Mikroračunalniški sistemi

Laboratorijske vaje
Študijsko leto 2008/2009

A. M.
11.1.2009

Vaja1: Uporaba GPIO linij

Izmenično prižiganje štirih svetlečih diod - delo s splošno namenskimi vhodno-izhodnimi linijami
V programskem jeziku C napišite program, ki bo na razvojni plošči ITLPC2138 izmenično prižigal štiri svetleče diode, kot je to izvedeno na avtomobilu glavnega junaka v nadaljevanki Night rider.
Potrebne datoteke:
• crt0.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)

Program:

```
#define REG32 (volatile unsigned int*)    //zamenjamo kodo v oklepaju za kodo pred njim
#define IO0PIN  (*(REG32(0xE0028000)))
#define IO0SET  (*(REG32(0xE0028004)))
#define IO0DIR  (*(REG32(0xE0028008)))
#define IO0CLR  (*(REG32(0xE002800C)))

void pl(int led);    //deklaracija funkcije pl
void cakaj();        //deklaracija funkcije cakaj
int main(void)
{

      //(*((volatile unsigned int*)(0xE0028008)))=0xF00;
      IO0DIR=0xF00;   // nastavimo IO0DIR na mestu diod na 1 (izhod),
      IO0CLR=0xF00;   // IO0CLR postavimo diode na 0 (ugasnemo jih ce so prižgane)

      int stevec=0;
      int smer=1;
      while(1)
      {
        if(smer==1) stevec++;
        else stevec--;

        if(stevec==3) smer=-1;
        if(stevec==0) smer=1;

        pl(stevec);   //kliče funkcijo pl

      }
}

void pl(int led)
{
      IO0CLR=0xF00;
      IO0SET=0x1<<(8+led);    //prižigamo led diodice
      cakaj(1000);
}

void cakaj(int ms)
{
int i,j;
for(i=0;i<ms;i++)
for(j=0;j<1000;j++);     //zaposlimo računalnik

}
```

Vaja1.1: Uporaba GPIO linij

Dodatek k programu vaje1.1:

```
int beritipko()
{
   int pinreg, tipka;

   pinreg=IO0PIN;
   pinreg=pinreg>>12;        //pomaknemo bite za 12 mest, tako da bojo tipke na zacetku
   pinreg=~pinreg;           // negiramo
   pinreg=pinreg&0xF ;       // naredimo operacijo AND, vse bite ,ki niso tipke postavimo na 0, biti tipk
se ohranijo

   if(pinreg==0x1) tipka=0;
   if(pinreg==0x2) tipka=1;
   if(pinreg==0x4) tipka=2;
   if(pinreg==0x8) tipka=3;
   return tipka;
}
```

GPIO LINIJE

GPIO (**G**eneral **P**urpose **I**nput **O**utput) linije so splošno namenske vhodno ihodne linije.

IO0PIN Z njih lahko beremo tre-nutno stanje na linijah neglede na to ali gre za vhodno ali izhodno linijo. Vendarto velja samo za primer kadar so linije konfigurirane kot splošno namenske (ang. General Purpose Input Output - GPIO). Če linije postavimo v katerikoli drug način delovanja, vrednosti v teh registrih niso veljavne.
(naslov IOPIN0 - 0xE002 8000 in IOPIN1 - 0xE002 801)
IO0SET Registra se uporabljata za dolo˘canje stanja izhodnih linij. V primeru ko je linija konfigurirana kot GPIO izhod, bo pisanje logične ’1’ v register postavilo ustrezno linijo na visok nivo. Pisanje logične ’0’ nima nikakršnega vpliva na dogajanje na izhodu. Kadar je linija konfigurirana kot GPIO vhod ali ima dodeljeno kakšno drugo funkcijo, je pisanje v IOSET register brez pomena.
(naslov: IOSET0 - 0xE002 8004 in IOSET1 - 0xE002 8014)
IO0DIR – Registra se uporabljata za določitev smeri vhodno-izhodne linije, kadar je ta konfigurirana kot GPIO. Logična ’0’ pomeni, da je linija konfigurirana kot vhod; logična ’1’ pa nastavi linijo kot izhod. Napetostni nivo na liniji vrat določamo s pisanjem v registra IOSET in IOCLR. Stanje na vratih je rezultat zadnjega vpisa.
(naslov IODIR0 - 0xE002 8008 in IODIR1 - 0xE002 8018)
IO0CLR – Kadar so linije na vratih nastavljene kot GPIO izhodi, jih je mogoče v nizko logično stanje postaviti s pisanjem v ta dva registra. Z vpisom logične ’1’ v IOCLR register resetiramo pripadajoči bit IOSET registru in postavimo izhod v nizko stanje. Če je linija konfigurirana kot GPIO vhod ali ima dodeljeno kakšno drugo funkcijo je pisanje v IOCLR register brez pomena.
(naslov: IOCLR0 - 0xE002 800C in IOCLR1 - 0xE002801C)

