```c
1    /**************************************************************************
2    * main.c                                                                  *
3    * Grubmüller Stefan, Marx Clemens                                         *
4    * May 2020                                                                *
5    *                                                                         *
6    * Function: This programm should use the sparkfun sensor and the          *
7    * proximity engine. At the beginning the LCD will show you the Text:       *
8    *                                                                         *
9    * WELCOME                                                                  *
10   * DEVICE LINKED / DISLINKED                                               *
11   *                                                                         *
12   * After sending data via the I2C - so data from the device/cortex can     *
13   * be transmitted and recieved - the LCD will show you a real time         *
14   * clock, the distance of a object near the threashold and if it is in     *
15   * threshold range. Example:                                               *
16   *                                                                         *
17   * IN RANGE   20-140                                                       *
18   * 12:04:31       50                                                       *
19   *                                                                         *
20   * This process will run paralell due to interrupts.                       *
21   * To recieve data from the slave/device you have to configure the         *
22   * devive registers shown in proximity.c.                                  *
23   *                                                                         *
24   * More information in the scritum by @JosefReisinger and in the           *
25   * Specifications by @ClemensMarx and @StefanGrubmueller or in the         *
26   * datasheet by @sparkfun:                                                 *
27   * https://cdn.sparkfun.com/datasheets/Sensors/Proximity/apds9960.pdf      *
28   **************************************************************************/
29
30   /* --------------------------- Main ----------------------------------*/
31   #include "proximity.h"
32   #include "stdlib.h"
33   #include <string.h>
34   #include <stdio.h>
35   int main()
36   {
37     // Initalisations
38     // set_clock_36MHz();           // set system clock to 36MHz
39     InitI2CPorts();                 // initialisation of GPIO ports (PB6 = SCL and PB7 = SDA)
40     i2c_init(&device, &SCL, &SDA);  // initialisation of I2C (extra library)
41     lcd_init();                     // initialisation of LCD
42     lcd_clear();                    // clear screen
43     uart_init(9600);                // 9600,8,n,1
44     uart_clear();                   // send clear string to VT 100 terminal
45
46
47     // PA1 as Input (external interrupt Pin of sparkfun sensor)
48     RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;  // enable clock for GPIOA (APB2 Peripheral clock enable
     register)
49     GPIOA->CRL &= 0xFFFFFF0F;            // set Port Pins PA1 to Pull Up/Down Input mode (50MHz) =
     Mode 8
50     GPIOA->CRL |= 0x00000080;
51     GPIOA->ODR |= 0x0002;
52
53
54     // starting text on LCD
55     lcd_set_cursor(0, 0);           // set position on LCD
56     lcd_put_string("WELCOME");      // write on LCD
57     check_device_con();             // is the device connected?
58     wait_ms(2000);                  // wait 2 seconds
59     lcd_clear();
60
61     // timer on lcd (real time clock)
62     milsek = 0;        // initalise milliseconds
63     TIM3_Config();     // start timer 3: Upcounter --> triggers every 0,1s an update interrupt
64
65
66
67     start_proximity_engine(); // set of configuration registers for proximity detection
68
69     EXTI_config(a, 1);         // external interrupt pin; triggers when int pin of sensor sends falling
     edge
70
71
72     // endless loop
73     while (1)
74     {
```

```
 75            clock_lcd();
 76
 77            // read data of PDATA regsiter (0x90)
 78            char pdata_w[] = {0x9C};
 79            i2c_write(&device, pdata_w, 1, END_WITHOUT_STOP);
 80            char pdata_r;
 81            i2c_read(&device, &pdata_r, 1);
 82
 83            // output of proximity data
 84            char buffer_i [8]= {0};              // set and clear buffer
 85            sprintf(buffer_i, "%d", pdata_r); // proximity data as int
 86            lcd_set_cursor(1,13);
 87            lcd_put_string(buffer_i);           // output of proximity data on lcd as int
 88
 89            //output of range (lower threshold to higher threshold)
 90            char threshold[2];
 91            sprintf(threshold, "%d-%d", LOWTHRES, HIGHTHRES);
 92            lcd_set_cursor(0,9);
 93            lcd_put_string(threshold);          // output of range on lcd as int
 94
 95            // check if data of proximity data register is inside range
 96            if (((unsigned char)pdata_r >= LOWTHRES) && ((unsigned char)pdata_r <= HIGHTHRES))
 97            {
 98              char en_reg[2] = {ENABLE_REG, SET_PIEN};            // set PIEN (enable Proximity Interrupt)
 99              i2c_write(&device, en_reg, 2, END_WITHOUT_STOP);  // set enable register
100            }
101        wait_ms(500);    // wait to avoid lecking image
102        lcd_clear();     // clear screen
103      }
104    }
105
```

```c
 1  /****************************************************************************
 2   * proximity.h                                                             *
 3   * Grubmüller Stefan, Marx Clemens                                         *
 4   * May 2020                                                                *
 5   *                                                                         *
 6   * Function: This header should define all the functions and variables     *
 7   * being used in different files.                                          *
 8   *                                                                         *
 9   * More information in the scritum by @JosefReisinger and in the           *
10   * Specifications by @ClemensMarx and @StefanGrubmueller or in the         *
11   * datasheet by @sparkfun:                                                 *
12   * https://cdn.sparkfun.com/datasheets/Sensors/Proximity/apds9960.pdf      *
13   ****************************************************************************/
14
15  #ifndef __PROXIMITY_H__
16  #define __PROXIMITY_H__
17
18  /* -------------------------Includes --------------------------------*/
19
20  #include <stm32f10x.h>
21  #include "armv10_std.h"
22  #include "i2c.h"                    // I2C library by Jakob Pachtrog
23  //#include <stm32f10x_i2c.h>        // I2C default library
24
25
26  /* ------------------------ defines ---------------------------------*/
27  // bitbanding for SDA (PB7)
28  #define GPIOB_IDR GPIOB_BASE + 2*sizeof(uint32_t)
29  #define SDA_IN *((volatile unsigned long *)(BITBAND_PERI(GPIOB_IDR,7))) //PB7 - Input
30
31  #define ENABLE_REG          0x80      // enable register
32  #define PERS_REG            0x8C      // persistance register
33  #define LOWTHRES_REG        0x89      // lower threashold
34  #define HIGHTHRES_REG       0x8B      // higher threashold
35  #define PROX_PULSE_REG      0x8E      // proximity pulse register
36  #define CONTROL_REG1        0x8F      // control register one
37  #define CONTROL_REG2        0x90      // control register two
38  #define STAT_REG            0x93      // status register
39  #define UPRIGHT_OFFSET_REG  0x9D      // proximity offset UP / RIGHT register
40  #define DLEFT_OFFSET_REG    0x9E      // proximity offset UP / RIGHT register
41  #define CONF_REG3           0x9F      // configuration register three
42  #define PROX_INT_CLEAR      0xE5      // proximity interrupt clear
43  #define CLEAR_ALL_INT       0xE7      // clear all non-gesture interrupts
44  #define SET_PIEN            0x25      // set Proximity interrupt enable
45  #define DEL_PIEN            0x05      // delete PIEN bit (PIEN = 0)
46
47  // Threshold- Low to High
48  // 0xFF...very near
49  // 0x00...very far
50  #define LOWTHRES            0x10      // lower threshold for proximity
51  #define HIGHTHRES           0xAC      // higher threashold for proximity
52
53  // Light Intensity
54  // Very Dark to very Bright
55  // Various Gains
56  #define GAIN_x1             0x00      // very bright conditions (GAIN = 1)
57  #define GAIN_x2             0x04      // not that bright but also not dark ~  (GAIN = 2)
58  #define GAIN_x4             0x08      // need of light (normal room condtions in evening) (GAIN = 4)
59  #define GAIN_x8             0x0C      // dark conditions (GAIN = 8)
60
61  // activates or deactivates the LED Boost option of sensor
