30021 Digital Instrumenteing
Gruppe 5

Temperaturmåler med Datalogger

Skrevet af

Kim Kofoed Nielsen s103624

Thomas Vintersborg s103632

D. 21. April 2013

Introduktion

Der er blevet givet 3 opaver, som er udført på en PIC16F887. Målet med disse opgaver er at opsætte en Temperatur logger der gemmer en digitalmåling samt analog måling på en EEPROM. Derudover skal disse målinger have et tids stamp for hvornår de er foretaget. Derfor skal der opsættes en I2C der styre en digital temperatursensor, EEPROM og Real Time Clock. Derudover skal der laves en bruger interface der styres fra computeren via seriel kommunikation, fra denne interface skal man kunne gå til menuen, indstille tid og dato på RTC, indstille tidsinterval imellem logning af målinger, starte logning, stoppe logning, læse log fra EEPROM og slette EEPROM data.

Problem formulering

Der skal designes en datalogger, der måler temperaturen i et rum, med en digital og en analog komponent. Disse målinger skal kunne gemmes i en extern EEPROM og skal kunnes læses fra igen. Dette skal kunne være styret af brugeren af programmet.

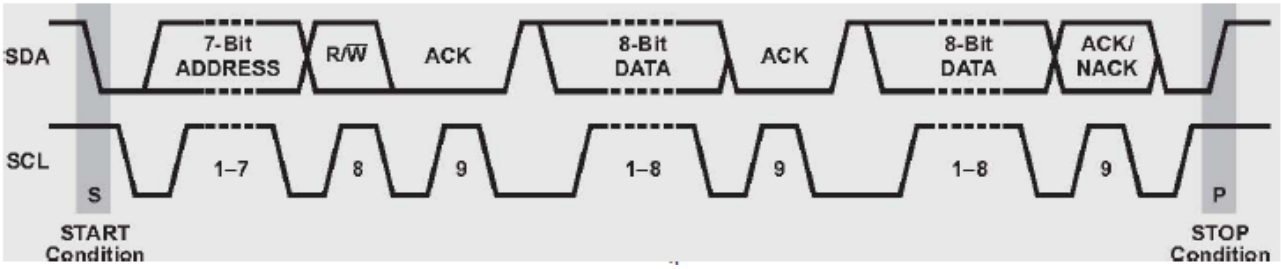
Dataloggeren skal kunne starte ved blot at sætte strøm til.

Problem begrænsning

Teori

I2C-kommunikation

I2C-kommunikation anvendes til at styre flere måleenheder. Den består af 2 linjer SDA(data linje) og SCL(Clock linje). For at I2C skal kunne fungere vælges en Master (mikrocontrolleren) samt slaves (måleenhederne). Masteren står for at styre SCL samt hvilke slaves skal være aktive og hvornår de skal sende data, dette gøres ved at give de forskellige slaves individuelle adresser som masteren så kalder når den skal kommunikere med den valgte slave. Protokollen starter med en startsekvens (SDA høj->lav mens SCL høj) hvorefter masteren sender den adresse som der ønskes at kommunikere med ud på SDA efterfulgt af enten et read eller write bit, der bestemmer om der skal skrives eller læses fra slaven. Herefter "frigøre" masteren SDA så slaven kan sende data på denne og venter på slaven skal ACK(acknowledge - godkende) at den har modtaget/forstået informationen, dette gøres ved at slaven sætter SDA lav i 9 SCL-clock perioder. Herefter vil masteren enten overtage SDA igen for at sende data hvis dette var ønsket eller lytte på SDA for at modtage den ønskede data. Efter hvert bit sendt/modtaget vil der blive sendt et ACK fra modparten for at sikre dataen er modtaget korrekt. Når masteren har udført de handlinger den skal og skal stoppe kommunikationen sender den en NAK(Not ACK) hvis den læste fra slaven hvorefter den sender en stopsekvens(SDA lav->høj mens SCL høj) eller hvis den sender data til slaven sender den kun stopsekvensen.



Hardware

Microkontroller PIC16F887

Micorkontrolleren styre I2C forbindelsen, ADC konverteringen af analog temperatur måling, seriel kommunikation med PC og initialisering af komponenter.

