Dílenská praxe

A4	3. Mikrovlnná trouba		
Tenk Jakub		1/16	Známka:
3. 11. 2021	Datum odevzdání:	1. 12. 2021	Odevzdáno:

Zadání:

Zpracujte ovládací program v programovacím jazyce C ovládající model mikrovlnné trouby tak, aby obsahoval nejméně tyto funkce:

1) simulujte provoz velmi jednoduché mikrovlnné trouby
2) na vestavěném displeji modelu zobrazuje jak dobu ohřevu, tak i teplotu „pokrmu“
3) na vestavěné klávesnici modelu umožní nastavit jak dobu ohřevu, tak i požadovanou teplotu.
4) na vestavěné klávesnici modelu umožní nastavit pracovní otáčky talíře
5) při „ohřevu“ průběžně zobrazuje metodou „countdown“ zbývající dobu ohřevu
6) sleduje a zobrazuje provozní a chybové stavy přípravku na monitoru PC

Propojení PC a Mikrovlnné trouby:

0x300 P1 (OUT)		**0x301 P2 (OUT)**		**0x300 P3 (IN)**	
Číslo pinu	**Číslo bitu**	**Číslo pinu**	**Číslo bitu**	**Číslo pinu**	**Číslo bitu**
8	0	1	0	12	0
9	1	2	1	7	1
10	2	3	2	14	2
11	3	4	3		3
20	4	5	4		4
19	5		5		5
16	6		6		6
15	7		7		7

Vývojový diagram:

Příloha 1 – vývojový diagram

Výpis programu:

Příloha 2 – výpis programu

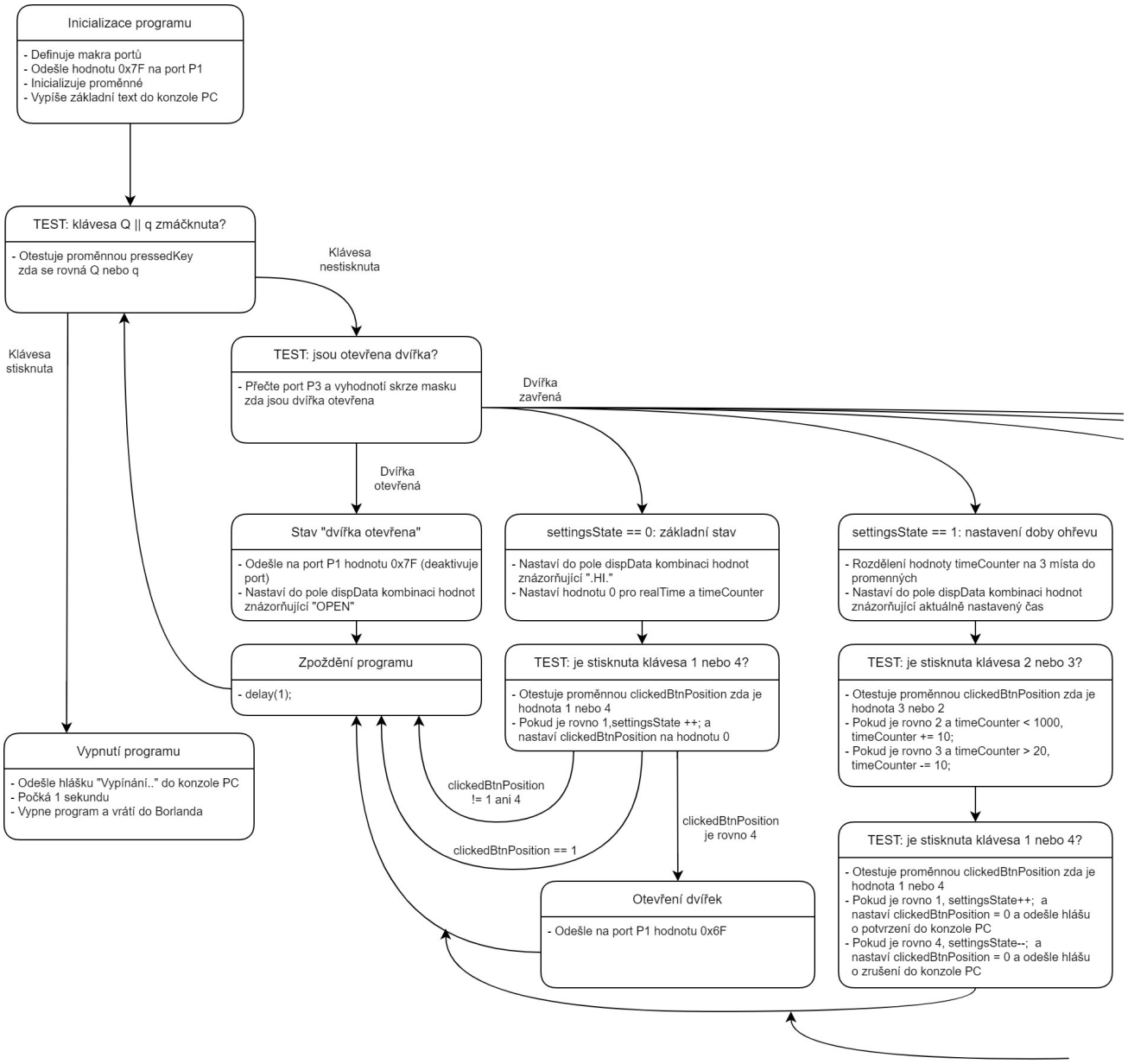
Závěr:

