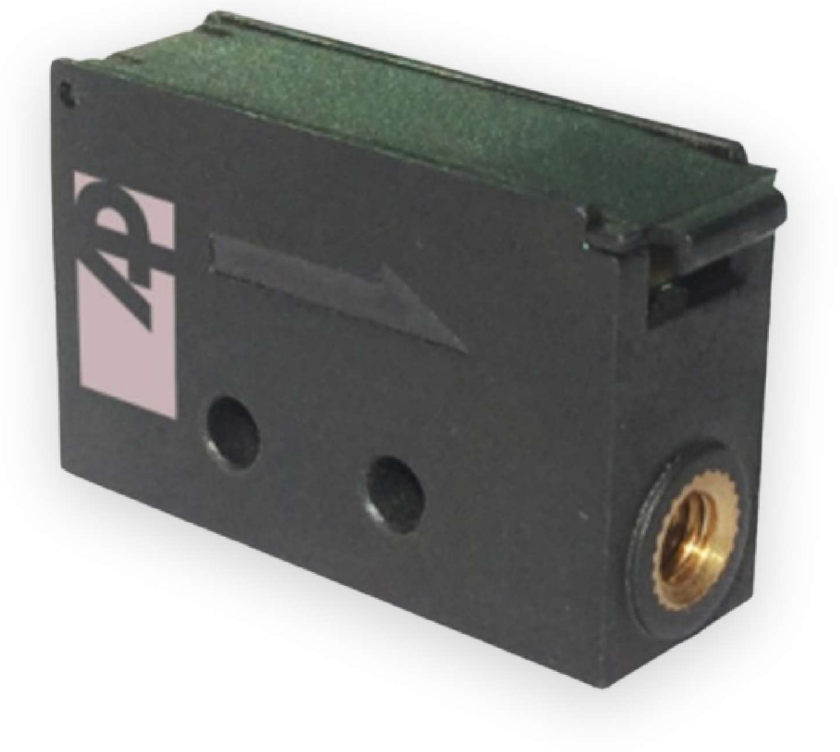
PFLOW2008

I2C Communication Guide



Angst+Pfister

Table of Contents

PFLOW2008

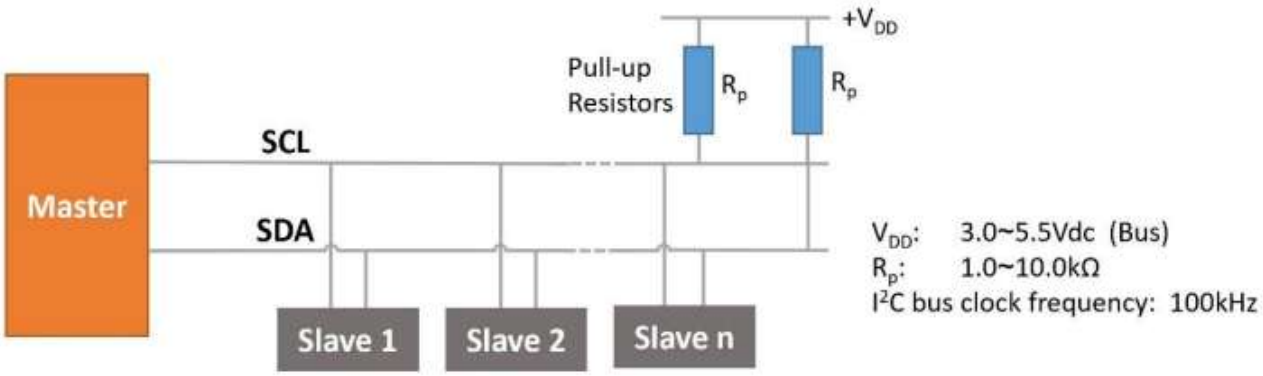
Contents

1 I2C Setup

PFLOW2008

1.1 Wiring diagram

The image below shows a the recommended wiring for the PFLOW2008. The PFLOW2008 requires pull-up resistors on both the SDA and SCL lines. The pull-up voltage should be between 3.0V and 5.5V. Warning: Do not connect the pull-up resistors to the PFLOW2008 supply voltage, this can damage most microcontrollers.



