Matyáš Vysloužil

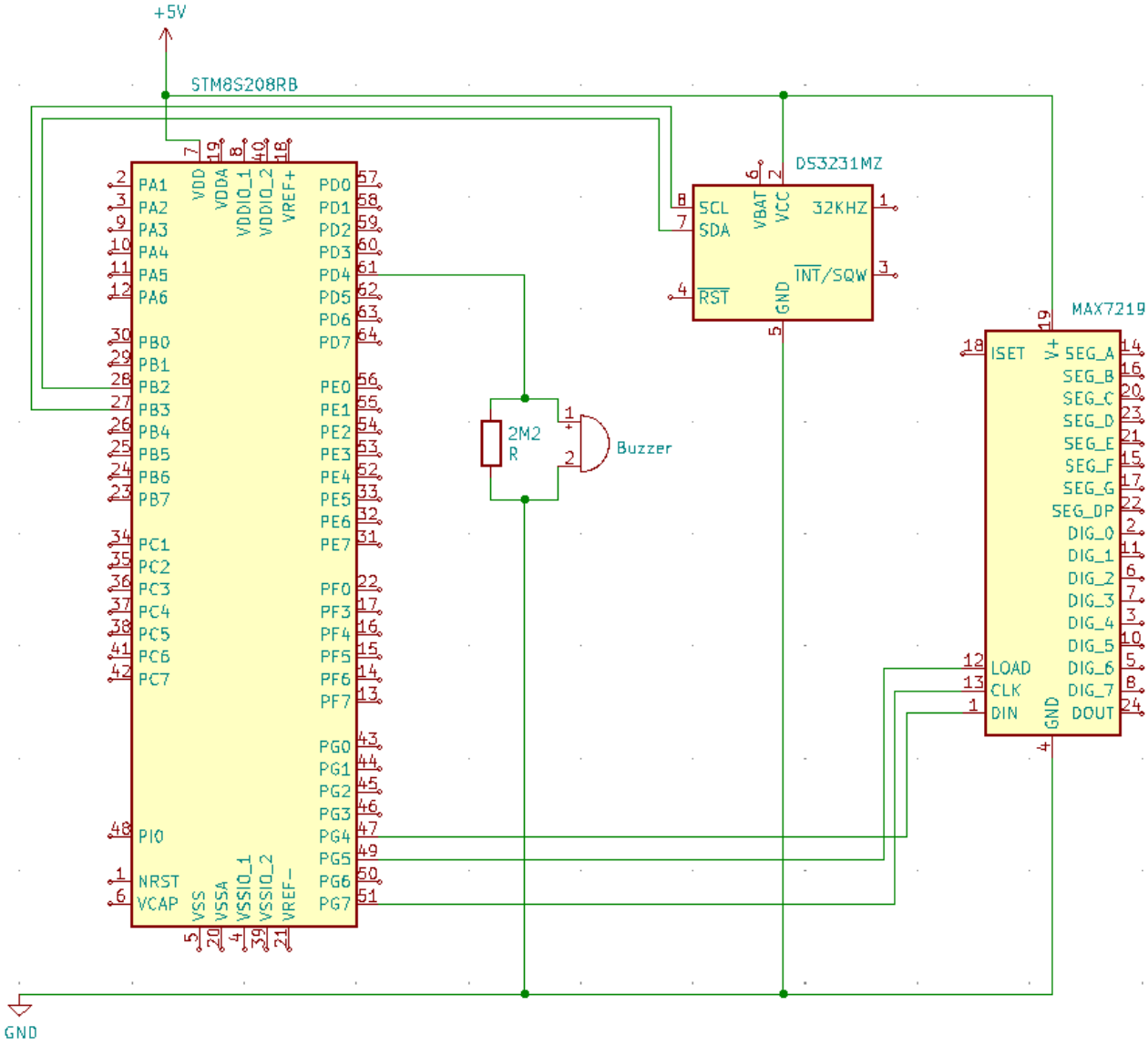
Elektronické hodiny s budíkem

samostatný projekt MIT

Zadání:

- Pomocí řady 7-segmentových displejů a procesoru STM8 vytvořte elektronické hodiny.
- Čas se bude nastavovat/synchronizovat automaticky pomocí RTC DS3231.
- Implementované zvonění/buzení pomocí piezo reproduktoru
- Aktuální čas a odpočet času budíku se bude zobrazovat na řadě 7-segmentových displejů

