DIGITAL ELECTRONICS 2 LAB ASSIGNMENT 7

Name: Demirkan Korbey Baglamac
Github Repository Link: Click Here

1)

Push button	PC0[A0] voltage	ADC value (calculated)	ADC value (measured)
Right	0 V	0	0
Up	0.495 V	101	101
Down	1.204 V	246	245
Left	1.97 V	403	402
Select	3.182 V	651	650
none	5 V	1023	1022

2)

Operation	Register(s)	Bit(s)	Description
Voltage reference	ADMUX	REFS1:0	00: AREF, Internal Vref turned off 01: Avcc voltage reference, 5V 10: Reserved 00: Internal 1.1V Vref with external capacitor at AREF pin
Input channel		MUX3:0	0000: ADC0, 0001: ADC1, 0010: ADC2, 0011: ADC3, 0100: ADC4, 0101: ADC5, 0110: ADC6, 0111: ADC7, 1000: ADC8 1001, 1010, 1011, 1100, 1101: reserved 1110: 1.1V (Vbg) 1111: 0V (GND)
ADC enable	ADCSRA	ADEN	1: ADC is turned on 0: ADC is turned off
Start conversion		ADSC	1: Starts conversion

			0: Writing 0 has no effect, when conversion is complete, it returns to 0 automatically.
ADC interrupt enable		ADIE	1: If I-bit in SREG is set, activates ADC Interrupt
ADC clock prescaler		ADPS2:0	Division Factor 000: 2 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128
ADC result	ADCL and ADCH	ADC9:0	These bits represent the result from the ADC conversion.

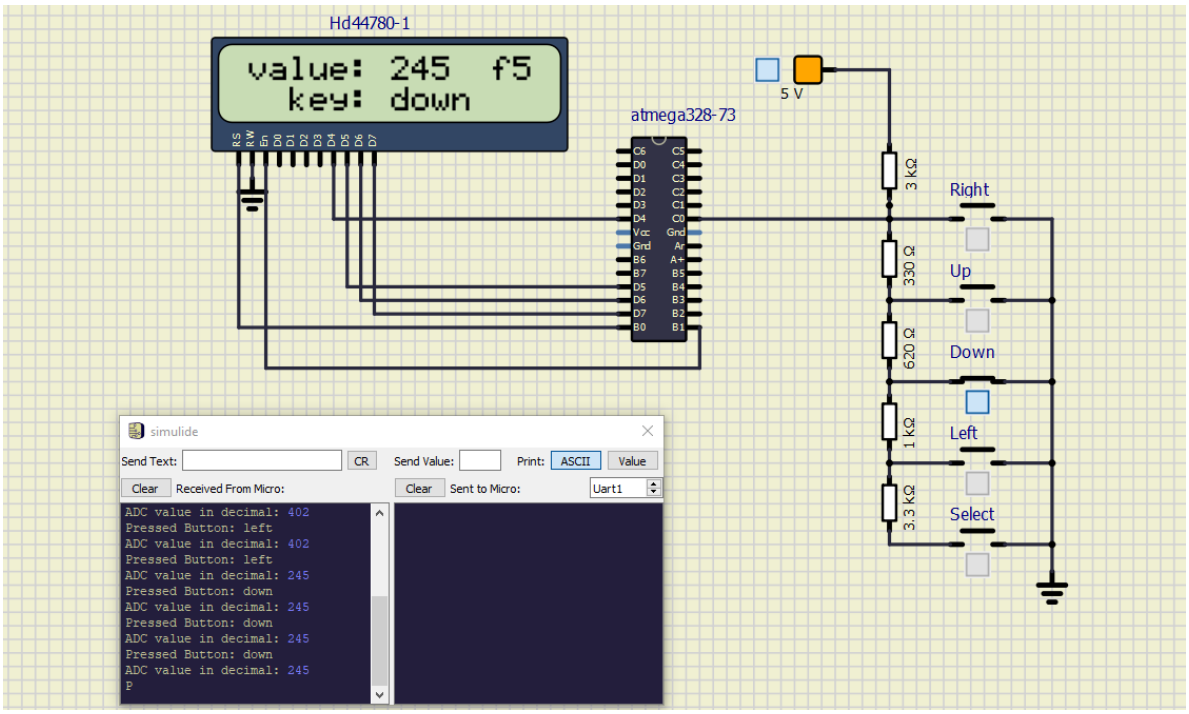
ADC_vect interrupt routine code:

I include the <string.h> library, to use strcpy function. (to change the value of the char array)

