Vysoká škola chemicko-technologická v Praze
Fakulta chemicko-inženýrská
Ústav počítačové a řídicí techniky

Aplikace mikroprocesorů

Desky Evb Display a Evb Keyboard

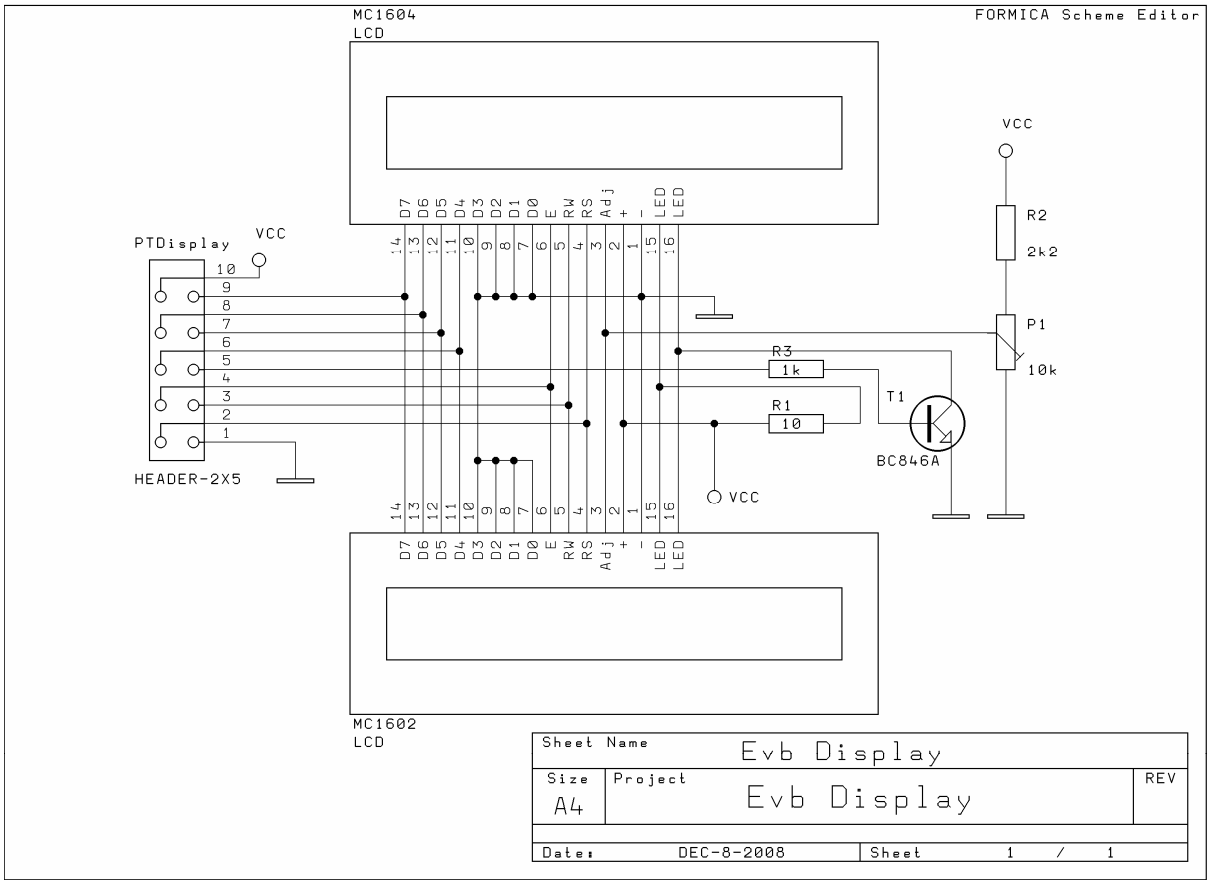
Návod k použití

Lukáš Lahoda 2010

Obsah

1 Deska Evb Display

Deska Evb Display je osazena čtyřřádkovým LCD displejem, délka každého řádku je 16 znaků. Schéma displeje uvádí *obr. 1*. Jak je na schématu vidět, samotný displej je ve skutečnosti složen ze dvou dvouřádkových displejů. V praxi to má své důsledky, a sice že třetí řádek je prodloužením prvního a čtvrtý řádek druhého řádku. Pokud tedy zapisujeme znaky na první řádek, tak po jeho zaplnění skočí kurzor na řádek č. 3, pak na řádek č. 2 a nakonec na řádek č. 4. Na to je nutné v programu pamatovat.



