Hodiny s obvodem RTC

Zpracoval: Petr Losert 4 A

**Blokové schéma zapojení**

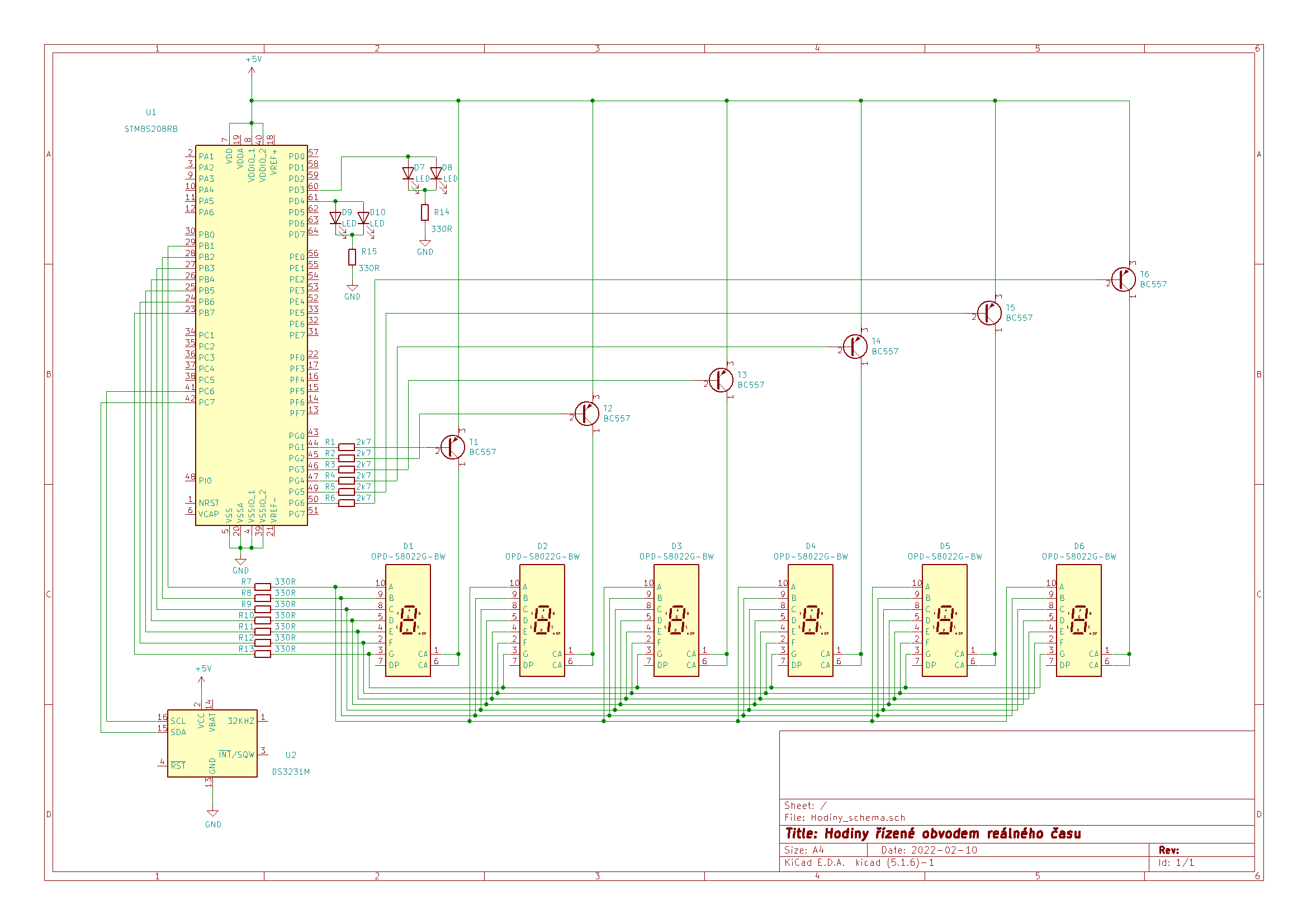


**Seznam použitých součástek**

|  |  |  |  |
| --- | --- | --- | --- |
| **Součástka** | **Typ/Hodnota** | **Počet** | **Poznámka** |
| Obvod RTC | DS3231 | 1 | - |
| STM kit | STM8S208RB | 1 | - |
| Rezistor | 330 Ω | 9 | Ochrana 7seg a LED |
| Rezistor | 2k7 | 6 | Bázové rezistory |
| 7 - segment displej | OPD-S8022G-BW | 6 | Společná anoda |
| PNP tranzistor | BC327-40 | 6 | Spínací tranzistor |
| LED dioda | 5 mm | 2 | Červená, 20 mA |
| LED dioda | 5 mm, hranatá | 2 | Žlutá, 20 mA |

**Slovní popis zapojení**

Celé zapojení jsem realizoval na dvou nepájivých polích. Jedno nepájivé pole je skládací a složil jsem ho přesně na velikost 7 – segmentových displejů. Prostor pro displeje bylo nutné vytvořit hlavně kvůli je jich velikosti. Použil jsem typ OPD-S8022G-BW, který má výšku digitu 20,40 mm, což je už větší typ displeje. Záměrem byla čitelnost i z větší vzdálenosti. Na složeném poli jsou dále 4 střídavě blikající LED diody umístěné mezi dvojicemi displejů. Pro každou barevnou dvojici je na poli ochranný rezistor. Druhé nepájivé pole je klasické pevně dané velikosti. Na toto pole jsem umístil ochranné rezistory pro 7 – segmentové displeje, spínací tranzistory PNP a bázové rezistory spínacích rezistorů. Dalším prvkem je obvod reálného času (RTC) obvod, který je do nepájivých polí připojen pro přivedení napájení do RTC obvodu. Vodiče I2C sběrnice SCL a SDA jsou z RTC obvodu připojeny rovnou na STM kit. Veškerá propojení mezi RTC obvodem, STM kitem a oběma nepájivými poli jsou realizovány pomocí propojovacích klasických kabelů. Vzhledem k typům zakončení jsem musel použít kabely samec-samec nebo samec-samice. Z STM kitu mám takto zapojeno napájení, báze tranzistorů (PG1 – PG6), jednotlivé segmenty (PB1 – PB7) a PWM piny (PD3 a PD4). K RTC obvodu jsou všechny 4 vodiče připájeny. Pro napájení celého zařízení využívám zdroj 5 V z STM kitu.

**Schéma zapojení**

**Vývojový diagram programu**

****

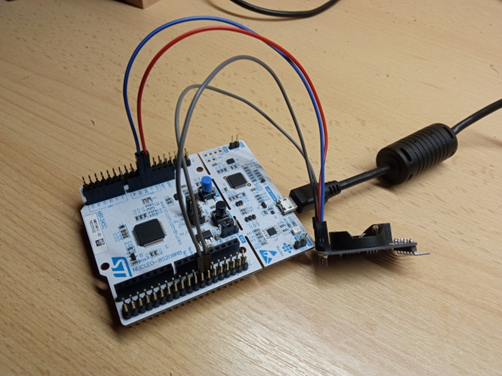
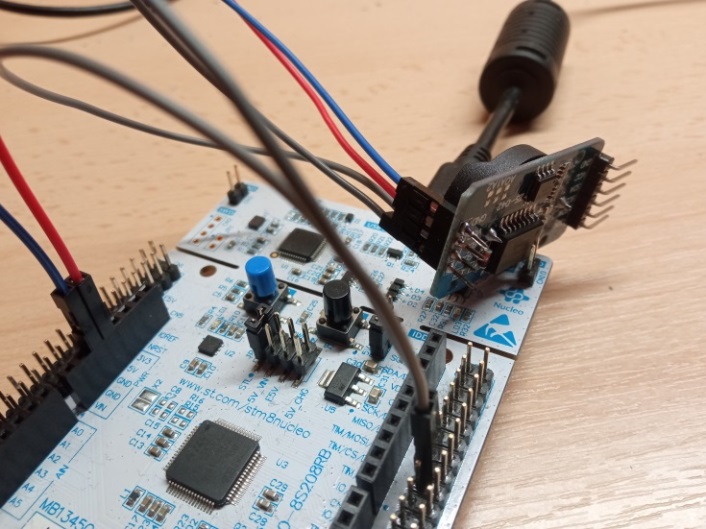
**Slovní popis funkce programu**

