RTC

ZÁVĚREČNÝ PROJEKT DO MIT

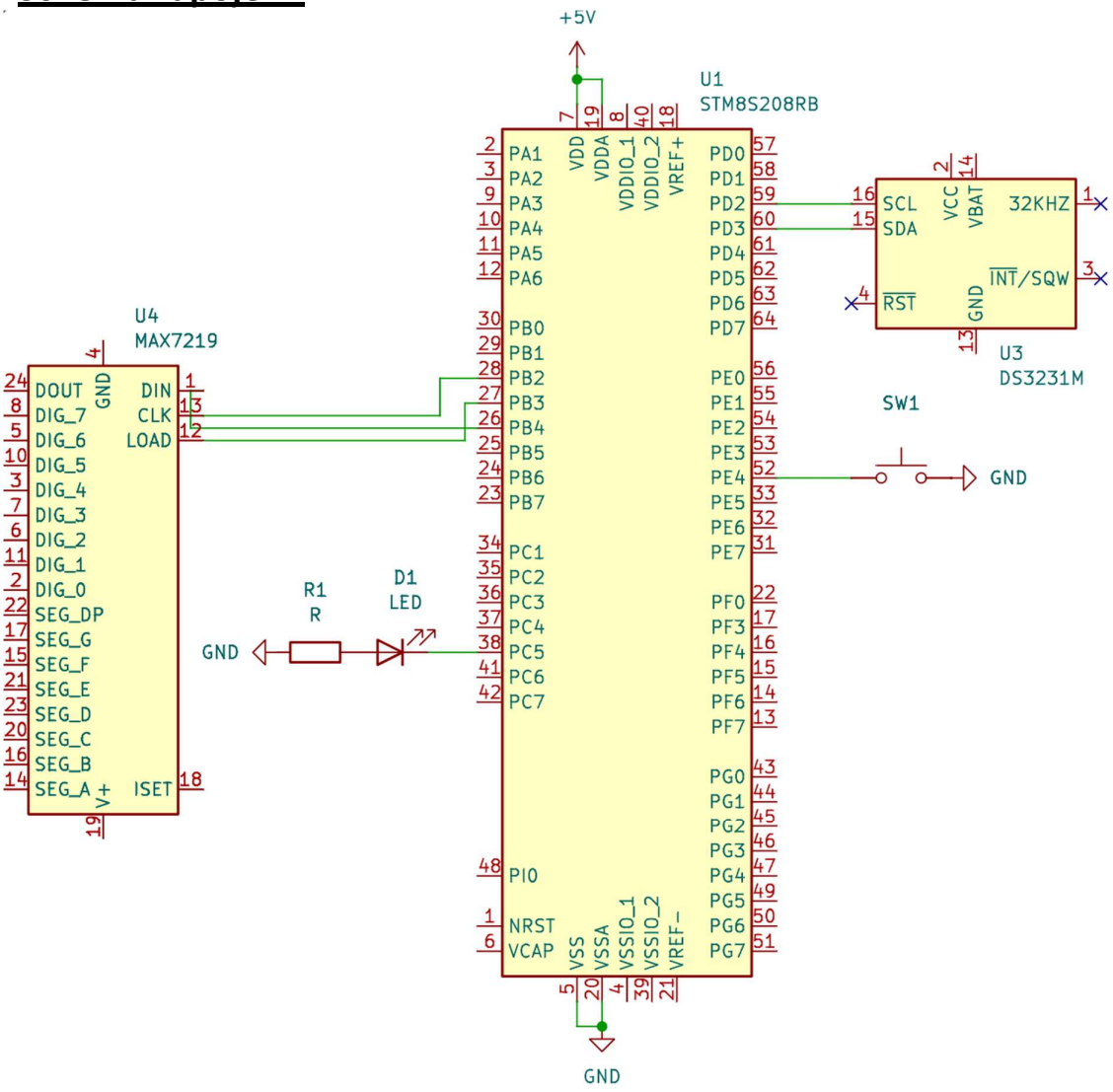
Zadání:

Zařízení ukazující reálný čas:

- Na LED displeji zobrazovat reálný čas
- Pomocí 2 tlačítek měníme čas

Použité součástky: RTC, LED display

Schéma zapojení:



Popis činnosti programu:

Princip programu má sloužit jako hodiny pro zobrazování reálného času.
Po zapojení k napájení se nám na LED displeji zobrazí čas, který nemusí být vždy přesný.
Pro přesnost času musíme zapnout python script v programu, který čas změní na ten správný.