Obr. 1. Schéma desky Evb Display

1.1 Propojení desky Evb IO s deskou EvbHCS08

Desku Evb Display je možné ovládat přes desku Evb HCS08. Obě desky je nutné propojit přes paralelní port za pomoci plochého desetižilového kabelu, který naleznete v laboratoři. V tomto návodu budeme dále předpokládat, že propojení je realizováno přes port PTB[1] desky EvbHCS08. Po propojení můžete přes piny portu PTB (PTB0 až PTB7) přistupovat k pinům displeje. Tabulka *tab. 1* uvádí, jak jsou si piny vzájemně přiřazeny. V tabulce je uveden i význam pinů displeje.

[1] POZOR! Na deskách EvbHCS08 přítomných v laboratoři AL02 je prohozeno označení portů PTA a PTB!

Tab. 1. Přiřazení bitů displeje bitům portu PTB a jejich význam

PTB	displej	význam
PTB0	RS	určuje, zda je zadáván znak, nebo příkaz
PTB1	RW	určuje, zda se bude číst, nebo zapisovat
PTB2	E	povoluje přístup k bitům D4 – D7
PTB3	LED	podsvícení displeje
PTB4	D4	příjem 0. a 4. bitu znaku
PTB5	D5	příjem 1. a 5. bitu znaku
PTB6	D6	příjem 2. a 6. bitu znaku
PTB7	D7	příjem 3. a 7. bitu znaku

1.2 Komunikace s displejem

Komunikace s displejem je řízena bity RS, RW a E a probíhá pomocí bitů D4 – D7, přes které můžeme odesílat řídicí příkazy, nebo znaky. O tom, zda se jedná o příkaz, nebo znak, rozhoduje hodnota bitu RS (0 – příkaz, 1 – znak).

1.3 Inicializace displeje

Abychom mohli zahájit komunikaci s displejem, musíme nejprve (1x na začátku programu) odeslat sérii výrobcem definovaných příkazů. Inicializace probíhá v následujících krocích

1) Nastavení všech pinů portu PTB pro zápis
2) Nastavit bit RS na 0 a zbývající bity na 1
3) Nastavit bit RS na 1 a zbývající bity na 0
4) Odeslat na displej příkazy v pořadí dle *tab. 2*. Postup pro odeslání příkazu je uveden v odst. 1.4.

Tab. 2. Inicializační příkazy displeje

pořadí	hodnota bin	hodnota hex
1	0b00000011	0x03
2	0b00000011	0x03
3	0b00000011	0x03
4	0b00000010	0x02
5	0b00101000	0x28
6	0b00001000	0x08
7	0b00000001	0x01
8	0b00000110	0x06
9	0b00101000	0x28
10	0b00001100	0x0C
11	0b00000110	0x06
12	0b00001100	0x0C

1.4 Odesílání příkazů a znaků na displej

Pro odeslání znaku či příkazu na displej je nutné dodržet následující postup:

1) bit RW nastavit na 0 (= zápis)
2) bit RS nastavit na 0, jedná-li se o příkaz, nebo na 1, jedná-li se o znak
3) bit E nastavit na 1
4) zápis horních 4 bitů bajtu
5) bit E nastavit na 0
6) bit E nastavit opět na 1
7) zápis dolních 4 bitů bajtu
8) bit E nastavit na 0

Na displeji je možné zobrazit znaky ASCII kódu 32 – 127.

1.5 Dostupné příkazy

V *tab. 2* jsou uvedeny příkazy, které je možné při práci s displejem použít.

Tab. 3. Dostupné příkazy pro displej

hodnota	význam
0b00000001	zapne displej
0b00001000	vypne displej
0b00000001	smaže obsah displeje
0b10000000	nastaví kurzor na začátek prvního řádku
0b11000000	nastaví kurzor na začátek druhého řádku
0b10000001	nastaví kurzor na 2.pozici prvního řádku (přičtením požadovaného čísla k příkazu 0b10000000 můžete kurzor nastavit na libovolnou pozici)
0b10010000	nastaví kurzor na začátek třetího řádku (1. řádek + 16 pozic)
0b11010000	nastaví kurzor na začátek čtvrtého řádku (2. řádek + 16 pozic)
0b00001100	kurzor neviditelný
0b00001101	kurzor podtržítko bliká
0b00001110	kurzor plný
0b00001111	kurzor plný bliká

1.6 Podsvícení displeje

Nastavením bitu LED na 1 se rozsvítí LED podsvícení displeje. Zápisem 0 naopak zhasne.

1.7 Programový pracovní rámec pro displej

Abychom mohli jednoduše ovládat displej, tj. přistupovat k jednotlivým řádkům, zapisovat znaky, řetězce nebo matice znaků, vytvoříme si v prostředí CodeWarrior několik funkcí, které tyto činnosti budou zajišťovat. Funkce můžete umístit do samostatných souborů, které uložíte do složky Sources v pracovním adresáři projektu. Tyto soubory poté nalinkujete do souboru main.c pomocí direktivy `#include "vas_soubor.c"`. Všechny funkce musí být do souborů umístěny ve stejném pořadí, v jakém jsou uváděny v tomto návodu.

1.7.1 Funkce pro čekání

Mezi jednotlivými příkazy je vždy nutné ponechat displeji čas na jejich zpracování. Vytvořte soubor wait.c a umístěte do něj následující 2 funkce, které umožňují realizovat časovou prodlevu volitelné délky.

```
void wait() {
      unsigned int j;
      for (j=0;j<200;j++) {
        __RESET_WATCHDOG();
      }
}

void waitMore(unsigned int times) {
   unsigned int i;
   for (i=0;i<times;i++) {
     wait();
   }
}
```

1.7.2 Definice konstant

V souboru main.c definujte následující konstanty představující bity displeje. Nalinkování externích souborů proveďte až po definici těchto konstant.

```
#include <hidef.h>
#include "derivative.h" //...

#define RS      PTBD_PTBD0
#define RW      PTBD_PTBD1
#define E       PTBD_PTBD2
#define LED     PTBD_PTBD3
#define D4      PTBD_PTBD4
#define D5      PTBD_PTBD5
#define D6      PTBD_PTBD6
#define D7      PTBD_PTBD7

#include "wait.c"
#include "disp.c" //...
```

Vytvořte si soubor disp.c ve složce Sources a umístěte do něj tyto dvě konstanty představující počet řádků a délku řádku displeje. Do tohoto souboru umisťujte i funkce pro displej (viz dále).

```
#define dispRowCount  4
#define dispColCount  16
```

1.7.3 Funkce pro odeslání bajtu

Nejprve vytvořte funkci `dispSend()` pro odeslání jednoho bajtu. Funkce přejímá bajt v podobě argumentu typu unsigned char a následně odešle na displej nejdříve horní a poté dolní 4 bity.

```
void dispSend (unsigned char byte) {

  E=1;
  D4=(byte&0b00010000)>>4;
  D5=(byte&0b00100000)>>5;
  D6=(byte&0b01000000)>>6;
  D7=(byte&0b10000000)>>7;
  E=0;

  E=1;
  D4=(byte&0b00000001)>>0;
  D5=(byte&0b00000010)>>1;
  D6=(byte&0b00000100)>>2;
  D7=(byte&0b00001000)>>3;
  E=0;

  wait();
}
```

1.7.4 Funkce pro odeslání příkazu/znaku

Ve funkci dispSend() jsme neurčili, odesíláme-li příkaz, nebo znak. Pro odeslání příkazů a znaků vytvoříme 2 zvláštní funkce, které přejmou bajt ve formě argumentu, nastaví bity RS a RW a předají funkci dispSend() bajt k odeslání. Nejdříve vytvořte funkci dispCommand() pro odeslání příkazu

```
void dispCommand (unsigned char command) {

  RW=0;
  RS=0;

  dispSend(command);

}
```

a poté funkci dispChar() pro odeslání znaku. Funkce kontroluje povolený ASCII rozsah.

```
void dispChar (unsigned char character) {

  if (character<32 || character>127) return;

  RW=0;
  RS=1;

  dispSend(character);

}
```