V hodinách je použit obvod reálného času DS3231 (RTC). Principem funkce je každých 100 ms vyčítat obsah RTC, ukládat ho a zpracovávat pomocí funkce na data zobrazitelná na 7 - segmentových displejích. V RTC obvodu je možné vyčíst mnoho informací o čase (od sekund až po století). Já jsem ve svých hodinách využil možnosti vyčíst, zpracovat a zobrazit informace od sekund až po hodiny, tedy celkem šest číselných údajů. Komunikace mezi STM kitem a obvodem RTC probíhá přes I2C sběrnici. Kromě napájení spojují STM kit a RTC obvod tedy ještě vodiče SCL (hodiny) a SDA (data), které tuto sběrnici tvoří, a po kterých běží komunikace mezi RTC obvodem a STM kitem. Veškerá komunikace s RTC obvodem probíhá pomocí dvou funkcí. Jedna slouží pro zápis dat do RTC obvodu (pro nastavení nějakého času) a druhá pro čtení dat z RTC obvodu. Pro každou z funkcí mám vytvořené odděleně pole o 7 bytech s daty nebo pro uložení dat. Po spuštění programu na pozadí běží dva timery. První (timer 3) slouží pro obsluhu displejů a druhý (timer 2) slouží pro PWM pro obsluhu LEDek mezi displeji.

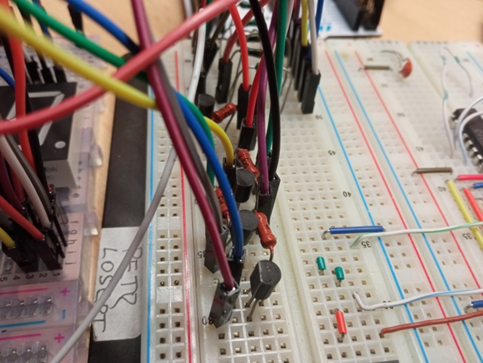
Zobrazení časových údajů na displeje jsem vyřešil pomocí rychlého přepínání mezi jednotlivými displeji (digity) pomocí PNP spínacích tranzistorů. Sepnu vždy tranzistor u digitu, na kterém chci něco zobrazit. Tím přivedu na displeji se společnou anodou na anodu plus a zobrazím danou číslici. Poté tranzistor uzavřu a sepnu další. Takto jednotlivé digity střídám pořád dokola.

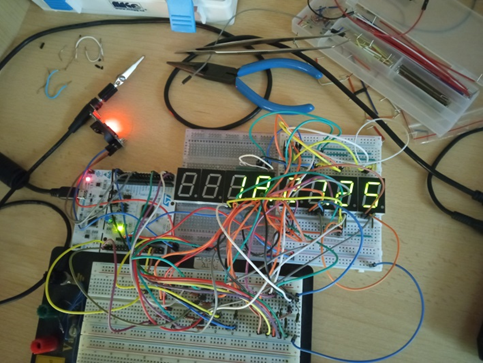
Samostatnou část projektu tvoří blikající LED diody, které slouží pouze jako doplňující prvek hodin. Blikání je realizované pomocí pulzně šířkové modulace (PWM), kterou ovládá timer 2. Základním principem PWM jsou změny střídy signálu v čase. Na základě těchto změn se nám tedy v čase mění průměrný výkon dodávaný do LED diod. Tyto změny výkonu vidíme jako změny jasu svitu LED diody. Pro obsluhu PWM je v programu také oddělená funkce.