Vaja2: Uporaba zbirnika

Napišite del programa v zbirniku, ki bo v dološenih šasovnih intervalih prižigal in ugašal svetlečo diodo. Interval zakasnitve naj bo dolg približno eno sekundo. LED diodo , ki je priključena na splošne vhodno-izhodne linije na razvojni plošči, izberite sami.

Potrebne datoteke:

• crt0.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)
• LCD.h (datoteka z ukazi za upravljanje LCD prikazovalnika)
• LCD.c (datoteka s funkcijami za delo z LCD prikazovalnikom)

Program napisan v zbirnem jeziku, ki na portu P0.12 priziga in ugasa LEDico

```
PINSEL0 = 0xE002C000
  IO0DIR = 0xE0028008
  IO0SET = 0xE0028004
  IO0CLR = 0xE002800C
  IO0PIN = 0xE0028000

.text
        .align  2
        .global _start
_start:
    bal ResetHandler    //;for debug
    bal .   //;handlerUndef
    bal .   //;SWI interrupt handler
    bal .   //;handlerPrefetchAbort
    bal .   //;handlerReserved
    bal .   //;handlerDataAbort
    bal .   //;handlerReserved
    bal .   //;handlerIRQ
    bal .   //; Remake to save FIQ access time.....

ResetHandler:

ldr  r0, =PINSEL0
ldr  r1, =0x0      // PINSELO
str r1,[r0]        //na naslov PINSEL0 shrani vrednost 0x0

ldr  r0, =IO0DIR
ldr  r1, =0xF00    //   nastavimo IO0DIR na mestu diod na 1 (izhod)
str r1,[r0]

ldr r3, =0x100

loop:

  ldr  r0, =IO0CLR
  ldr  r1, =0xF00    //   IO0CLR postavimo diode na 0 (ugasnemo jih ce so prižgane)
  str r1,[r0]

  ldr r2, =500000
  zak2:
    nop                // ne naredi nič
```

```
  SUBS R2,R2, # 1   // R2 = R2 - 1
 BNE zak2            // preveri če je rezultat nič, če ni se vrne na "zak2"

 ldr  r0, =IO0SET   //   prižigamo LEDice
 str r3,[r0]



 add r3,r3,r3
 subs r4,r3,#0x1000
 bne skok
  ldr r3,=0x100
 skok:

 ldr r2, =500000
  zak1:
  nop               // ne naredi nic
  SUBS R2,R2, # 1     // odšteje od R2 vrednost 1, preveri če je rezultat 0
 BNE zak1   //če rezultat ni 0, gre na zak1

bal loop
       .align  2
.bss
```

Nekaj ukazov:
LDR: naloži vrednost v register
STR shrani vrednost v register
BNE – skoči, če ni enako (ang. branch if not equal).
BEQ Enako Rezultat primerjave je enako ali nič

Vaja3: Uporaba LCD-ja

Napišite funkciji, ki bosta služili za pogon LCD prikazovalnika. Prva funkcija naj vrši pošiljanje ukazov, druga pa naj omogoča izpis teksta na prikazovalnik.
Potrebne datoteke:
• crt0.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)
• LCD.h (datoteka z ukazi za upravljanje LCD prikazovalnika)
• LCD.c (datoteka s funkcijami za delo z LCD prikazovalnikom)
• ADC.c (datoteka s funkcijami za delo z A/D pretvornikom)

VAJA3.c

```
#include "LPC2138.h"
#include "LCD.h"


void main(void)
{

  __LCD_init();   //inicializacija LCDja

  LCD_cmd(Cls);       //klic funkcije LCD_cmd
  waitms(2);                  //zakasnitev 2ms
  LCD_cmd(CoursorOff);  //ugasnemo kurzor na LCDju
  waitms(2);


  LCD_locate(1,1);            //vaja 3 zapšemo na LCD v prvo vrstico in prvi stolpec
  LCD_chr("vaja 3");

  LCD_locate(2,6);
  LCD_var(2306);
  waitms(100);

  LCD_var(0);

  int smer = 1;
  int i;
  while(1)
  {
    int i;

   for(i=0; i<10; i++)
   {
     if(smer == 1)
     {
      waitms(200);
      LCD_cmd(ShiftR);
```

```
      waitms(200);
     }

     if(smer == -1)
     {
       waitms(200);
       LCD_cmd(ShiftL);
       waitms(200);
     }
   }
   smer = smer * (-1);
   waitms(2);
  }



  //LCD_chr("ALEŠ");

  //LCD_cifra(12);
```

LCD.c

```
#include "LPC2138.h"
#include "LCD.h"

void __LCD_init(void)
{

  PINSEL0 = 0x0;
  IO0DIR |= (1<<D4)|(1<<D5)|(1<<D6)|(1<<D7)|(1<<RS)|(1<<E)|(1<<RW); // define LCD-Pins as
outputs
  LCD_port(0);
  waitms(40);
  reset(RS);
  set(D4); // init!
  set(D5);
  E_pulse();
  waitms(6);
  E_pulse(); // init!
  waitms(2);
  E_pulse(); // init!
  waitms(5);
  reset(D4); // set 4 bit mode
  E_pulse();
  LCD_cmd(0x328); // 4 bit mode, 1/16 duty, 5x8 font, 2lines
  LCD_cmd(0x0f); // display on
  LCD_cmd(0x06); // entry mode advance cursor
  LCD_cmd(0x01); // clear display and reset cursor
  waitms(1);
}

void LCD_cmd(int cmd)
{

  int pv;
```

```
  if(cmd>0xFF)
    cmd=cmd&0xFF;
  reset(RS);

  pv = cmd>>4;
  LCD_port(pv);
  E_pulse();
  pv = cmd & 0xF;
  LCD_port(pv);
  E_pulse();
  waitms(2);

}

void LCD_var(int var)
{

  int i, a, nic;
  nic = 0;

  for(i=9; i>=0; i--)
  {
    a = 0;
    while(var >= desetX(i))
    {
      var = var - desetX(i);
      a++;

    }
    if(a != 0)
      nic = 1;

    if(a != 0 || nic )
      LCD_cifra(a);

    if(i == 0 && nic == 0 && a==0)
      LCD_cifra(0);
  }


  /*int pv;

  if(var>0xFF)
    var = var & 0xFF;
  set(RS);

  pv = var>>4;
  LCD_port(pv);
  E_pulse();
  pv = var & 0xF;
  LCD_port(pv);
  E_pulse();
  waitms(2); */
```

```
}

int desetX(int x)        // vrne potenco števila (oziroma 10 na to potenco)
{
   int i = 0;
   int rez = 1;
   while(i < x)
   {
    rez = rez*10;
    i++;
   }
   return rez;
}
void LCD_cifra(int var)
{
  if (var<0) var=0;
  if (var>9) var=9;
  set(RS);
  LCD_port(0x3);    // pošljemo prve 4 bite (ki so vedno 0x3, ker so cifre med 0x30 in 0x39)
  E_pulse();
  LCD_port(var);    // pošljemo zadne 4 bite (ki so kar cifra ki jo vpišemo)
  E_pulse();
}
void LCD_chr(char data[])
{
  int i;
  int pv;
  set(RS);

  for(i=0; data[i]!=0; i++)
  {

   pv = data[i]>>4;
   LCD_port(pv);
   E_pulse();
   pv = data[i] & 0xF;
   LCD_port(pv);
    E_pulse();
    waitms(2);
   }
}
void LCD_locate(int row, int column)
{
   if (row == 1)
    LCD_cmd(0x80 | (column-1)&0x0f);

   if (row == 2)
    LCD_cmd(0xc0 | (column-1)&0x0f);
}
void LCD_port(int port_value)
{
if(port_value>0xF)              //port value je res 4 bitno
  port_value = 0xF & port_value;

IO0SET = port_value << 16;
IO0CLR = (~port_value & 0xF) << 16 ;
```

```
}


void E_pulse(void)

  set(E);
  waitms(2);
  reset(E);
  waitms(2);

}
void waitms(int ms)
{
  volatile int i,j;
  for (i=0;i<ms;i++)
    for (j=0;j<200;j++);
}
void set(int pin)
{
  IO0SET =0x1 << pin;
}
void reset(int pin)
{
   IO0CLR =0x1 << pin;
}
```

