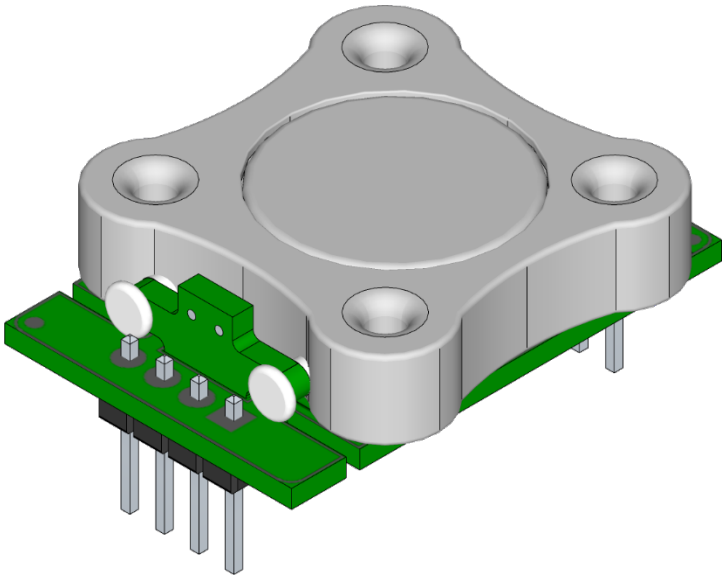
Backplane Systems Technology Pty Ltd
Backplane Systems Technology
TEL 02 9457 6400
sales@backplane.com.au
www.backplane.com.au
Proudly Australian-Owned Since 1989

Formaldehyde Gas Sensor Module IAGM1

Specification Sheet Rev.0.3



IAGM1 is a digital formaldehyde gas sensor. Idea for indoor air quality monitoring, security monitoring or wireless sensor networks to detect formaldehyde concentration near the installation location. This sensor is a proven and maintenance-free technology, designed for high performance and reliability.

Key Feature & Benefits:

- Fast response and recovery time:
- High stability & long life:
- Low cost but high performance:
- Wide operating ranges:
    - temperature: -10 to +50°C
    - humidity: 15 to 90%
    - VDD: 3.1V to 3.5V
- Hassle-free:

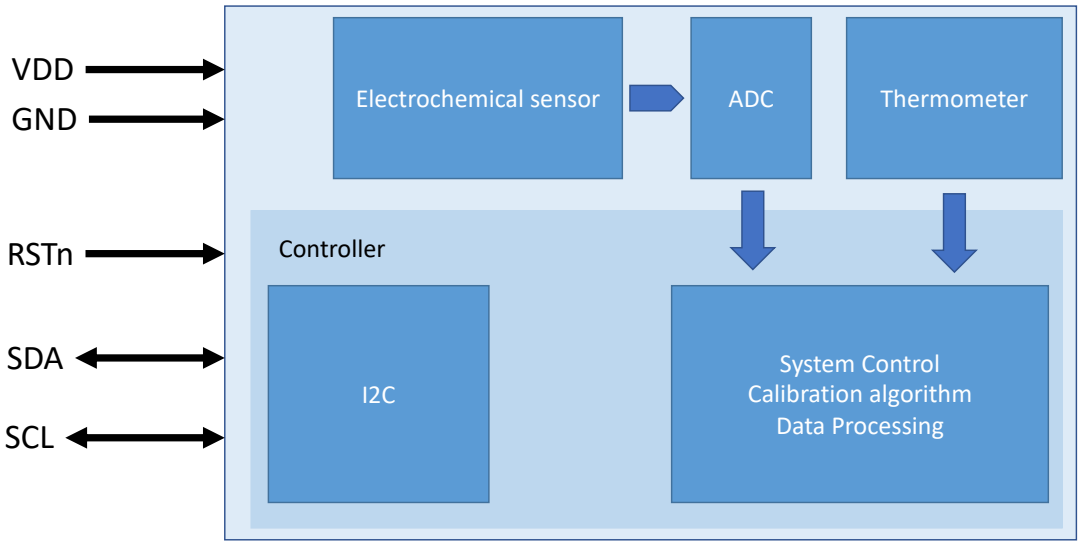
  on-chip data processing – no need for external libraries – no impact on MCU