**Fotodokumentace**

****

**Zhodnocení**

****

****

**Zhodnocení**

Hodiny jsem úspěšně realizoval a jsou funkční. Podařilo se mi vyřešit problém s tím, že se mi jednotlivé číslice z neznámého důvodu přepisovaly do sebe a číselné údaje byly nečitelné. Myslel jsem si, že chybu mám v programu, ale ten byl nakonec v pořádku. Hodiny jsem totiž zkusil přestavět, jestli jsem neudělal někde chybu v zapojení. Konkrétní chybu jsem sice neobjevil, ale vím, že to zapojením nakonec bylo. Nejpravděpodobnější je podle mě, že jsem omylem data určená pro daný displej připojil k více displejům současně. Cílem také bylo, aby byly displeje dostatečně velké a příjemné pro čtení času. Čitelnost času bych určitě označil jako velkou výhodu mého zapojení. Další výhodou je podle mě přesnost funkce. Pokud do hodin nahraju nějaký čas, hodiny spolehlivě fungují. Nevýhodou je velké množství použitých propojovacích vodičů, díky kterým může být zapojení trošku nepřehledné. Vodiče jsou však barevně rozlišeny a jako autor projektu se v zapojení dokážu ihned orientovat. Na desce s displeji jsem se snažil alespoň nějaká propojení realizovat pomocí speciálně vyrobených propojek, abych obyčejné vodiče nemusel používat všude. Úplně první prototyp hodin byl realizovaný pouze klasickými propojovacími vodiči. Další nevýhodou by mohlo být, že zobrazuju čas pouze po desítky hodin, když RTC obvod umožňuje uschovávat informace až po století. Myslím si však, že takto je to nejběžnější a klasické digitální hodiny obsahují většinou těchto šest informací. Bral bych to tedy spíš jako námět a výhled do budoucna, připojit ještě jedno obdobně složené pole a zpracovávat i další data, která RTC umožňuje. Projekt se tímto tedy ještě bude moct vylepšit.

Celý projekt jsem si rozdělil do určitých částí, které jsem postupně vykonával. Jedním z problémů bylo vyřešit zobrazování času na displeje, kde jsem musel vyřešit správné přepínání jednotlivých digitů. Řešení jsem nakonec našel ve spínání obyčejných PNP tranzistorů, kde každý postupně spíná jeden digit (displej). Dalším krokem pro mě bylo vyřešení komunikace mezi STM kitem a RTC obvodem pomocí I2C sběrnice. Zcela samostatný problém jsem řešil u PWM, která s hodinami přímo nesouvisí, ale využil jsem k ní další timer, takže mi běží dva timery současně.

V projektu jsem pracoval na dílčích částech, z nichž každá mě obohatila, a které jsem následně spojil dohromady. Nové pro mě bylo použití I2C sběrnice, se kterou jsem doteď neměl moc zkušeností, a také využití PWM.

**Přílohy**

**Program**

#include "stm8s.h"

#include "milis.h" //inkludování knihoven

#include "swi2c.h"

#define SEGMENT\_A\_PORT GPIOB

#define SEGMENT\_B\_PORT GPIOB

#define SEGMENT\_C\_PORT GPIOB //Makra pro jednotlivé segmenty (port B)

#define SEGMENT\_D\_PORT GPIOB

#define SEGMENT\_E\_PORT GPIOB

#define SEGMENT\_F\_PORT GPIOB

#define SEGMENT\_G\_PORT GPIOB

#define SEGMENT\_A\_PIN GPIO\_PIN\_1

#define SEGMENT\_B\_PIN GPIO\_PIN\_2

#define SEGMENT\_C\_PIN GPIO\_PIN\_3 //Makra pro jednotlivé segmenty (piny PB1 – PB7)

#define SEGMENT\_D\_PIN GPIO\_PIN\_4

#define SEGMENT\_E\_PIN GPIO\_PIN\_5

#define SEGMENT\_F\_PIN GPIO\_PIN\_6

#define SEGMENT\_G\_PIN GPIO\_PIN\_7

#define DIGIT\_1\_PORT GPIOG

#define DIGIT\_2\_PORT GPIOG //Makra pro spínání jednotlivých digitů (port G)

#define DIGIT\_3\_PORT GPIOG

#define DIGIT\_4\_PORT GPIOG

#define DIGIT\_5\_PORT GPIOG

#define DIGIT\_6\_PORT GPIOG

#define DIGIT\_1\_PIN GPIO\_PIN\_1

#define DIGIT\_2\_PIN GPIO\_PIN\_2 //Makra pro spínání jednotlivých digitů (piny PG1 – PG6)

