Dílenská praxe

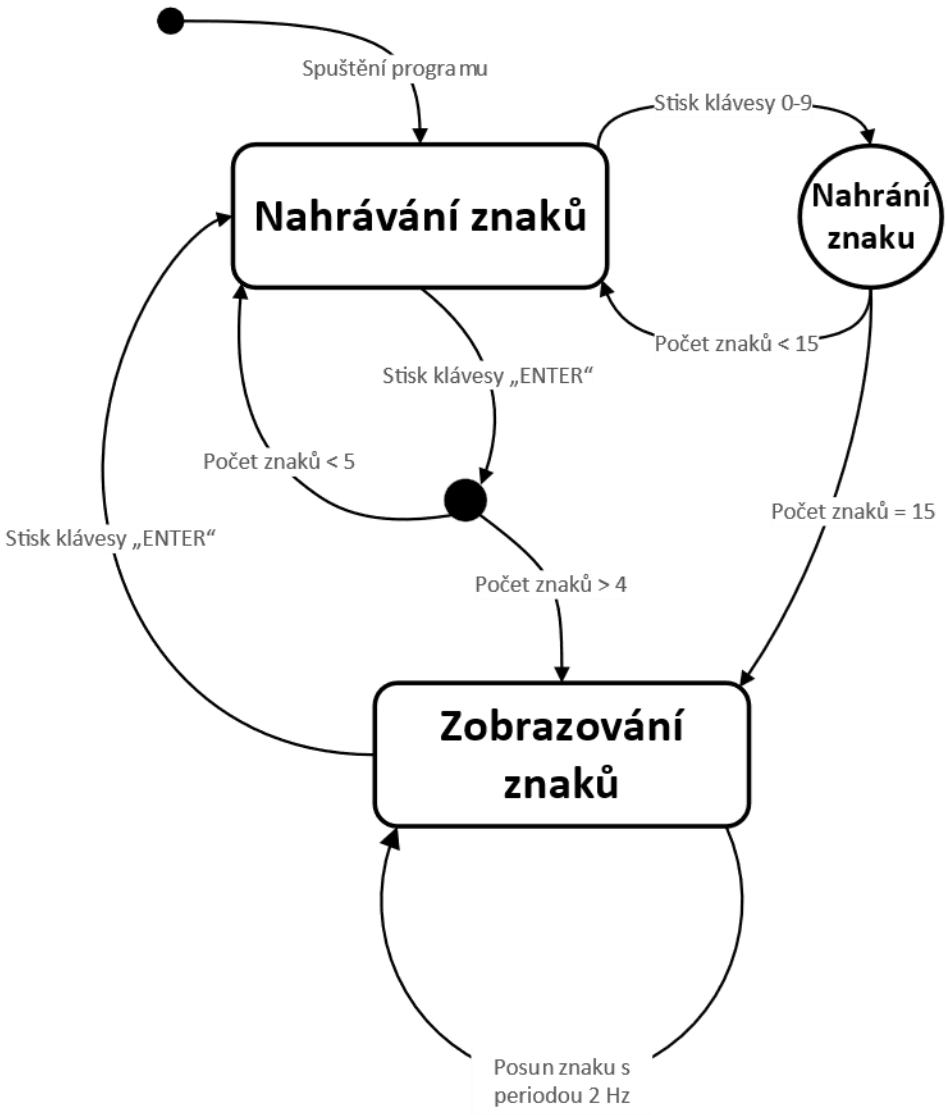
A4	2. Maticový displej		
Petřík Vít		1/12	Známka:
2.10. 2019	Datum odevzdání:	6.11. 2019	Odevzdáno:

Zadání:

Zpracujte program v programovacím jazyce JSA ATMega128 ovládající určený připojený maticový displej a klávesnici tak, aby obsahoval nejméně tyto funkce:

1) stisknuté tlačítko klávesnice se uloží do paměti modulu MB-ATmega128, minimálně 5
2) kláves, maximálně 15 kláves
3) každé klávese bude přiřazen vhodný zobrazovaný symbol
4) rozpoznání stavu vkládání znaků a stavu přehrávání vložených znaků ovládaných pomocí
5) maticové klávesnice klávesnice
6) přepínání mezi těmito režimy
7) využití všech vhodných HW možností přípravku MB-ATmega128.

Vývojový diagram:



Výpis programu:

Příloha 1 – assembler

Závěr:

Řešení úlohy funguje dle očekávání. Kdyby měl být program psán v jazyku C tak obsluhu klávesnice vyřeším algoritmem, ale v assembleru je to moc práce navíc a tak jsem upřednostnil rychlejší vyhotovení kódu před kvalitním zpracováním.

Přílohy:

- Příloha 1 – 9 stran

Příloha 1

```
/**********definice**********/

.nolist
.Include "m128def.inc"
.list

//definice prescaleru
    .EQU timer0_prescaler_noclock = 0
    .EQU timer0_prescaler_1 = 1
    .EQU timer0_prescaler_8 = 2
    .EQU timer0_prescaler_32 = 3
    .EQU timer0_prescaler_64 = 4
    .EQU timer0_prescaler_128 = 5
    .EQU timer0_prescaler_256 = 6
    .EQU timer0_prescaler_1024 = 7

    .org 0x0000
    rjmp start

    .org 0x0018
    jmp timer1a_ctc

    .org 0x001A
    jmp timer1b_ctc

    .org 0x001E
    jmp timer0_ctc

/**********konec definic**********/

/**********Datove struktury**********/

.CSEG
matrixArray:
    .DB 0, 0b01000001, 0b00101110, 0b00110110, 0b00111010, 0b01000001, 0, 0, 0, 0
    .DB 0, 0b01111111, 0b00111101, 0b00000000, 0b00111111, 0b01111111, 0, 0, 0, 0
    .DB 0, 0b00111110, 0b00011110, 0b00101110, 0b00110110, 0b00111001, 0, 0, 0, 0
    .DB 0, 0b00111110, 0b00110110, 0b00110110, 0b00110110, 0b01001001, 0, 0, 0, 0
    .DB 0, 0b01110000, 0b01110111, 0b01110111, 0b01110111, 0b01111111, 0, 0, 0, 0
    .DB 0, 0b00110000, 0b00110110, 0b00110110, 0b00110110, 0b01001110, 0, 0, 0, 0
    .DB 0, 0b01000001, 0b00110110, 0b00110110, 0b00110110, 0b01001111, 0, 0, 0, 0
    .DB 0, 0b01111110, 0b01111110, 0b00001110, 0b01110110, 0b01111000, 0, 0, 0, 0
    .DB 0, 0b01001001, 0b00110110, 0b00110110, 0b00110110, 0b01001001, 0, 0, 0, 0
    .DB 0, 0b01011001, 0b00110110, 0b00110110, 0b00110110, 0b01000001, 0, 0, 0, 0
    .DB 0, 0b00011100, 0b01101011, 0b01110111, 0b01101011, 0b00011100, 0, 0, 0, 0

.DSEG
input:
    .BYTE 15

inputIndex:
    .BYTE 1

playIndex:
```

```
    .BYTE 1

display:
    .BYTE 1

displayIndex:
    .BYTE 1

lastKey:
    .BYTE 1

stav:
    .BYTE 1

/***********************************/

/**********Startovaci sekvence**********/

.CSEG
start:
// nastaveni zasobniku
    ldi r16, low(RAMEND)
    out spl, R16
    ldi r16, high(RAMEND)
    out sph, R16

    ldi r16, 0b0000_1111
    out ddrb, r16

//inicializace a reset displeje
    ldi r16, 0xFF
    out ddre, r16
    ldi r16, 0b0111_1111
    out portE, r16
    cbi portE, 7

    call reset_delay

    sbi portE, 7

    call reset_delay

    cbi portE, 7

//vynulovani promennych
    ldi ZL, low(inputIndex*2)
    ldi ZH, high(inputIndex*2)
    ldi r16, 0
    st Z, r16

    ldi ZL, low(displayIndex*2)
    ldi ZH, high(displayIndex*2)
    ldi r16, 0
    st Z, r16

    ldi ZL, low(playIndex*2)
```

```
    ldi ZH, high(playIndex*2)
    ldi r16, 0
    st Z, r16

    ldi ZL, low(display*2)
    ldi ZH, high(display*2)
    ldi r16, 10
    st Z, r16

    ldi ZL, low(lastKey*2)
    ldi ZH, high(lastKey*2)
    ldi r16, 10
    st Z, r16

    ldi ZL, low(stav*2)
    ldi ZH, high(stav*2)
    ldi r16, 0
    st Z, r16

//nastaveni timeru
    ldi r16, (timer0_prescaler_64<<CS00)
    ldi r17, 255
    call timer0_Init

    call timer1_Init

    clr r15

// zapnuti interruptu
    sei

cerna_dira:
    rjmp cerna_dira

/**********Konec startovaci sekvence**********/


/**********Inicializace Timer0**********/

//r16 - precaler, r17 - OCR
timer0_Init:
    ldi r18, (1<<WGM01)
    or r18, r16
    out TCCR0, r18
    out OCR0, r17
    in r17, TIMSK
    ldi r16, (1<<OCIE0)
    or r16, r17
    out TIMSK, r16
ret

/**********Konec inicializace Timer0**********/

/**********Inicializace Timer1**********/

timer1_Init:
```

```
    ldi r16, high(100)
    out OCR1AH, r16
    ldi r16, low(100)
    out OCR1AL, r16
    ldi r16, high(14400)
    out OCR1BH, r16
    ldi r16, low(14400)
    out OCR1BL, r16
    ldi r16, (1<<WGM12)|(5<<CS10)
    out TCCR1B, r16
    in r16, TIMSK
    ori r16, (1<<OCIE1A)|(0<<OCIE1B)
    out TIMSK, r16
ret

/**********Konec inicializace Timer1**********/

/**********Zobrazovani na displeji**********/

timer0_ctc:
//Uchovani registru ve stacku
    push r16
    push r17
    push r18
//posouvaci impuls
    ldi r16, 0xFF
    out portE, r16
//vyzvednuti zobrazovneho znaku
    ldi ZL, low(display*2)
    ldi ZH, high(display*2)
    ld r16, Z
//vynasobime deseti abychom se dostali na spravny prvek pole
    ldi r17, 10
    mul r16, r17
    mov r16, r0
    ldi ZL, low(displayIndex*2)
    ldi ZH, high(displayIndex*2)
//pokud jsme v displeji dame nizsi frekvenci
    ld r17, Z
    cpi r17, 1
    brne doNotSetSlowerFrequency
    ldi r18, 255
    out OCR0, r18
    rjmp exitFromFrequencyChange
doNotSetSlowerFrequency:
//pokud jsme mimo displej nahodime na timer0 vesmirnou frekvenci
    cpi r17, 6
    brne exitFromFrequencyChange
    ldi r18, 1
    out OCR0, r18
exitFromFrequencyChange:
    add r16, r17
//pretekli jsem displej?
    inc r17
    cpi r17, 10
    brlo noOverflow
```

```
    ldi r17, 0
noOverflow:
    st Z, r17
    clr r17
//vyzvedneme adresu
    ldi ZL, low(matrixArray*2)
    ldi ZH, high(matrixArray*2)
//pricteme k index
    add ZL, r16
    adc ZH, r17
    lpm r16, Z
//hodime to na displej
    out portE, r16
//vratime hodnoty ze stacku do registru
    pop r18
    pop r17
    pop r16
reti

/**********Konec zobrazovani na displeji**********/

/**********Obsluha klavesnice + hlavni program**********/

timer1a_ctc:
    push r16
    push r17
    push r18
//nahrani posledniho stavu klavesnice
    ldi ZL, low(lastKey*2)
    ldi ZH, high(lastKey*2)
    ld r18, Z
    push r16
    ldi r17, 11

//1. radek
    ldi r16, 0b1111_1110
    out portB, r16
    call key_delay
    sbis pinB, 4
    ldi r17, 1
    sbis pinB, 5
    ldi r17, 2
    sbis pinB, 6
    ldi r17, 3

//2. radek
    ldi r16, 0b1111_1101
    out portB, r16
    call key_delay
    sbis pinB, 4
    ldi r17, 4
    sbis pinB, 5
    ldi r17, 5
    sbis pinB, 6
    ldi r17, 6
```

```
//3. radek
    ldi r16, 0b1111_1011
    out portB, r16
    call key_delay
    sbis pinB, 4
    ldi r17, 7
    sbis pinB, 5
    ldi r17, 8
    sbis pinB, 6
    ldi r17, 9

//4. radek
    ldi r16, 0b1111_0111
    out portB, r16
    call key_delay
    sbis pinB, 5
    ldi r17, 0
