```
#include <pic.h>
#include <pic1687x.h>

#define FOSC 12750000L
#define BAUD 9600L

#define PWMZYCLUS 50 //PWM-Zyclus 20ms = 50Hz
#define PWM1MS (unsigned int)(FOSC/(4*20*PWMZYCLUS))
#define PWMZA (unsigned int)(65536L - PWM1MS)
#define PWMZB (unsigned int)(65536L - 19 * PWM1MS)
#define CW 0
#define CCW 1
#define SLEEPTIMER 600000L
#define AUF 0   //Fenster
#define STOPP 1 //Fenster
#define ZU 2    //Fenster

volatile bit pwm_sync;
unsigned int i, j = 0;
volatile bit pwmphase;
volatile unsigned int richtung = PWM1MS / 2; // 0...PWM1MS entspricht
links...rechts
unsigned int temperatur1, temperatur2;
volatile unsigned char temp_ovf;
double temperatur;
unsigned int licht;
volatile unsigned int schritt; //Stellung Schrittmotor
volatile unsigned char rs232_buffer[16];
volatile unsigned char rs232_head = 0;
volatile unsigned char rs232_tail = 0;

void init_schrittmotor() {
  RC0 = 0;
  RC1 = 0;
  RC2 = 0;
  RC3 = 0;
  RC4 = 1; //L293D Enable
  schritt = 200;
  TRISC &= 0b11100000; //RC0...RC3 Output
  TRISA != 0b00000010; //RA1 Input (Endschalter)
}

void init_fenster() {
  RA2 = 0;
  TRISA &= 0b11111011; // RA2 Output für IR-Sender
}

void init_adc() {
  TRISA != 0b00000001; //RA0 Input
  ADCON1 = 0b11001110; //nur AN0 Speed FOSC/64
  CHS0 = 0; //Channel 0 = AN0
  CHS1 = 0;
  CHS2 = 0;
  ADCS1 = 1; //Convesion Clock FOSC/64
  ADCS0 = 0;
  ADON = 1; //ADC Power on
}

void init_pwm() {
  TRISC &= 0b11011111;  //RC5 Output
  T1CON = 0;
  TMR1L = (PWMZA - richtung) & 255;
  TMR1H = (PWMZA - richtung) >> 8;
  pwmphase = 0;
  TMR1ON = 1; //Start Timer
  TMR1IE = 1; //Timer1 Interrupt enable
}

init_led() {
  TRISB &= 0b11001111; // RB4+RB5 sind Ausgaenge fuer LED
  RB4 = 0;
  RB5 = 0;
}
```

```
void init_rs232() {
  // Geschwindigkeit setzten
  SPBRG = (int)(FOSC/(16*BAUD)) - 1;
  BRGH = 1;
  // Asynchrone Schnittstelle einschalten
  SYNC = 0;
  SPEN = 1;
  TXEN = 1;
  CREN = 1;
  RCIE = 1; //Enable Empfangs-Interrupt
}

init_termometer() {
  TRISB != 0b00000001; //RB0 ist Input
  RBPU = 1; //Kein Weak-Pullup
  INTEDG = 0;
  RBIE = 0;
}

void led_rot() {
  RB5 = 0;
  RB4 = 1;
}

void led_gelb() {
  RB5 = 1;
  RB4 = 0;
}

void led_off() {
  RB5 = 0;
  RB4 = 0;
}

void delay( unsigned int time) {
  unsigned int i;

  TMR2IE = 0; //Kein Timer2-Interrupt
  PR2 = (char)( FOSC / (4L * 16L * 1000L)); // 1 ms
  T2CON = 0b00000110;
  for( i = time; i; i--) {
    while( !TMR2IF);
    TMR2IF = 0;
  }
  TMR2ON = 0;
  TMR2IF = 0;
}

unsigned char read_lcd( unsigned char rs) {
  unsigned char temp;
  TRISD = 0b11111111; //RD0...RD7 Eingang
  RB1 = rs & 1;
  RB2 = 1; //Read
  RB3 = 1; //Enable fuer mindestens 300ns
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  temp = PORTD;
  RB3 = 0;  //Nicht enable fuer mindestens 200ns
  RB3 = 0;
  RB3 = 0;
  RB3 = 0;
  TRISD = 0b00000000; //RD0...RD7 Ausgang
  return( temp);
}

void write_lcd( unsigned char rs, unsigned char data) {
  RB1 = rs & 1;
  RB2 = 0; //Write
  PORTD = data;
```

```
  RB3 = 1; //Enable fuer mindestens 300ns
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  RB3 = 1;
  RB3 = 0;  //Nicht enable fuer mindestens 200ns
  RB3 = 0;
  RB3 = 0;
  RB3 = 0;
  for( j = 25; j; j--) RB3 = 0;
}

void init_lcd() {
  TRISD = 0b00000000; //RD0...RD7 Ausgang
  TRISB &= 0b11110001; //RB1...RB3 Ausgang
  PORTD = 0;
  PORTB &= 0b11110001; //RB1...RB3 = 0
  write_lcd( 0, 0b00110100); //Function set
  write_lcd( 0, 0b00000110); //Entry Mode Set
  write_lcd( 0, 0b00001111); //Display on
  write_lcd( 0, 0b00010000); //Display/Coursor Shift
  write_lcd( 0, 0b00000001); //Clear Display
  for( j = 1000; j; j--) RB3 = 0;
  write_lcd( 0, 0b00000010); //Coursor Home
  for( j = 1000; j; j--) RB3 = 0;
  write_lcd( 1, 'D');
  write_lcd( 1, 'L');
  write_lcd( 1, '1');
  write_lcd( 1, 'G');
  write_lcd( 1, 'L');
  write_lcd( 1, 'H');
  write_lcd( 1, ' ');
  write_lcd( 1, ' ');
  write_lcd( 1, '1');
  write_lcd( 1, '0');
  write_lcd( 1, '0');
  write_lcd( 1, '2');
  write_lcd( 1, '2');
  write_lcd( 1, '0');
  write_lcd( 1, '0');
  write_lcd( 1, '5');
}

void sende_rs232( char zeichen) {
  // Warten bis TXREG leer ist;
  while( TXIF == 0);
  // Neues Zeichen in TXREG laden;
  TXEN = 1;
  TXREG = zeichen;
}

signed int lese_rs232() {
  int temp;

  if( rs232_tail == rs232_head) return( -1);
  RCIE = 0;
  if( rs232_tail < 15) {
    temp = rs232_buffer[++rs232_tail];
  }
  else {
    rs232_tail = 0;
    temp = rs232_buffer[ rs232_tail];
  }
  RCIE = 1;
  return( temp);
}

unsigned char getch() {
  int temp;
  unsigned long sleeptimer = SLEEPTIMER;

  while( (temp = lese_rs232()) == -1) {
```

```
    delay(1);
    if( sleeptimer) {
      sleeptimer--;
      led_gelb();
    }
    else {
      RC4 = 0; //LD293 Disable
      led_off();
    }
  }
  RC4 = 1; //LD293 Enable
  led_rot();
  return( (unsigned char)temp);
}

void interrupt int_routine() {
  unsigned char temp;

  if( RCIF == 1) { //Serielles Zeichen empfangen;
    if(rs232_head < 15) {
      if( (rs232_head + 1) != rs232_tail) rs232_buffer[++rs232_head] = RCREG;
      else temp = RCREG;
    }
    else {
      if( rs232_tail > 0) {
        rs232_head = 0;
        rs232_buffer[ rs232_head] = RCREG;
      }
      else temp = RCREG;
    }
    RCIF = 0;
  }
  if( TMR0IF == 1) { // Timer 0 Interrupt
    temp_ovf++;
    TMR0IF = 0;
  }
  if( TMR1IF == 1) { //Timer 1 Interrupt
