DE2 lab 7

Václav Pastušek

xpastu02, 204437

Table with voltage divider

$$U_R = 5V \cdot \frac{0}{R_2 + 0} = 5V \cdot \frac{0}{3000 + 0} = 0V$$

$$U_U = 5V \cdot \frac{R_3}{R_2 + R_3} = 5V \cdot \frac{330}{3000 + 330} = 0.495V$$

$$U_D = 5V \cdot \frac{R_3 + R_4}{R_2 + R_3 + R_4} = 5V \cdot \frac{330 + 620}{3000 + 330 + 620} = 1.203V$$

$$U_L = 5V \cdot \frac{R_3 + R_4 + R_5}{R_2 + R_3 + R_4 + R_5} = 5V \cdot \frac{330 + 620 + 1000}{3000 + 330 + 620 + 1000} = 1.970V$$

$$U_{SEL} = 5V \cdot \frac{R_3 + R_4 + R_5 + R_6}{R_2 + R_3 + R_4 + R_5 + R_6} = 5V \cdot \frac{330 + 620 + 1000 + 3300}{3000 + 330 + 620 + 1000 + 3300} = 3.182V$$

$$U_{none} = 5V$$

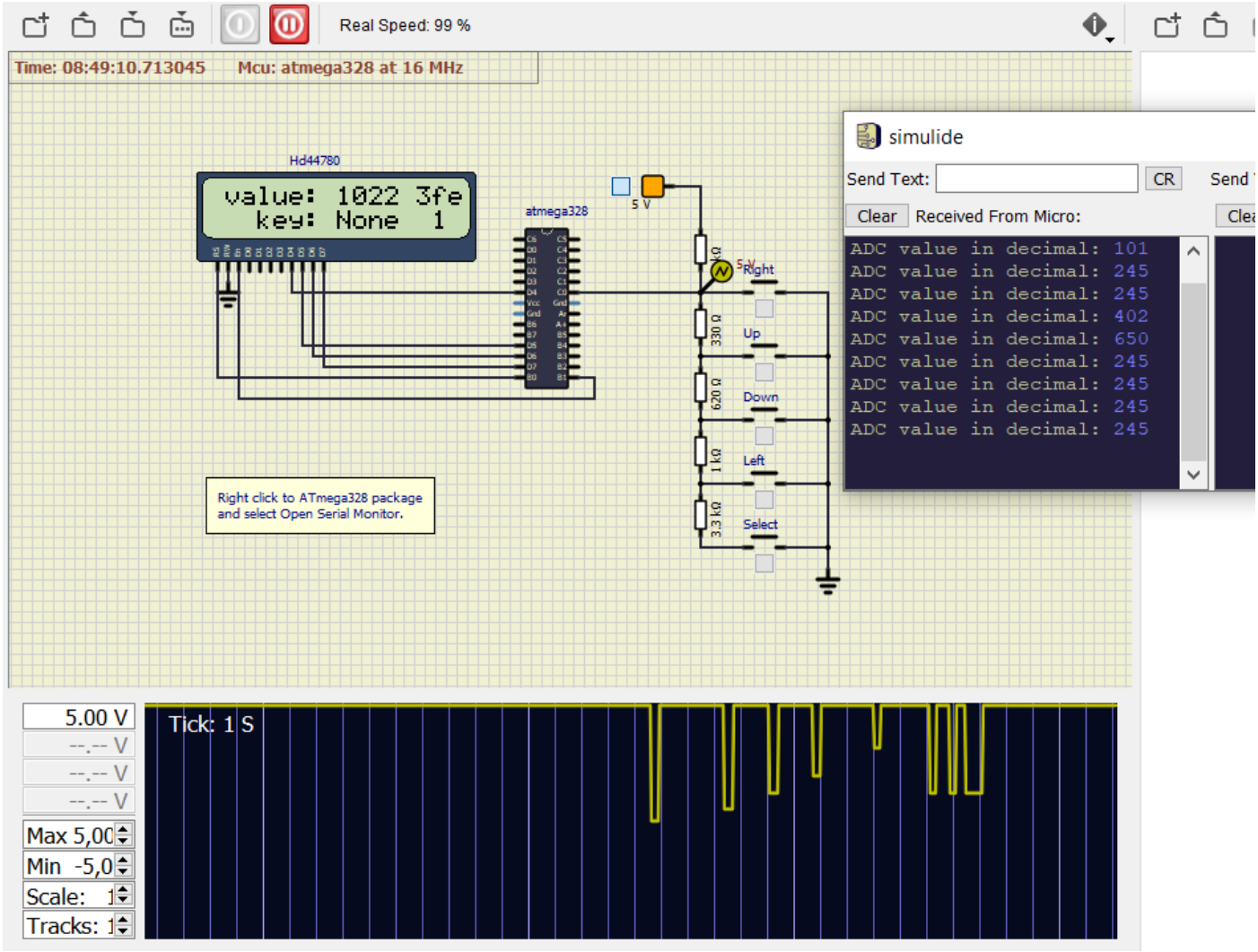
ADC values for all buttons

$$ADC = \frac{V_i}{V_{ref}} \cdot (2^n - 1) = \frac{V_i}{5} \cdot (2^{10} - 1)$$

Push button	PC0[A0] voltage	ADC value (calculated)	ADC value (measured)
Right	0	0	0
Up	0.495	101	101
Down	1.203	246	245
Left	1.970	403	402
Select	3.182	651	650
none	5	1023	1022

Operation	Register(s)	Bit(s)	Description
Voltage reference	ADMUX	REFS1:0	01: Avcc voltage reference, 5V