#define DIGIT\_3\_PIN GPIO\_PIN\_3

#define DIGIT\_4\_PIN GPIO\_PIN\_4

#define DIGIT\_5\_PIN GPIO\_PIN\_5

#define DIGIT\_6\_PIN GPIO\_PIN\_6

#define RTC\_ADRESS 0b11010000 //Makro pro adresu RTC obvodu

//PB0 - 1 1 1 1 1 1 1 1 1 1

//PB1 A 0 1 0 0 1 0 0 0 0 0

//PB2 B 0 0 0 0 0 1 1 0 0 0

//PB3 C 0 0 1 0 0 0 0 0 0 0

//PB4 D 0 1 0 0 1 0 0 1 0 0

//PB5 E 0 1 0 1 1 1 0 1 0 1

//PB6 F 0 1 1 1 0 0 0 1 0 0

//PB7 G 1 1 0 0 0 0 0 1 0 0

// 0 1 2 3 4 5 6 7 8 9

void read\_RTC(void); //funkce, která každých 100 ms vyčítá informace z RTC

void process\_RTC(void); //funkce, která nám vyčtené hodnoty převede

void GPIO\_Set(void); //funkce, která inicializuje potřebné piny

void init\_timer(void); //funkce, která nastaví timer 3

volatile uint8\_t error;

volatile uint8\_t RTC\_precteno[7]; // pole o délce 7 bytů, kam ukládám data o čase

volatile uint8\_t zapis[7]; //pole o délce 7 bytů, ze kterého zapisuju data do RTC

void set\_pwm(void); //funkce, která nastaví parametry Timeru 2

void pwm\_change(void); //funkce, která se stará o střídání LED diod a změny stříd

uint8\_t pomocna\_promenna1 = 0; //proměnné zajišťuje vstup do správné podmínky u PWM (aby se LEDky střídaly)

uint8\_t pomocna\_promenna2 = 0;

#define time 2 //čas mezi změnami střídy u PWM

#define TIMER\_TOP (16000-1) //strop časovače je 16000 impulzů

const uint16\_t znak[] = //pole obsahující znaky, které zapisuju

{

0b10000000, //0

0b11110011, //1

0b01001001, //2

0b01100001, //3

0b00110011, //4

0b00100101, //5

0b00000101, //6

0b11110001, //7

0b00000001, //8

0b00100001, //9

};

uint8\_t sec; //proměnné pro zpracování dat z RTC a zapsání dat na displeje

uint8\_t des\_sec;

uint8\_t min;

uint8\_t des\_min;

uint8\_t hod;

uint8\_t des\_hod;

uint8\_t zbytek\_hod;

void main(void){

CLK\_HSIPrescalerConfig(CLK\_PRESCALER\_HSIDIV1); // 16MHz z interního RC oscilátoru

init\_milis();

set\_pwm();

swi2c\_init();

GPIO\_Set();

init\_timer();

enableInterrupts(); //globálně povolí přerušení

zapis[0] = 0b00000000; //hodnoty zapisované do RTC obvodu

zapis[1] = 0b00000101;

zapis[2] = 0b00100011;

swi2c\_write\_buf(RTC\_ADRESS,0x00,zapis,3); //zápis dat do RTC obvodu

while (1){ //nekonečná smyčka

read\_RTC();

process\_RTC();

pwm\_change();

}

}

void pwm\_change(void){

static int16\_t strida2 = 0; //střída červených LED

static int16\_t strida1 = 0; //střída žlutých LED

static uint16\_t minuly\_cas = 0; //počáteční hodnota času pro porovnávání v podmínkách

if(pomocna\_promenna1 == 0 && pomocna\_promenna2 == 0){ //žluté LED se rozsvěcí

if((milis() - minuly\_cas) >= time){

strida1=strida1+200;

minuly\_cas=milis();

if(strida1 >= TIMER\_TOP){pomocna\_promenna1=1;}

TIM2\_SetCompare1(strida1); // nastav střídu kanálu 1 (TIM2\_CH1, PD4)