LCD.h

```
//LCD control lines
#define D4 16
#define D5 17
#define D6 18
#define D7 19
#define RS 20
#define E 21
#define RW 22//LCD commands
#define Cls 0x01
#define CoursorOn 0x0e
#define CoursorOff 0x0c
#define CoursorBlinkOn 0x0f
#define ShiftL 0x18
#define ShiftR 0x1c
void __LCD_init(void);
void LCD_cmd(int cmd);
void LCD_var(int var);
int desetX(int x);
void LCD_cifra(int var);
void LCD_chr(char data[]);
void LCD_locate(int row, int column);
void LCD_port(int port_value);
void E_pulse(void);
void waitms(int ms);
void set(int pin);
void reset(int pin);
```

Vaja 4: IRQ prekinitve

Z uporabo A/D pretvornika vgrajenega v mikrokrmilnik izdelajte voltmeter, ki bo meril napetost na potenciometru razvojne plošče. Rezultat merjenja prikažite na LCD prikazovalnik.

Potrebne datoteke:
• crt0.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)
• LCD.h (datoteka z ukazi za upravljanje LCD prikazovalnika)
• LCD.c (datoteka s funkcijami za delo z LCD prikazovalnikom)
• ADC.c (datoteka s funkcijami za delo z A/D pretvornikom)

```
#include "LPC2138.h"
#include "LCD.h"
#include "ADC.h"

#define reference 3264

int i;

void prekinitvenaFunkcija(void) __attribute__((interrupt("IRQ"))); // deklaracija nato pa povemo da se bo uporabljala kot prekinitev
void prekinitvenaFunkcija(void)
{
  LCD_cmd(Cls);
  waitms(1000);
  LCD_chr("V PREKINITVI");
  waitms(500);
  LCD_cmd(Cls);
  waitms(1000);

  for(i=0;i<6;i++)
  {
    IO0SET = 0xF00;
    waitms(2000);
    IO0CLR = 0xF00;
    waitms(5);
    LCD_locate(1,1);
    LCD_chr("Korak");
    LCD_locate(2,10);
    LCD_var(i);
  }

  LCD_cmd(Cls);
  waitms(1000);

  EXTINT = 0x4;
  VICVectAddr = 0x0;   // s tem ukazom  resetiramo prekinitev
```

```
}

void main(void) {



  //MAMCR = 0x02;              // MAM functions fully enabled
  //MAMTIM = 0x03;             // MAM fetch cycles are 3 CCLKs in duration
  //__PLL_init();              // boost crystal's 12 MHz up to 60 MHz
  int vol;
  __LCD_init();              // Inicializacija LCD-ja
  __ADC_init();   //inicializacija A/D pretvornika


  PINSEL0 = 0x0;
  PINSEL0 =(1<<31);
  IO0DIR |= 0xF00;
  IO0CLR = 0xF00;
  VICIntSelect = 0x0;
  EXTMODE = 0x0;
  VICVectCntl0 = 0x30;
  VICVectAddr0 = (unsigned long)prekinitvenaFunkcija;
  VICIntEnable = (1<<16);


  LCD_cmd(CursorOff);
  LCD_locate(1,1);
  LCD_chr("vaja 4");
  LCD_locate(2,1);
  LCD_chr("A/D pretvornik");
  waitms(2000);
  LCD_cmd(Cls);
  LCD_chr("V");

  //LCD_chr(ADC_read ());

  while(1)
  {
  LCD_locate(1,1);
  LCD_chr("Napetost");
  LCD_locate(2,11);
     vol = ADC_read()*3196/1023;
   voltage2LCD(vol) ;

  }
}
```