Anvendte porte

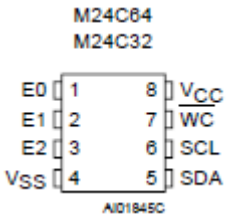
Forbundet Fra:	Forbundet Til:
PIC16F , pin 18 SCL	LM73 , pin 4 SMBCLK
	M24C64 , pin 6 SCL
	DS1340C , pin 1 SCL
PIC16F, pin 23 SDA	LM73 , pin 6 SMBDAT
	M24C64 , pin 5 SDA
	DS1340C , pin 16 SDA
PIC16F , pin 25 TX	FTDI, RX
PIC16F , pin 26 RX	FTDI, TX

Porte brugt til LCD skærm er ikke vist.

EEPROM M24C64

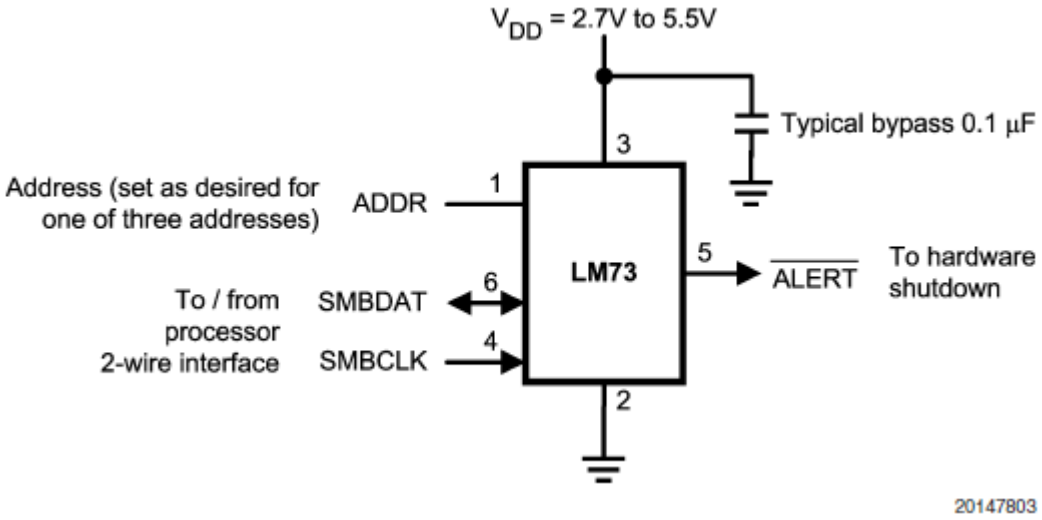
M24C64 er en EEPROM der anvendes til at logge de data som der bliver fortaget af temperaturmålerne samt RTC'en. EEPROM'en styres ved hjælp af en to lednings I2C seriel interface og kan rumme 8192 x 8bit data.

E0, E1, E2	Chip Enable
SDA	Serial Data
SCL	Serial Clock
$\overline{WC}$	Write Control
V_{CC}	Supply Voltage
V_{SS}	Ground