Applications:

- Building Automation / smart home / HVAC
    - Demand-controlled ventilation
    - Smart thermostats
- Home appliances
    - Air cleaners
    - Purifiers
- Air quality monitors
- IoT devices

Block diagram:

The IAGM1 digital formaldehyde gas sensor based on electrochemical technology, and a controller as shown in the functional block diagram below.



Specification:

The following figure details the electrical characteristics of the sensor.

Model	IAGM1	
Detection	Formaldehyde	
Principle	Electrochemical	
Measurement range	0.01 to 5	ppm
Resolution	0.01	ppm
Accuracy	±0.04±10% of reading	ppm
Response time(T90)	<30	sec
Operation temp.	-10 to 50	°C
Operation Humidity	15 to 90	%RH
Expected operating life	3	years
Power supply	3.1 to 3.5	V
Power consumption	<100	mW
Interface	I^2C	
Dimension(mm)	32(L) 25(W) 10(H)	mm

Interfering Substances:

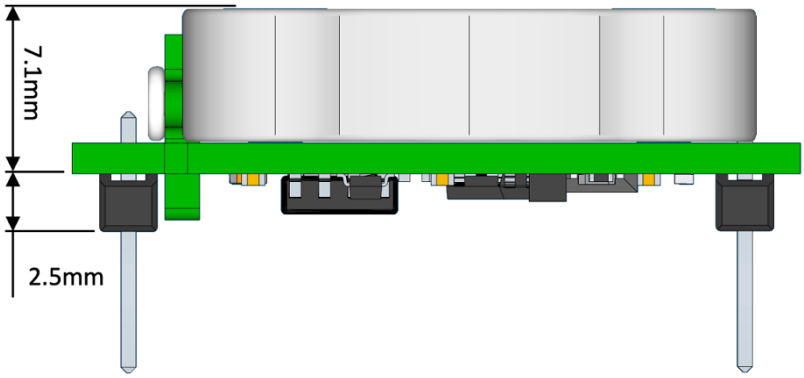
Below table shows the typical response of sensor to interference gases under standard test conditions.

Substance	Cross sensitivity (%)
CO	1
H2S	No data
H2	0.1
SO2	12
NO2	No data
NO	No data
Cl2	-3
C2H4	No data
NH3	0
CO2	0
Ethanol	45
Phenol	7
Water vapour	0*

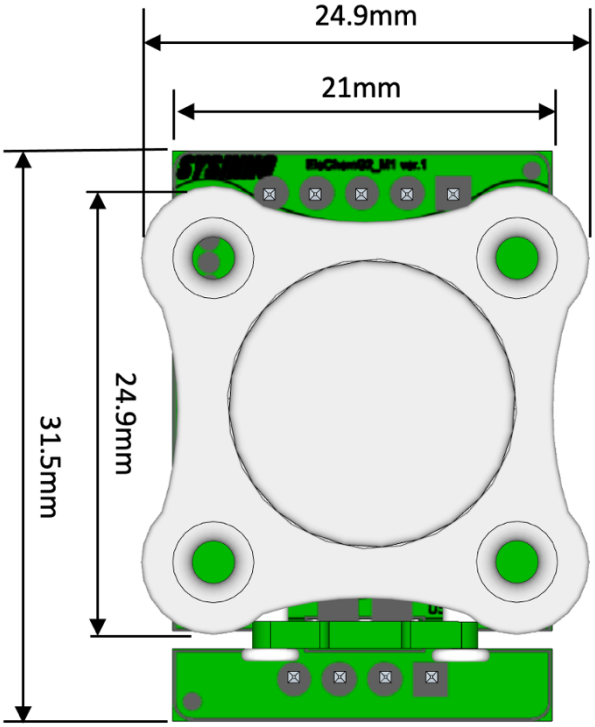
***NB: Within specified range. Step changes in %RH produce short term transient response**

