```c
/*****************************************************************************
* proximity.c                                                                *
* Grubmüller Stefan, Marx Clemens                                            *
* May 2020                                                                   *
*                                                                            *
* Function: In this file you can find any functions used in the main         *
* file.                                                                      *
*                                                                            *
* More information in the scritum by @JosefReisinger and in the              *
* Specifications by @ClemensMarx and @StefanGrubmueller or in the            *
* datasheet by @sparkfun:                                                    *
* https://cdn.sparkfun.com/datasheets/Sensors/Proximity/apds9960.pdf         *
*****************************************************************************/

/* ----------------------------Includes ---------------------------------*/
#include "proximity.h"


/*----------------------- Static Variables ----------------------------*/
int h, min, sek, milsek;  // used for the timer
char buffer[30];

I2C_PIN_CONF SCL =      // defenition of SCL (I2C1)
{
  .GPIOx = GPIOB,
  .Pin = 6              // Pin PB6
};

I2C_PIN_CONF SDA =      // defenition of SDA (I2C1)
{
  .GPIOx = GPIOB,
  .Pin = 7              // Pin PB7
};

I2C_Device device =
{
  .Adress = 0x39,       // hardware adress = 0x39
  .I2C_Pereph = I2C1,   // I2C1 peripherals
  .pclr = 8,            // 36 MHz perepherial clock rate
  .sclr = 80,           // 80 kHz SCL Clock Rate (0 - 400kHZ)
  .maxRiseTime = 200    // 200ns max rise time (0 - 300ns)
};




/* ----------------------- Initalitations -----------------------------*/

// init GPIO ports PB6 and PB7 for I2C
void InitI2CPorts(void)
{
  // activate GPIO clock
  RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

  // init PB6 (SCL) as AF Open Drain
  int tmp = GPIOB->CRL;
  tmp &= 0xF0FFFFFF;      // delete configuration for PB6
  tmp |= 0x0F000000;      // define port PB6 as AF Open Drain
  GPIOB->CRL = tmp;

  // init PB7 (SDA) as AF Open Drain
  tmp = GPIOB->CRL;
  tmp &= 0x0FFFFFFF;      // delete configuration for PB7
  tmp |= 0xF0000000;      // define Port PB7 as AF Open Drain
  GPIOB->CRL = tmp;
}




/* -------------------------- Interrupts ------------------------------*/

// interrupt service routine timer3 (General Purpose Timer)
void TIM3_IRQHandler(void)  //Timer 3, very 100ms
{
  TIM3->SR &=~0x01; // clear interrupt pending bit(precent Interrupt-trigger)
  milsek = milsek + 2;
```

```
78  }
79
80
81  // External Interrupt Service Routine PA1
82  void EXTI1_IRQHandler(void)//ISR
83  {
84    EXTI->PR |= (0x01 << 1); // reset pending bit EXT0 (otherwise ISR will repeat)
85
86    lcd_set_cursor(0,0);
87    sprintf(&buffer[0],"IN RANGE");               // if the object is in range
88    lcd_put_string(&buffer[0]);
89
90    char en_reg[2] = {ENABLE_REG, DEL_PIEN};      // disable proximity interupt
91    i2c_write(&device, en_reg, 2, END_WITH_STOP); // set enable register
92  }
93
94  // initialisation Timer3 (General Purpose Timer)
95  void TIM3_Config(void)
96  {
97    /*-------------------------- configuration timer 3 -------------------------*/
98    RCC->APB1ENR |= 0x0002; // timer 3 clock enable
99    TIM3->SMCR = 0x0000;    // timer 3 clock Selection: CK_INT wird verwendet
100   TIM3->CR1  = 0x0000;    // selcet timer mode: Upcounter --> counts 0 to value of autoreload-register
101
102   // Tck_INT = 27,78ns, Presc = 54 ---> Auto Reload value = 3,6Mio (=0xFFFF) --> 0,1s Update Event
103   TIM3->PSC = 23;       //value of prescalers (Taktverminderung)
104   TIM3->ARR = 0xFFFF;   //Auto-Reload Wert = Maximaler Zaehlerstand des Upcounters
105
106   /*-------------------- configuration Interrupt Timer 3  --------------------*/
107   TIM3->DIER = 0x01;          // enable Interrupt bei einem UEV (Überlauf / Unterlauf)
108   NVIC_init(TIM3_IRQn,3);     // enable Timer 3 Update Interrupt, Priority 3
109
110   /*--------------------------- start Timer 3  -------------------------------*/
111   TIM3->CR1 |= 0x0001;        // set counter-Enable bit
112 }
113
114
115 // EXTI1_config
116 // connecct PA1 with EXTI1, Interrupt at falling edge, priority 2
117 void EXTI_config(pin p, int n)
118 {
119   NVIC_init(n + 6, 2);    //init NVIC for EXTI Line1 (Position n+6, Priority 2)
120
121   RCC->APB2ENR |= 0x0001;      //AFIOEN  - Clock enable
122   AFIO->EXTICR[0] &= ~((0xF & ~p) << (4 * n)); //Interrupt-Line EXTIn mit Portpin PAn verbinden
123   // ...
124   EXTI->FTSR |= (0x01 << n);     //Falling Edge Trigger für EXITn Aktivieren
125   EXTI->RTSR &= ~(0x01 << n);    //Rising Edge Trigger für EXTIn Deaktivieren
126
127   EXTI->PR |= (0x01 << n);  //EXTI_clear_pending: Das Auslösen auf vergangene Vorgänge nach dem
    enablen verhindern
128   EXTI->IMR |= (0x01 << n);   // Enable Interrupt EXTIn-Line. Kann durch den NVIC jedoch noch
    maskiert werden
129 }
130
131
132 // NVIC_init(char position, char priority)
133 // initialisation of an interrupts in the Nested Vectored Interrupt
134 // Controller (set priority, prevent trigger, enable
    interrupt
135 // parameters: "position" = 0-67 (number of interrupt)
136 //             "priority" = 0-15 (priority of interrupt)
137 void NVIC_init(char position, char priority)
138 {
139   NVIC->IP[position]=(priority<<4); //Interrupt priority register: Setzen der Interrupt Priorität
140   //Interrupt Clear Pendig Register: prevent trigger after enable
141   NVIC->ICPR[position >> 0x05] |= (0x01 << (position & 0x1F));
142   //Interrupt Set Enable Register: Enable interrupt
143   NVIC->ISER[position >> 0x05] |= (0x01 << (position & 0x1F));
144 }
145
146 /* --------------------------- functions --------------------------------*/
147
148 // after 0.1 seconds an interrupt is being triggered (milsek ++)
149 // function prints output real clock on lcd in hh:mm:ss:z
    (hour:minute:second:millisecond)
150 void clock_lcd(void)
```

```
151   {
152     if(milsek==10)
153     {
154       milsek=0;
155       if(++sek==60)
156       {
157         sek=0;
158         if(++min==60)
159         {
160           min=0;
161           if(++h==24)
162           {
163             h=0;
164           }
165         }
166       }
167     }
168     else if(milsek>=10)   // error detection
169     {
170       milsek=0;
171     }
172
173     // output of lcd
174     sprintf(&buffer[0], "%02d:%02d:%02d:%d", h, min, sek, milsek);
175     lcd_set_cursor(1,0);
176     lcd_put_string(&buffer[0]);
177   }
178
179   void check_device_con()
180   {
181     // check if device is there and connected
182     char ack;
183     int i;
184     char dev_con[] = {"DEVICE CONNECTED"};                // device connected
185     char dev_discon[] = {"NOT CONNECTED"};        // device disconnected
186     char buffer[1] = {0x1};
187
188     i2c_write(&device, buffer, 1, END_WITH_STOP); // write bit to test gettin ack for checking
    connection
189     ack = SDA_IN;         // read one bit of data pin
190     lcd_set_cursor(1,0);  // read acknowledge
191     if (ack == 0x0)       // if there is an acknowledge
192     {
193       // output
194       for(i=0; dev_con[i] != '\0'; i++)
195       {
196         lcd_put_char(dev_con[i]);                 // device connected
197       }
198     }
199     else                  // if no ack detected
200     {
201       // output
202       for(i=0; dev_discon[i] != '\0'; i++)
203       {
204         lcd_put_char(dev_discon[i]);              // device disconnected
205       }
206     }
207   }
208
209
210   // start proximity engine (configuration of registers)
211   void start_proximity_engine()
212   {
213
214     // Enable Register (0x80):
215     // 00000101 (0x05)
216     // Bit 7 = 0: is being reserved as 0
217     // Bit 6 = 0: gesture enable (GEN)
218     // Bit 5 = 0: proximity interrupt enable (PIEN)
219     // Bit 4 = 0: ambient light sense (ALS) interrupt enable (AIEN)
220     // Bit 3 = 0: wait enable (WEN) actives wait feature
221     // Bit 2 = 1: proximity detect enable (PEN)
222     // Bit 1 = 0: ALS enable (AEN)
223     // Bit 0 = 1: Power ON (PON)
224     char en_reg[2] = {ENABLE_REG, 0x05};
225     i2c_write(&device, en_reg, 2, END_WITH_STOP); // set enable register
226
```

```
227     // Persistance Register (0x8C):
228     // 00000000 (0x0)
229     // Bit 7 : 4 = 0: Controls rate of proximity interrupt to host process
