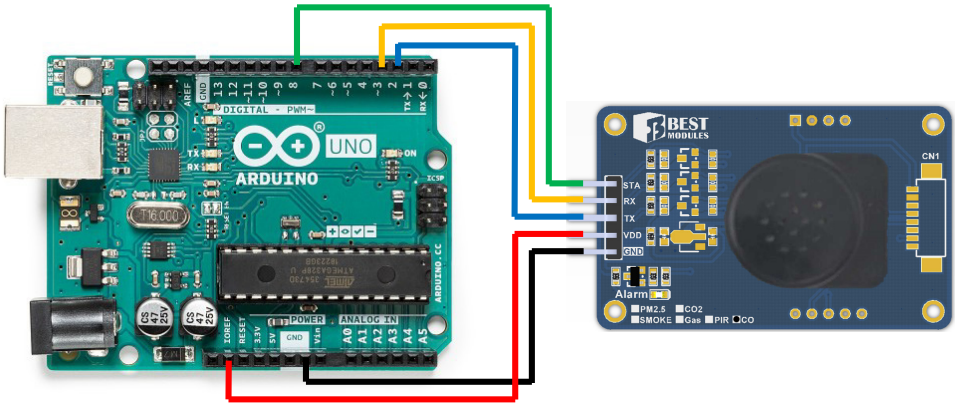
Method 2: Download before adding a ZIP library

Download method: Open the Best Modules official website (https://www.bestmodulescorp.com/bm22s3221-1.html) and download the BM22S3221-1 Library from "Arduino example program" under the "DOCUMENTS" menu.

Add .ZIP library: Arduino IDE → Sketch → Include Library → Add .ZIP Library....



Arduino Examples



Physical Connection Diagram

Example1: readStatusPin

Example function: Determine whether there is an alarm by querying the STATUS pin status

- When the module does not alarm, the LED (D13) flashing cycle is 1s
- When the module alarms, the LED (D13) flashing cycle is 0.2s

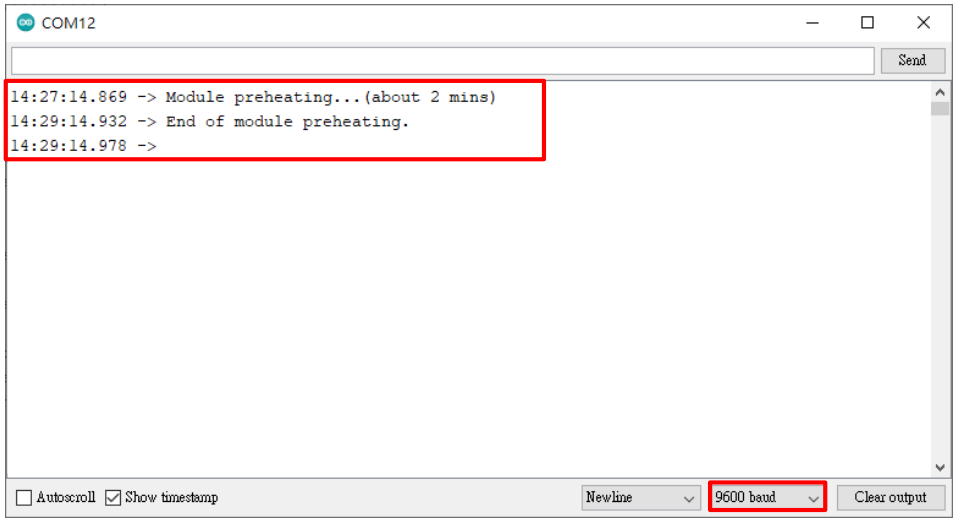
1. Open the example: Arduino IDE → File → Examples → Select Lib (BM22S3221-1) → Select example (readStatusPin)
2. Example description:
   a. Create object & initialise object

```
#include < BM22S3221-1.h> // Call the BM22S3221-1 library
#define LED (13)           // LED control Pin:13

/* Create object & Set Software serial pin */
BM22S3221_1 CO(8, 2, 3);  // Softeware serial: 8→STATUS, 2→RX, 3→TX
void setup()
{
  /* Module initialisation */
  CO.begin(); // Softeware serial initialisation (baud rate: 9600bps),
              // set the Pin 8 to input mode

  /* Configure serial monitor */
  Serial.begin(9600);     // Serial initialisation, baud rate: 9600bps

  /* LED control pin initialisation */
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);

  /* Wait for the module preheating to complete */
  Serial.println("Module preheating...(about 2 mins)");
  CO.preheatCountdown(); // Wait for the module preheating to complete
  Serial.println("End of module preheating.");
  Serial.println();
}
```

b. According to the STATUS (whether the module alarms or not) change the LED flashing cycle

```
void loop()
{
  if (CO.getSTATUS() == 0)
  {
    /* No alarm, LED flashing cycle is 1s */
    digitalWrite(LED, HIGH);
    delay(500);
    digitalWrite(LED, LOW);
    delay(500);
  }
  if (CO.getSTATUS() == 1)
  {
    /* Alarm, LED flashing cycle is 0.2s */
    digitalWrite(LED, HIGH);
    delay(100);
    digitalWrite(LED, LOW);
    delay(100);
  }
}
```

3. Open the serial monitor and select the baud rate to be 9600; the serial monitor will display as follows; prompt to observe the LED flashing status after the preheating is completed



Example2: readCOValue

Example function: Receive the module automatic output information per second, and print the CO relevant information to the serial monitor

1. Open the example: Arduino IDE → File → Examples → Select Lib (BM22S3221-1) → Select example (readCOValue)
2. Example description:

a. Create object & initialise object

```
#include < BM22S3221-1.h> // Call the BM22S3221-1 library

/* Create arrays and variables to store data */
uint8_t moduleInfo[32] = {0};
uint16_t ADValue, COValue, COAlarmThreshold;

/* Create object & Set Software serial pin */
BM22S3221_1 CO(8, 2, 3);  // Softeware serial: 8→STATUS, 2→RX, 3→TX
void setup()
{
  /* Initialise module */
  CO.begin(); // Software serial initialisation (baud rate: 9600bps),
              //set the Pin 8 to input mode
  /* Configure serial monitor */
  Serial.begin(9600);    // Initialise Serial, baud rate: 9600bps

  /* Wait for the module preheating to complete */
  Serial.println("Module preheating...(about 2 mins)");
  CO.preheatCountdown(); // Wait for the module preheating to complete
  Serial.println("End of module preheating.");
  Serial.println();
}
```

b. Receive the module automatic output information

```
void loop()
{
  if (CO.isInfoPackage() == true) // Poll whether the module sent data
                                  // has been received
  {
    CO.readInfoPackage(moduleInfo); // Read the module sent data to
                                    // moduleInfo[]
    printInfo(); // Print part of the module information
  }
}
```

c. Print part of the information according to the received data

```
void printInfo()
{
  /* Print the CO alarm threshold value (PPM) */
  Serial.print("CO alarm threshold: ");
  COAlarmThreshold = (moduleInfo[21] << 8) + moduleInfo[22];
  Serial.print(COAlarmThreshold);
  Serial.println(" PPM");

  /* Print the current CO concentration (PPM) */
  Serial.print("CO concentration: ");
  COValue = (moduleInfo[9] << 8) + moduleInfo[10];
  Serial.print(COValue);
  Serial.println(" PPM");

  /* Print the real-time A/D value */
  Serial.print("CO AD Value: ");
  ADValue = (moduleInfo[5] << 8) + moduleInfo[6];
  Serial.println(ADValue);
}
```

3. Open the serial monitor and select the baud rate to be 9600; the serial monitor will display as follows: