MIKROPROCESOROVÁ TECHNIKA

Třída: A3	Úloha č. 11	Název: Krokový motor	
Jméno: Jakub Tenk	Datum zadání: 20. 5. 2021	Datum odevzdání: 16. 6. 2021	Známka:

Zadání:

Vytvořte program pro obsluhu krokového motorku. KM se bude řízen DIP spínačem:
1: otáčení jedním směrem/druhým směrem
2: rychlost - pomalu/rychle
3: počet otáček - 2/4

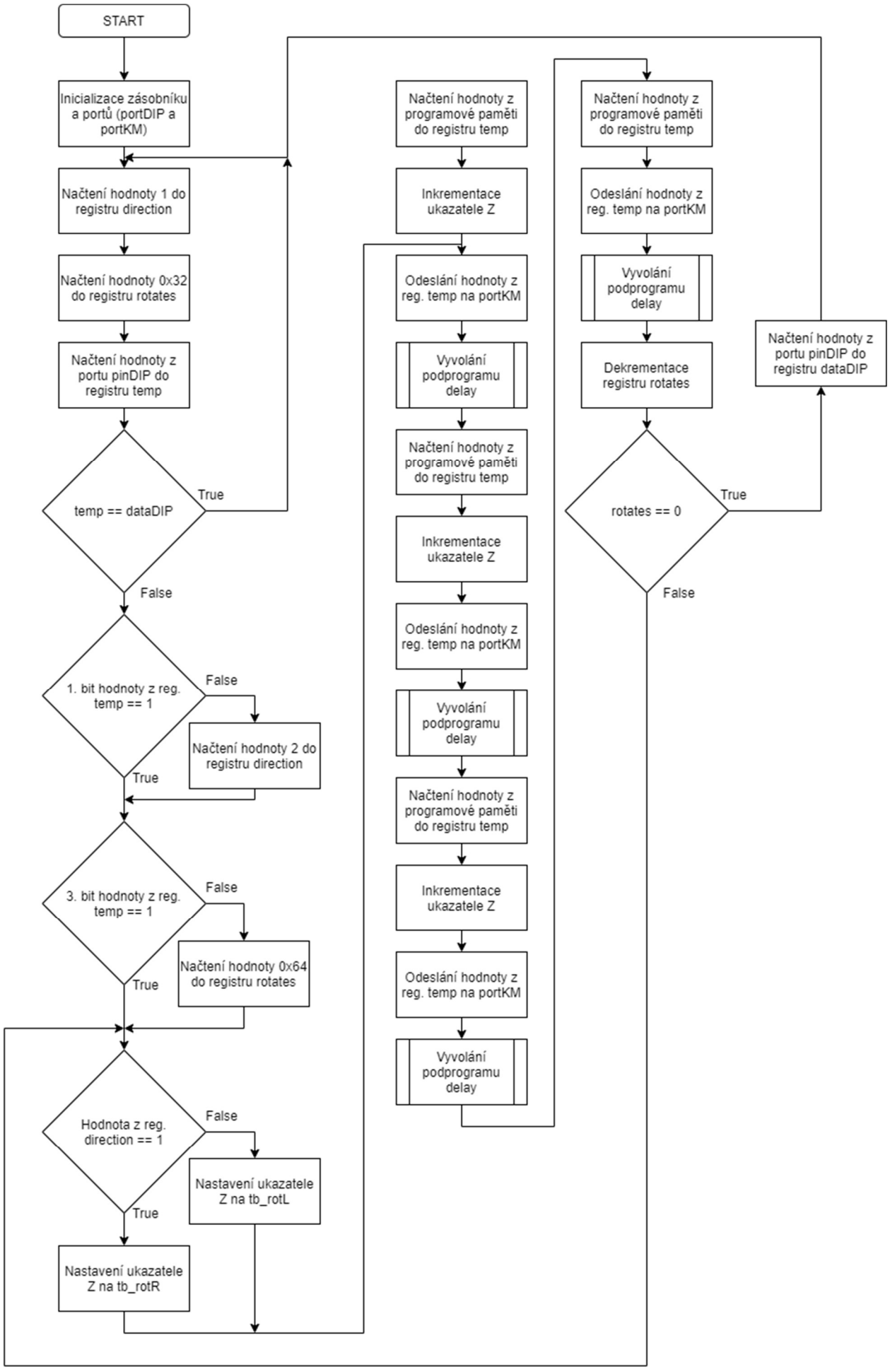
Postup (Hlavní program):