62  #define LED_BOOST_ON        0xA0      // additional current up to 200%
63  #define LED_BOOST_OFF       0x80      // non addtional current
64
65  extern int milsek;
66
67
68  /* ----------------------- Prototypes -------------------------------*/
69
70  // init ports (PA7 Open Drain)
71  void InitI2CPorts(void);
72
73  // interrupts
74  void NVIC_init(char position, char priority);
75  // Nestet Vector Interrupt Controller
76  void NVIC_init(char position, char priority);
77  // timer
```

```
78     void TIM3_Config(void);
79
80     // real time clock on lcd
81     void clock_lcd(void);
82     // check connection of device
83     void check_device_con(void);
84
85     // start proximity engine due to setting the register bits
86     void start_proximity_engine(void);
87     // reads data out of the Proximity data register (0x9C)
88     void read_data(void);
89
90     // transfer pins for function of EXTI_config
91     typedef enum { a, b, c, d, e, f, } pin;
92     // external interrupt
93     void EXTI_config(pin p, int n);
94
95     /* ------------------------- structures --------------------------------*/
96
97     extern I2C_Device device;
98
99     extern I2C_PIN_CONF SCL;     // defenition of SCL (I2C1)
100
101    extern I2C_PIN_CONF SDA;     // defenition of SDA (I2C1)
102
103    #endif
104
```

```c
1    /***************************************************************************
2     * proximity.c                                                             *
3     * Grubmüller Stefan, Marx Clemens                                         *
4     * May 2020                                                                *
5     *                                                                         *
6     * Function: In this file you can find any functions used in the main      *
7     * file.                                                                   *
8     *                                                                         *
9     * More information in the scritum by @JosefReisinger and in the           *
10    * Specifications by @ClemensMarx and @StefanGrubmueller or in the         *
11    * datasheet by @sparkfun:                                                 *
12    * https://cdn.sparkfun.com/datasheets/Sensors/Proximity/apds9960.pdf      *
13    ***************************************************************************/
14
15   /* -------------------------Includes -------------------------------*/
16   #include "proximity.h"
17
18
19   /*--------------------- Static Variables ---------------------------*/
20   int h, min, sek, milsek;  // used for the timer
21   char buffer[30];
22
23   I2C_PIN_CONF SCL =       // defenition of SCL (I2C1)
24   {
25     .GPIOx = GPIOB,
26     .Pin = 6             // Pin PB6
27   };
28
29   I2C_PIN_CONF SDA =       // defenition of SDA (I2C1)
30   {
31     .GPIOx = GPIOB,
32     .Pin = 7             // Pin PB7
33   };
34
35   I2C_Device device =
36   {
37     .Adress = 0x39,       // hardware adress = 0x39
38     .I2C_Pereph = I2C1,   // I2C1 peripherals
39     .pclr = 8,            // 36 MHz perepherial clock rate
40     .sclr = 80,           // 80 kHz SCL Clock Rate (0 - 400kHZ)
41     .maxRiseTime = 200    // 200ns max rise time (0 - 300ns)
42   };
43
44
45
46
47   /* --------------------- Initalitations ---------------------------*/
48
49   // init GPIO ports PB6 and PB7 for I2C
50   void InitI2CPorts(void)
51   {
52     // activate GPIO clock
53     RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
54
55     // init PB6 (SCL) as AF Open Drain
56     int tmp = GPIOB->CRL;
57     tmp &= 0xF0FFFFFF;      // delete configuration for PB6
58     tmp |= 0x0F000000;      // define port PB6 as AF Open Drain
59     GPIOB->CRL = tmp;
60
61     // init PB7 (SDA) as AF Open Drain
62     tmp = GPIOB->CRL;
63     tmp &= 0x0FFFFFFF;      // delete configuration for PB7
64     tmp |= 0xF0000000;      // define Port PB7 as AF Open Drain
65     GPIOB->CRL = tmp;
66   }
67
68
69
70
71   /* ----------------------- Interrupts ------------------------------*/
72
73   // interrupt service routine timer3 (General Purpose Timer)
74   void TIM3_IRQHandler(void)  //Timer 3, very 100ms
75   {
76     TIM3->SR &=~0x01; // clear interrupt pending bit(precent Interrupt-trigger)
77     milsek = milsek + 2;
```

```c
 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)
```

```c
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
```