Session 7

-

ADC and UART serial communication

Author: Guillermo Cortés Orellana

Teacher: Tomáš Frýza

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Lab assignment

1. **Preparation tasks**

    - Table with voltaje divider, calculated and measured ADC values for all buttons

Push button	**PC0[A0] voltage**	**ADC value (calculated)**	**ADC value (measured)**
Right	0 V	0	0
Up	0.495 V	101	101
Down	1.2 V	245	245
Left	1.96 V	401	402
Select	3.18 V	650	650
None	5 V	1023	1022

2. ADC

- Listing of ADC_vect interrupt routine with complete code for sending data to the LCD/UART and identification of the pressed Button

```
/**
 * ISR starts when ADC completes the conversion. Display value on LCD
 * and send it to UART.
 */
ISR(ADC_vect)
{
    uint16_t value = 0;
    char lcd_string[4] = "0000";    // Maximum value is 1023 -> We need 4 character in DEC

    value = ADC;                    // Copy ADC register result to 16-bit variable

    itoa (value, lcd_string, 10);   // Convert DEC value to string
    lcd_gotoxy(8,0);                // Display in position 'a'
    lcd_puts("    ");               // Clear position 'a'
    lcd_gotoxy(8,0);                // Display in position 'a'
    lcd_puts(lcd_string);

    /*We use this condition in order to avoid display continuously value
      in UART communication (when we don't push any button)*/
    if(value < 700){
        uart_puts("ADC value in DEC: ");
        uart_puts(lcd_string);
        uart_puts("\r\n");
    }

    itoa (value, lcd_string, 16);   // Convert HEX value to string
    lcd_gotoxy(13,0);               // Display in position 'b'
    lcd_puts("    ");               // Clear position 'b'
    lcd_gotoxy(13,0);               // Display in position 'b'
    lcd_puts(lcd_string);

    /*We use this condition in order to avoid display continuously value
      in UART communication (when we don't push any button)*/
    if(value < 700){
        uart_puts("ADC value in HEX: ");
        uart_puts(lcd_string);
        uart_puts("\r\n");
    }
```

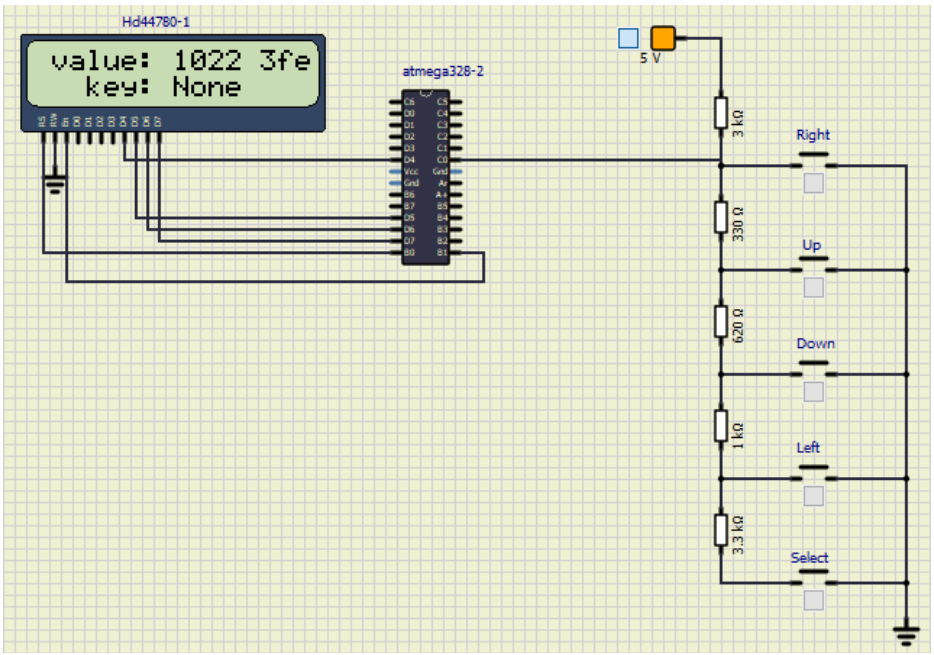
```
/**
 * 1. We will display in position 'c' of LCD which button is pushed.
 *    AVR ADC module has 10-bit resolution with +/-2LSB accuracy
 *
 * 2. We will use UART communication
 */

lcd_gotoxy(8, 1);               // Display in position 'c'
lcd_puts("      ");             // Clear position 'c'
lcd_gotoxy(8, 1);               // Display in position 'c'

/*We use this condition in order to avoid display continuously value
  in UART communication (when we don't push any button)*/
if(value < 700){
    if(value <= 2){

        // Right
        lcd_puts("Right");
        uart_puts("Button: Right");
        uart_puts("\r\n");

    } else if ((value >= 99)&(value <= 103)){

        // Up
        lcd_puts("Up");
        uart_puts("Button: Up");
        uart_puts("\r\n");


    }else if ((value >= 243)&(value <= 247)){

        // Down
        lcd_puts("Down");
        uart_puts("Button: Down");
        uart_puts("\r\n");

    }else if ((value >= 400)&(value <= 404)){

        // Left
        lcd_puts("Left");
        uart_puts("Button: Left");
        uart_puts("\r\n");

    }else if ((value >= 648)&(value <= 652)){

        // Select
        lcd_puts("Select");
        uart_puts("Button: Select");
        uart_puts("\r\n");
```

```
        }else if (value >= 1021){

            // None
            lcd_puts("None");
            uart_puts("Button: None");
            uart_puts("\r\n");

        }else{

            // ERROR
            lcd_puts("ERROR");
            uart_puts("Button: ERROR");
            uart_puts("\r\n");
        }
    }
}
```