1. Při spuštění program se inicializuje zásobník a porty.
2. Program načte základní nastavení do registrů direction, rotates.
3. Program přečte hodnotu z portu pinDIP a uloží ji do registru temp.
4. Pokud hodnota temp je rovna s hodnotou dataDIP, tak program pokračuje bodem 2.
5. Pokud první bit hodnoty z registru temp bude roven log. 0, tak se načte do registru direction hodnota 2.
6. Pokud třetí bit hodnoty z registru temp bude roven log. 0, tak se načte do registru rotates hodnota 0x64.
7. Program skočí na návěští direct.
8. Pokud hodnota z registru direction je rovna 1, tak program pokračuje bodem 9.
9. Načtení programové paměti s daty pro otáčení motoru na pravou stranu, pokračuje bodem 10.
10. Načtení programové paměti s daty pro otáčení motoru na levou stranu, pokračuje bodem 10.
11. Načtení hodnoty z programové paměti s následnou inkrementací a odeslání jej na portKM,
12. Vyvolání podprogramu delay.
    (Bod 10. a 11. se opakuje 4krát za sebou)
13. Dekrementace registru rotates.
14. Pokud hodnota z registru rotates není rovna nule, tak pokračuje bodem 6.
15. Program přečte hodnotu z portu pinDIP a uloží ji do registru dataDIP
16. Skok na návěští start (na začátek programu).

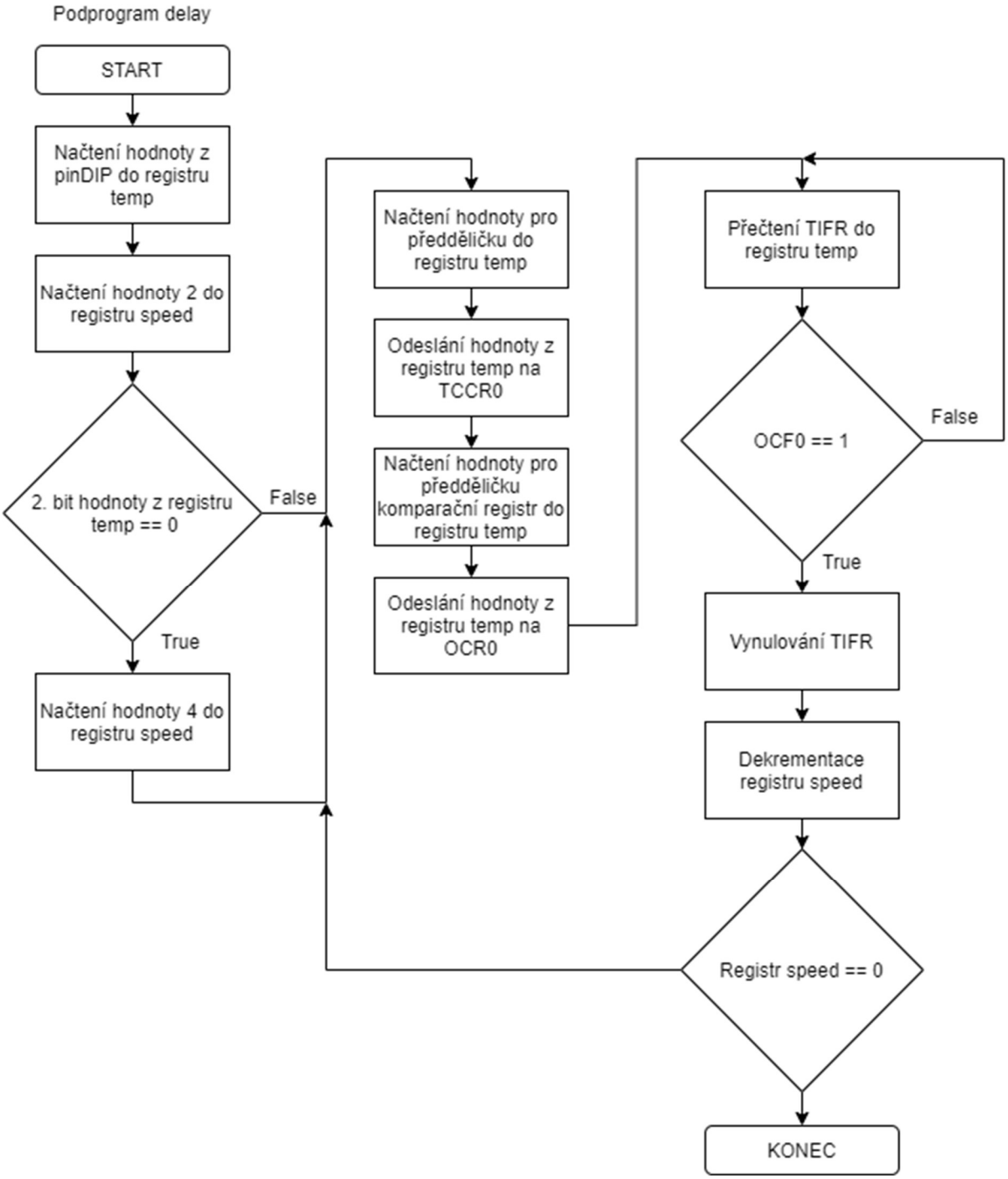
Postup (Podprogram delay):