Dimensions:

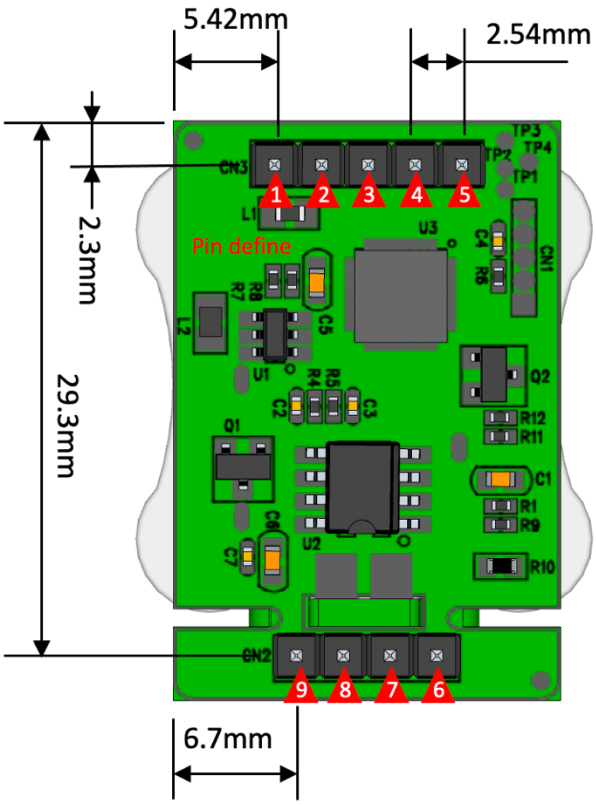
Side view



Top view



Bottom view



Pin assignment:

Pins	Name	Type	Function
1	V_{DD}	Supply	Power supply (3.3V)
2	SDA	Input / Output	I2C bus Bi-Directional data
3	SCL	Input / Output	I2C bus Bi-Directional clock
4	GND	Supply	Ground
5	RSTn	Input	Reset pin(Low Voltage reset)
6	GND	Supply	Ground
7	NC	Input / Output	Reserved
8	NC	Input / Output	Reserved
9	NC	---	Reserved

I2C Communication:

■ I2C description:

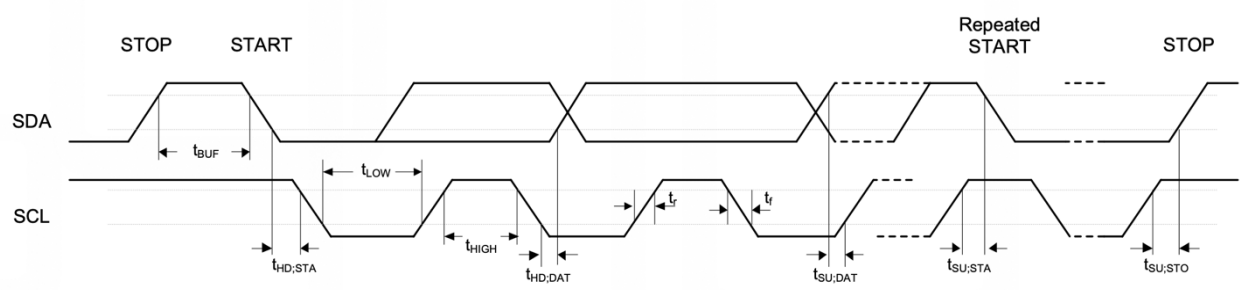
This sensor is an I2C slave device with a fixed 7-bit address 5Dh. The I2C interface supports standard (100kbit/s), and fast (400kbit/s) mode. Details on I2C protocol is according to I2C-bus specifications [UM10204, I2C-bus specification and user manual, Rev. 6, 4 April 2014].

The device applies all mandatory I2C protocol features for slaves: START, STOP, Acknowledge and 7-bit slave address. None of the other optional features (10-bit slave address, general call, software reset or Device ID) are supported, nor are the master features (Synchronization, Arbitration, START byte).