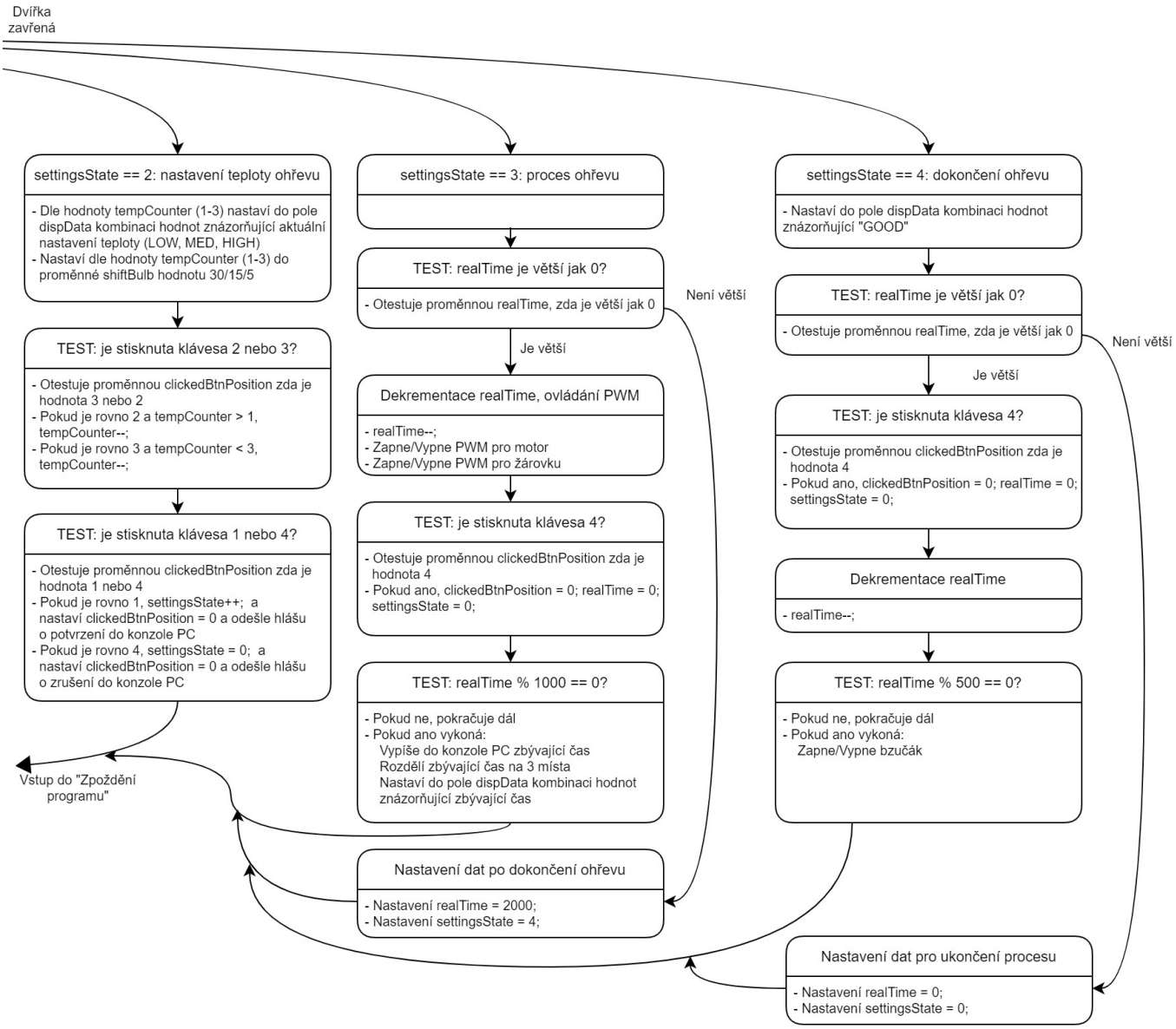
Ze začátku jsem měl problém zkompilovat hotový kód, protože kompilátor neznal nové způsoby cyklů for(), které fungují na nové verzi jazyka C. To byl trošku problém, ale ne takový, že by mi to zabránilo mikrovlnku zprovoznit. Přepsal jsem kód, předělal jsem část algoritmu a otestoval. Po otestování jsem měl sestavený funkční kód pro mikrovlnnou troubu, dle zadání.

