```c
/**
 ********************************************************************************
   * Main
 ********************************************************************************
   * @author  Samuel Ruhl, Alexander Meier
   * @date    2017-04-04
   * @file    main.c
   * @brief   Initializing of the XMC4700 components and displaying Things on the LCD
 ********************************************************************************
 **/


#include <DAVE.h>                 //Declarations from DAVE Code Generation (includes SFR declaration)

#include "spi.h"
#include "ft800.h"
#include "BME280def.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>


uint8_t start = 0;
uint32_t temp = 0;
char buf[50];
char buf2[50];
char buf3[50];
char buf4[50];
char buf5[50];
char buf6[50];

float temprature, press, hum;

uint32_t result_CO2;
uint32_t result_CO;
uint32_t result_CH4;

int interrupt_sign ;

/**

 * @brief main() - Application entry point
 *
 * <b>Details of function</b><br>
 * This routine is the application entry point. It is invoked by the
```

```
     device startup code. It is responsible for
46   * invoking the APP initialization dispatcher routine - DAVE_Init() and
     hosting the place-holder for user application
47   * code.
48   */
49
50   void sysDms(uint32_t millisec)
51   {
52       for(int i = OSCHP_GetFrequency() * millisec /1000; i > 0 ;i--);
53   }
54
55
56
57   /* Init function for an 5" LCD display */
58   uint8_t initFT800(void){
59       uint8_t dev_id = 0;                  // Variable for holding the read
         device id
60
61       ms_delay(50);
62       PORT3->OUT &= ~(1<<13);             // Set the PDN pin low
63       ms_delay(50);                       // Delay 50 ms for stability
64       PORT3->OUT |= (1<<13);              // Set the PDN pin high
65       ms_delay(50);                       // Delay 50 ms for stability
66
67       //WAKE
68       HOST_CMD_ACTIVE();
69       ms_delay(500);
70
71       //Ext Clock
72       HOST_CMD_WRITE(CMD_CLKEXT);         // Send CLK_EXT Command (0x44)//
           Set FT800 for external clock
73       ms_delay(5);                        //give some time to process
74
75       //PLL (48M) Clock
76       HOST_CMD_WRITE(CMD_CLK48M);         // Send CLK_48M Command (0x62)//
           Set FT800 for 48MHz PLL
77       ms_delay(5);        //give some time to process
78
79
80       ms_delay(5);
81
82       HOST_CMD_WRITE(0x40);
83       ms_delay(5);
84
85       // Now FT800 can accept commands at up to 30MHz clock on SPI bus
86
87       //Read Dev ID
88       dev_id = HOST_MEM_RD8(REG_ID);      // Read device id       //REG_ID
           is read only and always 0x7C
89
90       //while(HOST_MEM_RD8(REG_ID) != 0x7C);
91       if(dev_id != 0x7C)                  // Device ID should always be 0x7C
92       {
```

```
 93          return 1;
 94      }
 95
 96      HOST_MEM_WR8(REG_GPIO, 0x00);           // Set REG_GPIO to 0 to turn
           off the LCD DISP signal
 97      HOST_MEM_WR8(REG_PCLK, 0x00);           // Pixel Clock Output disable
 98
 99      // End of Wake-up FT800
100
101      //*************************************
102      // Initialize Display
103
104      HOST_MEM_WR16(REG_HCYCLE, 548);         // Set H_Cycle to 548
105      HOST_MEM_WR16(REG_HOFFSET, 43);         // Set H_Offset to 43
106      HOST_MEM_WR16(REG_HSYNC0, 0);           // Set H_SYNC_0 to 0
107      HOST_MEM_WR16(REG_HSYNC1, 41);          // Set H_SYNC_1 to 41
108      HOST_MEM_WR16(REG_VCYCLE, 292);         // Set V_Cycle to 292
109      HOST_MEM_WR16(REG_VOFFSET, 12);         // Set V_OFFSET to 12