LCD.c

```
#include "LPC2138.h"
#include "LCD.h"

void __LCD_init(void)
{
```

```
  IO0DIR |= (1<<D4)|(1<<D5)|(1<<D6)|(1<<D7)|(1<<RS)|(1<<E)|(1<<RW); // define LCD-Pins as outputs
  LCD_port(0);
  waitms(40);
  reset(RS);
  set(D4); // init!
  set(D5);
  E_pulse();
  waitms(6);
  E_pulse(); // init!
  waitms(2);
  E_pulse(); // init!
  waitms(5);
  reset(D4); // set 4 bit mode
  E_pulse();
  LCD_cmd(0x28); // 4 bit mode, 1/16 duty, 5x8 font, 2lines
  LCD_cmd(0x0f); // display on
  LCD_cmd(0x06); // entry mode advance cursor
  LCD_cmd(0x01); // clear display and reset cursor
  waitms(1);
}

void LCD_cmd(int cmd)
{

  int pv;

  reset(RS); // LCD command mode

  pv = cmd>>4;
  LCD_port(pv); // high bits
  E_pulse();
  pv = cmd&0x0f;
  LCD_port(pv); // low bits
  E_pulse();
  waitms(1);
}

void LCD_var(int var)
{
  int a, i,set;
  set = 0;
  for(i = 9; i >= 0; i--)
  {
    a = 0;
    while(var >= desetX(i))
    {
      var -= desetX(i);
      a++;
    }
    if(a != 0)
    {
      set = 1;
  }
  if(a != 0 || set)
  {
```

```
    LCD_vars(a);
   }
   if(a == 0 && i == 0 && set == 0)
   {
    LCD_vars(a);
   }
  }

  waitms(5);
}

int desetX(int x)
{
  int i = 0;
  int rez = 1;
  while(i < x)
  {
    rez *= 10;
    i++;
  }
  return rez;
}

void LCD_vars(int var)
{
  set(RS); // LCD data mode
  LCD_port((0x30|var)>>4); // high bits | ASCII mask
  E_pulse();
  LCD_port((0x30|var)&0x0f); // low bits | ASCII mask
  E_pulse();
}

void LCD_chr(char data[])
{
  int i;
  int pv;
  int ch;
  set(RS); // LCD data mode
  for(i = 0; data[i] != 0; i++)
  {
    ch=data[i];
    pv=ch>>4;
    LCD_port(pv); // high bits
    E_pulse();
    pv=ch&0x0f;
    LCD_port(pv); // low bits
    E_pulse();
  }

  waitms(5);
}

void LCD_locate(int row, int column)
{
  reset(RS);
  if(row == 1)
```

```
  {
    LCD_cmd(0x80 | (column - 1)&0x0f);
  }
  else
  {
    LCD_cmd(0xc0 | (column - 1)&0x0f);
  }

  waitms(5);
}

void LCD_port(int port_value)
{
  if(port_value > 0x0f ) //LCD port is only 4-bit long
    port_value = port_value&0x0f; //we don't want to mess other bits
  IO0SET = port_value << 16;
  IO0CLR = (~port_value&0x0f) << 16;
}

void E_pulse(void)
{
  set(E);
  int a;
  for(a=0;a<200;a++); //delay app. 50 us
  reset(E);
  for(a=0;a<200;a++); //delay app. 50 us
}

void waitms(int ms)
{
  volatile int i,j;
  for (i=0;i<ms;i++)
    for (j=0;j<170;j++);
}

void set(int pin)
{
  IO0SET = (1<<pin);
}

void reset(int pin)
{
  IO0CLR = (1<<pin);
}
```

ADC.c

```
#include "LPC2138.h"

void __ADC_init() {

 PINSEL1 |= 0x01<<22;   // P0.27 --> pina 23:22 --> 0:1 --> konfiguracija A/D pretvornika

}
int ADC_read (void) { /* vrača vrednosti od 0...1023 (10 bitni rezultat)  Get ADC Value and set PWM */

   int val, adcr;
```

```
  adcr = (1<<24) | (1<<21) ;  //postavimo 24  bit na 1 (START) in 21 bit na 1(PDN pomeni pretvorba)
  AD0CR |=adcr;   // je enako kot -->   AD0CR=AD0CR | adcr ;
  PCONP &=~(1<<12);   // pišemo ničlo na dvanajsti bit
  do
  {
    val = AD0DR;
  }while( (val &(1<<31))==0 ); // preverimo če je bit 31 na 1, kar pomeni pretvorba končana

  val = (val >> 6) & 0x3FF;
  PCONP |= (1 << 12);   // izklop napajanja A/D pretvornika
  AD0CR &=~adcr;        // ustavimo

  return val;

}
```