1.7.5 Funkce pro výběr řádku

Vytvořte funkci dispSetRow() pro nastavení kurzoru na začátek řádku, jehož číslo se bude předávat přes argument funkce.

```
void dispSetRow (unsigned int row) {

  switch(row) {
    case 1:
      dispCommand(0b10000000);
      break;

     case 2:
      dispCommand(0b11000000);
      break;

     case 3:
      dispCommand(0b10000000+dispColCount);
      break;

     case 4:
      dispCommand(0b11000000+dispColCount);
  }

}
```

1.7.6 Funkce pro inicializaci displeje

Na začátku programu musíte displej inicializovat odesláním série příkazů. Funkci dispInit() zavolejte vždy na začátku funkce main(). Přidejte navíc funkci dispClear(), která odesílá příkaz na smazání displeje.

```
void dispClear() {
  dispCommand(0b00000001);
}

void dispInit() {

  PTBDD=0b11111111;
  PTBD=0b11111110;
  PTBD=0b00000001;

  dispCommand(0b00000011); //0x03
  dispCommand(0b00000011); //0x03
  dispCommand(0b00000011); //0x03
  dispCommand(0b00000010); //0x02

  dispCommand(0b00101000); //0x28
  dispCommand(0b00001000); //0x08
  dispCommand(0b00000001); //0x01
  dispCommand(0b00000110); //0x06
  dispCommand(0b00101000); //0x28
  dispCommand(0b00001100); //0x0C
  dispCommand(0b00000110); //0x06
  dispCommand(0b00001100); //0x0C

  LED=1;

  dispClear();
}
```

1.7.7 Funkce pro odeslání řetězce nebo matice znaků na displej

Funkce dispString() přejímá řetězec, jehož délka je rovna počtu znaků na řádek displeje. Tento řetězec vypíše na řádek určený funkcí dispSetRow().

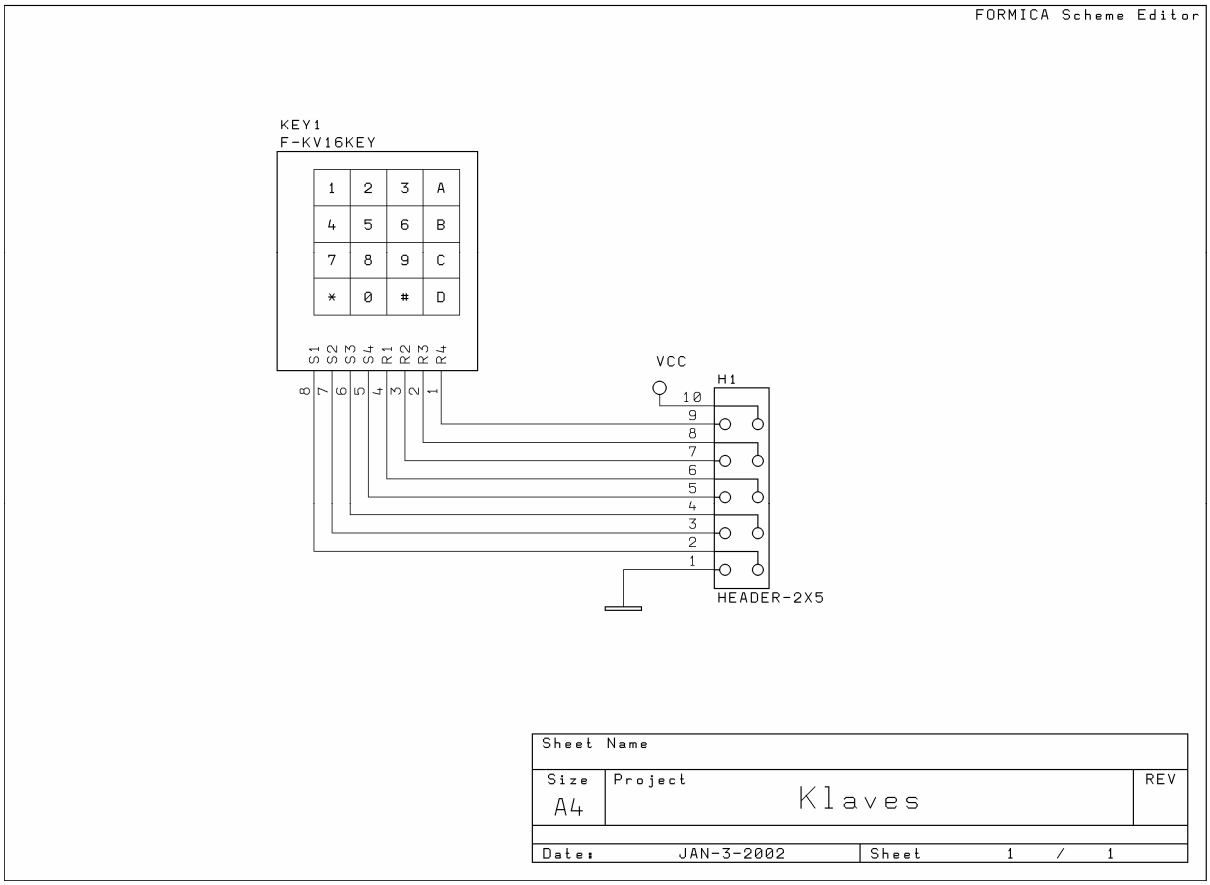
```
void dispString(char str[dispColCount]) {

  unsigned int i;

  for (i=0;i<dispColCount;i++) {
    dispChar(str[i]);
  }

}
```