You can find the code on my GitHub:

https://github.com/GuicoRM/Digital-Electronics-2

- Screenshot of SimulIDE circuit when “Power Circuit” is applied

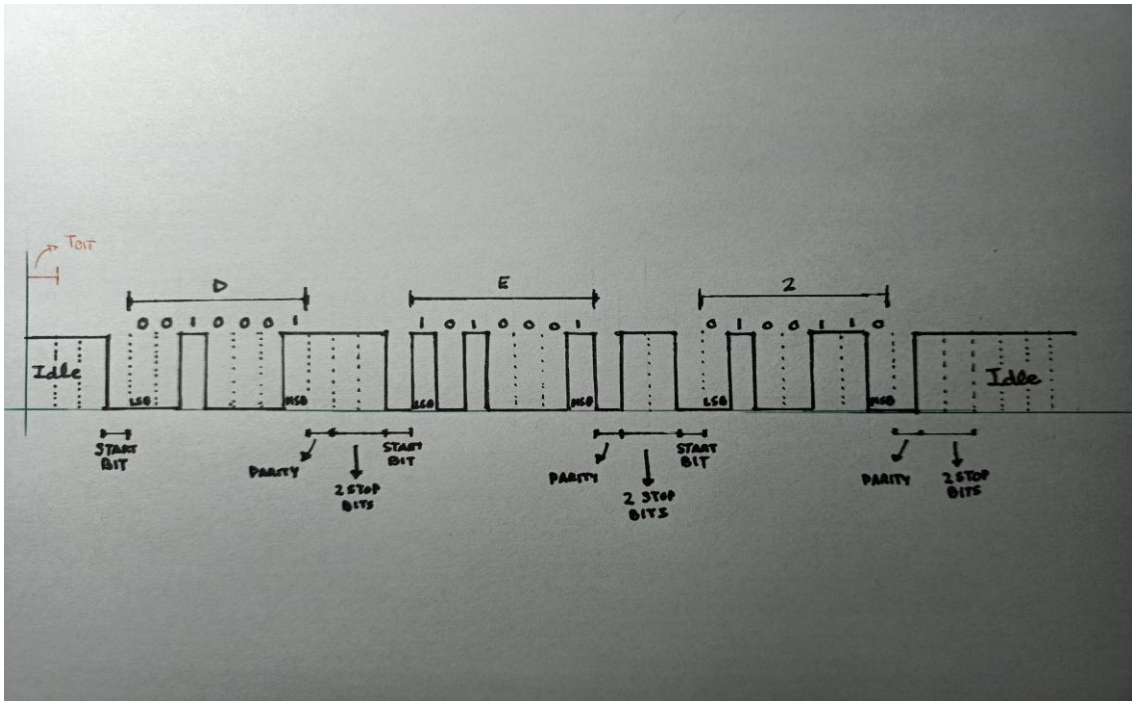


3. **UART**

- (Hand-drawn) picture of UART signal when transmitting data DE2 in 4800 7O2 mode (7 data bits, odd parity, 2 stop bits, 4800 Bd)

If we consider 4800 Bd $\rightarrow T_{BIT} = \frac{1}{4800} \rightarrow$ **T_{BIT} = 208 us**

DE2 = 100 0100 – 100 0101 – 011 0010 (with 7 bits)



- Listing of code for calculating/displaying parity bit

I couldn't find out the solution of the code. I have several ideas about how to solve it but I don't know how to codify them.

1. We could work with frame data

2. We could work with the variable 'value' which is equals to 'ADC' in order to count the number of '1' and '0' and find out the parity

3. We could solve the problem working in rudimentary way which only would be valid for this example: first, we could display different values of voltage; secondly we could count the number of '1' and '0' and finally decide if we chose odd or even parity and display it