Input channel	ADMUX	MUX3:0	0000: ADC0, 0001: ADC1, …
ADC enable	ADCSRA	ADEN	1: enables the ADC, 0: the ADC is turned off (0: while a conversion is in progress --> terminate conversion)
Start conversion	ADCSRA	ADSC	Single Conversion mode: 1: to start each conversion; Free Running mode: 1: to start the first conversion.
ADC interrupt enable	ADCSRA	ADIE	1: ADC Conversion Complete Interrupt is activated
ADC clock prescaler	ADCSRA	ADPS2:0	000: Division fator 2, 001: 2, 010: 4, …
ADC result	ADCH, ADCL	ADC9:0	ADLAR=1: ADCH9:2, ADCL1:0, result is left adjusted
			ADLAR=0: ADCH9:8, ADCL7:0, result is right adjusted

Function name	Function parameters	Description	Example
uart_init	UART_BAUD_SELECT(9600, F_CPU)	Initialize UART to 8N1 and set baudrate to 9600 Bd	uart_init(UART_BAUD_SELECT(9600, F_CPU));
uart_getc	void	Get received byte from ringbuffer	uart_getc();
uart_putc	data	Put byte to ringbuffer for transmitting via UART.	uart_putc('A');
uart_puts	* s	Put string to ringbuffer for transmitting via UART.	uart_puts("Hello world");

