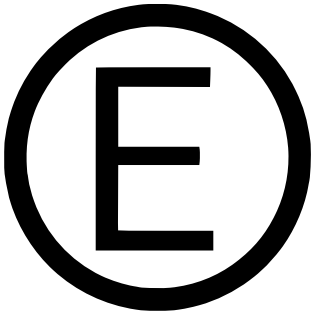
AH3 air speedometer module
Software 2nd revision manual



Barsotion ROy, 2025

Table of contents

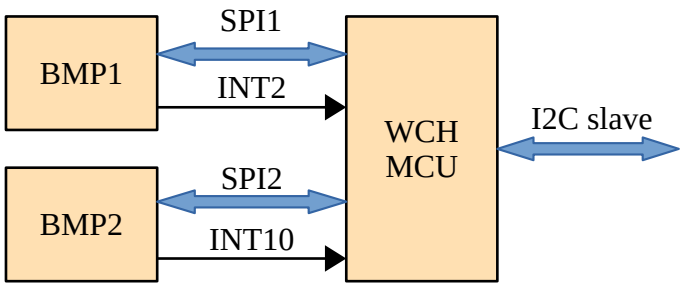
Overview

The Barsotion AH3 airspeed module is an I2C device created to measure atmosphere pressure, airspeed, vertical speed, altitude, air temperature.

This documents describes the work with the AH3 boards with the 1st version of processing microcontroller software.

Structural organization

The module contains 2 BMP388 temperature and pressure sensors and CH32V305 processing microcontroller. The microcontroller reads rata from the sensors via SPI, process it and can give the results to host via I2C interface.



The BMP388 sensors are numerated: 1 and 2. The 1st sensor is a static pressure and temperature sensor, the 2nd is a dynamic sensor.

The microcontroller process the data: it calculates altitude according the static sensor barometer data, differentiates altitude for getting vertical speed, also it can analyze the difference in static and dynamic pressures to get an air speed value. The read temperature and pressure data can be filtered by 1-dimentional Kalman filters. The calibration methods allows to get the filter start parameters and provide the quality of outgoing air speed data.

Future updates [TODO]

The host can set up the AH3 module depending of it's needs. Both of static and dynamic sensors can be disabled if it is necessary. In this (1st) software version host cannot power down any BMP388 sensor, it will still work if even the according enable bit is cleared, so disabling the BMP388 sensor does not affect on the power consumption. However the sensor power down option while disabled will be added in future software versions.

The BMP388 sensors interrupt request signals are wired to the microcontroller's GPIOs. Theoretically, the data reading can be processed via interrupts, but this software does not use it. The program reads the BMP388s' interrupt status registers waiting for the data ready bits setting up. The sensors interrupt handling will be added in future versions, that can also reduce the power consumption.

Sensors settings

The BMP388 maximum ODR value is 200 Hz. The processing microcontroller uses this ODR for providing fast reaction on the external events.

The software does not use the BMP388's data oversampling function. The oversampling can help to get more accurate output value, however it strongly limits the real output data rate.

The software uses BMP388's IIR-filter with coefficient 7. It is a compromise between data rate and accuracy.

The host cannot configure the BMP388 sensors settings for this software revision.

Applications

This airspeed module can be used in copters, model rockets, little airplanes, automobiles and other mobile machines and devices.

Device characteristics

Parameter	Min	Nom	Max
Power supply, V	3.0	3.3	3.6
Power consimption, mA	-	-	16
I2C logic level, V	-	3.3	5
I2C speed, kHz	-	400	1000
Main CPU frequency, MHz	-	96	-
BMP388 sensor ODR, Hz	-	200	-
Pressure oversampling	-	1x	-
Temperature oversampling	-	1x	-
BMP388 IIR-filter coefficient	-	7	-