LM73

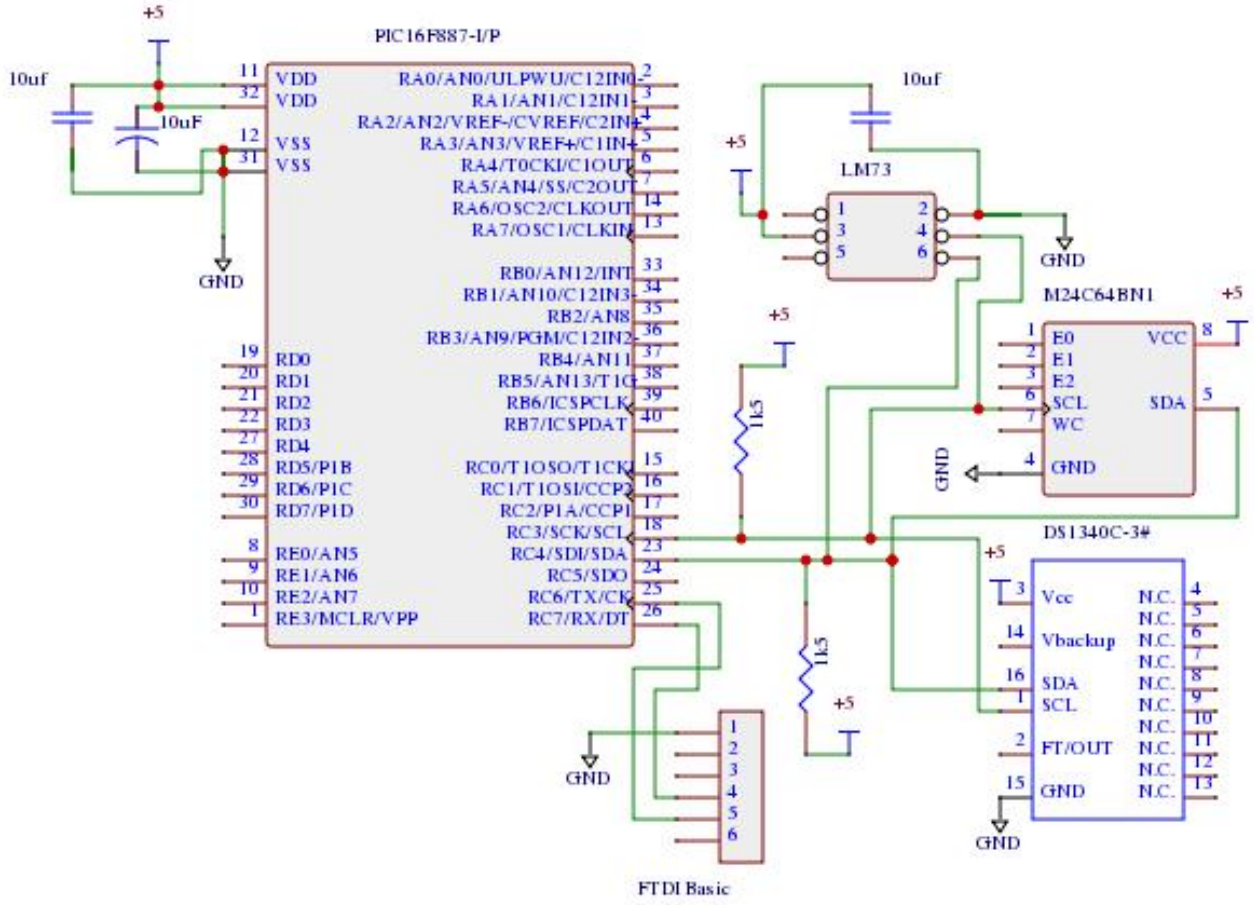
LM73 er en digital temperatur måler som anvendes til at optage de digitale målinger. Denne sensor kan indstille følsomheden på temperaturmålinger ved at levere 11 eller 14 bit hvilket resultere i enten 0.25°C/LSB eller 0.03125°C/LSB, der anvendes 14 bit til vores opgave. Derudover anvender LM73 også en SMBus og en I2Ckompatipel to lednings interface som der bruges af mikrokontrolleren for at styre flere digitale komponenter på en gang



LM35

LM35 er en analog temperatur måler som anvendes til at optage de analog målinger for projektet. Denne sensor måler lineært med +10mV/°C og kan fungere i temperaturer fra -55 til +150°C, med en præcision på ±0.75°C

Samlet Diagram



Test of software

Softwaren er testet med en udskrift fra terminalen. En udskrift ses nedenunder.

M // 'm' blev trykket og menuen popper op på skærmen
menu
Press d for setting the date
Press t for setting logging time
press l for logging data
press s for stopping the data logging
press r for reading the logged data to the eeprom
press e for erasing the logged data.
T // 't' blev trykket og bruger skal nu vælge en værdi mellem 0-99 sekunder.
Set logging interval
03 // brugeren vælger 3 sekunder
Log interval is: 3.00
L // 'l' blev trykket, brugeren har nu valgt at logge temperature

Når brugeren vælger logging, kommer dataen ud på følgende format:

Add; Addressen i epprommen, H; time tal, M; minut tal, S; sekund tal, Digital_temp; den digitale temperatur, Analog_temp, den analoge temperatur.

Logging

add: 10 H: 0 M: 20 S: 58	Digital_temp: 24.07	Analog_temp: 0.00
add: 20 H: 0 M: 21 S: 2	Digital_temp: 24.09	Analog_temp: 0.00
add: 30 H: 0 M: 21 S: 5	Digital_temp: 24.09	Analog_temp: 0.00
add: 40 H: 0 M: 21 S: 9	Digital_temp: 24.14	Analog_temp: 0.00
add: 50 H: 0 M: 21 S: 12	Digital_temp: 24.12	Analog_temp: 0.00
add: 60 H: 0 M: 21 S: 16	Digital_temp: 24.12	Analog_temp: 0.00
add: 70 H: 0 M: 21 S: 19	Digital_temp: 24.12	Analog_temp: 0.00
add: 80 H: 0 M: 21 S: 23	Digital_temp: 24.09	Analog_temp: 0.00
add: 90 H: 0 M: 21 S: 26	Digital_temp: 24.12	Analog_temp: 0.00
add: 100 H: 0 M: 21 S: 30	Digital_temp: 24.12	Analog_temp: 0.00
add: 110 H: 0 M: 21 S: 33	Digital_temp: 24.09	Analog_temp: 0.00
add: 120 H: 0 M: 21 S: 37	Digital_temp: 24.12	Analog_temp: 0.00
add: 130 H: 0 M: 21 S: 40	Digital_temp: 24.15	Analog_temp: 0.00
add: 140 H: 0 M: 21 S: 44	Digital_temp: 24.12	Analog_temp: 0.00

s // 's' er blevet trykket, for at stoppe logningen
Stopping logging
R // 'r' er blevet trykket for at læse dataen fra EEPROM'EN
Reading the log

H: 0 M: 20 S: 58	Digital: 24.09	Analog: 0
H: 0 M: 21 S: 2	Digital: 24.09	Analog: 0
H: 0 M: 21 S: 5	Digital: 24.09	Analog: 0
H: 0 M: 21 S: 9	Digital: 24.15	Analog: 0
H: 0 M: 21 S: 12	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 16	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 19	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 23	Digital: 24.09	Analog: 0
H: 0 M: 21 S: 26	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 30	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 33	Digital: 24.09	Analog: 0
H: 0 M: 21 S: 37	Digital: 24.12	Analog: 0
H: 0 M: 21 S: 40	Digital: 24.15	Analog: 0

E // 'e' er blevet valgt for at slette dataen på EEPROM'en
Erasing the log

Konklusion

Der kan konkluderes at følgende funktioner fungere:

- Datalogning; Dataloggeren kan skrive data til EEPROM og læse derfra
- Temperatur; Dataloggeren kan læse temperaturen fra den digitale temperatur sensor, dog virker den analog komponent ikke og den er derfor fjernet fra systemet.

- Real tids ur: Dataloggeren kan logge læse fra uret og bruge læsningen som tidsstempel til loggningen. Dog når der skrives en dato og tid til dataloggeren vil den ikke modtage dette.
- Bruger interfacet; Virker og lader brugeren vælge ud fra menuen. Hvis brugeren dog ved et uheld kommer til at trykke på en tast, så stoppes logningen og dataloggeren gør så nu hvad brugeren har trykket på.

