Dílenská praxe

A4	5. Robot Nisa 600		
Petřík Vít		1/9	Známka:
18. 12. 2019	Datum odevzdání:	29. 1. 2020	Odevzdáno:

Zadání:

Zpracujte program v programovacím jazyce C ovládající robotickou ruku tak, aby obsahoval nejméně tyto funkce:

1) ovládání pohybu jednotlivých pohybových os robota pomocí zvolených kláves klávesnice počítače
2) hlídání mezních poloh pohybu robota (a to jak s využitím HW senzorů, tak i SW)
3) sledování chybových stavů
4) vhodná indikace stavu a polohy robotické ruky na monitoru počítače

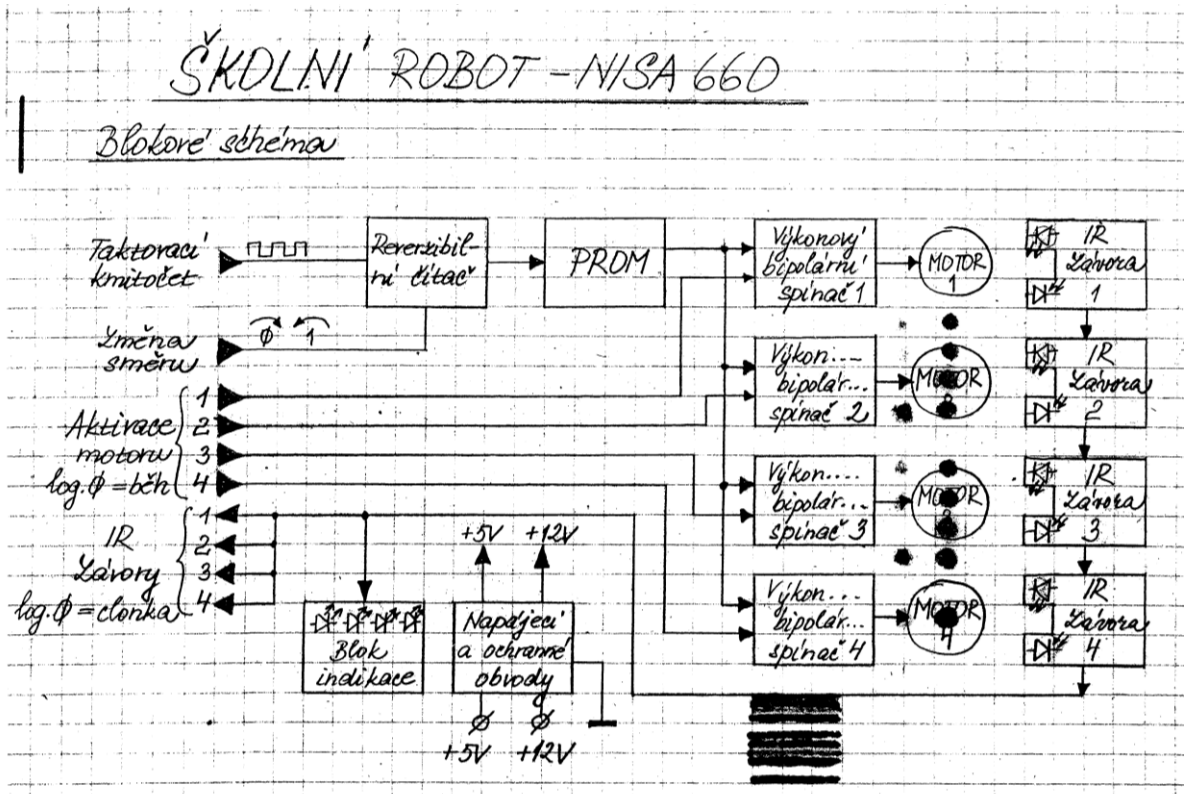
Postup:

- Nadefinování datové struktury.
- Zapsání aktuátorů do datové struktury.
- Vytvoření stavového automatu s 2 stavy.
  - Uvedení ruky do výchozí pozice.
  - Ovládání ruky pomocí klávesnice.
- Zapsání kontroly klávesnice a následnou akci zapnutí pohonu.