Ukaza:
&=~(...) **pišemo ničlo na določeno mesto**
|=(...) **pišemo enico na določeno mesto**

Vaja 5: Uporaba A/D pretvornika

Napišite program, ki bo upravljal z UART serijskim vmesnikom mikrokrmilnika. Z njim pošljite izbran niz znakov na terminal osebnega računalnika.
Potrebne datoteke:
• crt1.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)

VAJA5.c

```
#include "LPC2138.h"
#include "LCD.h"

#define reference 3264
void voltage2LCD(a);
void voltage2UART(a);

#define TEMT (1<<6)
#define LINE_FEED 0xA
#define CARRIAGE_RET 0xD

int vol;

void main(void)
{
  //MAMCR = 0x02;              // MAM functions fully enabled
  //MAMTIM = 0x03;             // MAM fetch cycles are 3 CCLKs in duration
  //__PLL_init();              // boost crystal's 12 MHz up to 60 MHz

  __LCD_init();                // Inicializacija LCD-ja
  __ADC_init();                // Inicializacija ADC-ja

  __UART1_init();
```

```
  LCD_cmd(Cls);
  LCD_locate(1,1);
  LCD_chr("UPORABA");
  LCD_locate(2,1);
  LCD_chr("UART komunikacije");
  waitms(1000);

 LCD_cmd(Cls);              // Glavni meni
 LCD_locate(1,1);
 LCD_chr("Napetost: P0.27");
 LCD_locate(2,11);
 LCD_chr("V");

 while(1)                           // Glavna zanka programa
 {                                  // Vsakih 100ms beremo ADC
   vol = (reference*ADC_read())/1023;   // in vrednost prikažemo na LCD
   voltage2LCD(vol);
   voltage2UART(vol);

   //UART_read();   //branje porta

   LCD_locate(2,1);

   LCD_znak( UART_read() );
   waitms(1000);
 }

}

void voltage2LCD(a)                // Pretvorba napetosti za prikaz na LCD-ju
{
   LCD_locate(2,5);
   int edst,i;                  //enice, desetice, stotice, tisocice
   for(i = 3; i >= 0; i--)
   {
     edst = 0;
     while(a > desetX(i))
     {
       a -= desetX(i);
       edst++;
     }
     if(i == 2)
     {
       LCD_chr(".");              // postavimo decimalno piko
     }
     if(edst == 0)
     {
       LCD_chr("0");
     }
     else
     {
       LCD_var(edst);
     }
   }
}
```

```
void voltage2UART(a)                // Pretvorba napetosti za prikaz na LCD-ju
{
   int edst,i;                  //enice, desetice, stotice, tisocice
   for(i = 3; i >= 0; i--)
   {
     edst = 0;
     while(a > desetX(i))
     {
       a -= desetX(i);
       edst++;
     }
     if(i == 2)
     {
       UART_send('.');              // postavimo decimalno piko
     }
     if(edst == 0)
     {
       UART_send('0');
     }
     else
     {
       UART_send(edst+48);
     }
   }

   UART_send(LINE_FEED);

}
```

UART.c

```
#include "LPC2138.h"

void __UART1_init()
{
  PINSEL0 = 0x50000;

  U1FCR = 0x7;       // resetiramo fifo register
  U1LCR = 0x83;      //določimo parametre komunikacije

  U1DLL = 0x14;      // določimo hitrost komunikacije
  U1DLM = 0x0;       // določimo hitrost komunikacije

  U1LCR = 0x3;       //določimo da lahko dostopamo do registrov
}
/* Funkcija, ki piˇse na serijska vrata. */
int UART_send(int ch)
{
   while((U1LSR >> 6) & 0x1 == 0x0);
   return (U1THR = ch);     //register za pošiljanje
}
/* Funkcija, ki bere iz serijskih vrat. */
int UART_read(void)
{
  while((U1LSR & 0x1) == 0x0) ;  //če se pojavi nov podatek gremo v zanko
```

```
   return (U1RBR) ;
}
```

Vaja 6: Uporaba PWM izhoda – krmiljenje enosmernega motorja

Napišite program, ki bo preko PWM modula krmilil enosmernimotor. V uporabniškem vmesniku naj bo s tipkami mogoče nastaviti frekvenco vrtenja motorja.

