```c
1  /**********************************************************************
2   *
3   * Analog-to-digital conversion with displaying result on LCD and
4   * transmitting via UART.
5   * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
6   *
7   * Copyright (c) 2018-2020 Tomas Fryza
8   * Dept. of Radio Electronics, Brno University of Technology, Czechia
9   * This work is licensed under the terms of the MIT license.
10  *
11   **********************************************************************/
12
13  /* Includes ----------------------------------------------------------*/
14  #include <avr/io.h>          // AVR device-specific IO definitions
15  #include <avr/interrupt.h>   // Interrupts standard C library for AVR-GCC
16  #include "timer.h"           // Timer library for AVR-GCC
17  #include "lcd.h"             // Peter Fleury's LCD library
18  #include <stdlib.h>          // C library. Needed for conversion function
19  #include "uart.h"            // Peter Fleury's UART library
20  #include "stdbool.h"
21  #ifndef F_CPU
22  #define F_CPU 16000000
23  #endif
24
25  /* Function definitions ----------------------------------------------*/
26  /**
27   * Main function where the program execution begins. Use Timer/Counter1
28   * and start ADC conversion four times per second. Send value to LCD
29   * and UART.
30   */
31  int main(void)
32  {
33      // Initialize LCD display
34      lcd_init(LCD_DISP_ON);
35      lcd_gotoxy(1, 0); lcd_puts("value:");
36      lcd_gotoxy(3, 1); lcd_puts("key:");
37      lcd_gotoxy(8, 0); lcd_puts("a");     // Put ADC value in decimal
38      lcd_gotoxy(13,0); lcd_puts("b");     // Put ADC value in hexadecimal
39      lcd_gotoxy(8, 1); lcd_puts("c");     // Put button name here
40
41      // Configure ADC to convert PC0[A0] analog value
42      // Set ADC reference to AVcc
43      ADMUX |= (1 << REFS0);
44      ADMUX &= ~(1 << REFS1);
45
46      // Set input channel to ADC0
47      ADMUX &= ~((1 << MUX0) | (1 << MUX1) | (1 << MUX2) | (1 << MUX3));
48
49      // Enable ADC module
50      ADCSRA |= (1 << ADEN);
51
52      // Enable conversion complete interrupt
53      ADCSRA |= (1 << ADIE);
```

```c
54
55      // Set clock prescaler to 128
56      ADCSRA |= (1 << ADPS0) | (1 << ADPS1) | (1 << ADPS2);
57
58      // Configure 16-bit Timer/Counter1 to start ADC conversion
59      // Enable interrupt and set the overflow prescaler to 262 ms
60      TIM1_overflow_262ms();
61      TIM1_overflow_interrupt_enable();
62
63      // Initialize UART to asynchronous, 8N1, 9600
64      uart_init(UART_BAUD_SELECT(9600, F_CPU));
65
66      // Enables interrupts by setting the global interrupt mask
67      sei();
68
69      // Infinite loop
70      while (1)
71      {
72          /* Empty loop. All subsequent operations are performed exclusively
73           * inside interrupt service routines ISRs */
74      }
75
76      // Will never reach this
77      return 0;
78  }
79
80  /* Interrupt service routines ----------------------------------------*/
81  /**
82   * ISR starts when Timer/Counter1 overflows. Use single conversion mode
83   * and start conversion four times per second.
84   */
85  ISR(TIMER1_OVF_vect)
86  {
87      // Start ADC conversion
88      ADCSRA |= (1 << ADSC);
89
90  }
91
92  /* ------------------------------------------------------------------ */
93  /**
94   * ISR starts when ADC completes the conversion. Display value on LCD
95   * and send it to UART.
96   */
97  ISR(ADC_vect)
98  {
99      // WRITE YOUR CODE HERE
100     uint16_t value;
101     char lcd_string[5];
102     char parity = 0;
103     value = ADC;
104     bool b[8]; //bits
105
106     for (int j = 0;  j < 8;  j++)
```

```c
107                b[j] = 0 != (value & (1 << j));
108
109        // Print parity bit
110        parity = b[0]^b[1]^b[2]^b[3]^b[4]^b[5]^b[6]^b[7];
111        lcd_gotoxy(14, 1);
112        itoa(parity, lcd_string, 10);
113        lcd_puts(lcd_string);
114
115        // Print on LCD in decimal
116        itoa(value, lcd_string, 10);
117        lcd_gotoxy(8, 0);
118        lcd_puts("    ");
119        lcd_gotoxy(8, 0);
120        lcd_puts(lcd_string);
121
122        if(value < 700)
123        {
124            // Send to uart in decimal
125            uart_puts("ADC value in decimal: ");
126            uart_puts(lcd_string);
127            uart_puts("\n");
128        }
129        itoa(value, lcd_string, 16);
130        lcd_gotoxy(13, 0);
131        lcd_puts("   ");
132        lcd_gotoxy(13, 0);
133        lcd_puts(lcd_string);
134
135
136        // Print what is pressed
137        lcd_gotoxy(8, 1);
138        lcd_puts("     ");
139        if(value >= 1023-8)
140        {
141            lcd_gotoxy(8, 1);
142            lcd_puts("None  ");
143        }
144        else if(value >= 651-8)
145        {
146            lcd_gotoxy(8, 1);
147            lcd_puts("Select");
148        }
149        else if(value >= 403-8)
150        {
151            lcd_gotoxy(8, 1);
152            lcd_puts("Left  ");
153        }
154        else if(value >= 246-8)
155        {
156            lcd_gotoxy(8, 1);
157            lcd_puts("Down  ");
158        }
159        else if(value >= 101-8)
```

```c
160         {
161             lcd_gotoxy(8, 1);
162             lcd_puts("Up");
163         }
164     else
165         {
166             lcd_gotoxy(8, 1);
167             lcd_puts("Right ");
168         }
169
170 }
```