230     // Bit 3  : 0 = 0: Controls rate of clear channel interrupt to host process
231     char pers_reg[2] = {PERS_REG, 0x00};
232     i2c_write(&device, pers_reg, 2, END_WITH_STOP);
233
234
235     // Proximity Interrupt Threshold Register - Low (0x89):
236     // set as define LOWTHRES in proximity.h
237     // Bit 7 : 0 = ?? (adjustable)
238     char low_threshold[2] = {LOWTHRES_REG, 0x00};
239     i2c_write(&device, low_threshold, 2, END_WITH_STOP);
240
241
242     // Proximity Interrupt Threshold Register - High (0x8B):
243     // set as define HIGHTHRES in
    proximity.h
244     // Bit 7 : 0 = ?? (adjustable)
245     char high_threshold[2] = {HIGHTHRES_REG, HIGHTHRES};
246     i2c_write(&device, high_threshold, 2, END_WITH_STOP);
247
248
249     // Proximity Pulse Register (0x8E):
250     // 01111111 (7F)
251     // Bit 7 : 6 = 01
252     // Bit 5 : 0 = 1: 111111
253     char prox_pulse_reg[2] = {PROX_PULSE_REG, 0x7F};
254     i2c_write(&device, prox_pulse_reg, 2, END_WITH_STOP);
255
256
257     //  Control Register One(0x8F):
258     //  0000??00
259     //  Bit 7 : 6 = 10
260     //  Bit 5 : 4 = reserved as 0
261     //  Bit 3 : 2 = LIGHT_INTENSITY set in defines
262     //  Bit 1 : 0 = 0
263     char control_reg1[2] = {CONTROL_REG1, GAIN_x2};
264     i2c_write(&device, control_reg1, 2, END_WITH_STOP);
265
266     //  Control Register Two(0x90):
267     // 10??0000
268     // Bit 7 = 1
269     // Bit 6 = 0
270     // Bit 5 : 4 = LED_BOOST_ON/LED_BOOST_OFF
271     // Bit 3 : 0 = reserved as 0
272     char control_reg2[2] = {CONTROL_REG2, LED_BOOST_OFF};
273     i2c_write(&device, control_reg2, 2, END_WITH_STOP);
274
275
276     // Status Register(0x93):
277     // 01100010 (0x62)
278     // The read-only Status Register provides the status of the device.
279     // Bit 7 = 0  Clear Photodiode Saturation
    (CPSAT)
280     // Bit 6 = 1  Indicates that an analog saturation event occurred
    (PGSAT)
281     // Bit 5 = 1  Proximity Interrupt. This bit triggers an interrupt if PIEN in ENABLE is set
    (PINT)
282     // Bit 4 = 0  ALS Interrupt (AINT)
283     // Bit 3 =    reserved as 0
284     // Bit 2 = 0  Gesture Interrupt (GINT)
285     // Bit 1 = 1  Proximity Valid (PVALID)
286     // Bit 0 = 0  ALS Valid (AVALID)
287     char stat_reg[2] = {STAT_REG, 0x62};
288     i2c_write(&device, stat_reg, 2, END_WITH_STOP);
289
290
291     // Proximity Offset UP / RIGHT Register(0x9D)
292     // 00000000 (0x00)
293     // Bit 7 : 0 = 0x0
294     char ur_offset_reg[2] = {UPRIGHT_OFFSET_REG, 0x00};
295     i2c_write(&device, ur_offset_reg, 2, END_WITH_STOP);
296
297
298     // Proximity Offset DOWN / LEFT Register(0x9E)
299     // 00000000 (0x00)
```

```
300     // Bit 7 : 0 = 0x0
301     char dl_offset_reg[2] = {DLEFT_OFFSET_REG, 0x00};
302     i2c_write(&device, dl_offset_reg, 2, END_WITH_STOP);
303
304
305     //Configuration Register Three(0x9F):
306     // 00000000 (0x00)
307     // select which photodiodes are used for proximity
308     // Bit 7 : 6 = reserved as 0
309     // Bit 5 = 0 use all diodes
310     // Bit 4 = 0 sleep after interrupt
311     // Bit 3 = 0 Proximity Mask UP Enable
312     // Bit 2 = 0 Proximity Mask LEFT Enable
313     // Bit 1 = 0 Proximity Mask LEFT Enable
314     // Bit 0 = 0 Proximity Mask RIGHT Enable
315     char conf_reg3[2] = {CONF_REG3, 0x00};
316     i2c_write(&device, conf_reg3, 2, END_WITH_STOP);
317
318
319   //  // Proximity Interrupt Clear (0xE5):
320   //  // 00000000 (0x00)
321   //  // Bit 7 : 0 = 0x0
322   //  char prox_int_clear[2] = {PROX_INT_CLEAR, 0x00};
323   //  i2c_write(&device, prox_int_clear, 2, END_WITH_STOP);
324
325
326   //  // Clear All Non-Gesture Interrupts (0xE7):
327   //  // 00000000 (0x00)
328   //  // Bit 7 : 0 = 0x0
329   //  char clear_all_int[2] = {CLEAR_ALL_INT, 0x00};
330   //  i2c_write(&device, clear_all_int, 2, END_WITH_STOP);
331   }
332
```