Přílohy:

- Příloha 1 – 2 strany
- Příloha 2 – 12 strana

Příloha 1:


Inicializace programu
- Definuje makra portů
- Odešle hodnotu 0x7F na port P1
- Inicializuje proměnné
- Vypíše základní text do konzole PC
TEST: klávesa Q || q zmáčknuta?
- Otestuje proměnnou pressedKey zda se rovná Q nebo q
Klávesa nestisknuta
Klávesa stisknuta
TEST: jsou otevřena dvířka?
- Přečte port P3 a vyhodnotí skrze masku zda jsou dvířka otevřena
Dvířka zavřená
Dvířka otevřená
Stav "dvířka otevřena"
- Odešle na port P1 hodnotu 0x7F (deaktivuje port)
- Nastaví do pole dispData kombinaci hodnot znázorňující "OPEN"
settingsState == 0: základní stav
- Nastaví do pole dispData kombinaci hodnot znázorňující ".HI."
- Nastaví hodnotu 0 pro realTime a timeCounter
settingsState == 1: nastavení doby ohřevu
- Rozdělení hodnoty timeCounter na 3 místa do promenných
- Nastaví do pole dispData kombinaci hodnot znázorňující aktuálně nastavený čas
Zpoždění programu
- delay(1);
TEST: je stisknuta klávesa 1 nebo 4?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 1 nebo 4
- Pokud je rovno 1,settingsState ++; a nastaví clickedBtnPosition na hodnotu 0
TEST: je stisknuta klávesa 2 nebo 3?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 3 nebo 2
- Pokud je rovno 2 a timeCounter < 1000, timeCounter += 10;
- Pokud je rovno 3 a timeCounter > 20, timeCounter -= 10;
Vypnutí programu
- Odešle hlášku "Vypínání.." do konzole PC
- Počká 1 sekundu
- Vypne program a vrátí do Borlanda
clickedBtnPosition != 1 ani 4
clickedBtnPosition == 1
clickedBtnPosition je rovno 4
TEST: je stisknuta klávesa 1 nebo 4?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 1 nebo 4
- Pokud je rovno 1, settingsState++; a nastaví clickedBtnPosition = 0 a odešle hlášu o potvrzení do konzole PC
- Pokud je rovno 4, settingsState--; a nastaví clickedBtnPosition = 0 a odešle hlášu o zrušení do konzole PC
Otevření dvířek
- Odešle na port P1 hodnotu 0x6F