Register map

Name	Address	Description
WHO_AM_I	0x00	Always read as 0x74.
SOFT_REV	0x01	Soft revision code, for this version 0x02.
ERR	0x02	Error status register, see ERR.
CONFIG	0x03	Config register, see CONFIG.
STATUS	0x04	Status flag register, see STATUS.
TEMP1_0	0x05	Static sensor temperature data, 4-byte float.
TEMP1_1	0x06	
TEMP1_2	0x07	
TEMP1_3	0x08	
PRES1_0	0x09	Static sensor pressure data, 4-byte float.
PRES1_1	0x0A	
PRES1_2	0x0B	
PRES1_3	0x0C	
TEMP2_0	0x0D	Dynamic sensor temperature data, 4-byte float.
TEMP2_1	0x0E	
TEMP2_2	0x0F	
TEMP2_3	0x10	
PRES2_0	0x11	Dynamic sensor pressure data, 4-byte float.
PRES2_1	0x12	
PRES2_2	0x13	
PRES2_3	0x14	
ASPEED_0	0x15	Calculated air speed value, 4-byte float.
ASPEED_1	0x16	
ASPEED_2	0x17	
ASPEED_3	0x18	
ALT_0	0x19	Calculated altitude.
ALT_1	0x1A	
ALT_2	0x1B	
ALT_3	0x1C	

VSPEED_0	0x1D	Calculated vertical speed.
VSPEED_1	0x1E	
VSPEED_2	0x1F	
VSPEED_3	0x20	

WHO_AM_I

7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	0

The register is always read as 0x74.

SOFT_REV

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

The register is always read as 0x01.

ERR

7	6	5	4	3	2	1	0
Reserved			ERR_BMP2_DATA	ERR_BMP1_DATA	ERR_I2C	ERR_BMP2_INIT	ERR_BMP1_INIT

- ERR_BMP1_INIT: Static sensor initialization error.
- ERR_BMP2_INIT: Dynamic sensor initialization error.
- ERR_I2C: I2C protocol error.
- ERR_BMP1_DATA: Static sensor data incorrect value error (is not supported in this version).
- ERR_BMP2_DATA: Dynamic sensor data incorrect value error (is not supported in this version).

CONFIG

7	6	5	4	3	2	1	0
ASPD_CAL	ASPD_EN	BMP2_CAL	BMP1_CAL	BMP2_FILT	BMP1_FILT	BMP2_EN	BMP1_EN

- BMP1_EN: Set to 1 enables static sensor data reading.
- BMP2_EN: Set to 1 enables dynamic sensor data reading.
- BMP1_FILT: Set to 1 enables static sensor data filtering by 1-dimentional Kalman algorithm (the option is available only after calibrating).
- BMP2_FILT: Set to 1 enables dynamic sensor data filtering by 1-dimentional Kalman algorithm (the option is available only after calibrating).
- BMP1_CAL: Set to starts static sensor calibration. Cleared automatically after the calibration has finished. While the calibration time goes on, the BUSY bit in STATUS register is set to 1.

- BMP2_CAL: Set to starts dynamic sensor calibration. Cleared automatically after the calibration has finished. While the calibration time goes on, the BUSY bit in STATUS register is set to 1.
- ASPD_EN: Set to 1 enables air speed calculations. The option is only available if both of BMP1_EN and BMP2_EN bits are set to 1. Note that the barometer sensors should be calibrated before air speed calculations starts, in the other way the output data will contain a non-normed offset.
- ASPD_CAL: Set to 1 starts the airspeed calibrating.

STATUS

7	6	5	4	3	2	1	0
BUSY	ASPD_DRDY	Reserved				BMP2_DRDY	BMP1_DRDY

- BMP1_DRDY: is set to 1 when the first sensor temperature and pressure data are ready to be read. The bit is cleared automatically when the STATUS register is read.
- BMP2_DRDY: is set to 1 when the second sensor temperature and pressure data are ready to be read. The bit is cleared automatically when the STATUS register is read.
- ASPD_DRDY: is set to 1 when the air speed value is ready to be read. The bit is cleared automatically when the STATUS register is read.
- BUSY: is set when any sensor is calibrating. While the calibrating time the AH3 module do not update other sensor output data and air speed calculations are stopped too. The bit is cleared automatically when the calibration done.