Propojení PC a Mikrovlnné trouby:

PORT 2 - output		PORT 3 – input optické závory	
Pin	Význam	Pin	Význam
0	Takt	0	Otáčení základny
1	Směr	1	Hlavní rameno
2	Otáčení základny	2	Rameno chapadla
3	Hlavní rameno	3	Chapadlo
4	Rameno chapadla	4	-
5	Chapadlo	5	-
6	-	6	-
7	-	7	-

Schéma zapojení:

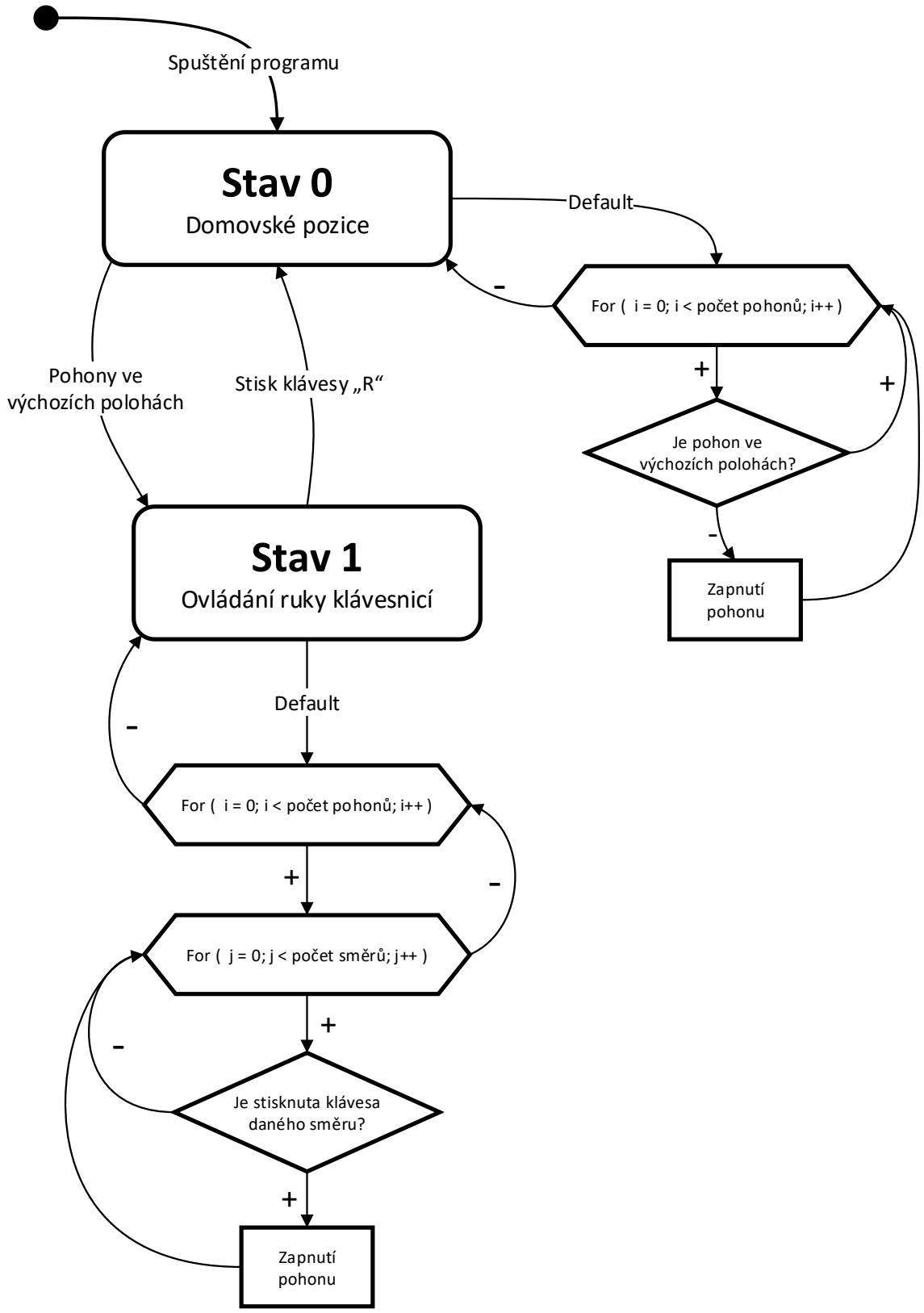


Základní technické údaje

Napájení: +5V/0,5A pro logiku a +12V/1,5A pro motory
Vstupy a výstupy: všechny v úrovních logiky TTL
Maximální taktovací kmitočet: 450Hz, střída 1:1, (určuje rychlost otáčení)
Motor je aktivní při úrovni log. Ø na aktivačním vstupu
(Motorů může současně běžet více, avšak všechny jedním směrem)
Výstupy IR závor mají úroveň log. Ø a indikační LED na panelu svítí, pokud clonka prochází závorou
(Všechny závory zaclněny = výchozí stav)

Směr otáčení (výpod č.2)	log.1	log.Ø
Otáčení horizontální	po směru hod. ruč.	proti směru hod. ruč.
Hlavní rameno	dolů	nahoru
Chapadlo	zavřít	otevřít
Rameno s chapadlem	nahoru	dolů

Vývojový diagram:



Výpis programu:

Příloha 1 – C++

Závěr:

Program funguje tak jak má a je spolehlivý. V programu jsem využil poměrně nezvyklé datové konstrukce s nadefinovaným polem structů. Tato konstrukce mi umožnila vytvořit jeden for cyklus který funguje na všechny pohony robotické ruky.

Přílohy:

- Příloha 1 – 4 strany

Příloha 1

```
/*Zapojeni
Port P3 - input
0 bit       1 bit       2 bit           3 bit
zakladna    Hl.rameno   rameno chapadla chapadlo
Port P2 - output
0 bit   1 bit           2 bit       3 bit       4 bit               5 bit
takt    smer otaceni    zakladna    Hl.rameno   rameno  chapadla    chapadlo*/

#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <dos.h>

//pohon, ktery nema koncak
#define NOLIMIT -1

//porty
#define P1 0x300
#define P2 0x301
#define P3 0x300
#define P4 0x301

#define CLOCK_BIT 0
#define DIRECTION_BIT 1

//sipky na klávesnici
#define UP 72
#define DOWN 80
#define LEFT 75
#define RIGHT 77

//trida citace, je vyuzit pro generovani taktu
class counter
{
private:
    int count, resolution, prescaler, prescalerCount;

public:
    //nastaveni citace
    //resolution, prescaler
    void begin(int a, int b)
    {
        resolution = a;
        prescaler = b;
        count = 0;
        prescalerCount = 0;
    }

    //cyklus citace
    void run()
    {
        if (prescalerCount >= prescaler)
        {
            count++;
            prescalerCount = 0;
        }
        if (count + 1 > resolution)
        {
            count = 0;
        }
```

```
        prescalerCount++;
    }

    //vrati hodnotu citace
    int getCount()
    {
        return count;
    }
};

//definice structu pro smery pohonu
typedef struct
{
    char name[50];
    char limit;
    char key;
} direction;

//definice structu pro jeden pohon
typedef struct
{
    char name[50];
    direction directions[2];
    char bit;
} actuator;

//nadefinovani vsech pohonu do jednoho pole structu "actuator"
actuator actuators[4] = {
    {"Otaceni zakladny", {{"Proti hodinovym rucickam", NOLIMIT, 'a'}, {"hodinovych rucicek"
, NOLIMIT, 'd'}}, 2},
    {"Hlavni rameno", {{"nahoru", 1, 's'}, {"dolu", NOLIMIT, 'w'}}, 3},
    {"Rameno chapadla", {{"dolu", NOLIMIT, UP}, {"nahoru", 2, DOWN}}, 4},
    {"Chapadlo", {{"Otevrit", 3, LEFT}, {"zavrit", NOLIMIT, RIGHT}}, 5}};

//vrati pocet milisekund od startu programu
unsigned long millis()
{
    return (clock() * 1000) / CLOCKS_PER_SEC;
}

int main(void)
{
    //inicializace proměnných
    char pressedKey;
    char state = 1;
    char in;
    char out;
    unsigned long keyMillis = 0;
    counter clock;
    char kbhitChange = 0;
    clock.begin(2, 150);    //nastaveni citace 2-pocet poloh; 150-preddelicka
    while (1)
    {
        //zjisteni stisknute klavesy
        if (kbhit())
        {
            pressedKey = getch();
            kbhitChange = 1;
        }
        else
        {
            kbhitChange = 0;
```

```
            if ((millis() - keyMillis) > 100)
                pressedKey = 0;
        }
        out = 0xFF;
        //precteni vstupniho portu
        in = inport(P3);
        char i;
        //stavovy diagram
        switch (state)
        {
            //stav poslani pohonu do vychozich poloh
        case 0:
            char enable = -1;
            //projeti vsech pohonu
            for (i = 0; i < sizeof(actuators) / sizeof(actuator); i++)
            {
                //ve vsech smerech
                for (char j = 0; j < 2; j++)
                {
                    //pokud ma pohon a dany smer nadefinovany koncovy spinac a spinac je za