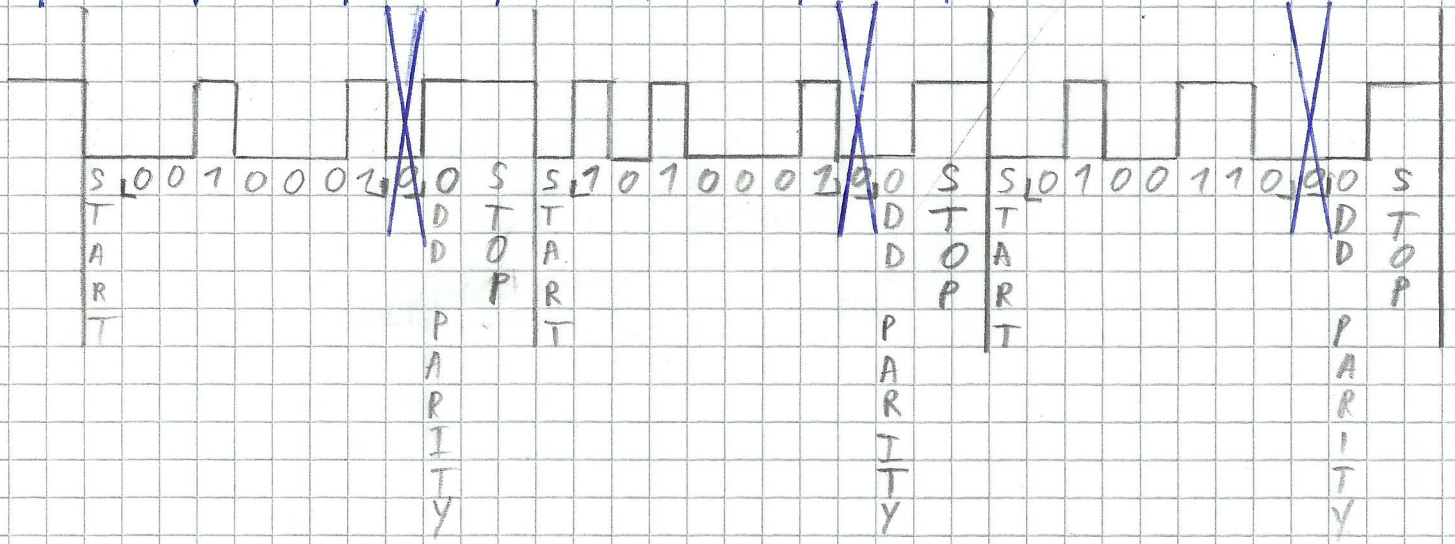

Real Speed: 99 %
Time: 08:49:10.713045
Mcu: atmega328 at 16 MHz
Hd44780
value: 1022 3fe
key: None 1
atmega328
5 V
5 V
Right
Up
Down
Left
Select
330 Ω
620 Ω
1 kΩ
3.3 kΩ
Right click to ATmega328 package and select Open Serial Monitor.
simulide
Send Text:
CR
Send
Clear
Received From Micro:
ADC value in decimal: 101
ADC value in decimal: 245
ADC value in decimal: 245
ADC value in decimal: 402
ADC value in decimal: 650
ADC value in decimal: 245
ADC value in decimal: 245
ADC value in decimal: 245
ADC value in decimal: 245
5.00 V
--.-- V
--.-- V
--.-- V
Max 5,00
Min -5,0
Scale: 1
Tracks: 1
Tick: 1 S

DATA: DE2 4800 7o2 (7 data bits, odd parity, 2 stop bits, 4800 Bd)

Václav Kaňkovský
xkanko02
204437

DE2: 0x44 0x45 0x32

| ~~0~~ 1 0 0 D 0 1 0 0 | ~~0~~ 1 0 0 E 0 1 0 1 | ~~0~~ 1 1 2 0 0 1 0 |



$\frac{1}{4800} = 208,\overline{3}\ \mu s$

```c
/***********************************************************************
 *
 * Analog-to-digital conversion with displaying result on LCD and
 * transmitting via UART.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czechia
 * This work is licensed under the terms of the MIT license.
 *
 **********************************************************************/

/* Includes ----------------------------------------------------------*/
#include <avr/io.h>         // AVR device-specific IO definitions
#include <avr/interrupt.h>  // Interrupts standard C library for AVR-GCC
#include "timer.h"          // Timer library for AVR-GCC
#include "lcd.h"            // Peter Fleury's LCD library
#include <stdlib.h>         // C library. Needed for conversion function
#include "uart.h"           // Peter Fleury's UART library
#include "stdbool.h"
#ifndef F_CPU
#define F_CPU 16000000
#endif

/* Function definitions ----------------------------------------------*/
/**
 * Main function where the program execution begins. Use Timer/Counter1
 * and start ADC conversion four times per second. Send value to LCD
 * and UART.
 */
int main(void)
{
    // Initialize LCD display
    lcd_init(LCD_DISP_ON);
    lcd_gotoxy(1, 0); lcd_puts("value:");
    lcd_gotoxy(3, 1); lcd_puts("key:");
    lcd_gotoxy(8, 0); lcd_puts("a");    // Put ADC value in decimal
    lcd_gotoxy(13,0); lcd_puts("b");    // Put ADC value in hexadecimal
    lcd_gotoxy(8, 1); lcd_puts("c");    // Put button name here

    // Configure ADC to convert PC0[A0] analog value
    // Set ADC reference to AVcc
    ADMUX |= (1 << REFS0);
    ADMUX &= ~(1 << REFS1);

    // Set input channel to ADC0
    ADMUX &= ~((1 << MUX0) | (1 << MUX1) | (1 << MUX2) | (1 << MUX3));

    // Enable ADC module
    ADCSRA |= (1 << ADEN);

    // Enable conversion complete interrupt
    ADCSRA |= (1 << ADIE);
```

```
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++)
```

```
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)
```

```
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 }
```