Dvířka zavřená
settingsState == 2: nastavení teploty ohřevu
- Dle hodnoty tempCounter (1-3) nastaví do pole dispData kombinaci hodnot znázorňující aktuální nastavení teploty (LOW, MED, HIGH)
- Nastaví dle hodnoty tempCounter (1-3) do proměnné shiftBulb hodnotu 30/15/5
TEST: je stisknuta klávesa 2 nebo 3?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 3 nebo 2
- Pokud je rovno 2 a tempCounter > 1, tempCounter--;
- Pokud je rovno 3 a tempCounter < 3, tempCounter--;
TEST: je stisknuta klávesa 1 nebo 4?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 1 nebo 4
- Pokud je rovno 1, settingsState++; a nastaví clickedBtnPosition = 0 a odešle hlášu o potvrzení do konzole PC
- Pokud je rovno 4, settingsState = 0; a nastaví clickedBtnPosition = 0 a odešle hlášu o zrušení do konzole PC
Vstup do "Zpoždění programu"
settingsState == 3: proces ohřevu
TEST: realTime je větší jak 0?
- Otestuje proměnnou realTime, zda je větší jak 0
Není větší
Je větší
Dekrementace realTime, ovládání PWM
- realTime--;
- Zapne/Vypne PWM pro motor
- Zapne/Vypne PWM pro žárovku
TEST: je stisknuta klávesa 4?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 4
- Pokud ano, clickedBtnPosition = 0; realTime = 0; settingsState = 0;
TEST: realTime % 1000 == 0?
- Pokud ne, pokračuje dál
- Pokud ano vykoná:
Vypíše do konzole PC zbývající čas
Rozdělí zbývající čas na 3 místa
Nastaví do pole dispData kombinaci hodnot znázorňující zbývající čas
Nastavení dat po dokončení ohřevu
- Nastavení realTime = 2000;
- Nastavení settingsState = 4;
settingsState == 4: dokončení ohřevu
- Nastaví do pole dispData kombinaci hodnot znázorňující "GOOD"
TEST: realTime je větší jak 0?
- Otestuje proměnnou realTime, zda je větší jak 0
Není větší
Je větší
TEST: je stisknuta klávesa 4?
- Otestuje proměnnou clickedBtnPosition zda je hodnota 4
- Pokud ano, clickedBtnPosition = 0; realTime = 0; settingsState = 0;
Dekrementace realTime
- realTime--;
TEST: realTime % 500 == 0?
- Pokud ne, pokračuje dál
- Pokud ano vykoná:
Zapne/Vypne bzučák
Nastavení dat pro ukončení procesu
- Nastavení realTime = 0;
- Nastavení settingsState = 0;

Příloha 2:

```
/*
    clickedBtnPosition = 1  ==> MODE
    clickedBtnPosition = 2  ==> sipka nahoru
    clickedBtnPosition = 3  ==> sipka dolu
    clickedBtnPosition = 4  ==> SET

    Do pameti znaku
    pin 1  - P2 0
    pin 2  - P2 1
    pin 3  - P2 2
    pin 4  - P2 3
    pin 5  - P2 4

    Klavesnice/multiplex
    pin 8  - P1 0
    pin 9  - P1 1
    pin 10 - P1 2
    pin 11 - P1 3
    pin 12 - P3 0

    Zámek
    pin 20 - P1 4

    Motor
    pin 19 - P1 5

    Žárovka
    pin 16 - P1 6

    Zvuk
    pin 15 - P1 7

    Dvirka zaverna?
    pin 7  - P3 1

    Snimac teploty
    pin 14 - P3 2
*/

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

```
#define P2OUT 0x301
#define P4IN 0x301
#define P1OUT 0x300
#define P3IN 0x300

int dispData[4] = { 0x1C, 0x11, 0x01, 0x1C }; // Data pro displej (defaultne =
.HI.)