}

}

if(pomocna\_promenna1 == 1 && pomocna\_promenna2 == 0){ //žluté LED zhasínají

if((milis() - minuly\_cas) >= time){

strida1=strida1-200;

minuly\_cas=milis();

if(strida1 <= 0)

{

pomocna\_promenna2=1;

}

TIM2\_SetCompare1(strida1); // nastav střídu kanálu 1 (TIM2\_CH1, PD4)

}

}

if(pomocna\_promenna1 == 1 && pomocna\_promenna2 == 1){ //červené LED se rozsvěcí (dvakrát pomaleji než žluté)

if((milis() - minuly\_cas) >= time){

strida2=strida2+100;

minuly\_cas=milis();

if(strida2 >= TIMER\_TOP){pomocna\_promenna1=0;}

TIM2\_SetCompare2(strida2); // nastav střídu kanálu 2 (TIM2\_CH2, PD3)

}

}

if(pomocna\_promenna1 == 0 && pomocna\_promenna2 == 1){ //červené LED zhasínají (dvakrát pomaleji než žluté)

if((milis() - minuly\_cas) >= time){

strida2=strida2-100;

minuly\_cas=milis();

if(strida2 <= 0)

{

pomocna\_promenna2=0;

}

TIM2\_SetCompare2(strida2); // nastav střídu kanálu 2 (TIM2\_CH2, PD3)

}

}

}

void set\_pwm(void){

GPIO\_Init(GPIOD, GPIO\_PIN\_4, GPIO\_MODE\_OUT\_PP\_LOW\_SLOW); //nastavím piny pro PWM

GPIO\_Init(GPIOD, GPIO\_PIN\_3, GPIO\_MODE\_OUT\_PP\_LOW\_SLOW);

TIM2\_TimeBaseInit(TIM2\_PRESCALER\_1, TIMER\_TOP); //inicializace časové základny

TIM2\_OC1Init( //kanál 1 (TM2\_CH1)

TIM2\_OCMODE\_PWM1,

TIM2\_OUTPUTSTATE\_ENABLE,

0,

TIM2\_OCPOLARITY\_HIGH); //Rozsvěcím úrovní HIGH (oba kanály timeru 2)

TIM2\_OC2Init( //kanál 2 (TM2\_CH2)

TIM2\_OCMODE\_PWM1,

TIM2\_OUTPUTSTATE\_ENABLE,

0,

TIM2\_OCPOLARITY\_HIGH);

// aktivuji na všech kanálech preload (zajišťuje změnu střídy bez nežádoucích efektů)

TIM2\_OC1PreloadConfig(ENABLE);

TIM2\_OC2PreloadConfig(ENABLE);

TIM2\_OC3PreloadConfig(ENABLE);

TIM2\_Cmd(ENABLE); // spustíme timer

}

INTERRUPT\_HANDLER(TIM3\_UPD\_OVF\_BRK\_IRQHandler, 15){ //funkce pro obsluhu displejů

static uint8\_t digit = 1;

TIM3\_ClearITPendingBit(TIM3\_IT\_UPDATE);

if(digit == 1){

GPIO\_WriteHigh(DIGIT\_6\_PORT,DIGIT\_6\_PIN);

GPIO\_WriteLow(DIGIT\_1\_PORT,DIGIT\_1\_PIN);

GPIO\_Write(SEGMENT\_A\_PORT,znak[sec]); //sekundy

//GPIO\_WriteHigh(DIGIT\_1\_PORT,DIGIT\_1\_PIN);

}else if(digit == 2){

GPIO\_WriteHigh(DIGIT\_1\_PORT,DIGIT\_1\_PIN);

GPIO\_WriteLow(DIGIT\_2\_PORT,DIGIT\_2\_PIN);

GPIO\_Write(SEGMENT\_B\_PORT,znak[des\_sec]); //desítky sekund

//GPIO\_WriteHigh(DIGIT\_2\_PORT,DIGIT\_2\_PIN);

}else if(digit == 3){

GPIO\_WriteHigh(DIGIT\_2\_PORT,DIGIT\_2\_PIN);

GPIO\_WriteLow(DIGIT\_3\_PORT,DIGIT\_3\_PIN);