Potrebne datoteke:
• crt1.s (zagonska datoteka)
• LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)
• LCD.h (datoteka z ukazi za upravljanje LCD prikazovalnika)
• KEY.h (datoteka z definicijami tipk)
• MotorBoard.h (datoteka z definicijami linij priklju˘cenih na MotorPCB)
• LCD.c (datoteka s funkcijami za delo z LCD prikazovalnikom)
• KEY.c (datoteka s funkcijami za delo s tipkami na razvojni plošči ITLPC2138)
• PWM.c (datoteka s funkcijami za delo s PWM modulom)

VAJA_PWM.c

```
#include "LPC2138.h"
#include "LCD.h"

#include "MotorBoard.h"

#define   DIR   1
void LCD_ShowDuty(void);
void PWM_out(int val);

int duty = 0;

int main(void)
{

  //MAMCR = 0x02;              // MAM functions fully enabled
  //MAMTIM = 0x03;             // MAM fetch cycles are 3 CCLKs in duration
  //__PLL_init();              // boost crystal's 12 MHz up to 60 MHz

  //HCTL2022 init

  IO0DIR |= (1<<DIR) | (1<<OE) | (1<<RST) | (1<<SEL1) | (1<<SEL2) ;
  IO0SET = (1<<OE);
  IO0CLR = (1<<DIR);


  __LCD_init();                // inicializacija LCDja
  __PWM_init();          //inicializacija  PWM
  __ADC_init();          // inicializacija ADC

  LCD_cmd(Cls);                // uvodni pozdrav na LCDju
```

```
  LCD_cmd(CursorOff);
  LCD_chr("Vaja 7  Uporaba ");
  LCD_locate(2,1);
  LCD_chr("   PWM izhoda   ");
  waitms(2000);

  LCD_cmd(Cls);
  LCD_chr("PWM duty cycle  ");
  LCD_locate(2,1);
  LCD_chr(" -    100 %   + ");

  LCD_ShowDuty();

  while(1)
  {
    duty = (100*ADC_read())/1023;
    if (duty > 100)
       duty = 100;
    if (duty < 0)
       duty = 0;

     LCD_ShowDuty();     //prikaže vrednost duty
     PWM_out(duty);
  }

}

void LCD_ShowDuty()
{
    LCD_locate(2,7);
    if(duty < 100)
      LCD_chr(" ");
    if(duty < 10)
      LCD_chr(" ");
    LCD_var(duty);
}

void PWM_out(int val)
{

    if (val < 0)
      val = 0;
    if (val > 100)
      val = 100;

    val = val*15;

    PWMMR1 = val;
    PWMLER = 0x2;

}
```

PWM.c

```
#include "LPC2138.h"
#include "PWM.h"
```

```
void __PWM_init (void)
{
//  VICVectAddr8 = (unsigned)PWM0_isr;        /* Set the PWM ISR vector address */
//  VICVectCntl8 = 0x00000028;                /* Set channel */
//  VICIntEnable = 0x00000100;                /* Enable the interrupt */

//  VPBDIV = 0x01;


   PINSEL0 |= 0x2;
   PCONP |= (1<<5);   //PWM vklop napajanja

   PWMPR =0x1;       //delilno razmerje glavne ure in PWM ure. če damo na 1 bosta tekli enako.
   PWMPCR =0x200;      //PWM1 output enabled
   PWMMCR = 0x3;

   PWMMR0 = 1500;       // dolžina cikla/ periode
   PWMMR1 = 0;          // določa dolžino pulza, od 0 do PWMMR0

   PWMLER = 0x3;
   PWMTCR = 0x02;
   PWMTCR = 0x09;
```

Vaja 7: Uporaba PWM izhoda - regulacija enosmernega motorja

Dopolnite program iz prejšnjega primera tako da bo izvajal zaprtozančno regulacijo hitrosti vrtenja enosmernega motorja.