Appendix A; Kode

```
#include "test82.h"
#include "lcd16216.c"
#define GREEN_LED PIN_A0

// Register tabel
#define LM_write     0x90 //Fortæller registeret at der skal skrives til komponenten
#define LM_read      0x91 //Fortæller registeret at der skal læses fra komponenten
#define LM_Config    0x01 //adressen der peger på opsætning
#define LM_c_s       0x04 //adressen som fortæller om der skal læses

// Variabler
int i, digit,  log_interval, int_flag= 0;
int16 sec, sec_old, min, hour, day, year, month, dow;
char key;
int16 data,interval_int;
int8  high_byte, low_byte;
int16 temperature,reading=0, address = 0,eeprom_read_digital,eeprom_read_analog;
int16 eeprom_read_sec, eeprom_read_min, eeprom_read_hour;
float avg_temp_digital, temp_digital,temp_analog,adc_val,interval;


// Funktioner
void temp_read();
void avg_temp();
void adc_read();
void write_ext_eeprom(int16 address, BYTE data);
BYTE read_ext_eeprom (int16 address);
BYTE b2bcd(BYTE binary_value) ;
BYTE bcd2b(BYTE bcd_value) ;
void write_int16_ext_eeprom(int16 address, int16 data);
int16 read_int16_ext_eeprom(int16 address);
void set_date_time(BYTE day, BYTE month, BYTE year, BYTE dow, BYTE hour, BYTE min, BYTE sec) ;
void clock_date_read();
int get_num();
void set_date();
void ds1340_init();


#int_RDA
void keyin() //Interrupt der starter når PIC'en modtager en karakter i uarten.
{
  disable_interrupts(int_rda);
  key = getc(); //Lægger karakteren ned i en variable
  int_flag = 1; // Sætter interrupt flaget til høj
  putc(key); // udskriver karakteren til terminalen
  printf("\n");
  enable_interrupts(int_rda);
```

```
}


void main()
{
  int buf[17];
  int j;

  output_float(PIN_C3); // Sætter Pin C3 og Pin C4 til at
  output_float(PIN_C4);

  // Opsætning til ADC
  setup_vref(0xE2);
  setup_adc(ADC_CLOCK_DIV_32);
  setup_adc_ports(sAN7 | VSS_VDD);
  set_adc_channel(7);

  setup_spi(SPI_SS_DISABLED);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);// This device COMP currently not supported by the PICWizard
  //Setup_Oscillator parameter not selected from Intr Oscillator Config tab
  setup_oscillator(OSC_8MHz, 0);
  // TODO: USER CODE!!
  port_b_pullups (TRUE);       //enabler interna pullup modstand på alle b porte.
  enable_interrupts(global);
  enable_interrupts(int_rda);

  lcd_init();
  lcd_clear();

  // Tjekker på I2C bussen for brugte addresser og udskriver dem
 /* for(i=0; i<112; i++)
  {
    i2c_start();
    if( ! i2c_write(i<<1) )
    {
     //  printf("%X\n",i<<1);
    }
    i2c_stop();
  }*/


  delay_ms(1000);
          printf("press 'm' for menu \n");


  while(1)
  {
    if(int_flag)
    {
      int_flag = 0;
      switch(key)
      {
        case 'm':
          printf("menu \n");
```

```
    printf("Press d for setting the date \n");
    printf("Press t for setting logging time \n");
    printf("press l for logging data \n");
    printf("press s for stopping the data logging \n");
    printf("press r for reading the logged data to the eeprom \n");
    printf("press e for erasing the logged data. \n");
  break;
  case 'd':
    printf("Set Date\n");
    clock_date_read();
    set_date();
    printf("The date is: %lu - %lu - %lu",day, month,year);

  break;
  case 't':
  // denne case spørger brugern om
    printf("Set logging interval\n");
    log_interval = get_num()*10;
    log_interval += get_num();
    interval = (float)(log_interval) * 1000;
    interval_int = (int16)interval;
    printf("Log interval is: %3.2f \n",interval/1000);
  break;
  case 'l':
    printf("Logging \n");
     i2c_start();         // Starter kommunikation
      i2c_write(LM_write);  // Fortæller at der skal skrives til et register
      i2c_write(LM_c_s);    // får adgang til control/status register
      i2c_write(0x60);      // skriver til control/status registeret og
                    // opsætter til en temp opløsning på 13 bit + sign

      i2c_start();         // Starter kommunikatio
      i2c_write(LM_write);  // Fortæller at der skal skrives til et register
      i2c_write(LM_Config); // får adgang til Configuration register
      i2c_write(0x40);      // opstiller configuration
      i2c_stop();          // slutter kommunikation

    while(key == 'l')
    {
      clock_date_read(); // Læser klokken og datoen fra DS1340
      delay_ms(100);