GPIO\_Write(SEGMENT\_C\_PORT,znak[min]); //minuty

// GPIO\_WriteHigh(DIGIT\_3\_PORT,DIGIT\_3\_PIN);

}else if(digit == 4){

GPIO\_WriteHigh(DIGIT\_3\_PORT,DIGIT\_3\_PIN);

GPIO\_WriteLow(DIGIT\_4\_PORT,DIGIT\_4\_PIN);

GPIO\_Write(SEGMENT\_D\_PORT,znak[des\_min]); //desítky minut

//GPIO\_WriteHigh(DIGIT\_4\_PORT,DIGIT\_4\_PIN);

}else if(digit == 5){

GPIO\_WriteHigh(DIGIT\_4\_PORT,DIGIT\_4\_PIN);

GPIO\_WriteLow(DIGIT\_5\_PORT,DIGIT\_5\_PIN);

GPIO\_Write(SEGMENT\_E\_PORT,znak[hod]); //hodiny

//GPIO\_WriteHigh(DIGIT\_5\_PORT,DIGIT\_5\_PIN);

}else if(digit == 6){

GPIO\_WriteHigh(DIGIT\_5\_PORT,DIGIT\_5\_PIN);

GPIO\_WriteLow(DIGIT\_6\_PORT,DIGIT\_6\_PIN);

GPIO\_Write(SEGMENT\_F\_PORT,znak[des\_hod]); //desítky hodin

//GPIO\_WriteHigh(DIGIT\_6\_PORT,DIGIT\_6\_PIN);

}

digit++; //inkrementuju proměnnou digit o 1

if(digit > 6){ //pokud je digit = 7, nastaví se zpět na 1

digit = 1;

}

}

void read\_RTC(void){

static uint16\_t last\_time=0; // každých 100ms přečte obsah RTC

if(milis() - last\_time >= 100){

last\_time = milis();

error=swi2c\_read\_buf(RTC\_ADRESS,0x00,RTC\_precteno,7);

}

}

void process\_RTC(void){ //zpracování dat z RTC

sec = (RTC\_precteno[0] & 0b00001111); //sekundy

des\_sec = ((RTC\_precteno[0] >> 4) & 0b00001111); //desítky sekund

min = (RTC\_precteno[1] & 0b00001111); //minuty

des\_min = ((RTC\_precteno[1] >> 4) & 0b00001111); //desítky minut

hod = (RTC\_precteno[2] & 0b00001111); //hodiny

des\_hod = ((RTC\_precteno[2] >> 4) & 0b00000011); //desítky hodin

zbytek\_hod = ((RTC\_precteno[2] >> 4) & 0b00001111); //zbytek dat hodin

}

void GPIO\_Set(void){ //Inicializace pinů, nastavuju na HIGH, jelikož rozsvěcím nulou

GPIO\_Init(SEGMENT\_A\_PORT,SEGMENT\_A\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_B\_PORT,SEGMENT\_B\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_C\_PORT,SEGMENT\_C\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_D\_PORT,SEGMENT\_D\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_E\_PORT,SEGMENT\_E\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_F\_PORT,SEGMENT\_F\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(SEGMENT\_G\_PORT,SEGMENT\_G\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_1\_PORT,DIGIT\_1\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_2\_PORT,DIGIT\_2\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_3\_PORT,DIGIT\_3\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_4\_PORT,DIGIT\_4\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_5\_PORT,DIGIT\_5\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

GPIO\_Init(DIGIT\_6\_PORT,DIGIT\_6\_PIN,GPIO\_MODE\_OUT\_PP\_HIGH\_SLOW);

}

void init\_timer(void){

TIM3\_TimeBaseInit(TIM3\_PRESCALER\_16,1999); // clock 1MHz, strop 5000 => perioda přetečení 5 ms

TIM3\_ITConfig(TIM3\_IT\_UPDATE, ENABLE); // povolíme přerušení od update události (přetečení) timeru 3

TIM3\_Cmd(ENABLE); // spustíme timer 3

}

// pod tímto komentáoem nic nemiote

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval : None

\*/

void assert\_failed(u8\* file, u32 line)

{

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* Infinite loop \*/

while (1)

{

}

}

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/