    TMR1ON = 0; //Stopp Timer
    if( pwmphase = !pwmphase) {
      TMR1L = (PWMZB + richtung) & 0x0ff;
      TMR1H = (PWMZB + richtung) >> 8;
      RC5 = 0;
      pwm_sync = 1; //Jetzt kommt 18..19ms kein Timer1-Interrupt (für
Temperaturmessung)
    } else {
      TMR1L = (PWMZA - richtung) & 0x0ff;
      TMR1H = (PWMZA - richtung) >> 8;
      RC5 = 1;
      pwm_sync = 0;
    }
    TMR1ON = 1; //Start Timer
    TMR1IF = 0;
  }
}

zeige_temperatur(double temperatur) {
  write_lcd( 0, 0b10000000);
  write_lcd( 1, (char)(temperatur / 10.0) + '0');
  write_lcd( 1, (char)(temperatur) % 10 + '0');
  write_lcd( 1, '.');
  write_lcd( 1, (char)(temperatur * (double)10.0) % 10 + '0');
  write_lcd( 1, 0xdf);
  write_lcd( 1, 'C');
  write_lcd( 1, ' ');
}

double lese_temperatur() {
  double temperatur;
  temperatur1 = 0;
  temperatur2 = 0;
  T0CS = 0; //CLKOUT benutzen

  for( i = 1; i; i--) { //war30
```

```
    while( !pwm_sync); //Mit PWM synchronisieren
    pwm_sync = 0;
    PEIE = 0; //Keine Pherepherie Interrupts

    while( !RB0); //Warten bis Impuls
    while( RB0);
    while( !RB0);
    di();
    TMR0 = 0;
    temp_ovf = 0;
    TMR0IF = 0;
    TMR0IE = 1; //Bitte Timer Interrupts
    ei();
    while( RB0);
    di();
    temperatur1 += TMR0;
    temperatur1 += (unsigned int)temp_ovf << 8;
    ei();
    while( !RB0);
    di();
    temperatur2 += TMR0;
    temperatur2 += (unsigned int)temp_ovf << 8;
    ei();

    TMR0IE = 0; //Keine Timer Interrupts
    PEIE = 1; //Perepherie Interrupts erlauben
  }

  temperatur = (((double)temperatur1 / (double)temperatur2) *
(double)212.7659574) - (double)68.08510638;
  return( temperatur);
}

zeige_licht(unsigned int licht) {
  write_lcd( 0, 0b10000000);
  write_lcd( 1, (char)((licht%10000) / 1000) + '0');
  write_lcd( 1, (char)((licht%1000) / 100) + '0');
  write_lcd( 1, (char)((licht%100) / 10) + '0');
  write_lcd( 1, (char)(licht%10) + '0');
  write_lcd( 1, ' ');
  write_lcd( 1, 'L');
  write_lcd( 1, 'X');
}

zeige_richtung(unsigned int richtung) {
  write_lcd( 0, 0b10000000);
  write_lcd( 1, 'V');
  write_lcd( 1, 'R');
  write_lcd( 1, ' ');
  write_lcd( 1, (char)((richtung) / 1000) + '0');
  write_lcd( 1, (char)((richtung%1000) / 100) + '0');
  write_lcd( 1, (char)((richtung%100) / 10) + '0');
  write_lcd( 1, (char)(richtung%10) + '0');
  write_lcd( 1, ' ');
}

unsigned int lese_licht() {
  unsigned int licht;
  ADGO = 1; //AD-Wandlung starten
  while( ADGO); //Warten bis gewandelt
  licht = ((unsigned int)(ADRESH) << 8) + ADRESL;
  return(licht);
}

void schrittmotor_schritt( unsigned char richtung) {
  if( (richtung == CW) && (RA1 == 1)) {//CW nur wenn Endschalter nicht
geoeffnet!