      adc_read();   // Læser temperaturen fra ADC'en
      avg_temp();    // Læser temperaturen fra LM73

      output_high (GREEN_LED);
   // Skriver temperaturene og klokken til eeprommen.
   // Hver måling fylder 10 bytes
              write_int16_ext_eeprom(address,temperature);// skriver digital temp til eeprom
            write_int16_ext_eeprom(address+=2,reading);//skriver analog temp til eeprom
      write_int16_ext_eeprom(address+=2,hour);
      write_int16_ext_eeprom(address+=2,min);
      write_int16_ext_eeprom(address+=2,sec);
      address+=2;
      output_low (GREEN_LED);

   // udskriver dataen på LCD displayet.
       sprintf (buf, "h: %lu M: %lu S: %lu", hour,min,sec);
```

```
            lcd_gotoxy(1,1);
            lcd_print (buf);

           sprintf (buf, "d: %3.2f a: %2.2f ",avg_temp_digital, temp_analog);
            lcd_gotoxy(1,2);
            lcd_print (buf);

        // udskriver dataen til
          printf("add: %lu H: %lu M: %lu S: %lu \t",address,hour,min,sec);
          printf("Digital_temp: %f \t",avg_temp_digital);
          printf("Analog_temp: %f \n",temp_analog);

          delay_ms(interval_int);
        }
      break;
      case 's':
        printf("\n Stopping logging \n");
      break;
      case 'r':
        printf("Reading the log \n");
      // henter dataen fra eeprommen og udskriver den til terminalen
        for(j=0; j<(address-10);)
        {
          eeprom_read_digital  =  read_int16_ext_eeprom(j);
          eeprom_read_analog   =  read_int16_ext_eeprom(j+=2);
          eeprom_read_hour     =  read_int16_ext_eeprom(j+=2);
          eeprom_read_min      =  read_int16_ext_eeprom(j+=2);
          eeprom_read_sec      =  read_int16_ext_eeprom(j+=2);
          j+=2;
          printf("H: %lu M: %lu S: %lu \t",eeprom_read_hour,eeprom_read_min,eeprom_read_sec);
          printf("Digital: %f \t", (float)eeprom_read_digital/128);
          printf("Analog: %lu \n", eeprom_read_analog);

        }
      break;
      case 'e':
        printf("\nErasing the log\n");
        address = 0;
      break;
      default:
        printf("\nWrong key, press 'm' for menu \n");
      break;
    }
  }
 }
}

/*   Denne funktion bruger karakteren fra interruptet
  og herefter tjekkes karakteren med en switchcase
  */
int get_num()
{
  digit = 0;
while(!int_flag)   //venter på et input fra brugeren.
{

}
```

```
  int_flag = 0;
                                                    // Tjekker karakteren for en tal værdi
                                                    // Hvis den er et tal, bliver den gemt i variablen "digit"
                                                    // Hvis den er et bogstab eller symbol bliver digital sat til nul
      switch(key)
      {
        case '0':
          digit = 0;
        break;
        case '1':
          digit = 1;
        break;
        case '2':
          digit = 2;
        break;
        case '3':
          digit = 3;
        break;
        case '4':
          digit = 4;
        break;
        case '5':
          digit = 5;
        break;
        case '6':
          digit = 6;
        break;
        case '7':
          digit = 7;
        break;
        case '8':
           digit = 8;
        break;
        case '9':
           digit = 9;
        break;
        default:
          break;
      }
    return digit;
}
/*                  Dennne funktion læser temperaturen fra LM73
                    på I2C bussen */
void temp_read()
{
  i2c_start();
  i2c_write(LM_write);
  i2c_write(0);
  i2c_start();
  i2c_write(LM_read);
  high_byte = i2c_read(); // gemmer MSB i high_byte
  low_byte = i2c_read(0); // gemmer LSB i low_byte
  i2c_stop();

  // nedestående kode sammen sætter high og low_byte til en 16 bits variable
  // og udregner temperaturen. Der divideres med 128 pga opløsningen.
  temperature = high_byte << 8;
```

```
  temperature += low_byte;
  temp_digital = (float) temperature/128;
}


/*                Denne funktion tager temperaturen fra temp_read
                  og tager 8 målinger og udregner gennemsnittet
                  for at få en bedre og stabil værdi*/
void avg_temp()
{

  for(i=0;i<7;i++)
  {
    temp_read();
    avg_temp_digital += temp_digital;
    high_byte = 0, low_byte = 0, temp_digital=0;

  }
  avg_temp_digital = avg_temp_digital/8;
}