110      HOST_MEM_WR16(REG_VSYNC0, 0);           // Set V_SYNC_0 to 0
111      HOST_MEM_WR16(REG_VSYNC1, 10);          // Set V_SYNC_1 to 10
112      HOST_MEM_WR8(REG_SWIZZLE, 0);           // Set SWIZZLE to 0
113      HOST_MEM_WR8(REG_PCLK_POL, 1);          // Set PCLK_POL to 1
114      HOST_MEM_WR8(REG_CSPREAD, 1);           // Set CSPREAD to 1
115      HOST_MEM_WR16(REG_HSIZE, 480);          // Set H_SIZE to 480
116      HOST_MEM_WR16(REG_VSIZE, 272);          // Set V_SIZE to 272
117
118      /* configure touch & audio */
119      HOST_MEM_WR8(REG_TOUCH_MODE, 0x03);         //set touch on: continous
120      HOST_MEM_WR8(REG_TOUCH_ADC_MODE, 0x01);     //set touch mode:
           differential
121      HOST_MEM_WR8(REG_TOUCH_OVERSAMPLE, 0x0F);   //set touch oversampling
           to max
122      HOST_MEM_WR16(REG_TOUCH_RZTHRESH, 5000);    //set touch resistance
           threshold
123      HOST_MEM_WR8(REG_VOL_SOUND, 0xFF);          //set the volume to
           maximum
124
125      /* write first display list */
126      HOST_MEM_WR32(RAM_DL+0, CLEAR_COLOR_RGB(0,0,0));  // Set Initial Color
            to BLACK
127      HOST_MEM_WR32(RAM_DL+4, CLEAR(1,1,1));            // Clear to the
           Initial Color
128      HOST_MEM_WR32(RAM_DL+8, DISPLAY());               // End Display List
129
130      HOST_MEM_WR8(REG_DLSWAP, DLSWAP_FRAME);           // Make this display
            list active on the next frame
131
132      HOST_MEM_WR8(REG_GPIO_DIR, 0x80);                 // Set Disp GPIO
           Direction
133      HOST_MEM_WR8(REG_GPIO, 0x80);                     // Enable Disp (if
           used)
134      HOST_MEM_WR16(REG_PWM_HZ, 0x00FA);                // Backlight PWM
           frequency
```

```
135     HOST_MEM_WR8(REG_PWM_DUTY, 0x80);                   // Backlight PWM
          duty
136
137     HOST_MEM_WR8(REG_PCLK, 0x05);                       // After this
          display is visible on the LCD
138
139     return 0;
140 }
141
142 /* Clear Screen */
143 void clrscr(void)
144 {
145     cmd(CMD_DLSTART);
146     cmd(CLEAR_COLOR_RGB(0,0,0));
147     cmd(CLEAR(1,1,1));
148     cmd(DISPLAY());
149     cmd(CMD_SWAP);
150 }
151
152
153
154 void screen_var(){
155
156     //clrscr();
157     cmd(CMD_DLSTART);
158     cmd(CLEAR_COLOR_RGB(0,0,0));
159     cmd(CLEAR(1,1,1));
160     cmd_gradient(0,0,0xA1E1FF, 0,250,0x000080);
161     start = 1;
162     cmd(COLOR_RGB(0x00,0x00,0x00));
163     cmd_text(240,35, 30,OPT_CENTERX, "Luftanalysesystem");
164
165
166     cmd(COLOR_RGB(255,255,255));
167
168     int vor = (int) temprature  ;
169     int nach = (int)(temprature * 100) % 100;
170     int vor2 =  press / 100;
171     int nach2 = (int) press  % 100 ;
172
173     int vor3 = (int) hum  ;
174     int nach3 = (int)(hum * 100) % 100;
175
176     sprintf(buf, "%d,%d Grad C", vor, nach);
177     sprintf(buf2, "%d,%d hPa", vor2, nach2);
178     sprintf(buf3, "%d,%d %%", vor3, nach3);
179
180     cmd_text(10,200, 27,0, "Temperature");
181     cmd_text(120,200, 27,0, buf);
182     cmd_text(10,220, 27,0, "Luftdruck");
183     cmd_text(120,220, 27,0, buf2);
184     cmd_text(10,240, 27,0, "Luftfeuchte");
185     cmd_text(120,240, 27,0, buf3);
```

```
186
187
188     sprintf(buf4, "%d", result_CO2);
189     sprintf(buf5, "%d", result_CO);
190     sprintf(buf6, "%d", result_CH4);
191
192     cmd_text(10,100, 27,0, "CO2");
193     cmd_text(10,120, 27,0, "CO");
194     cmd_text(10,140, 27,0, "CH4");
195     cmd_text(120,100, 27,0, buf4);
196     cmd_text(120,120, 27,0, buf5);
197     cmd_text(120,140, 27,0, buf6);