Funkce dispMatrix() zobrazí celou matici znaků. Argumentem je 2D pole o rozměrech (počet řádků) x (počet sloupců) displeje.

```
void dispMatrix(unsigned char matrix[dispRowCount][dispColCount]) {

  unsigned int i;

  for (i=0;i<dispRowCount;i++) {
    dispSetRow(i+1);
    dispString(matrix[i]);
  }

}
```

2 Deska Evb Keyboard

Deska Evb Keyboard je osazena maticovou klávesnicí o čtyřech řádcích a čtyřech sloupcích. Schéma desky je uvedeno na *obr. 2*.



Obr. 2. Schéma desky Evb Keyboard

2.1 Propojení desky Evb IO s deskou EvbHCS08

Tuto desku propojte přes paralelní port PTA s deskou EvbHCS08. K ovládání klávesnice je k dispozici 8 bitů S1 – S4 indikujících stisk tlačítka ve sloupci 1 – 4 a R1 – R4 indikujících stisk tlačítka na řádku 1 – 4. Tyto bity jsou určeny výhradně pro čtení a jejich propojení s bity portu PTA uvádí *tab. 4*. Pozor – řádky jsou číslovány zdola nahoru a sloupce zprava doleva!
.

Tab. 4. Přiřazení bitů klávesnice bitům portu PTA a jejich význam

PTB	displej	význam
PTA0	S1	indikace stisku tlačítka v 1. řádku
PTA1	S2	indikace stisku tlačítka v 2. řádku
PTA2	S3	indikace stisku tlačítka v 3. řádku

PTA3	S4	indikace stisku tlačítka v 4. řádku
PTA4	R1	indikace stisku tlačítka v 1. sloupci
PTA5	R2	indikace stisku tlačítka v 2. sloupci
PTA6	R3	indikace stisku tlačítka v 3. sloupci
PTA7	R4	indikace stisku tlačítka v 4. sloupci

Pro čtení znaku z klávesnice je potřeba nastavit všechny piny PTA pro čtení a následně číst vždy nejprve řádek a poté sloupec. Pokud odečteme hodnotu 1, znamená to, že příslušná klávesa nebyla stisknuta, 0 znamená, že v daném řádku/sloupci byla klávesa stisknuta.

2.2 Programový pracovní rámec pro klávesnici

Podobně jako u displeje i pro klávesnici vytvoříme několik funkcí, které nám později usnadní práci.