/*                Læser fra ADC'en og udregner en temperatur.*/
void adc_read()
{
  reading = read_adc();
  adc_val = (float) reading;
  temp_analog = (4.8327/1024.0)*adc_val;
}

void write_ext_eeprom(int16 address, BYTE data)
{
  int8 status;
  i2c_start();
  i2c_write(0xa0); // i2c adressen for for EEPROM, skrive mode
  i2c_write((address>>8)&0x1f); // MSB af data adressen, max 8 kB
  i2c_write(address); // LSB af data adressen
  i2c_write(data); // data byte skrives til EEPROM
  i2c_stop();
  // vent på at EEPROM er færdig med at skrive
  i2c_start(); // restart
  status = i2c_write(0xa0); //få acknowledge tegnet fra EEPROM

  while(status == 1) // hvis acknowledge tegnet ikke er modtaget fra EEPROM, vent.
  {
    i2c_start();
    status=i2c_write(0xa0); // gentager indtil status er nul.
  }

  i2c_stop();
}

BYTE read_ext_eeprom (int16 address)
{
  BYTE data;
  i2c_start();
  i2c_write(0xa0); // i2c address for EEPROM, write mode
  i2c_write((address>>8)&0x1f); // MSB of data address, max 8kB
```

```
  i2c_write(address); // LSB of data address
  i2c_start();
  i2c_write(0xa1); // i2c address for EEPROM, read mode
  data=i2c_read(0); // read byte, send NACK
  i2c_stop();
  return(data);
}

void write_int16_ext_eeprom(int16 address, int16 data)
{

  for(i = 0; i < 2; ++i)
  {
    write_ext_eeprom(address + i, *((int8 *)(&data) + i));
  }
}

int16 read_int16_ext_eeprom(int16 address)
{

  for(i = 0; i < 2; ++i)
  {
    *((int8 *)(&data) + i) = read_ext_eeprom(address + i);
  }
  return(data);
}


void clock_date_read()
{
    i2c_start();
    i2c_write(0xd0); //Addressen, write mode
    i2c_write(0x00); //Register
    i2c_start();
    i2c_write(0xd1);              //read mode, læser datoen fra DS1340.
                                  // og gemmer læsningerne i variabler.
                                  // læsningerne er codet i binary coded decibel
                                  // og bliver derfor konverteret med bcd2b.
    sec = bcd2b(i2c_read());
    min = bcd2b(i2c_read());
    hour = bcd2b(i2c_read());
    dow = bcd2b(i2c_read());
    day = bcd2b(i2c_read());
    month = bcd2b(i2c_read());
    year = bcd2b(i2c_read());
    i2c_stop();

}

void set_date_time(BYTE day, BYTE month, BYTE year, BYTE dow, BYTE hour, BYTE min, BYTE sec)
{

  i2c_start();
  i2c_write(0xD0);              // I2C write address
  i2c_write(0x00);              // Start at REG 0 - Seconds
  i2c_write(b2bcd(sec));   // REG 0
  i2c_write(b2bcd(min));   // REG 1
```

```
  i2c_write(b2bcd(hour));  // REG 2
  i2c_write(b2bcd(dow));   // REG 3
  i2c_write(b2bcd(day));   // REG 4
  i2c_write(b2bcd(month)); // REG 5
  i2c_write(b2bcd(year));  // REG 6

  i2c_stop();
}

/*                  Denne funktion bruges til set_date_time for at sætte de forskellige tidspunkter. */
void set_date()
{
    printf("set seconds\n");
    sec = (get_num())*10;
    sec += get_num();

    printf("set minuts\n");
    min = get_num()*10;
    min += get_num();

    printf("set hours\n");
    hour = get_num()*10;
    hour += get_num();

    printf("set day of week, 1 is monday, 7 is sunday\n");
    dow = get_num();

    printf("set date\n");
    day = get_num()*10;
    day += get_num();

    printf("set month\n");
    month = get_num()*10;
    month += get_num();

    printf("set year\n");
    year = get_num()*10;
    year += get_num();

}

/*  Nedestående to funktioner er lånt fra en DS1340 driver på nettet
                    B2BCD omregner en binær værdi til en binary coded decimal
                    BCD2B omregner en binary coded værdi til en binær værdi
                    kilde: ccsinfo.com/forum/viewtopic.php?p=172486 */
BYTE b2bcd(BYTE binary_value)
{
 BYTE temp;
 BYTE retval;

 temp = binary_value;
 retval = 0;

 while(1)
 {
  // Get the tens digit by doing multiple subtraction
  // of 10 from the binary value.
```

```
   if(temp >= 10)
   {
    temp -= 10;
    retval += 0x10;
   }
   else // Get the ones digit by adding the remainder.
   {
    retval += temp;
    break;
   }
 }
  return(retval);
}

BYTE bcd2b(BYTE bcd_value)
{
 BYTE temp;

 temp = bcd_value;
 // Shifting upper digit right by 1 is same as multiplying by 8.
 temp >>= 1;
 // Isolate the bits for the upper digit.
 temp &= 0x78;

 // Now return: (Tens * 8) + (Tens * 2) + Ones

 return(temp + (temp >> 2) + (bcd_value & 0x0f));
}
```