int door_btn_masks[2] = { 0x02, 0x01 }; // Masky - dvere senzor, btn
int segments_masks[4] = { 0x7E, 0x7D, 0x7B, 0x77 }; // 1. segment, 2. segment, 3.
segment, 4. segment
int pwmMasks[6] = { 0x5F, 0x7F, 0x1F, 0x3F, 0x00, 0x00 }; // PWM masky -> MOTOR
ON; MOTOR OFF i ZAROVKA OFF; ZAROVKA ON i MOTOR ON; ZAROVKA ON; BUZZER ON; BUZZER
OFF;

int positionAsciiNumber = 48; // Konstanta pro znak '0'

// Funkce pro konverzi znaku na cislo pro displej
int getChar(char wantedChar) {
    switch (wantedChar) {
        case '1':
            return 0x01;
        case '2':
            return 0x02;
        case '3':
            return 0x03;
        case '4':
            return 0x04;
        case '5':
            return 0x05;
        case '6':
            return 0x06;
        case '7':
            return 0x07;
        case '8':
            return 0x08;
        case '9':
            return 0x09;
        case '0':
            return 0x00;
        case 'A':
            return 0x0A;
        case 'B':
            return 0x0B;
        case 'C':
            return 0x0C;
```

```
        case 'D':
            return 0x0D;
        case 'E':
            return 0x0E;
        case 'F':
            return 0x0F;
        case 'G':
            return 0x10;
        case 'H':
            return 0x11;
        case 'I':
            return 0x01;
        case 'J':
            return 0x12;
        case 'L':
            return 0x13;
        case 'N':
            return 0x14;
        case 'n':
            return 0x15;
        case 'O':
            return 0x00;
        case 'P':
            return 0x16;
        case 'R':
            return 0x17;
        case 'T':
            return 0x18;
        case 'U':
            return 0x19;
        case 'Y':
            return 0x1A;
        case '.':
            return 0x1C;
        default:
            return 0x00;
    }
}

// Funkce rozdeli cislo na 3 casti pro displej
void splitIntToThreeDigits(int number, int& digit1, int& digit2, int& digit3) {
    digit3 = number % 10;
    number /= 10;

    digit2 = number % 10;
    number /= 10;
```

```
    digit1 = number % 10;
    number /= 10;
}

// Obsluha tlacitek a displeje
int controlDisplayBtns(int clickedBtnPosition, int& isSegmentOn, int&
segmentDelayCounter, int& position, int& lastPushedBtn, int& testBtnDelay) {
    clickedBtnPosition = 0;

    if (position < 4) {
        if (isSegmentOn) {
            outportb(P1OUT, segments_masks[position]);  // Precteni tlacitka
            outportb(P2OUT, dispData[position]);    // Odeslani symbolu na displej
            int tmp = inportb(P3IN);

            delay(1);

            // Maskovani prectene hodnoty a ziskani informace zda je tlacitko
stisknuto
            if ((tmp &= door_btn_masks[1]) == 1)
            {
                // Podminka pro detekci nabezne hrany tlacitka + delay mezi
kontrolami tlacitek (80 * delta T = +-160ms)
                if (lastPushedBtn == clickedBtnPosition && testBtnDelay > 0) {
                    clickedBtnPosition = 0;
                    testBtnDelay--;
                }
                else {
                    clickedBtnPosition = position + 1;
                    lastPushedBtn = clickedBtnPosition;
                    testBtnDelay = 80;
                }
            }

            // Odecitani prubehu cyklu, kdy se zobrazuje segment
            if (segmentDelayCounter > 0) segmentDelayCounter--; else isSegmentOn =
0;
        } else {
            position++;
            isSegmentOn = 1;
            segmentDelayCounter = 2;
        }
    } else position = 0;

    return clickedBtnPosition;
```

```
}

// PWM funkce, pro motor i zarovku
int togglePWM(int shiftMotor, int shiftBulb, int isBulbOn, int& isMotorOn, int&
motorToggleCounterOn, int& bulbToggleCounterOn)
{
    motorToggleCounterOn++;

    if (motorToggleCounterOn >= shiftMotor) isMotorOn = 0; // Vypne motor
    else if (motorToggleCounterOn >= 10) { // Zapne motor
        isMotorOn = 1;
        motorToggleCounterOn = 0;
    }