Zdrojový kód:

```
#include <stdbool.h>
#include <stm8s.h>
#include "main.h"
#include "milis.h"
#include "uart1.h"
#include "swi2c.h"
#include <stdio.h>
#include "max7219.h"

#define RTC_ADDR 0x68 << 1
#define DIN_PORT GPIOB
#define DIN_PIN GPIO_PIN_4
#define CS_PORT GPIOB
#define CS_PIN GPIO_PIN_3
#define CLK_PORT GPIOB
#define CLK_PIN GPIO_PIN_2



void display(uint8_t adress, uint8_t data)
{
    uint8_t mask ;
    LOW(CS);            // začátek přenosu

    /* Odesílání adresy */
    mask = 128;
    mask = 0b10000000;
    mask = 1 << 7;              //posun masky
    while(mask){
        if(mask & adress){      //nachystám data
            HIGH(DIN);
        }
        else{
            LOW(DIN);
        }

        HIGH(CLK);
        mask = mask >> 1;
        LOW(CLK);
    }

    /* odesílání dat */
    mask = 0b10000000;
    mask = 1 << 7;              //posun masky
    while(mask){
```

```
        if(mask & data){       //nachystám data
            HIGH(DIN);
        }
        else{
            LOW(DIN);
        }

        HIGH(CLK);
        mask = mask >> 1;
        LOW(CLK);
    }


    HIGH(CS);  // konec přenosu
}
void display_setup(void)
{
    display(DECODE_MODE, 0b11011011);  // zapnutí znakové sady na jednotlivých
digitech
    display(SCAN_LIMIT,7);             // chci všech 8 cifer
    display(INTENSITY, 9);             // chci intenzitu 1, aby to málo svítilo
    display(DISPLAY_TEST, DISPLAY_TEST_OFF);
    display(SHUTDOWN,SHUTDOWN_ON);
    display(DIGIT0,0x0F);
    display(DIGIT1,0x0F);
    display(DIGIT2,0x01);
    display(DIGIT3,0x0F);
    display(DIGIT4,0x0F);           //Zhasne digit 4 a další obdobně
    display(DIGIT5,0x01);
    display(DIGIT6,0x0F);
    display(DIGIT7,0x0F);
}
void sync_time_from_serial(void)
{
    uint8_t time_data[3] = {0, 0, 0};
    while (UART1_GetFlagStatus(UART1_FLAG_RXNE) == RESET) {}
    uint8_t start_byte = UART1_ReceiveData8();
    while (UART1_GetFlagStatus(UART1_FLAG_RXNE) == RESET) {}
        time_data[2] = UART1_ReceiveData8();
        while (UART1_GetFlagStatus(UART1_FLAG_RXNE) == RESET) {}
        time_data[1] = UART1_ReceiveData8();
        while (UART1_GetFlagStatus(UART1_FLAG_RXNE) == RESET) {}
        time_data[0] = UART1_ReceiveData8();
swi2c_write_buf(RTC_ADDR, 0x00, time_data, 3);
}
void init(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
    GPIO_Init(DIN_PORT, DIN_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(CS_PORT, CS_PIN, GPIO_MODE_OUT_PP_HIGH_SLOW);
    GPIO_Init(CLK_PORT, CLK_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(BTN_PORT, BTN_PIN, GPIO_MODE_IN_FL_NO_IT);
    GPIO_Init(LED_PORT, LED_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
```

```
    init_milis();
    init_uart1();
    swi2c_init();

}
int main(void)
{

    init();
    uint32_t time = 0;
    uint8_t precteno[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    uint8_t zapsano[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };


    display_setup();


    printf("\nScan I2C bus:\n");
    printf("Recover: 0x%02X\n", swi2c_recover());
    for (uint8_t addr = 0; addr < 128; addr++) {
        if (swi2c_test_slave(addr << 1) == 0) {