Schéma zapojení



Vývojový diagram

Popis činnosti programu

Tento kód pro mikrořadič STM8 slouží k zobrazení aktuálního času na sedmisegmentovém displeji a k odpočítávání času, které je spuštěno pomocí tlačítka. Po spuštění se mikrořadič inicializuje, včetně nastavení hodin na 16 MHz a konfigurace GPIO pinů pro komunikaci s čipem MAX7219, tlačítkem a piezo reproduktorem. Následně se inicializují moduly pro časování (milis), sériovou komunikaci (UART) a I2C (swi2c).

Čip MAX7219 je nakonfigurován pro zobrazení znaků na všech osmi číslicích displeje s nízkou intenzitou osvětlení. Na sériový port se poté vypíše seznam všech I2C zařízení připojených na sběrnici.

V hlavní smyčce se každou sekundu načítá aktuální čas z RTC (real-time clock) a zobrazuje se na sedmisegmentovém displeji. Pokud je stisknuto tlačítko, spustí se odpočítávání. Každou minutu se kontroluje stav odpočítávání. Pokud je odpočet aktivní, snižuje se počet minut, a pokud nejsou žádné minuty, snižuje se počet hodin. Po dosažení nuly se aktivuje piezo reproduktor, který začne vydávat přerušovaný zvuk (500 ms zapnuto, 500 ms vypnuto). Tento stav trvá do nekonečna, dokud ho uživatel nevypne.

Tento kód tedy zobrazuje aktuální čas z RTC, umožňuje spuštění odpočítávání pomocí tlačítka a po skončení odpočítávání vydává zvukový signál prostřednictvím piezo reproduktoru.

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"
#include "delay.h"

// Definice portů a pinů pro komunikaci s MAX7219
#define DIN_PORT GPIOG
#define DIN_PIN GPIO_PIN_4
#define CS_PORT GPIOG
#define CS_PIN GPIO_PIN_5
#define CLK_PORT GPIOG
#define CLK_PIN GPIO_PIN_7

// Definice portu a pinu pro piezo reproduktor
#define PZ_PORT GPIOD
#define PZ_PIN GPIO_PIN_4
```

```
// Makra pro ovládání piezo reproduktoru
#define PZ_ON GPIO_WriteHigh(PZ_PORT, PZ_PIN);
#define PZ_OFF GPIO_WriteLow(PZ_PORT, PZ_PIN);
#define PZ_REVERSE GPIO_WriteReverse(PZ_PORT, PZ_PIN);

// Inicializační funkce
void init(void) {
    // Nastavení hodin MCU na 16MHz
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);

    // Inicializace GPIO pinů pro komunikaci s MAX7219
    GPIO_Init(DIN_PORT, DIN_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(CS_PORT, CS_PIN, GPIO_MODE_OUT_PP_HIGH_SLOW); // Start high
    GPIO_Init(CLK_PORT, CLK_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);

    // Inicializace GPIO pinu pro tlačítko
    GPIO_Init(BTN_PORT, BTN_PIN, GPIO_MODE_IN_FL_NO_IT);

    // Inicializace GPIO pinu pro piezo reproduktor
    GPIO_Init(PZ_PORT, PZ_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);

    // Inicializace dalších modulů
    init_milis();
    init_uart1();
    swi2c_init();
}

// Funkce pro zasílání dat na MAX7219
void display(uint8_t address, uint8_t data) {
    uint8_t mask;

    LOW(CS); // Začátek přenosu

    /* Zasílání adresy */
    mask = 1 << 7;
    while (mask) {
        if (address & mask) { // Příprava dat
            HIGH(DIN);
        } else {
            LOW(DIN);
        }
        HIGH(CLK);
        mask = mask >> 1;
        LOW(CLK);
    }

    /* Zasílání dat */
    mask = 1 << 7;
    while (mask) {
        if (data & mask) { // Příprava dat
            HIGH(DIN);
```

```
        } else {
            LOW(DIN);
        }
        HIGH(CLK);
        mask = mask >> 1;
        LOW(CLK);
    }