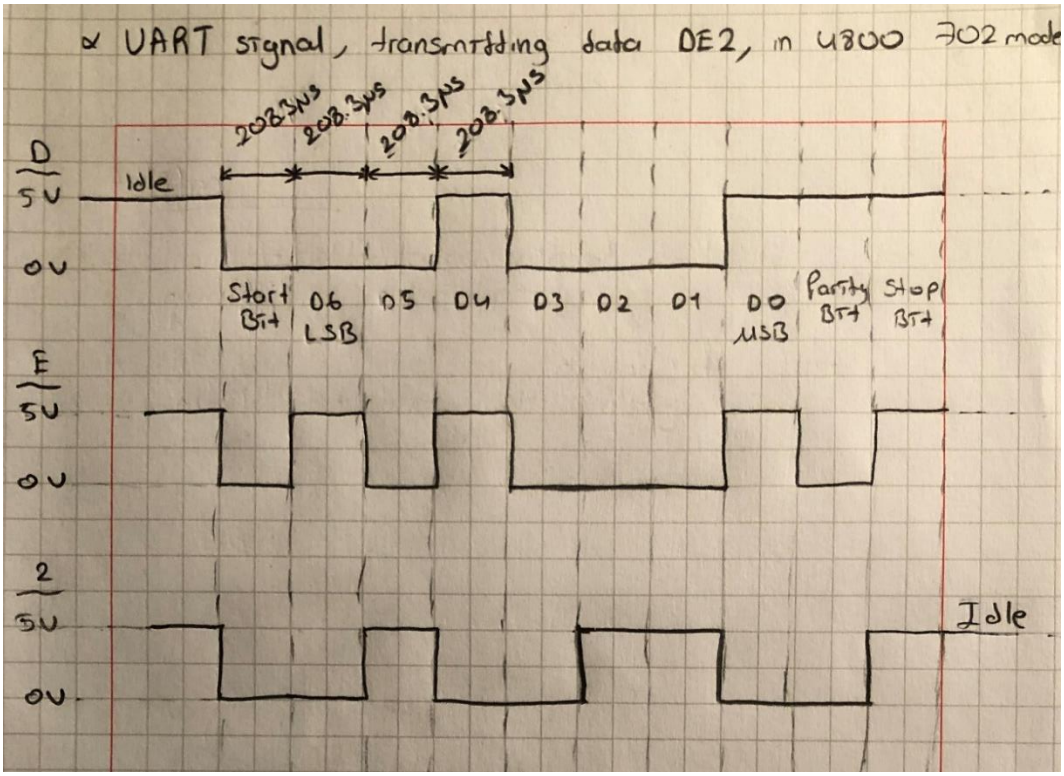
```
/* ------------------------------------------------------------------*/
/**
 * ISR starts when ADC completes the conversion. Display value on LCD
 * and send it to UART.
 */
ISR(ADC_vect)
{
	// WRITE YOUR CODE HERE
	uint16_t value = 0;
	char lcd_string[4] = "0000";
	char pressed_button[6] = "      ";

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

	// Displaying ADC result as decimal
	itoa(value, lcd_string, 10);
	lcd_gotoxy(8,0);
	lcd_puts("    ");
	lcd_gotoxy(8,0);
	lcd_puts(lcd_string);

	// Determining the pressed key
	// None of the buttons are pressed
	if(value > 700)
		strcpy(pressed_button, "none");

	// Select button is pressed
	if(value < 700 && value > 452)
		strcpy(pressed_button, "select");


	// Left button is pressed
	if(value < 452 && value > 295)
		strcpy(pressed_button, "left");
```

```
        // Down button is pressed
        if(value < 295 && value > 151)
                strcpy(pressed_button, "down");

        // Up button is pressed
        if(value < 151 && value > 50)
                strcpy(pressed_button, "up");

        // Right button is pressed
        if(value < 50)
                strcpy(pressed_button, "right");

        //Displaying the pressed button on LCD
        lcd_gotoxy(8,1);
        lcd_puts("      ");
        lcd_gotoxy(8,1);
        lcd_puts(pressed_button);

        // UART
        if (value < 700)
        {
                uart_puts("ADC value in decimal: ");
                uart_puts(lcd_string);
                uart_puts("\r\n");
                uart_puts("Pressed Button: ");
                uart_puts(pressed_button);
                uart_puts("\r\n");
        }

        // Displaying ADC result as hexadecimal
        itoa(value, lcd_string, 16);
        lcd_gotoxy(13,0);
        lcd_puts("   ");
        lcd_gotoxy(13,0);
        lcd_puts(lcd_string);
}
```

Screenshot of the circuit:



3)



Listing of code for calculating/displaying parity bit:

I assume we are using even parity.

```
ISR(ADC_vect)
{
	…
	…
	… Same code, written in the ADC_vect interrupt routine code section.
	…
	…

	// Calculating Parity Bit for ADC value
	// Lets assume we use even parity bit

	uint8_t cnt = 0;

	while(value > 0)
	{
		if(value & 1)
			cnt++;

		value = value >> 1;
	}

	// Displaying Parity Bit for ADC value
	if((cnt % 2) == 0)
		cnt = 0;
	else
		cnt = 1;

	itoa(cnt, lcd_string, 10);
	lcd_gotoxy(15,1);
	lcd_puts(lcd_string);
}
```

Screenshot of the circuit: