Arduino-Codes PFLOW4008

I2C address change:

```
////////////////////////////////////////////////
// Angst+Pfister Sensors & Power AG           //
// PFLOW4008 - Arudino I2C Change Address Code //
// Frey Lars                                  //
// 02.04.2024 - Revision 0.1                  //
////////////////////////////////////////////////

#include <Wire.h>
#include <Arduino.h>

// The old address of the sensor (by default 0x01) and the desired new address
uint8_t ADDR_old = 0x01;
uint8_t ADDR_new = 0x05;

/**
 * Check if the address change was successful with an I2C scanner
 * @param void
 * @return void
 */
void I2C_scan() {

  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ ) {

    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0) {

      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4) {
```

```
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

}

/**
 * Check if the address change was successful by requesting
 * data from the new sensor address
 * @param void
 * @return void
 */
void check() {

  ADDR_new = ADDR_new >> 1;

  Wire.beginTransmission(ADDR_new);
  Wire.write(0x00);
  Wire.write(0x3A);
  Wire.endTransmission(false);

  Wire.requestFrom(ADDR_new, 6);

  if(Wire.available() != 0) {
    Serial.print("Address succesfully changed from 0x");
    Serial.print(ADDR_old, HEX);
    Serial.print(" to 0x");
    Serial.println(ADDR_new, HEX);
  }
  else {
    Serial.println("Error: No device found at new address");
  }
}

/**
 * Calculate CRC8 for new address using polynomial 0x07.
 *
 * @param data new address
 * @return Calculated CRC8 value.
 */
uint8_t calculateCRC8(uint8_t data) {

  uint8_t crc = 0; // Initial crc value
```

```
  crc ^= data;
  for (int j = 0; j < 8; j++) {
    if (crc & 0x80) {
      crc = (crc << 1) ^ 0x07;  // CRC-8 polynomial
    }
    else {
      crc <<= 1;
    }
  }
  return crc;
}

void setup() {
  Wire.begin();           // Initialize I2C
  Serial.begin(9600);     // Initialize serial communication
  while (!Serial);        // Wait for the serial port to connect

  //bit shift address one step left because the slave device reads the address
as 15 bits
  ADDR_new = ADDR_new << 1;

  //begins transmission with sensor
  Wire.beginTransmission(ADDR_old);

  //command sent to slave device: 0x00, 0xA4, 0x00, new address (see
application note)
  Wire.write(0x00);
  Wire.write(0xA4);

  Wire.write(0x00);
  Wire.write(ADDR_new);

  //calculates crc from the value of the new address
  uint8_t crc = calculateCRC8(ADDR_new);

  //Send CRC
  Wire.write(crc);
  Wire.endTransmission();

  //Both options to check if change was successful
  check();
  //I2C_scan();
}

void loop() {

}
```

Print serial number:

```
////////////////////////////////////////////////////
// Angst+Pfister Sensors & Power AG               //
// PFLOW4008 - Arudino I2C Read serial number Code //
// Frey Lars                                      //
// 02.04.2024 - Revision 0.1                      //
////////////////////////////////////////////////////

#include <Wire.h>
#include <Arduino.h>

// Constants for device address and data size
#define I2C_ADDRESS 0x01
#define DATA_SIZE 2

// Buffer for storing read bytes from I2C
uint8_t readByte[18];

/**
 * Calculate CRC8 for data array using polynomial 0x07.
 *
 * @param data Pointer to data array.
 * @param size Size of the data array.
 * @return Calculated CRC8 value.
 */
uint8_t calculateCRC8(uint8_t *data, int size) {

  uint8_t crc = 0; // Initial crc value

  // Iterate over each byte of data
  for (int i = 0; i < size; i++) {
    // Process each bit in the byte
    crc ^= data[i];
    for (int j = 0; j < 8; j++) {
      if (crc & 0x80) {
        crc = (crc << 1) ^ 0x07;  // CRC-8 polynomial
      } else {
        crc <<= 1;
      }
    }
  }
  return crc;
}

void setup() {

  Wire.begin();           // Initialize I2C
  Serial.begin(9600);     // Initialize serial communication
}
```

```
void loop() {

  // Begin communication with sensor
  Wire.beginTransmission(I2C_ADDRESS);

  // Send the command to read serial number to sensor
  Wire.write(0x00);
  Wire.write(0x30);
  Wire.endTransmission(false);

  // Request 18 bytes of data from sensor (12 bytes address, 6 bytes crc)
  Wire.requestFrom(I2C_ADDRESS, 18);

  // Check if there are any bytes ready to be sent
  while (Wire.available()) {
    // Fill the buffer with the bytes sent
    for (int i = 1; i <= 18; i++) {
      readByte[i] = Wire.read();
    }
  }

  // Go through all bytes in the buffer
  for (int i = 1; i < 18; i++) {

    // Separate crc bytes
    if (i % 3 == 0) {

      // Create array of the two preceding bytes of the crc byte
      uint8_t data[DATA_SIZE] = { readByte[i - 2], readByte[i - 1] };
      // Calculate crc
      uint8_t calculatedCRC = calculateCRC8(data, DATA_SIZE);
      // Crc byte received from sensor
      uint8_t recievedCRC = readByte[i];

      // Compare calculated and received CRC, output error message if they're
not the same
      if (calculatedCRC != recievedCRC) {
        Serial.println("CRC error");
        break;
      }
    }

    // Print serial number if crc is correct
    else {
      Serial.write(readByte[i]);
    }
  }
  Serial.print("\n\n");
```

```
  Serial.println("-----------------------------------------");

  // Wait for one second
  delay(1000);
}
```