1 I2C Setup

1.2 Command Overview

The PFLOW2008 offers seven I2C commands, which are described in the table below. The commands are single-byte commands, followed by data bytes if required.

COMMAND BYTE (HEX)	LENGTH (INT8)	COMMAND NAME	READ/WRITE	NOTES
05H	1	I2C Address	Write	Bit 0 is the R/W flag bit; Bit 1~ Bit 7 are available.
0BH	1	Filter Depth	Write	Int8 data byte, 0 - 254.
1CH	1	Flow Offset	Write	1 byte dummy data, ensure no-flow condition.
82H	12	Serial Number	Read	ASCII
83H	4	Flow Rate	Read	Flow = (B0 | B1 | B2 | B3) / 1000
85H	1	I2C Address	Read	7-bit I2C Address.
8BH	1	Filter Depth	Read	Filter depth setpoint, 0 - 254.

1 I2C Setup

1.3 Additional Information

For electrical characteristics and additional information on the PFLOW2008 I2C bus, please refer to the User manual at PFLOW2008 User Manual

2 I2C Functions

2.1 Read Serial Number

The following function is used to read and print the Serial Number of the PFLOW2008. The Serial Number is a 12-byte ASCII string that uniquely identifies the device. The function writes the Serial Number to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void get_serial_number(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x82;
        uint8_t RX_Buffer[12] = {0};

        HAL_StatusTypeDef status;
        char serial[13];

        status = HAL_I2C_Mem_Read(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, RX_Buffer, 12, 1000);
        if(status != HAL_OK) {
                snprintf(msg, len, "I2C transmission failed, status = %d, error = 0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
                return;
        }

        memcpy(serial, RX_Buffer, 12);
        serial[12] = '\0';

        snprintf(msg, len, "Serial number: %s\r\n", serial);
}
```

2.2 Read Flow Data

The following function is used to read and print the flow rate from the PFLOW2008. The flow data is returned as 4 bytes. The flow is calculated using the formula: Flow = (int32)(data[0] << 24 | data[1] << 16 | data[2] << 8 | data[3]) / 1000.0) For the PFLOW2008, the flow rate is in sccm (standard cubic centimeters per minute), for the PFLOW2008E, the flow rate is in SLPM (standard liters per minute). The converted flow rate is printed to the provided message buffer. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void get_flow_data(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x83;
        uint8_t RX_Buffer[4] = {0};

        HAL_StatusTypeDef status;

        status = HAL_I2C_Mem_Read(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, RX_Buffer, 4, 1000);
        if(status != HAL_OK) {
                snprintf(msg, len, "Flow I2C transmission failed, status = %d, error = 0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
                return;
        }

        int32_t raw_flow = (int32_t)(((uint32_t)RX_Buffer[0] << 24) | ((uint32_t)RX_Buffer[1] << 16) |
((uint32_t)RX_Buffer[2] << 8)  | ((uint32_t)RX_Buffer[3]));
        int32_t whole = raw_flow / 1000;
        int32_t frac = raw_flow % 1000;

        if (frac < 0) {frac = - frac;}

        snprintf(msg, len, "Flow: %ld.%03ld SCCM\r\n", (long)whole, (long)frac);

}
```

2 I2C Functions

2.3 Read Filter Depth

The following function is used to read and print the Filter Depth of the PFLOW2008. The Filter Depth is one-byte integer in the range of 0 - 254. The function writes the Filter Depth to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void get_filter_depth(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x8B;
        uint8_t filter_depth;

