FilipPaul sync		065500c 1 minute ago History
..		
.vscode	sync	3 days ago
TEX	sync	3 days ago
include	sync	3 days ago
lib	No commit message	3 days ago
pictures	sync	7 minutes ago
src	sync	7 minutes ago
test	sync	3 days ago
.gitignore	sync	3 days ago
platformio.ini	sync	3 days ago
readme.md	sync	1 minute ago
simultion.simu	No commit message	3 days ago
simultion_backup.simu	sync	7 minutes ago

readme.md

Lab 7

Preparation Task

$$V_{PC0[A_0]}(Right) = 5V \cdot \frac{0}{R_2 + 0} = 0V$$
$$V_{PC0[A_0]}(Up) = 5V \cdot \frac{330\Omega}{330\Omega + 3k\Omega} = 0.495V$$
$$V_{PC0[A_0]}(Down) = 5V \cdot \frac{950\Omega}{950 + 3k\Omega} = 1.203V$$
$$V_{PC0[A_0]}(Left) = 5V \cdot \frac{1950\Omega}{1950\Omega + 3k\Omega} = 1.970V$$
$$V_{PC0[A_0]}(Select) = 5V \cdot \frac{5250\Omega}{5250\Omega + 3k\Omega} = 3.182V$$

When none of pushbutton is pressed the voltage is 5V

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

ADC

Operation	Register(s)	Bit(s)	Description
Voltage reference	ADMUX	REFS1:0	01: AVcc voltage reference, 5V
Input channel	ADMUX	MUX3:0	see fig. bellow
ADC enable	ADCSRA	7-ADEN	if 1: enable
Start conversion	ADSCRA	6-ADSC	write this bit to one to start each conversion. In free running mode, write this bit to one to start the first conversion
ADC interrupt enable	ADCSRA	3-ADIE	When this bit is written to one and the I-bit in SREG is set, the ADC conversion complete interrupt is activated.
ADC clock prescaler	ADCSRA	ADPS2:0	see fig. bellow
ADC result	ADCL and ADCH (depends on ADLAR)	ADC9:0	result

Table 23-4. Input Channel Selections

MUX3..0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Table 23-5. ADC Prescaler Selections

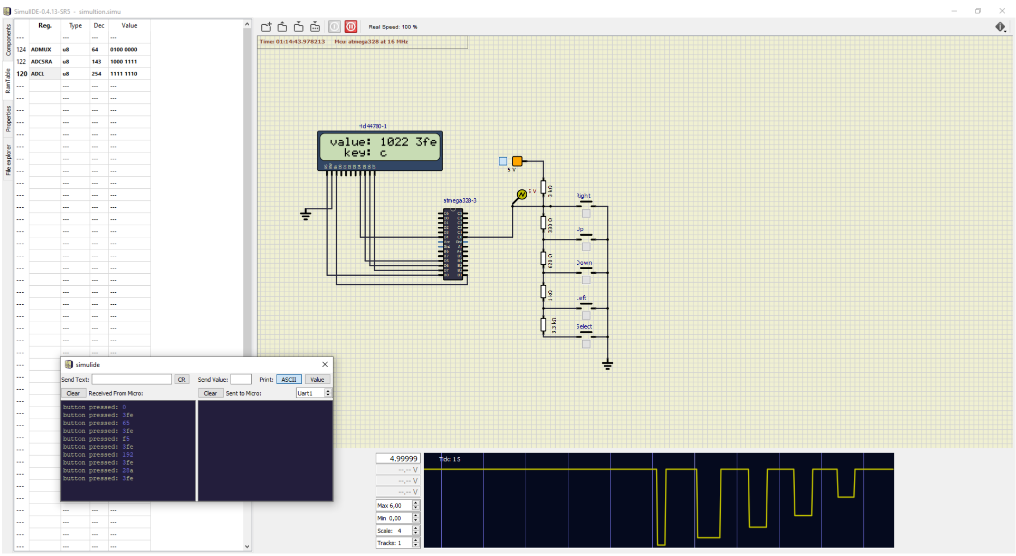
ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

UART

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`	-	Returns in the lower byte the received character and in the higher byte the last receive error. UART_NO_DATA is returned when no data is available	`uart_getc()`
`uart_putc`	unsigned char data	data byte to be transmitted	`uart_putc(unsigned char data)`
`uart_puts`	const char* s	s string to be transmitted	`uart_puts(const char* s)`

UART DE2

Simulation



main.cpp

For some reason, whenever myParity(uint16_t message) is called from ISR, the program stops working. That's the reason why a function is called from while() with delay. I'm not sure, why it behaves like that... Maybe because the microcontroller has not enough time to update the display and send UART message?

```
/***********************************************************************
 *
 * 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 <Arduino.h>
#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

/* Function definitions ----------------------------------------------*/
// returns 1 if odd (1,3.) and 0 if even(0,2...)
    bool myParity(uint16_t message){
    bool parity = 0;
    while (message) //while message == 1
    {
        parity =   ~parity;
        message =  message & (message -1);
    }
    return parity;
    }

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 = 010-0000
    //ADCSRA = 1000 1111
    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);
    // Set clock prescaler to 128
    ADCSRA |= ((1<<ADPS0)|(1<<ADPS1)| (1<<ADPS2));

    // Configure 16-bit Timer/Counter1 to start ADC conversion
    // Enable interrupt and set the overflow prescaler to 262 ms
    TIM1_overflow_262ms();
    TIM1_overflow_interrupt_enable();

    // Initialize UART to asynchronous, 8N1, 9600
    uart_init(UART_BAUD_SELECT(9600,F_CPU));

    // Enables interrupts by setting the global interrupt mask
    sei();

    // Infinite loop
    while (1)
    {
        _delay_ms(100);

    if( myParity(ADC)){
        lcd_gotoxy(12,1);
        lcd_puts(" ");
        lcd_gotoxy(13,1);
        lcd_puts("odd");
        uart_puts(" parity: odd");
        uart_puts("\n");
    }
    else
    {
        lcd_gotoxy(12,1);
        lcd_puts("even");
        uart_puts(" parity: even");
        uart_puts("\n");
    }
    }


    // Will never reach this
    return 0;
}

/* Interrupt service routines ----------------------------------------*/
/**
 * ISR starts when Timer/Counter1 overflows. Use single conversion mode
 * and start conversion four times per second.
 */
ISR(TIMER1_OVF_vect)
{
    // Start ADC conversion
    ADCSRA |= (1<<ADSC);

}

/* -------------------------------------------------------------------*/
/**
 * ISR starts when ADC completes the conversion. Display value on LCD
 * and send it to UART.
 */
ISR(ADC_vect)
{

    // print ADC value
    static uint16_t value = 0;
    static uint16_t pre_value = 0; // to avoid flashing display
    char lcd_string[2] = "";
```

```
        value = ADC;    // Copy ADC result to 16-bit variable
        if(value !=  pre_value){   //clear display whenever value change
        lcd_gotoxy(8,0);
        lcd_puts("    ");
        lcd_gotoxy(13,0);
        lcd_puts("    ");
        itoa(value, lcd_string, 10);    // Convert decimal value to string
        lcd_gotoxy(8,0);
        lcd_puts(lcd_string);
        itoa(value, lcd_string, 16);
        lcd_gotoxy(13,0);
        lcd_puts(lcd_string);
        //print data to serial in HEX monitor via UART
        uart_puts("button pressed: ");
        uart_puts(lcd_string);
        uart_puts("\n");

        }
        pre_value = value;
    }
```