1. Program přečte hodnotu z portu pinDIP a uloží ji do registru temp.
2. Načte hodnotu 2 do registru speed.
3. Pokud druhý bit hodnoty z registru temp je roven log. 0, tak se načte do registru speed hodnota 4.
4. Program nastaví před děličku a komparační hodnoty.
5. Program přečte hodnotu z TIFR a uloží ji do registru temp.
6. Pokud hodnota OCF0 není rovna 1, tak program pokračuje bodem 5.
7. Program vynuluje TIFR a dekrementuje registr speed.
8. Pokud hodnota z registru speed není rovna 0, tak program pokračuje bodem 4.
9. Program se vrátí do hlavního programu.

Vývojový diagram (hlavní program):



Vývojový diagram (podprogram delay):



Ukázka z kódu:

```
; >>                                  <<
; >>      Obsluha krokového motoru    <<
; >>        Vytvořil Jakub Tenk       <<
; >>         github.com/DrGumik       <<
; >>                                  <<

.nolist
.include "m128def.inc"
.list

.def temp = R16
.def rotL = R17
.def rotR = R18
.def poc_opak = R19
.def speed = R20
.def rotates = R21
.def direction = R22
.def dataDIP = R23
.equ portKM = portD     ; >> Port krokového motoru
.equ ddrKM = ddrD
.equ portDIP = portB
.equ ddrDIP = ddrB      ; >> Port DIP spínače
.equ pinDIP = pinB

    ldi temp, LOW(RAMEND)
    out SPL, temp
    ldi temp, HIGH(RAMEND)
    out SPH, temp

    ldi temp, 0x0F
    out ddrKM, temp

    ldi temp, 0x00
    out ddrDIP, temp

start:
    ldi direction, 1            ; >> Základní nastavení -> otáčení vlevo, rychlé otáčení, 2 otáčky
    ldi rotates, 0x32

    in temp, pinDIP             ; >> Přečtení portu s DIP spínačem

    cp temp, dataDIP
    breq start

    sbrs temp, 0                ; >> Změna směru, 1 = vlevo, 2 = vpravo
    ldi direction, 2
    sbrs temp, 2                ; >> Změna počtu otáček, 0x32 = 2 otáčky, 0x64 = 4 otáčky,
    ldi rotates, 0x64
    jmp direct

direct:                         ; >> Rozpoznání směru otáčení
    cpi direction, 1
    breq rotateL
    jmp rotateR

rotateL:                        ; >> Načtení programové paměti s daty pro otáčení na levou stranu
    ldi ZL, LOW(tb_rotL<<1)
    ldi ZH, HIGH(tb_rotL<<1)
    jmp process
rotateR:                        ; >> Načtení programové paměti s daty pro otáčení na pravou stranu
    ldi ZL, LOW(tb_rotR<<1)
    ldi ZH, HIGH(tb_rotR<<1)
    jmp process

process:                        ; >> Proces otáčení motoru
    lpm temp, Z+
    out portKM, temp
    call delay

    lpm temp, Z+
    out portKM, temp
    call delay

    lpm temp, Z+
    out portKM, temp
    call delay

    lpm temp, Z
    out portKM, temp
    call delay

    dec rotates
    brne direct
    in dataDIP, pinDIP
    jmp start
```

```
delay:                          ; >> Podprogram delay – dvě rychlosti (řídí se čísly 2 a 4)
    in temp, pinDIP             ; >> Přečtení portu s DIP spínačem
    ldi speed, 4
    sbrs temp, 1                ; >> Změna rychlosti, 4 = rychle, 6 = pomalu
    ldi speed, 6
set_timer:
    ldi temp, 0b0000_1101
    out TCCR0, temp
    ldi temp, 255
    out OCR0, temp
compare:
    in temp, TIFR
    sbrs temp, 1
    jmp compare
    out TIFR, temp
    dec speed
    brne set_timer
    ret

tb_rotL:    ; >> Data pro otáčení motoru na levou stranu
.db 0b0000_0001, 0b0000_0100, 0b0000_0010, 0b0000_1000

tb_rotR:    ; >> Data pro otáčení motoru na pravou stranu
.db 0b0000_1000, 0b0000_0010, 0b0000_0100, 0b0000_0001
```

Závěr:

Tato úloha z mého hlediska nebyla zas tak těžká, jen byl problém, že jsem program testoval jen jednou, protože jsme se museli u motoru prostřídat. Po rozšíření zadání jsme během praxí šli s Martinem Štěchem do školy otestovat znovu program, ale bohužel krokové motory byly rozbité. Program tedy není na sto procent otestovaný, ale měl by být funkční podle zadání.