    bulbToggleCounterOn++;

    if (bulbToggleCounterOn >= shiftBulb) isBulbOn = 0; // Vypne zarovku
    else if (bulbToggleCounterOn >= 10) { // Zapne zarovku
        isBulbOn = 1;
        bulbToggleCounterOn = 0;
    }

    int tmp = pwmMasks[1];
    if (isBulbOn && isMotorOn) tmp = pwmMasks[2];
    else if (isBulbOn && !isMotorOn) tmp = pwmMasks[3];
    else if (!isBulbOn && isMotorOn) tmp = pwmMasks[0];

    outportb(P1OUT, tmp);

    return isBulbOn;
}

// Zapinani/Vypinani bzucaku
int toggleBuzzer(int isBuzzerOff) {
    int tmp;

    if (isBuzzerOff) {
        isBuzzerOff = 0;
        tmp = pwmMasks[5];
    } else {
        isBuzzerOff = 1;
        tmp = pwmMasks[4];
    }

    outportb(P1OUT, tmp);
```

```
    return isBuzzerOff;
}

// Kontrola stavu dvirek
int checkDoor(void) {
    int tmp = inportb(P3IN);

    if ((tmp &= door_btn_masks[0]) == 1) return 0;
    else return 1;
}

// Otevreni dvirek (maska 0x6F)
void openDoor(void) {
    printf("\n\rDvirka mirkovlnky otevreny!");
    outportb(P1OUT, 0x6F);
}

// Kontrola stavu klavesnice u PC
int checkKeyboard(int shiftMotor, char& pressedKey) {
    if (kbhit())
    {
        pressedKey = tolower(getch());

        switch (pressedKey) {
            case 'w':
                if (shiftMotor < 50) {
                    shiftMotor += 5;
                    printf("\n\r\n\rRychlost otaceni zvysena o 10%");
                    printf("\n\rAktualni rychlost: %i%", shiftMotor*2);
                } else printf("\n\r!! Nelze zvysit rychlost otaceni !!");
                break;
            case 's':
                if (shiftMotor > 5) {
                    shiftMotor -= 5;
                    printf("\n\r\n\rRychlost otaceni snizena o 10%");
                    printf("\n\rAktualni rychlost: %i%", shiftMotor*2);
                } else printf("\n\r!! Nelze snizit rychlost otaceni !!");
                break;
            case 'q':
                break;
            default:
                printf("\n\r!! Stiskl jsi nepovolenou klavesu !!");
                break;
        }
    }
```

```
    return shiftMotor;
}


// Hlavni cyklus programu, delta T = +-2ms
int main()
{
    // Deaktivace celeho portu P1
    outportb(P1OUT, 0x7F);

    int settingsState = 0; // Stav programu
    int timeCounter = 0; // Counter casu
    int tempCounter = 3; // Counter teploty (3 moznosti teploty - HIGH, MED, LOW)
    int realTime; // Realny cas prepocitany z timeCounter

    int shiftMotor = 25; // Strida PWMka motoru, ze zacatku (50%)
    int isMotorOn = 0; // Stav motoru
    int motorToggleCounterOn = 10; // Counter pro zapinani motoru

    int shiftBulb = 15; // Strida PWMka svetla, ze zacatku (50%)
    int isBulbOn = 0; // Stav zarovky
    int bulbToggleCounterOn = 10; // Counter pro zapinani zarovky

    int isBuzzerOff = 0; // Stav buzzeru

    int segmentDelayCounter = 0; // Counter pro zapinani/vypinani segmentu
    int isSegmentOn = 0; // Stav segmentu
    int lastPushedBtn = 0; // Posledni stisknute tlacitko
    int testBtnDelay = 80; // Delka po kterou nebude "testovat" tlacitko
    int position = 0; // Pozice segmentu

    char pressedKey;

    printf("\x1B[2J\x1B[H"); // Vymazani obrazovky
    printf("\n\r[##################################################]");
    printf("\n\r Zapinani programu Mikrovlnka, vytvoril Jakub Tenk");
    printf("\n\r[--------------------------------------------------]");
    printf("\n\r Zakladni ovladani: ");
    printf("\n\r   Stisk klavesy W nebo w  ->  zvysi rychlost otaceni talire o 10%
");
    printf("\n\r   Stisk klavesy S nebo s  ->  snizi rychlost otaceni talire o 10%
");
    printf("\n\r   Stisk klavesy Q nebo s  ->  ukonci program ");
    printf("\n\r[--------------------------------------------------]");
    printf("\n\r Cekani na interakci uzivatele s mikrovlnkou.");
    printf("\n\r[##################################################]");
```