    HIGH(CS); // Konec přenosu
}

// Nastavení MAX7219
void setup_max7219(void) {
    display(DECODE_MODE, 0b11111111); // Zapnutí znakové sady na každé číslici
    display(SCAN_LIMIT, 7); // Použití všech 8 číslic
    display(INTENSITY, 1); // Nízká intenzita
    display(DISPLAY_TEST, DISPLAY_TEST_OFF);
    display(SHUTDOWN, SHUTDOWN_ON);
}

// Funkce pro zobrazení času na displeji
void show_time(uint8_t hours, uint8_t minutes, uint8_t hours1, uint8_t minutes1) {
    display(DIGIT7, hours1 / 10);
    display(DIGIT6, (hours1 % 10 | 0x80));
    display(DIGIT5, minutes1 / 10);
    display(DIGIT4, minutes1 % 10);
    display(DIGIT3, hours / 10);
    display(DIGIT2, (hours % 10 | 0x80)); // Přidání 0x80 k číslici pro zapnutí tečky
    display(DIGIT1, minutes / 10);
    display(DIGIT0, minutes % 10);
}

// Funkce pro zjištění, zda je tlačítko stisknuto
bool is_button_pressed(void) {
    return (GPIO_ReadInputData(BTN_PORT) & BTN_PIN) == 0; // Předpokládá se tlačítko s aktivní
nízkou úrovní
}

// Hlavní funkce programu
int main(void) {
    uint32_t last_time_check = 0; // Poslední čas kontroly v milisekundách
    uint32_t last_minute_check = 0; // Poslední čas kontroly v milisekundách
    uint8_t hours1 = 6; // Počáteční hodiny nastavené na 6
    uint8_t minutes1 = 30; // Počáteční minuty nastavené na 30
    uint8_t precteno[7] = {0, 0, 0, 0, 0, 0, 0}; // Pole pro načtení času z RTC
    uint8_t err;
    bool countdown_started = false; // Příznak spuštění odpočtu

    // Inicializace systému a MAX7219
    init();
    setup_max7219();
```

```
// Výpis na sériovou linku
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");

// Hlavní smyčka programu
while (1) {
    uint32_t current_time = milis(); // Aktuální čas v milisekundách

    // Kontrola aktuálního času a stisknutí tlačítka každou sekundu
    if (current_time - last_time_check > 1000) {
        last_time_check = current_time;

        // Načtení aktuálního času z RTC a zobrazení na číslicích 0-3
        err = swi2c_read_buf(0x68 << 1, 0x00, precteno, 7);
        if (err == 0) {
            uint8_t hours = (precteno[2] >> 4) * 10 + (precteno[2] & 0x0F);
            uint8_t minutes = (precteno[1] >> 4) * 10 + (precteno[1] & 0x0F);
            show_time(hours1, minutes1, hours, minutes);
            // Výpis času z RTC na sériovou linku v BCD formátu
            printf("%d%d.%d%d. 20%d%d %d%d:%d%d:%d%d \n",
                   precteno[4] >> 4, precteno[4] & 0x0F,
                   precteno[5] >> 4, precteno[5] & 0x0F,
                   precteno[6] >> 4, precteno[6] & 0x0F,
                   precteno[2] >> 4, precteno[2] & 0x0F,
                   precteno[1] >> 4, precteno[1] & 0x0F,
                   precteno[0] >> 4, precteno[0] & 0x0F);
        }

        // Kontrola stisknutí tlačítka každou sekundu
        if (is_button_pressed() && !countdown_started) {
            countdown_started = true; // Spuštění odpočtu
        }
    }

    // Logika odpočtu každou minutu
    if (current_time - last_minute_check > 60000) {
        last_minute_check = current_time;

        if (countdown_started) {
            if (minutes1 > 0) {
                minutes1--; // Snížení minut
            } else if (hours1 > 0) {
                hours1--; // Snížení hodin a nastavení minut na 59
                minutes1 = 59;
```

```
                }
            }
            // Kontrola, zda odpočet dosáhl nuly
             if (countdown_started && hours1 == 0 && minutes1 == 0) {
                while (1){ // Nekonečná smyčka
                    PZ_ON;
                    delay_ms(500); // 500 ms zapnuto
                    PZ_OFF;
                    delay_ms(500); // 500 ms vypnuto
                }
            }
        }
    }
}

/*------------------------------  Assert -----------------------------------*/
#include "__assert__.h"
```

https://github.com/MatY51/MIT-projekt