```c
/**
 ********************************************************************************
   * C Library for the BME280 Sensor
 ********************************************************************************
   * @author  Samuel Ruhl, Alexander Meier
   * @date    2017-04-04
   * @file    BME280.c
   * @brief   Contains Functions for using the BME280
 ********************************************************************************
**/

#include "BME280def.h"


int32_t t_fine;
bme280_calib_data bme280_calib;

uint8_t rx_buff[4];
uint8_t xx = 0;

void BME280_select(void){
    PORT0->OUT &= ~(1<<3);
}

void BME280_deselect(void){
    for(int i = 0; i< 100;i++);
    PORT0->OUT |= 1<<3;
}

void spi_send8(uint8_t *data){
    SPI_MASTER_Transmit(&SPI_MASTER_1, &data, 1 );
    while(SPI_MASTER_1.runtime->tx_busy);
}

void spi_send16(uint8_t *data){
    SPI_MASTER_Transmit(&SPI_MASTER_1, &data, 2 );
    while(SPI_MASTER_1.runtime->tx_busy);
}

void spi_send24(uint8_t *data){
    SPI_MASTER_Transmit(&SPI_MASTER_1, &data, 3 );
    while(SPI_MASTER_1.runtime->tx_busy);
}

void spi_tx8_bme280(uint8_t addr, uint8_t data){
    //See Datasheet BME280 p.33     MSB of address is "0"
    addr &= 0x7F;
```

```
48      uint8_t tx[2] = {addr, data};
49
50      BME280_select();
51
52      SPI_MASTER_Transmit(&SPI_MASTER_1, &tx[0], 2 );
53      while(SPI_MASTER_1.runtime->tx_busy);
54
55      BME280_deselect();
56  }
57
58  uint8_t spi_recive8(){
59      uint8_t data = 0;
60      SPI_MASTER_Receive(&SPI_MASTER_1, &rx_buff[0], 1 );
61      while(SPI_MASTER_1.runtime->rx_busy);
62      return rx_buff[0];
63  }
64
65  uint8_t spi_recive16(){
66      uint8_t data = 0;
67      SPI_MASTER_Receive(&SPI_MASTER_1, &rx_buff[0], 2 );
68      while(SPI_MASTER_1.runtime->rx_busy);
69      return (uint16_t)(rx_buff[0]<<8)|(rx_buff[1]);
70  }
71
72  uint32_t spi_recive24(){
73      uint8_t data = 0;
74      SPI_MASTER_Receive(&SPI_MASTER_1, &rx_buff[0], 3 );
75      while(SPI_MASTER_1.runtime->rx_busy);
76      return (uint32_t)(rx_buff[0]<<16)|(rx_buff[1]<<8)|(rx_buff[2]);
77  }
78
79  uint8_t spi_rx8_bme280(uint8_t addr){
80      //See Datasheet BME280 p.33     MSB of address is "1"
81      addr |= 0x80;
82      uint8_t rx_data;
83      BME280_select();    //CS Low active
84
85      spi_send8(addr);
86      rx_data = spi_recive8();
87
88      BME280_deselect();
89      return rx_data;
90  }
91
92  uint16_t spi_rx16_bme280(uint8_t addr){
93      //See Datasheet BME280 p.33     MSB of address is "1"
94      addr |= 0x80;
95      //uint8_t rx_data[2];
96      BME280_select();        //CS Low active
97      spi_send8(addr);
98      //rx_data[0] = spi_recive8();
99      //rx_data[1] = spi_recive8();
100     uint16_t rdata = spi_recive16();
```

```
101
102     BME280_deselect();
103     //return (uint16_t)((rx_data[0]<<8)|(rx_data[1])) ;
104     return rdata;
105 }
106
107 uint32_t spi_rx24_bme280(uint8_t addr){
108     //See Datasheet BME280 p.33     MSB of address is "1"
109     addr |= 0x80;
110
111     BME280_select();        //CS Low active
112
113     spi_send8(addr);
114     uint32_t rdata = spi_recive24();
115
116     BME280_deselect();
117
118     return rdata;
119 }
120
121 uint16_t spi_rx16_bme280_LE(uint8_t addr){
122     uint16_t tmp = spi_rx16_bme280(addr);
123     return (tmp >> 8) | (tmp << 8);
124 }
125
126 int16_t spi_rxS16_bme280(uint8_t addr){
127     return (int16_t) spi_rx16_bme280(addr);
128 }
129
130 int16_t spi_rxS16_bme280_LE(uint8_t addr){
131     return (int16_t) spi_rx16_bme280_LE(addr);
132 }
133
134
135 void BME280_readCoefficients(void)
136 {
137     bme280_calib.dig_T1 = spi_rx16_bme280_LE(BME280_REGISTER_DIG_T1);
138     bme280_calib.dig_T2 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_T2);
139     bme280_calib.dig_T3 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_T3);
140
141     bme280_calib.dig_P1 = spi_rx16_bme280_LE(BME280_REGISTER_DIG_P1);
142     bme280_calib.dig_P2 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P2);
143     bme280_calib.dig_P3 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P3);
144     bme280_calib.dig_P4 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P4);
145     bme280_calib.dig_P5 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P5);
146     bme280_calib.dig_P6 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P6);
147     bme280_calib.dig_P7 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P7);
148     bme280_calib.dig_P8 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P8);
149     bme280_calib.dig_P9 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_P9);
150
151     bme280_calib.dig_H1 = spi_rx8_bme280(BME280_REGISTER_DIG_H1);
152     bme280_calib.dig_H2 = spi_rxS16_bme280_LE(BME280_REGISTER_DIG_H2);
153     bme280_calib.dig_H3 = spi_rx8_bme280(BME280_REGISTER_DIG_H3);
```

```
154     bme280_calib.dig_H4 = (spi_rx8_bme280(BME280_REGISTER_DIG_H4) << 4) |
          (spi_rx8_bme280(BME280_REGISTER_DIG_H4+1) & 0xF);
155     bme280_calib.dig_H5 = (spi_rx8_bme280(BME280_REGISTER_DIG_H5+1) << 4)
          | (spi_rx8_bme280(BME280_REGISTER_DIG_H5) >> 4);
156     bme280_calib.dig_H6 = (int8_t)spi_rx8_bme280(BME280_REGISTER_DIG_H6);
157 }
158
159
160 void BME280_init(void){
161     spi_tx8_bme280(0xF2,0x01) ; //OSR for humidity
162     spi_tx8_bme280(0xF4,0xFF) ; //MAX oversampling and normal Mode
163     spi_tx8_bme280(0xF5,0xA0) ; //Config: 1s, IIR off, SPI 3 wire disabled
164 }
165
166
167 /
      ************************************************************************
      ************************************************************************
      */
168 /*!