```
    while(pressedKey != 'q' && pressedKey != 'Q') {
        // Zjisteni stavu dvirek
        int doorState = checkDoor();
        // Obsluha tlacitek a displeje
        int clickedBtnPosition = controlDisplayBtns(clickedBtnPosition,
isSegmentOn, segmentDelayCounter, position, lastPushedBtn, testBtnDelay);
        // Zjisteni stavu PC klavesnice
        shiftMotor = checkKeyboard(shiftMotor, pressedKey);


        if (doorState == 0) {

            // Obsluha a nastaveni mikrovlnky
            switch(settingsState) {
                case 0: // Zakladni stav (default)
                    // Nastaveni slova .HI. na displeji
                    dispData[0] = getChar('.');
                    dispData[1] = getChar('H');
                    dispData[2] = getChar('I');
                    dispData[3] = getChar('.');

                    // Vynulovani casu
                    realTime = 0;
                    timeCounter = 0;

                    // Postup do nastaveni casu
                    if (clickedBtnPosition == 1) {
                        printf("\n\rPrechod na nastaveni doby ohrevu!");
                        clickedBtnPosition = 0;
                        settingsState++;
                    }

                    // Otevreni dveri
                    if (clickedBtnPosition == 4) {
                        openDoor();
                        clickedBtnPosition = 0;
                    }

                    break;
                case 1: // Stav nastaveni doby ohrevu
                    // Pricteni casu o +10s k aktualni hodnote, nastaveno do max.
1000 sekund
                    if (clickedBtnPosition == 2 && timeCounter < 1000) {
                        clickedBtnPosition = 0;
```

```
                    timeCounter += 10;
                }

                // Odecteni casu o -10s od aktualni hodnoty, minimalni cas
nastaven na 20 sekund
                if (clickedBtnPosition == 3 && timeCounter > 20) {
                    clickedBtnPosition = 0;
                    timeCounter -= 10;
                }

                int digit1, digit2, digit3;
                splitIntToThreeDigits(timeCounter, digit1, digit2, digit3);

                dispData[0] = getChar('.');
                dispData[1] = getChar(positionAsciiNumber + digit1);
                dispData[2] = getChar(positionAsciiNumber + digit2);
                dispData[3] = getChar(positionAsciiNumber + digit3);

                // Potvrzeni casu -> pokracuje se na nastaveni teploty
                if (clickedBtnPosition == 1) {
                    clickedBtnPosition = 0;
                    printf("\n\rNastavena doba ohrevu byla potvrzena, presun
na nastaveni teploty.");
                    realTime = timeCounter * 1000; // Realny cas v ms
                    settingsState++;
                }

                // Zrusi nastavovani casu -> vraci se na zakladni stav
                if (clickedBtnPosition == 4) {
                    clickedBtnPosition = 0;
                    printf("\n\rNastavovani doby ohrevu bylo zruseno,
mikrovlnka ceka na interakci s uzivatelem.");
                    settingsState--;
                }

                break;
            case 2: // Stav nastaveni teploty
                // Nastaveni teploty mezi LOW, MEDIUM, HIGH
                if (clickedBtnPosition == 2 && tempCounter > 1) {
                    clickedBtnPosition = 0;
                    tempCounter--;
                }

                if (clickedBtnPosition == 3 && tempCounter < 3) {
                    clickedBtnPosition = 0;
                    tempCounter++;
```