The Host System, as an I2C master, can directly read or write values to one of the registers by first sending the single byte register address. This sensor implements “auto increment” which means that it is possible to read or write multiple bytes* (e.g. read multiple DATA_X bytes) in a single transaction.

***NB: Please do not read or write more than 16 bytes.**

■ I2C timing information:



SYMBOL	PARAMETER	STANDARD MODE[1][2]		FAST MODE[1][2]		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{LOW}	SCL low period					µs
t_{HIGH}	SCL high period					µs
$t_{SU;STA}$	Repeated START condition setup time					µs
$t_{HD;STA}$	START condition hold time	4	-	0.6	-	µs
$t_{SU;STO}$	STOP condition setup time	4	-	0.6	-	µs
t_{BUF}	Bus free time	4.7[3]	-	1.2[3]	-	µs
$t_{SU;DAT}$	Data setup time	250	-	100	-	ns
$t_{HD;DAT}$	Data hold time	0[4]	3.45[5]	0[4]	0.8[5]	µs
t_r	SCL/SDA rise time	-	1000	20+0.1Cb	300	ns
t_f	SCL/SDA fall time	-	300	-	300	ns
C_b	Capacitive load for each bus line	-	400	-	400	pF

1. Guaranteed by design, not tested in production.

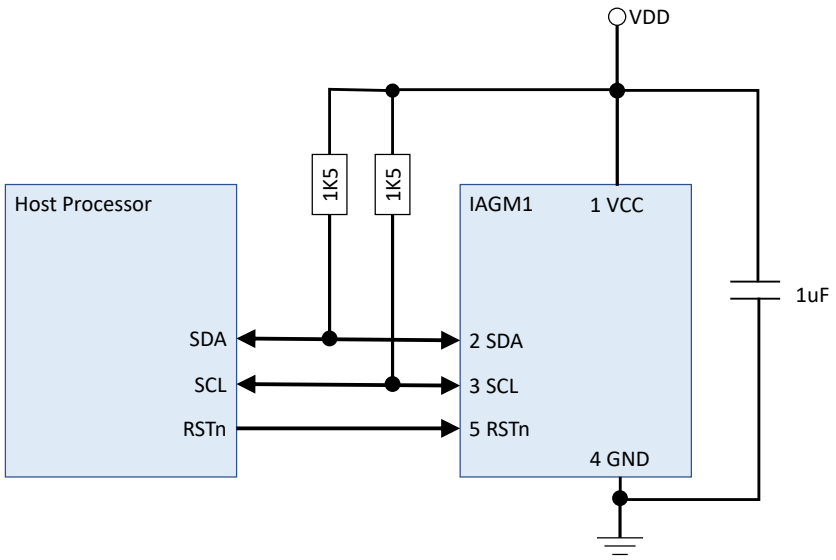
2. I2C controller must be retriggered immediately at slave mode after receiving STOP condition.

3. The device must internally provide a hold time of at least 300 ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.

4. The maximum hold time of the Start condition has only to be met if the interface does not stretch the low period of SCL signal.

■ I2C operation circuitry:

The recommended application circuit for the sensor I2C interface operation is shown in below.



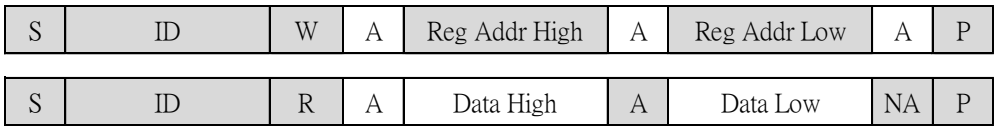
■ I2C Access Protocol:

I2C write operation:

TBD

I2C read operation:

Write register address first, then read data.