Potrebne datoteke:

- crt1.s (zagonska datoteka)
- LPC2138.h (datoteka z naslovi vseh registrov mikrokrmilnika LPC2138)
- LCD.h (datoteka z ukazi za upravljanje LCD prikazovalnika)
- MotorBoard.h (datoteka z definicijami linij priključenih na MotorPCB)
- LCD.c (datoteka s funkcijami za delo z LCD prikazovalnikom)
- PWM.c (datoteka s funkcijami za delo s PWM modulom)

VAJA_PWMreg.c

```
#include "LPC2138.h"
#include "LCD.h"
#include "MotorBoard.h"

#define     DIR     1       //smer vrtenja motorja
#define     K       2
#define     DT      100
signed long duty = 0;
signed long duty_reg = 0;
void PWM_out(signed long val);
void __HCTL2022_init(void);
unsigned long ReadHCTL(void);
void ResetHCTL(void);
void delay(void);
signed long rezultat = 0;
signed long referenca = 0;
signed long e = 0;
```

```
int main(void)
{
  IO0DIR |= 1<<DIR;
  IO0SET = 1<<DIR;
  IO0CLR = 1<<DIR;

  __PWM_init();              //inicializacija PWM linij
  __HCTL2022_init();         //inicializacija HCTL

 duty = 25;                  //motor deluje s 25% moči
 PWM_out(duty);
 waitms(1000);               //zakasnitev 1s, da motor pride do polnih vrtljejev

 ResetHCTL();
 waitms(DT);
 referenca = (signed long) (ReadHCTL())&0xffff;

  while(1)
  {
  ResetHCTL();
  waitms(DT);
  rezultat = (signed long) (ReadHCTL())&0xffff;
  e = referenca - rezultat;
  duty_reg = duty - e/DT*K;
  PWM_out(duty_reg);

  }
}

unsigned long ReadHCTL(void)
{
  unsigned long  ReturnData = 0;  //enable HCTL2022 IO lines
  IOCLR0 = (1<<OE);

  delay();

   //set MSB byte
  IOCLR0 = (1<<SEL1); IOSET0 = (1<<SEL2);
  //minimal delay of 50ns
  delay();
  //Read MSB byte and shif it up 16 times
  ReturnData = ((IOPIN0&0xff00)<<16);

  //set 2nd byte
  IOSET0 = (1<<SEL1); IOSET0 = (1<<SEL2);
  //minimal delay of 50ns
  delay();
  //Read 2nd byte shif it up 8 times and add previous result
  ReturnData += ((IOPIN0&0xff00)<<8);

  //set 3rd byte
  IOCLR0 = (1<<SEL1); IOCLR0 = (1<<SEL2);
  //minimal delay of 50ns
  delay();
  //Read 3nd byte and add previous result
```

```
  ReturnData += (IOPIN0&0xff00);

  //set LSB byte
  IOSET0 = (1<<SEL1); IOCLR0 = (1<<SEL2);
  //minimal delay of 50ns
  delay();
  //Read LSB byte shifti down 8 times and add previous result
  ReturnData += ((IOPIN0&0xff00)>>8);


  //HCTL2022 lines to high Z state
  IOSET0 = (1<<OE);

  return ReturnData;
}

void __HCTL2022_init(void)
{

  IODIR0 |= (1<<OE)|(1<<RST)|(1<<SEL1)|(1<<SEL2); //set selected pins as outputs
  IODIR0 &=~(1<<UD);
  IODIR0 &=~0xff00;


  ResetHCTL();  //HCTL2022 reset

}
void ResetHCTL(void)
{
  //enable HCTL2022 IO lines
  IOCLR0 = (1<<OE);
  delay();
  IOCLR0 = (1<<RST);
  delay();
  delay();
  IOSET0 = (1<<RST);

  // HCTL2022 lines to high Z state
  IOSET0 = (1<<OE);
}
void PWM_out(signed long val)
{
    val *= 10;

    if(val < 0)
      val = 0;
    if(val > 1000)
      val = 1000;

    PWMMR1 = val;              // set PWM duty cycle
    PWMLER = 0x2;              // enable shadow latch for match1
}
// short delay
void delay(void)
{
  //waitms(1);
```

```
  int j = 5;
  while(j--);
}
```