Output data format

All the output data are in the standard IEEE-754 4-byte float format, used in the C language. That means:

0th register:

7	6	5	4	3	2	1	0
Mantissa							Man. LSb

1st register:

15	14	13	12	11	10	8	7
Mantissa							

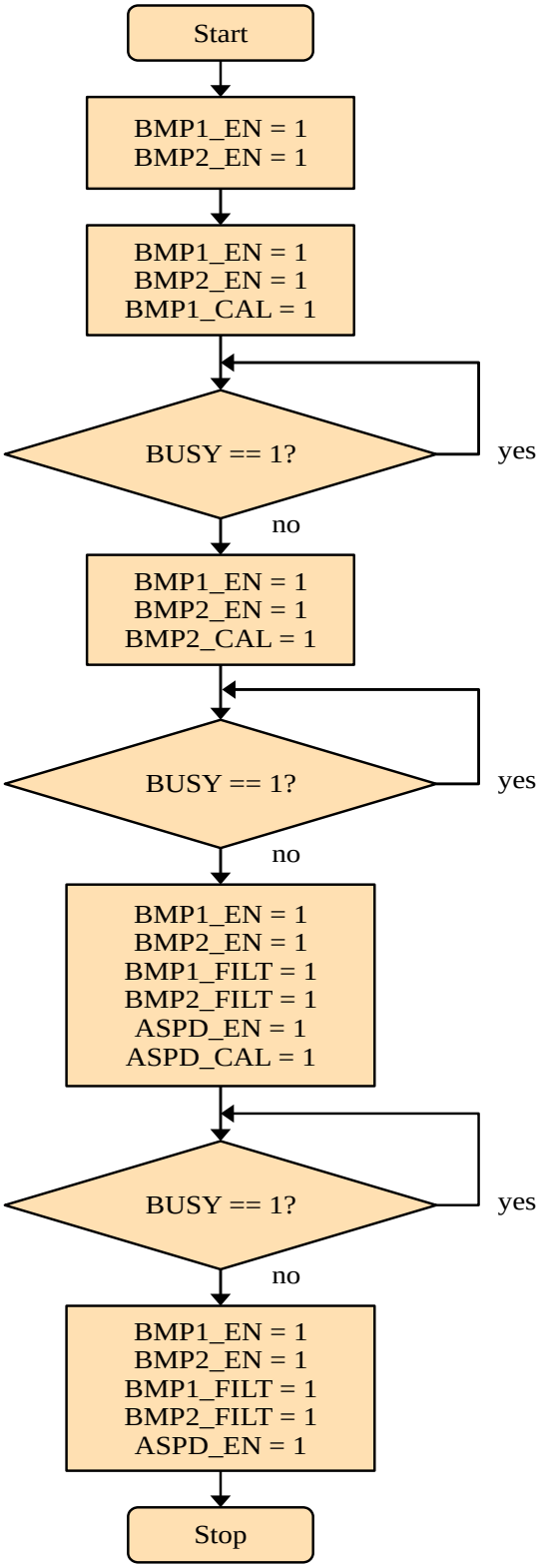
2nd register:

23	22	21	20	19	18	17	16
Exp. LSb	Man. MSb	Mantissa					

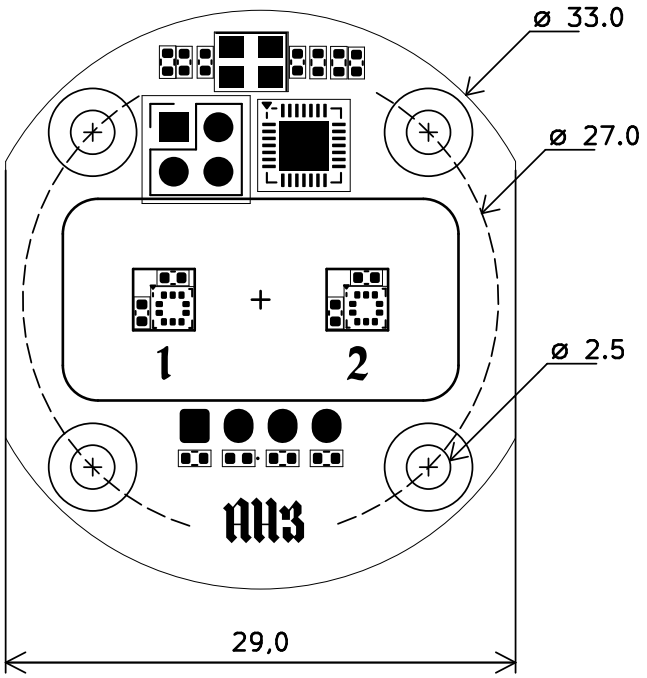
3rd register:

31	30	29	28	27	26	25	24
Sign	Exp. MSb	Exponent					

Initialization algorithm fair speed calculation enabled



Dimentions



2nd revision

AH3 software 2nd revision was presented 30th June 2025 after the error was found in internal BMP388 driver. The P1 and P2 correction coefficients was read incorrectly. The error was fixed in new version. The 1st version is deprecated, it is recommended to use 2nd revision.

```
@@ -91,17 +91,45 @@ uint8_t BMP388_t::readCalibrationData()
91        _par.t1 = (float)(((uint16_t)data[1] << 8) | ((uint16_t)data[0]));
92        _par.t2 = (float)(((uint16_t)data[3] << 8) | ((uint16_t)data[2]));
93        _par.t3 = (float)((int8_t)data[4]);
94  -     _par.p1 = (float)(((int16_t)data[6] << 8) | ((int16_t)data[5]));
95  -     _par.p2 = (float)(((int16_t)data[8] << 8) | ((int16_t)data[7]));
96  -     _par.p3 = (float)((int8_t)data[9]);
97        _par.p4 = (float)((int8_t)data[10]);
98        _par.p5 = (float)(((uint16_t)data[12] << 8) | ((uint16_t)data[11]));
99        _par.p6 = (float)(((uint16_t)data[14] << 8) | ((uint16_t)data[13]));
100       _par.p7 = (float)((int8_t)data[15]);
101       _par.p8 = (float)((int8_t)data[16]);
102 -     _par.p9 = (float)(((int16_t)data[18] << 8) | ((int16_t)data[17]));
103       _par.p10 = (float)((int8_t)data[19]);
104       _par.p11 = (float)((int8_t)data[20]);
```

```
91        _par.t1 = (float)(((uint16_t)data[1] << 8) | ((uint16_t)data[0]));
92        _par.t2 = (float)(((uint16_t)data[3] << 8) | ((uint16_t)data[2]));
93        _par.t3 = (float)((int8_t)data[4]);
94  +     _par.p1 = (float)(int16_t)(((int16_t)data[6] << 8) | ((int16_t)data[5]));
95  +     _par.p2 = (float)(int16_t)((int16_t)data[8] << 8) | ((int16_t)data[7]));
96  +     _par.p3 = (float)((uint8_t)data[9]);
97        _par.p4 = (float)((int8_t)data[10]);
98        _par.p5 = (float)(((uint16_t)data[12] << 8) | ((uint16_t)data[11]));
99        _par.p6 = (float)(((uint16_t)data[14] << 8) | ((uint16_t)data[13]));
100       _par.p7 = (float)((int8_t)data[15]);
101       _par.p8 = (float)((int8_t)data[16]);
102 +     _par.p9 = (float)(int16_t)(((int16_t)data[18] << 8) | ((int16_t)data[17]));
103       _par.p10 = (float)((int8_t)data[19]);
104       _par.p11 = (float)((int8_t)data[20]);
```

Revision history

Date	Modification
Jun 30, 2025	BMP388 driver bug fixed.
Mar 8, 2025	New dimention picture, revision history addition.
Jan 21, 2025	Document creation.

Contacts

If you have any questions, feel free to write via email or Telegram.

Email: barsotion@yandex.ru

Telegram: @barsybarsevich