169     @brief  Formulas as described in Datasheet BME280 p.23 and p24
170 */
171 /
      ************************************************************************
      ************************************************************************
      */
172
173 float BME280_readTemperature(void)
174 {
175     int32_t var1, var2;
176
177     int32_t adc_T = spi_rx24_bme280(BME280_REGISTER_TEMPDATA);
178     if (adc_T == 0x800000) // value in case temp measurement was disabled
179         return 0;
180     adc_T >>= 4;
181
182     var1 = ((((adc_T>>3) - ((int32_t)bme280_calib.dig_T1 <<1))) *
183             ((int32_t)bme280_calib.dig_T2)) >> 11;
184
185     var2 = (((((adc_T>>4) - ((int32_t)bme280_calib.dig_T1)) *
186               ((adc_T>>4) - ((int32_t)bme280_calib.dig_T1))) >> 12) *
187             ((int32_t)bme280_calib.dig_T3)) >> 14;
188
189     t_fine = var1 + var2;
190
191     float T = (t_fine * 5 + 128) >> 8;
192     return T/100;
193 }
194
195
196 float BME280_readPressure(void) {
197     int64_t var1, var2, p;
198
199     //readTemperature(); // must be done first to get t_fine
200
```

```
201     int32_t adc_P = spi_rx24_bme280(BME280_REGISTER_PRESSUREDATA);
202     if (adc_P == 0x800000) // value in case pressure measurement was
          disabled
203         return 0;
204     adc_P >>= 4;
205
206     var1 = ((int64_t)t_fine) - 128000;
207     var2 = var1 * var1 * (int64_t)bme280_calib.dig_P6;
208     var2 = var2 + ((var1*(int64_t)bme280_calib.dig_P5)<<17);
209     var2 = var2 + (((int64_t)bme280_calib.dig_P4)<<35);
210     var1 = ((var1 * var1 * (int64_t)bme280_calib.dig_P3)>>8) +
211            ((var1 * (int64_t)bme280_calib.dig_P2)<<12);
212     var1 = (((((int64_t)1)<<47)+var1))*((int64_t)bme280_calib.dig_P1)>>33;
213
214     if (var1 == 0) {
215         return 0; // avoid exception caused by division by zero
216     }
217     p = 1048576 - adc_P;
218     p = (((p<<31) - var2)*3125) / var1;
219     var1 = (((int64_t)bme280_calib.dig_P9) * (p>>13) * (p>>13)) >> 25;
220     var2 = (((int64_t)bme280_calib.dig_P8) * p) >> 19;
221
222     p = ((p + var1 + var2) >> 8) + (((int64_t)bme280_calib.dig_P7)<<4);
223     return (float)p/256;
224 }
225
226
227
228 float  BME280_readHumidity(void) {
229     //readTemperature(); // must be done first to get t_fine
230
231     int32_t adc_H = spi_rx24_bme280(BME280_REGISTER_HUMIDDATA);
232     if (adc_H == 0x8000) // value in case humidity measurement was
          disabled
233         return 0;
234
235     int32_t v_x1_u32r;
236
237     v_x1_u32r = (t_fine - ((int32_t)76800));
238
239     v_x1_u32r = (((((adc_H << 14) - (((int32_t)bme280_calib.dig_H4) << 20)
          -
240                    (((int32_t)bme280_calib.dig_H5) * v_x1_u32r)) +
                   ((int32_t)16384)) >> 15) *
241                 (((((((v_x1_u32r * ((int32_t)bme280_calib.dig_H6)) >> 10)
                      *
242                    (((v_x1_u32r * ((int32_t)bme280_calib.dig_H3)) >>
                   11) + ((int32_t)32768))) >> 10) +
243                  ((int32_t)2097152)) * ((int32_t)bme280_calib.dig_H2) +
                   8192) >> 14));
244
245     v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r >> 15) * (v_x1_u32r >> 15)) >>
          7) *
```

```
246                                 ((int32_t)bme280_calib.dig_H1)) >> 4));
247
248     v_x1_u32r = (v_x1_u32r < 0) ? 0 : v_x1_u32r;
249     v_x1_u32r = (v_x1_u32r > 419430400) ? 419430400 : v_x1_u32r;
250     float h = (v_x1_u32r>>12);
251     return  h / 1024.0;
252 }
253
254
255
256
```