        HAL_StatusTypeDef status;

        status = HAL_I2C_Mem_Read(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, &filter_depth, 1, 1000);

        if(status != HAL_OK) {
                snprintf(msg, len, "Filter depth I2C transmission failed, status = %d, error = 0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
                return;
        }

        snprintf(msg, len, "Current filter depth: %d\r\n", filter_depth);
}
```

2.4 Read I2C Address

The following function is used to read and print the I2C address of the PFLOW2008. The I2C address is a 7-bit value, shifted right by one bit to fit into the 8-bit I2C address format. The function writes the I2C address to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void get_i2c_addr(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x85;
        uint8_t i2c_Addr;

        HAL_StatusTypeDef status;

        status = HAL_I2C_Mem_Read(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, &i2c_Addr, 1, 1000);

        if(status != HAL_OK) {
                snprintf(msg, len, "Address read I2C transmission failed, status = %d, error = 0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
                return;
        }

        snprintf(msg, len, "I2C address: 0x%02X\r\n", (i2c_Addr >> 1));
}
```

2 I2C Functions

2.5 Set New I2C Address

The following function is used to set a new I2C address for the PFLOW2008. The I2C address is a 7-bit value, it needs to be shifted left by one bit before transmitting it to the sensor. The function writes the new I2C address to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void set_new_i2c_address(I2C_HandleTypeDef *hi2c, uint8_t current_addr, uint8_t new_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x05;
        uint8_t data;

        HAL_StatusTypeDef status;

        data = (uint8_t)(new_addr << 1);
        status = HAL_I2C_Mem_Write(hi2c, current_addr, CMD, I2C_MEMADD_SIZE_8BIT, &data, 1, 1000);

        if (status != HAL_OK) {
                snprintf(msg, len, "Set I2C address failed, status=%d error=0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
            return;
        }

        snprintf(msg, len, "I2C address change command sent: old=0x%02X new=0x%02X written_byte=0x%02X\r\n", current_addr,
new_addr, data);
}
```

2.6 Set Flowrate Offset

The following function is used to set the flowrate offset for the PFLOW2008. The function writes a confirmation message to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void set_flowrate_offset(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, char* msg, uint16_t len) {
        uint8_t CMD = 0x1C;
        uint8_t data = 0x00;

        HAL_StatusTypeDef status;

        status = HAL_I2C_Mem_Write(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, &data, 1, 1000);

        if (status != HAL_OK) {
                snprintf(msg, len, "Set flowrate offset address failed, status=%d error=0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
            return;
        }

        snprintf(msg, len, "Flowrate offset reset.\r\n");
}
```

2 I2C Functions

2.7 Set New Filter Depth

The following function is used to set a new filter depth for the PFLOW2008. The function writes a confirmation message to the provided message buffer, which can be used for printing and logging. If Errors occur during the I2C transmission, an error message is written to the message buffer instead.

```
void set_filter_depth(I2C_HandleTypeDef *hi2c, uint8_t i2c_addr, uint8_t filter_depth, char* msg, uint16_t len) {
        uint8_t CMD = 0x0B;
        uint8_t data = filter_depth;

        HAL_StatusTypeDef status;

        status = HAL_I2C_Mem_Write(hi2c, i2c_addr, CMD, I2C_MEMADD_SIZE_8BIT, &data, 1, 1000);

        if (status != HAL_OK) {
                snprintf(msg, len, "Set filter depth failed, status=%d error=0x%08lX\r\n", status,
HAL_I2C_GetError(hi2c));
            return;
        }

        snprintf(msg, len, "New filter depth %u written to sensor\r\n", data);
}
```

3 Revision History

Revision History

VERSION	DATE	AUTHOR	CHANGE
1.0	2026-06-29	Frey Lars	Document created.

We are here for you. Addresses and Contacts.

Headquarter Switzerland:

Angst+Pfister Sensors and Power AG
Thurgauerstrasse 66
CH-8050 Zurich

Phone +41 44 877 35 00
sensorsandpower@angst-pfister.com

Office Germany:

Angst+Pfister Sensors and Power Deutschland GmbH
Edisonstraße 16
D-85716 Unterschleißheim

Phone +49 89 374 288 87 00
sensorsandpower.de@angst-pfister.com

Office North America:

Angst+Pfister North America Inc.
10391 Brecksville Rd.
US-Brecksville, OH 44141

Phone +1 440 375-5212
info.apus@angst-pfister.com

Scan here and get an overview of personal contacts!

https://sensorsandpower.angst-pfister.com/en/