198
199     cmd(DISPLAY());
200     cmd(CMD_SWAP);
201
202
203 }
204
205 //                                                                              ↵
      ############################################################################ ↵
      ####################################
206
207 void val_to_buf(void){
208     int vor = (int) temprature ;
209     int nach = (int)(temprature * 100) % 100;
210
211     int vor2 =  press / 100;
212     int nach2 = (int) press  % 100;
213
214     int vor3 = (int) hum ;
215     int nach3 = (int)(hum * 100) % 100;
216
217     sprintf(buf, "%d,%d", vor, nach);
218     sprintf(buf2, "%d,%d", vor2, nach2);
219     sprintf(buf3, "%d,%d %%", vor3, nach3);
220
221     sprintf(buf4, "%d", result_CO2);
222     sprintf(buf5, "%d", result_CO);
223     sprintf(buf6, "%d", result_CH4);
224 }
225
226
227 void luft_warm(void){
228     val_to_buf();
229     cmd(CMD_DLSTART);
230     cmd(CLEAR_COLOR_RGB(0,0,0));
231     cmd(CLEAR(1,1,1));
232     cmd_gradient(362,134, 0xb7172c, 447,235, 0xf43b16);
233     cmd(COLOR_RGB(243,234,249));
234     cmd_text(178,15,28, 1536, "Meier - Ruhl'sche Luftanalyse");
235
236
```

```
237       cmd(COLOR_RGB(243,228,238));
238       cmd_text(80,72, 31, 1536, buf);
239       cmd_text(224,72,31, 1536, "Grad C");
240
241
242
243       cmd(COLOR_RGB(247,247,247));
244       cmd_text(80,130, 27, 1536, buf2);
245       cmd_text(129,130,27, 1536, "hPa");
246
247
248       cmd(COLOR_RGB(137,7,9));
249       cmd(LINE_WIDTH(16));
250       cmd(BEGIN(RECTS));
251       cmd(VERTEX2F(304,2496));
252       cmd(VERTEX2F(4224,4080));
253       cmd(END());
254
255
256       cmd(COLOR_RGB(248,248,248));
257       cmd_text(140,178, 28, 1536, buf4);
258       cmd_text(49,178,28, 1536, "CO2");
259
260
261       cmd(COLOR_RGB(248,248,248));
262       cmd_text(140,236, 28, 1536, buf6);
263       cmd_text(49,236,28, 1536, "CH4");
264
265
266       cmd(COLOR_RGB(245,245,245));
267       cmd_text(140,207, 28, 1536, buf5);
268       cmd_text(44,207,28, 1536, "CO");
269
270
271       cmd(COLOR_RGB(170,18,7));
272       cmd(LINE_WIDTH(16));
273       cmd(BEGIN(RECTS));
274       cmd(VERTEX2F(-96,496));
275       cmd(VERTEX2F(6144,512));
276       cmd(END());
277
278
279       cmd(POINT_SIZE(532));
280       cmd(cmd(COLOR_RGB(112,15,7)));
281       cmd(BEGIN(FTPOINTS));
282       cmd(VERTEX2F(4304,4016));
283       cmd(END());
284
285
286       cmd(POINT_SIZE(462));
287       cmd(COLOR_RGB(112,33,17));
288       cmd(BEGIN(FTPOINTS));
289       cmd(VERTEX2F(6608,1392));
```

```
290      cmd(END());
291
292
293      cmd(LINE_WIDTH(16));
294      cmd(COLOR_RGB(112,12,8));
295      cmd(BEGIN(LINES));
296      cmd(VERTEX2F(6640,1312));
297      cmd(VERTEX2F(6640,4352));
298
299
300      cmd(POINT_SIZE(462));
301      cmd(COLOR_RGB(112,33,17));
302      cmd(BEGIN(FTPOINTS));
303      cmd(VERTEX2F(5520,2768));
304      cmd(END());
305
306
307      cmd(LINE_WIDTH(16));
308      cmd(COLOR_RGB(112,12,8));
309      cmd(BEGIN(LINES));
310      cmd(VERTEX2F(5552,2816));
311      cmd(VERTEX2F(5552,5856));
312
313
314      cmd(POINT_SIZE(462));
315      cmd(COLOR_RGB(112,33,17));
316      cmd(BEGIN(FTPOINTS));
317      cmd(VERTEX2F(7680,336));
318      cmd(END());
319
320
321      cmd(DISPLAY());
322      cmd(CMD_SWAP);
323  }
324
325  void luft_kalt(){
326      val_to_buf();
327      cmd(CMD_DLSTART);
328      cmd(CLEAR_COLOR_RGB(0,0,0));
329      cmd(CLEAR(1,1,1));
330
331      cmd_gradient(332,141, 0x61cdff, 416,245, 0x002040);
332
333
334      cmd(COLOR_RGB(5,30,67));
335      cmd(LINE_WIDTH(16));