Read flow data:

```
//////////////////////////////////////////
// Angst+Pfister Sensors & Power AG       //
// PFLOW4008 - Arudino I2C Read Flow Code //
// Frey Lars                              //
// 02.04.2024 - Revision 0.1              //
//////////////////////////////////////////

#include <Wire.h>
#include <Arduino.h>

// Constants for device address and data size
#define I2C_ADDRESS 0x01
#define DATA_SIZE 2

// Buffer for storing read bytes from I2C
uint8_t readByte[6];

/**
 * Calculate CRC8 for data array using polynomial 0x07.
 *
 * @param data Pointer to data array.
 * @param size Size of the data array.
 * @return Calculated CRC8 value.
 */
uint8_t calculateCRC8(uint8_t *data, int size) {

  uint8_t crc = 0; // Initial CRC value

  // Iterate over each byte of data
  for (int i = 0; i < size; i++) {
    // Process each bit in the byte
    crc ^= data[i];
    for (int j = 0; j < 8; j++) {
      if (crc & 0x80) {
        crc = (crc << 1) ^ 0x07;  // CRC-8 polynomial
      }
      else {
        crc <<= 1;
      }
    }
  }
  return crc;
}

void setup() {

  Wire.begin();           // Initialize I2C
  Serial.begin(9600);     // Initialize serial communication
```

```
  while(!Serial);         // Wait for the serial port to connect

}

void loop() {

  // Start communication with sensor
  Wire.beginTransmission(I2C_ADDRESS);

  // Send command to read flow (see application note)
  Wire.write(0x00);
  Wire.write(0x3A);
  Wire.endTransmission(false);

  // Request six bytes of data from the sensor
  Wire.requestFrom(I2C_ADDRESS, 6);

  // Check if the sensor has data ready
  while(Wire.available()) {
    //Fill data in the buffer
    for(int i = 0; i < 6; i++) {
      readByte[i] = Wire.read();
    }
  }

  // Extract flow data calculate crc checksums
  uint8_t data1[DATA_SIZE] = {readByte[0], readByte[1]};
  uint8_t data2[DATA_SIZE] = {readByte[3], readByte[4]};

  uint8_t crc1 = calculateCRC8(data1, DATA_SIZE);
  uint8_t crc2 = calculateCRC8(data2, DATA_SIZE);

  // Compare calculated crc with received crc
  if(crc1 == readByte[2] && crc2 == readByte[5]) {

    // Putting together all bytes containing the flow
    long Flow = (long)readByte[0] << 24 | (long)readByte[1] << 16 |
                (long)readByte[3] << 8 | (long)readByte[4];

    // Convert flow data to sscm
    float convertedFlow = Flow / 1000.0;

    // Print flow
    Serial.print(convertedFlow, 3);
    Serial.println(" SLPM");
  }

  // Output error message if crc is wrong
  else {
    Serial.println("CRC error occured. Data might be invalid.");
```

```
  }

  // Wait one second before next measurement
  delay(1000);

}
```

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/