2.2.1 Definice konstant

Do souboru main.c umístěte definice potřebných konstant a naliknujte soubor kb.c, který vytvořte ve složce Sources a do kterého umístíte funkce pro klávesnici. Dále je potřeba nalinkovat soubor wait.c obsahující funkce pro zpoždění (odst. 1.7.1). Váš soubor main.c by měl začínat nějak takto:

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

//konstanty pro klávesnici:
#define S1      PTAD_PTAD0
#define S2      PTAD_PTAD1
#define S3      PTAD_PTAD2
#define S4      PTAD_PTAD3
#define R1      PTAD_PTAD4
#define R2      PTAD_PTAD5
#define R3      PTAD_PTAD6
#define R4      PTAD_PTAD7

//další konstanty, např. pro displej

#include "wait.c"
#include "kb.c"
//další include

//další kód
```

2.2.2 Funkce pro detekci stisku klávesy

Pro zjištění, která klávesa byla stisknuta, vytvoříme tři funkce: kbGetCol() pro zjištění řádku, kbGetRow() pro zjištění sloupce a kbGetChar() pro určení, jaký znak odpovídá dané kombinaci řádek-sloupec. Dále je potřeba vytvořit globální proměnnou kbPressed, která se v okamžiku stisku klávesy nastaví na 1 a zamezí tak opakovanému vkládání znaku po dobu, kdy klávesa zůstává stisknutá. Pokud bychom chtěli číst více stisknutých kláves zároveň, bylo by nutné kód upravit. Kód z tohoto odstavce umístěte do souboru kb.c.

```
unsigned char kbPressed=0;

unsigned char kbGetRow() {

  PTADD= 0b11110000;    //řádky
  PTAPE= 0b00001111;

  waitMore(5);

  if (!S1) return 4;
  else if (!S2) return 3;
  else if (!S3) return 2;
  else if (!S4) return 1;
  return 0;

}

unsigned char kbGetCol() {

  PTADD= 0b00001111;    //sloupce
  PTAPE= 0b11110000;

  waitMore(5);

  if (!R1) return 4;
  else if (!R2) return 3;
  else if (!R3) return 2;
  else if (!R4) return 1;
  return 0;

}

unsigned char kbGetChar() {

  unsigned char row;
  unsigned char col;

  if(!(row=kbGetRow()) || !(col=kbGetCol()) {
    return (kbPressed=0);
  }

  if (!kbPressed) {
    kbPressed=1;
    switch (row) {
      case 1:
        switch(col) {
          case 1: return '1';
          case 2: return '2';
          case 3: return '3';
          case 4: return 'A';
        }
      case 2:
        switch(col) {
          case 1: return '4';
          case 2: return '5';
          case 3: return '6';
          case 4: return 'B';
        }
      case 3:
        switch(col) {
          case 1: return '7';
          case 2: return '8';
          case 3: return '9';
          case 4: return 'C';
        }
```

```
      case 4:
        switch(col) {
          case 1: return '*';
          case 2: return '0';
          case 3: return '#';
          case 4: return 'D';
        }
    }
  }

  return 0;

}
```

Čtení sloupce a řádku se děje sekvenčně, protože současné čtení není možné. Čtené bity je nutné nastavit pro čtení a ostatní bity pro zápis. Pro čtené bity se navíc musí nastavit pullup enable na 1. Čteme nejprve horní a pak dolní čtyři bity portu. Mezi nastavením registrů portu a samotným čtením hodnoty jeho pinů je nutné ponechat určitou časovou prodlevu.

Samostatný úkol

S využitím funkcí uvedených v tomto návodu realizujte psaní znaků na displej. Program upravte tak, aby po zaplnění řádku kurzor skočil na správný nový řádek a aby se displej po zaplnění smazal a kurzor nastavil na začátek prvního řádku.

3 Ukázkový program – Had

V této kapitole demonstrujeme využití funkcí, které jsme vytvořili pro manipulaci s displejem a klávesnicí. Vytvoříme hru „Had“ známou především z raných mobilních telefonů. Zde uvedená verze je poněkud zjednodušená, jelikož had bude mít pevnou délku a nebude přijímat žádnou potravu a růst. Bude se pouze pohybovat a měnit směr pomocí kláves 2, 4, 6, 8. Jeho tělo bude tvořeno znaky „o“. Funkce pro hada umístíme do samostatného souboru snake.c a ve funkci main() zajistíme jejich volání v nekonečné smyčce, přičemž v každé iteraci nejprve zjistíme, byla-li stisknuta směrová klávesa a pak posuneme hada o 1 jednotku.

Had bude mít délku 4 znaky a souřadnice jeho segmentů budou uloženy v globálním 2D poli o čtyřech prvcích, přičemž každý prvek bude dvouprvkové pole uchovávající souřadnice, tj. řádek a sloupec segmentu. Celková situace se bude zobrazovat jako matice znaků o rozměrech displeje, kde znaky o souřadnicích hadových segmentů budou vyplněny znakem „o“ a ostatní prvky prázdným znakem (mezerou).

snake.c:

```
#define snakeLength 4

unsigned int snakeDirection=0; //0-east, 1-north, 2-west, 3-south

unsigned char snake[4][2]={{2,5},{2,6},{2,7},{2,8}}; //počáteční stav

unsigned char plane[dispRowCount][dispColCount]; //pracovní plocha


void snakeInit() {
  //vyplní pracovní plochu mezerami

  unsigned int i;
  unsigned int j;

  for (i=0;i<=dispRowCount;i++) {
    for (j=0;j<=dispColCount;j++) {
      plane[i][j]=' ';
    }
  }

}

void snakeCheckDirection() {
  //zjistí, zda byla stisknuta směrová klávesa a případně změní směr

  unsigned char key=kbGetChar();

  switch (key) {

    case '6':
      if (snakeDirection==1 || snakeDirection==3) {
        snakeDirection=0;
      }
      break;

    case '2':
      if (snakeDirection==0 || snakeDirection==2) {
        snakeDirection=1;
      }
      break;
```

```
    case '4':
      if (snakeDirection==1 || snakeDirection==3) {
        snakeDirection=2;
      }
      break;

    case '8':
      if (snakeDirection==0 || snakeDirection==2) {
        snakeDirection=3;
      }

  }

}

void snakeMove() {
  //posune hada o 1 jednotku zvoleným směrem

  unsigned char begin[2];
  unsigned char end[2];

  unsigned int i;

  begin[0]=snake[3][0];
  begin[1]=snake[3][1];
  end[0]=snake[0][0];
  end[1]=snake[0][1];

  plane[end[0]][end[1]]=' ';

  switch (snakeDirection) {

    case 0: //east
      if (begin[1]==dispColCount-1) begin[1]=0;
      else begin[1]++;
      break;

    case 2: //west
      if (begin[1]==0) begin[1]=dispColCount-1;
      else begin[1]--;
      break;


    case 1: //north
      if (begin[0]==0) begin[0]=dispRowCount-1;
      else begin[0]--;
      break;

    case 3: //south
      if (begin[0]==dispRowCount-1) begin[0]=0;
      else begin[0]++;
      break;

  }

  for (i=0;i<3;i++) {
    snake[i][0]=snake[i+1][0];
    snake[i][1]=snake[i+1][1];
    plane[snake[i][0]][snake[i][1]]='o';
  }

  snake[3][0]=begin[0];
  snake[3][1]=begin[1];
  plane[begin[0]][begin[1]]='o';
```

```
  dispMatrix(plane);

}
```

main.c:

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */

//display constants
#define RS      PTBD_PTBD0
#define RW      PTBD_PTBD1
#define E       PTBD_PTBD2
#define LED     PTBD_PTBD3 //podsvícení (1=svítí)
#define D4      PTBD_PTBD4
#define D5      PTBD_PTBD5
#define D6      PTBD_PTBD6
#define D7      PTBD_PTBD7

//keyboard constants
#define S1      PTAD_PTAD0
#define S2      PTAD_PTAD1
#define S3      PTAD_PTAD2
#define S4      PTAD_PTAD3
#define R1      PTAD_PTAD4
#define R2      PTAD_PTAD5
#define R3      PTAD_PTAD6
#define R4      PTAD_PTAD7

#include "wait.c"
#include "disp.c"
#include "kb.c"
#include "snake.c"


void main(void) {

  SOPT_COPE=0; //zákaz watchdog

  dispInit();
  snakeInit();

  for(;;) {

    snakeCheckDirection();
    snakeMove();

    waitMore(50); //určuje rychlost hada

  }

}
```

4 Literatura

[1] Freescale Semiconductor MC9S08GB/GT Data Sheet