            printf("0x%02X \n", addr);
        }
    }
    printf("------------- scan end --------------------\n");

    /*     Nastavení času v RTC */

    // RTC používá BCD kód, proto používám HEXA
    zapsano[0] = 0x00;          // sekundy
    zapsano[1] = 0x39;          // minuty
    zapsano[2] = 0x10;          // hodiny
    zapsano[3] = 0x01;          // den v týdnu
    zapsano[4] = 0x03;          // den
    zapsano[5] = 0x06;          // měsíc
    zapsano[6] = 0x24;          // rok
/*
    while(!PUSH(BTN));
    printf("Zápis do RTC StatusCode: %X\n",  swi2c_write_buf(0x68 <<1 , 0x00,
zapsano, 7));
*/

    //sync_time_from_serial();



    while (1) {
        if (milis() - time > 100) {
            time = milis();

            swi2c_read_buf(0x68 << 1, 0x00, precteno, 7);
            printf(" %d%d:%d%d:%d%d \n",
```

```
                    precteno[2] >> 4, precteno[2] & 0x0F,
                    precteno[1] >> 4, precteno[1] & 0x0F,
                    precteno[0] >> 4, precteno[0] & 0x0F);

            display(DIGIT0,  precteno[0] & 0x0F);
            display(DIGIT1,  precteno[0] >> 4);
            display(DIGIT3,  precteno[1] & 0x0F);
            display(DIGIT4,  precteno[1] >> 4);
            display(DIGIT6,  precteno[2] & 0x0F);
                display(DIGIT7,  precteno[2] >> 4);




        }
        if (PUSH(BTN)) {
            HIGH(LED);
            sync_time_from_serial();
            LOW(LED);
        }
        uint8_t trash = UART1_ReceiveData8();
    }
}




INTERRUPT_HANDLER(UART1_RX_IRQHandler, 18)
{
}
```

```
import serial
import serial.tools.list_ports
import ntplib
from datetime import datetime, timezone, timedelta


def int_to_bcd(n):
    return (n // 10) << 4 | (n % 10)


def get_ntp_time():
    c = ntplib.NTPClient()
    response = c.request('pool.ntp.org')
    dt = datetime.fromtimestamp(response.tx_time, timezone.utc)
    dt = dt + timedelta(hours=2)

    hours_bcd = int_to_bcd(dt.hour)
    minutes_bcd = int_to_bcd(dt.minute)
    seconds_bcd = int_to_bcd(dt.second)
```

```
    print(f"Hours: {hex(hours_bcd)[2:]}, Minutes: {hex(minutes_bcd)[2:]},
Seconds: {hex(seconds_bcd)[2:]}")

    return hours_bcd, minutes_bcd, seconds_bcd


def find_microcontroller_port():
    ports = list(serial.tools.list_ports.comports())
    for p in ports:
        print(p.description)
        if "STM" in p.description:
            return p.device
    return None


def send_time_to_microcontroller(lol=0):
    port = find_microcontroller_port()
    print(port)
    if port is None:
        print("Microcontroller not found")
        print(list(serial.tools.list_ports.comports()))
        return
    if lol:
        hours_bcd, minutes_bcd, seconds_bcd = get_ntp_time()
        print(f"Sending time to microcontroller: {hours_bcd}, {minutes_bcd},
{seconds_bcd}")

        with serial.Serial(port, 115200, timeout=1) as ser:
            ser.write(bytes([0xff, hours_bcd, minutes_bcd, seconds_bcd]))
 #testmode
    else:
        with serial.Serial(port, 115200, timeout=1) as ser:
            ser.write(bytes([0xff, 0x10, 0x00, 0x11]))


send_time_to_microcontroller(1)
```