clonen
                    if (actuators[i].directions[j].limit != NOLIMIT && (in & (1 << actuator
s[i].directions[j].limit)))
                    {
                        //pokud neni zadny jiny pohon aktivovan
                        if (enable == -1)
                        {
                            out &= ~(!j << DIRECTION_BIT);
                            out &= ~(1 << actuators[i].bit);
                            enable = j;
                        }
                        //pokud je smer uz aktivovaneho pohon stejny
                        //jako smer aktualniho pohonu muzeme ho zapnout
                        else if (enable == j)
                        {
                            out &= ~(1 << actuators[i].bit);
                        }
                    }
                }
            }
            //pokud jsme nespustili zadny pohon, znamena to, ze jsme vychozich polohach
            //a muzeme stav 0 opustit a prejit do ovladani ramena klavesnici
            if (enable == -1)
                state = 1;
            break;
        case 1:
            //klavesou 'r' muzeme spustit stav presunu do vychozich poloh
            if (pressedKey == 'r')
            {
                state = 0;
            }
            //projeti vsech pohonu
            for (i = 0; i < sizeof(actuators) / sizeof(actuator); i++)
            {
                //ve vsech smerech
                for (char j = 0; j < 2; j++)
                {
                    //pokud je zmacknuta klavesa, ktera je prirazena pohonu
                    if (actuators[i].directions[j].key == pressedKey)
                    {
                        //zobrazeni informace na displeji
                        if (kbhitChange)
```

```
                        {
                            if ((millis() - keyMillis) > 500)
                            {
                                printf("Kapitane mame dotek na pohonu: %s\n\r", actuators[i
].name);
                                printf("ve smeru: %s\n\r", actuators[i].directions[j].name)
;
                            }
                            keyMillis = millis();
                        }
                        //zjistime zda jsme na koncovem spinaci
                        if (actuators[i].directions[j].limit != NOLIMIT && !(in & (1 << act
uators[i].directions[j].limit)))
                        {
                            printf("Jsme na limetu\n\r");
                        }
                        //pokud ne, pohon zapneme
                        else
                        {
                            out &= ~(!j << DIRECTION_BIT);
                            out &= ~(1 << actuators[i].bit);
                        }
                        //Hihi, goto :-)
                        goto breakOutOfLoops;
                    }
                }
            }
        breakOutOfLoops:
            break;
        }
        //pridame do vystupniho bytu takt
        out &= ~(clock.getCount() << CLOCK_BIT);
        //hodime byte na port a aktualizujeme clock
        outportb(P2, out);
        clock.run();
        if (pressedKey == 27)
        {
            break;
        }
    }
    //konec programu a navrat do Borlanda
    while (!kbhit())
        ;
    return 0;
}
```