    sbis pinB, 7
    ldi r17, 10

//detekce "hrany"
    cp r17, r18
    brne hrana
    rjmp return
hrana:
    st Z, r17
    cpi r17, 11
    brne stisk
    rjmp return
stisk:
//nahrani stavu
    ldi ZL, low(stav*2)
    ldi ZH, high(stav*2)
    ld r16, Z
stav0:
    cpi r16, 0
    brne stav1
    cpi r17, 10
    breq hotkey0
//pokud je stav 0 a neni stisknuto hotkey
//nahrajeme klavesu do pameti
    clr r18
    ldi YL, low(input*2)
    ldi YH, high(input*2)
    ldi ZL, low(inputIndex*2)
    ldi ZH, high(inputIndex*2)
    ld r16, Z
    add YL, r16
    adc YH, r18
    st Y, r17
    inc r16
//pokud je v pameti 15 znaku vynutime rezim zobrazovani
    cpi r16, 15
    breq hotkey0
    st Z, r16
    rjmp return
```

```
hotkey0:
    ldi ZL, low(inputIndex*2)
    ldi ZH, high(inputIndex*2)
    ld r16, Z
//zkontrolujeme, zda je opravdu nahrnao 5 znaku a vice
    cpi r16, 5
    brlo return
//povolime prehravaci interrupt
    in r16, TIMSK
    ori r16, (1<<OCIE1B)
    out TIMSK, r16
//vyresetujeme promennou
    ldi ZL, low(playIndex*2)
    ldi ZH, high(playIndex*2)
    ldi r16, 0
    st Z, r16
//nahrajeme novy stav
    ldi ZL, low(stav*2)
    ldi ZH, high(stav*2)
    ldi r16, 1
    st Z, r16
    rjmp return
stav1:
    cpi r16, 1
    brne return
//zkontrolujeme, zda je stisknut hotkey
    cpi r17, 10
    breq hotkey1
    rjmp return
hotkey1:
//vyresetujeme promenne a zakazeme interrupt
    in r16, TIMSK
    andi r16, ~(1<<OCIE1B)
    out TIMSK, r16
    ldi ZL, low(display*2)
    ldi ZH, high(display*2)
    ldi r16, 10
    st Z, r16
    ldi ZL, low(inputIndex*2)
    ldi ZH, high(inputIndex*2)
    ldi r16, 0
    st Z, r16
//nahrajeme novy stav
    ldi ZL, low(stav*2)
    ldi ZH, high(stav*2)
    st Z, r16
    rjmp return
return:
    pop r18
    pop r17
    pop r16
reti

/**********Konec obluhy klavesnice a hlavniho programu**********/

/**********Rezim zobrazovni**********/
```

```
timer1b_ctc:
    push r16
    push r17
    push r18
//nacteme adresu vstupniho rezetce
//a indexu
    ldi YL, low(input*2)
    ldi YH, high(input*2)
    ldi ZL, low(playIndex*2)
    ldi ZH, high(playIndex*2)
//pricteme hodnotu indexu k vstupnimu retezci
    ld r16, Z
    add YL, r16
    adc YH, r18
    ld r17, Y
//nahrani znaku na displej
    ldi YL, low(display*2)
    ldi YH, high(display*2)
    st Y, r17
//porovnani, zdali uz jsme na konci vstupniho retezcu
//pokud jsme, tak vynulujeme index
    ldi YL, low(inputIndex*2)
    ldi YH, high(inputIndex*2)
    ld r17, Y
    inc r16
    cp r16, r17
    brlo noOverflow3
    ldi r16, 0
noOverflow3:
//index nahrajeme zpatky do adresy
    st Z, r16
    inc r16
    pop r18
    pop r17
    pop r16
reti

/**********Konec rezimu zobrazovni**********/

/**********reset delay**********/

reset_delay:
    push r18
    push r19
    ldi  r18, 50
    ldi  r19, 199
L1: dec  r19
    brne L1
    dec  r18
    brne L1
    pop r19
    pop r18
ret

/**********Konec reset delay**********/
```

```
/**********keyboard delay**********/

key_delay:
    push r18
    ldi  r18, 5
L2: dec  r18
    brne L2
    pop r18
ret

/**********Konec keyboard delay**********/
```