```
                    }

                    switch(tempCounter) {
                        case 1: // Vypis HIGH
                            dispData[0] = getChar('H');
                            dispData[1] = getChar('I');
                            dispData[2] = getChar('G');
                            dispData[3] = getChar('H');
                            shiftBulb = 30;
                            break;
                        case 2: // Vypis MED
                            dispData[0] = getChar('N');
                            dispData[1] = getChar('N');
                            dispData[2] = getChar('E');
                            dispData[3] = getChar('D');
                            shiftBulb = 15;
                            break;
                        case 3: // Vypis LOW
                            dispData[0] = getChar('L');
                            dispData[1] = getChar('O');
                            dispData[2] = getChar('U');
                            dispData[3] = getChar('U');
                            shiftBulb = 5;
                            break;
                    }

                    // Potvrzeni nastavene teploty -> pokracuje se na proces
ohrevu
                    if (clickedBtnPosition == 1) {
                        clickedBtnPosition = 0;
                        printf("\n\rNastavena teplota byla potvrzena, zacina ohrev
pokrmu.");
                        tempCounter = tempCounter * 10 + 50;
                        settingsState++;
                    }

                    // Zrusi nastavovani teploty -> vraci se na zakladni stav
                    if (clickedBtnPosition == 4) {
                        clickedBtnPosition = 0;
                        printf("\n\rNastavovani teploty bylo zruseno, mikrovlnka
ceka na interakci s uzivatelem.");
                        settingsState = 0;
                    }
                    break;
                case 3: // Stav procesu ohrevu
                    if (realTime > 0) {
```

```
                        if (clickedBtnPosition == 4) {
                            clickedBtnPosition = 0;
                            realTime = 0;
                            settingsState = 0;
                        }

                        realTime -= 1;

                        // Obsluha PWM
                        isBulbOn = togglePWM(shiftMotor, shiftBulb, isBulbOn,
isMotorOn, motorToggleCounterOn, bulbToggleCounterOn);

                        // Obsluha vypisu zbyvajiciho casu
                        if ((realTime % 1000) == 0) {
                            int remainingTime = realTime / 1000; // 30000ms =>
30sec .... 10000ms => 10sec

                            printf("\n\rZbyvajici cas ohrevu je %i sekund.",
remainingTime);

                            int digit1, digit2, digit3;
                            splitIntToThreeDigits(remainingTime, digit1, digit2,
digit3);

                            // Promitne zbyvajici cas v sekundach na displeji
                            dispData[0] = getChar('.');
                            dispData[1] = getChar(positionAsciiNumber + digit1);
                            dispData[2] = getChar(positionAsciiNumber + digit2);
                            dispData[3] = getChar(positionAsciiNumber + digit3);
                        }

                    } else {
                        realTime = 2000;
                        settingsState = 4;
                    }

                    break;
                case 4: // Stav dokonceni ohrevu, vypne pwm, vypise "GOOD" a 4x
pipne bzucak
                    dispData[0] = getChar('G');
                    dispData[1] = getChar('O');
                    dispData[2] = getChar('O');
                    dispData[3] = getChar('D');

                    if (realTime > 0) {
                        if (clickedBtnPosition == 4) {
```

```
                        clickedBtnPosition = 0;
                        realTime = 0;
                        settingsState = 0;
                    }
                    realTime -= 1;
                    if ((realTime % 500) == 0) {
                        toggleBuzzer(isBuzzerOff); // Zapne/vypne bzucak
                    }

                } else {
                    realTime = 0;
                    settingsState = 0;
                }

                break;
            }
        }
        else {
            // Dvirka otevrena, na displeji vypise "OPEN"
            outportb(P1OUT, 0x7F); // Deaktivuje P1, dokud se nezavrou dvirka
            settingsState = 0;

            dispData[0] = getChar('O');
            dispData[1] = getChar('P');
            dispData[2] = getChar('E');
            dispData[3] = getChar('N');
        }

        delay(1); // Zajisti zpozdeni 1ms (tudiz delta T = +-2ms i s instrukcemi)
    }

    printf("\n\r\n\r[############################]");
    printf("\n\r Vypinani programu Mikrovlnka");
    printf("\n\r[############################]");
    delay(1000);

    return 0;
}
```