336      cmd(BEGIN(RECTS));
337      cmd(VERTEX2F(288,2608));
338      cmd(VERTEX2F(4208,4192));
339      cmd(END());
340
341
342      cmd(POINT_SIZE(357));
```

```
343       cmd(COLOR_RGB(34,43,205));
344       cmd(BEGIN(FTPOINTS));
345       cmd(VERTEX2F(6048,1600));
346       cmd(END());
347
348
349       cmd(COLOR_RGB(146,146,146));
350       cmd_text(140,178, 28, 1536, buf4);
351       cmd_text(49,178,28, 1536, "CO2");
352
353
354       cmd(COLOR_RGB(141,141,141));
355       cmd_text(140,207, 28, 1536, buf5);
356       cmd_text(44,207,28, 1536, "CO");
357
358
359       cmd(COLOR_RGB(142,142,142));
360       cmd_text(140,236, 28, 1536, buf6);
361       cmd_text(49,236,28, 1536, "CH4");
362
363
364       cmd(COLOR_RGB(140,131,137));
365       cmd_text(80,72, 31, 1536, buf);
366       cmd_text(224,72,31, 1536, "Grad C");
367
368
369       cmd(COLOR_RGB(134,134,134));
370       cmd_text(80,130, 27, 1536, buf2);
371       cmd_text(129,130,27, 1536, "hPa");
372
373
374       cmd(COLOR_RGB(10,52,170));
375       cmd(LINE_WIDTH(16));
376       cmd(BEGIN(RECTS));
377       cmd(VERTEX2F(-96,496));
378       cmd(VERTEX2F(6144,512));
379       cmd(END());
380
381
382       cmd(COLOR_RGB(123,118,126));
383       cmd_text(154,14,28, 1536, "Meier - Ruhl'sche Luftanalyse");
384
385
386       cmd(LINE_WIDTH(16));
387       cmd(COLOR_RGB(0,56,112));
388       cmd(BEGIN(LINES));
389       cmd(VERTEX2F(6080,1536));
390       cmd(VERTEX2F(7664,16));
391
392
393       cmd(DISPLAY());
394       cmd(CMD_SWAP);
395   }
```

```
396
397
398 int main(void)
399 {
400   DAVE_STATUS_t status;
401
402   status = DAVE_Init();           /* Initialization of DAVE APPs  */
403   PORT3->IOCR12 |= (0x10 << 3) ; //Init the GPIO CSS 3.12 (Displ) Pin as
        General Purpose Output (see RM p.2789 (chapter 28 table 26.5))
404   PORT3->OUT |= 1<<12;
405
406   PORT3->IOCR12 |= (0x10 << 11) ; //Init the GPIO PD_N 3.13 (Displ) Pin as
        General Purpose Output (see RM p.2789 (chapter 28 table 26.5))
407   PORT3->OUT |= 1<<13;
408
409   PORT0->IOCR0 |= (0x10<<27);   //Init the GPIO CSS (BME280) Pin as
        General Purpose Output (see RM p.2789 (chapter 28 table 26.5))
410   PORT0->OUT |= 1<<3;
411
412   BME280_init();
413   BME280_readCoefficients();
414
415   while(initFT800());
416   sysDms(500);
417   ADC_MEASUREMENT_StartConversion(&ADC_MEASUREMENT_0);  //Start ADC
        conversation
418
419
420   clrscr();
421
422
423   if(status != DAVE_STATUS_SUCCESS)
424   {
425     /* Placeholder for error handler code. The while loop below can be
          replaced with an user error handler. */
426     XMC_DEBUG("DAVE APPs initialization failed\n");
427
428     while(1U)
429     {
430
431     }
432   }
433
434   /* Placeholder for user application code. The while loop below can be
        replaced with user application code. */
435   while(1U)
436   {
437       temprature = BME280_readTemperature();
438       press = BME280_readPressure();
439       hum = BME280_readHumidity();
440
441       luft_warm();
442       ms_delay(50);
```

```
443
444   }
445 }
446
447
448 void Adc_Measurement_Handler(void)
449 {
450     /*Read out conversion results*/
451     result_CO2   = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_CO2_handle);
452     result_CO    = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_CO_handle);
453     result_CH4   = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_CH4_handle);
454
455     /*Re-trigger conversion sequence*/
456     ADC_MEASUREMENT_StartConversion(&ADC_MEASUREMENT_0);
457
458     interrupt_sign = 1 ;
459
460 }
461
462
```