Master to Slave
Slave to Master

ID = 5Dh (CO Module 7-bits I2C address)
S = START condition
P = STOP condition
A = acknowledge (SDA LOW)
NA = not acknowledge (SDA HIGH)
W = WRITE (SDA LOW)
R = READ (SDA HIGH)

I2C register:

Address	Data	Size	Access	Description
0000h	PID	8	Read	Product ID Return device product code. (ASCII Code format.) e.g. IAGM1
0008h	FW	8	Read	Firmware version Return device firmware version. (ASCII Code format.)
0010h	SNO	16	Read	Serial Number Return device serial Number. (ASCII Code format.)
0020h	Status	1	Read	Module status 0: Normal operation 1: Warm-Up phase 2: Error case
0021h	HCHO	2	Read	Formaldehyde gas concentration unit: 0.01 ppm e.g. 100 = 1 ppm

■ Example Code:

```
#define SlaveDeviceID              0x5D
#define ProductID_RegAddr          0x00
#define FWVersion_RegAddr          0x08
#define SNO_RegAddr                0x10
#define HCHOData_RegAddr           0x21
#define HCHOData_Length            2
#define INI_HCHO_ppm               0          //unit 0.01ppm
#define UL_HCHO_ppm                500        //unit 0.01ppm
#define LL_HCHO_ppm                0          //unit 0.01ppm
#define IIROrder                   4
#define HCHO_Slope                 1     //Declare it as a variable for runtime calibration.
#define HCHO_Offset                0     //Declare it as a variable for runtime calibration.
#define     BYTE0(arg) (*((Uchar *)&(arg)     + 0))
#define     BYTE1(arg) (*((Uchar *)&(arg)     + 1))
//==========================================================================
// Function    :I2C_ReadData
// Format      :void  I2C_ReadData(uint8_t DeviceID, uint16_t RegAddr, uint8_t *i2cbuf, uint8_t Length)
// Explain     :Read device data via i2c
// Parameter   :DeviceID : I2C slave device address
//              RegAddr  : Data register address
//              i2cbuf   : Store the data read from the device.
//              Length   : Indicate the number of reading bytes.
// Return      :Error code: 0: success, !=0: fail
//==========================================================================
extern uint32_t I2C_ReadData(uint8_t DeviceID, uint16_t RegAddr, uint8_t *i2cbuf, uint8_t Length);
int16_t    ErrorCount = 0;
int32_t    HCHOIIR = INI_HCHO_ppm;
int32_t    iTmpHCHO, HCHOData;
uint8_t    I2CBuffer[16];  //for I2C buffer
int main(void)
{
   //Do system initialization based on your host MCU.
   system_init();
   //Below is an example showing how to read HCHO module data once per second.
   while(1)
   {

      if(I2C_ReadData(SlaveDeviceID, HCHOData_RegAddr, I2CBuffer, HCHOData_Length) != 0) //Read HCHO module data
      {
         //Read HCHO data fail! Do some error process, below is an example.
         ErrorCount++;
      }
      else
      {
         ErrorCount = 0;
         BYTE1(iTmpHCHO)=I2CBuffer[0];
         BYTE0(iTmpHCHO)=I2CBuffer[1];
      }
      //Add some data filters here, below is an example.
      if (HCHOIIR <= INI_HCHO_ppm)
         HCHOIIR = iTmpHCHO;
      else
         HCHOIIR = ((HCHOIIR * (IIROrder-1)) + iTmpHCHO) / IIROrder;//IIR filter (3/4 Old + 1/4 New)

      HCHOData = HCHOIIR * HCHO_Slope + HCHO_Offset; //Calibration mechanisms

      if (HCHOData > UL_HCHO_ppm) HCHOData = UL_HCHO_ppm; //Clamp upper limit
      if (HCHOData < LL_HCHO_ppm) HCHOData = LL_HCHO_ppm; //Clamp lower limit

      if (ErrorCount)
         printf("Read HCHO Data Fail!\n");
      else
         printf("Read HCHO Data OK!   HCHO:%4.2f ppm\n", HCHOData/100); //unit 0.01ppm
      HAL_Delay(1000); //delay 1000ms
   }
}
```