    schritt = 0;
    return;
  }

  delay( 10);
  if( richtung == CW) schritt--;
```

```
  else schritt++;

  switch( schritt & 7) {
       case 0: RC0 = 1;
               RC1 = 0;
               RC2 = 0;
               RC3 = 0;
               break;
       case 1: RC0 = 1;
               RC1 = 1;
               RC2 = 0;
               RC3 = 0;
               break;
       case 2: RC0 = 0;
               RC1 = 1;
               RC2 = 0;
               RC3 = 0;
               break;
       case 3: RC0 = 0;
               RC1 = 1;
               RC2 = 1;
               RC3 = 0;
               break;
       case 4: RC0 = 0;
               RC1 = 0;
               RC2 = 1;
               RC3 = 0;
               break;
       case 5: RC0 = 0;
               RC1 = 0;
               RC2 = 1;
               RC3 = 1;
               break;
       case 6: RC0 = 0;
               RC1 = 0;
               RC2 = 0;
               RC3 = 1;
               break;
       case 7: RC0 = 1;
               RC1 = 0;
               RC2 = 0;
               RC3 = 1;
              break;
  }
}

schrittmotor_zeige_richtung(unsigned int richtung) {
  write_lcd( 0, 0b10000000);
  write_lcd( 1, 'H');
  write_lcd( 1, 'S');
  write_lcd( 1, ' ');
  write_lcd( 1, (char)((richtung) / 100) + '0');
  write_lcd( 1, (char)((richtung % 100) / 10) + '0');
  write_lcd( 1, (char)(richtung%10) + '0');
  write_lcd( 1, ' ');
  write_lcd( 1, ' ');
}

schrittmotor_referenzfahrt()
{
 schritt = 1000; //Position unbekannt
 while( schritt) {
   schrittmotor_schritt(CW);
 }
}

schrittmotor_fahre( unsigned int wohin) {
  if( wohin > 0 && wohin < 351) {
    while( wohin != schritt) {
      if( wohin > schritt) schrittmotor_schritt(CCW);
      else schrittmotor_schritt(CW);
    }
  }
```

```
}

fenster_sende_bit( unsigned char wert) {
  unsigned char k;
  for( k = (wert > 0) ? 39 : 13; k; k--) {
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 1; RA2 = 1; RA2 = 1;      RA2 = 1;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
    RA2 = 0; RA2 = 0; RA2 = 0;      RA2 = 0;
  }
}

fenster_sende( unsigned char byte1, unsigned char byte2, unsigned char byte3) {
  unsigned char k;
  PR2 = 225;
  TMR2IE = 0;
  TMR2IF = 0;
  T2CON = 0b00101101;  // Timer2 on, Postscale = 6; Prescale = 4
  for( k = 128; k; k >>= 1) {
    while( !TMR2IF);
    fenster_sende_bit(byte1 & k);
    TMR2IF = 0;
  }
  for( k = 128; k; k = k >>= 1) {
    while( !TMR2IF);
    fenster_sende_bit(byte2 & k);
    TMR2IF = 0;
  }
  for( k = 128; k; k = k >>= 1) {
    while( !TMR2IF);
    fenster_sende_bit(byte3 & k);
    TMR2IF = 0;
  }
  T2CON = 0b00000000; //Timer2 off
  ei();
}

fenster( unsigned int kommando, unsigned char kanal) {
  // Der Kode der Fernbedienung muss auf 1100000000 stehen
  switch( kommando) {
    case AUF:   switch(kanal) {
                  case '3': fenster_sende( 0b00110000, 0b00110000, 0b00000111);
                            delay(25);
                            fenster_sende( 0b00110000, 0b00110000, 0b00000111);
                            break;
                  case '2': fenster_sende( 0b00101000, 0b00110000, 0b00001101);
                                                        delay(25);
                                                        fenster_sende(
0b00101000, 0b00110000, 0b00001101);
                            break;
                  case '1': fenster_sende( 0b00100100, 0b00110000, 0b00001000);
                            delay(25);
                                                        fenster_sende(
0b00100100, 0b00110000, 0b00001000);
                            break;
```

```
                  default:  break;
                }
                write_lcd( 0, 0b10000000);
                                write_lcd( 1, 'A');
                write_lcd( 1, 'u');
                write_lcd( 1, 'f');
                write_lcd( 1, ' ');
                write_lcd( 1, kanal);
                write_lcd( 1, ' ');
                write_lcd( 1, ' ');
                write_lcd( 1, ' ');
                break;
    case STOPP: switch( kanal) {
                  case '3': fenster_sende( 0b10110000, 0b00110000, 0b00001101);
                            delay(25);
                            fenster_sende( 0b10110000, 0b00110000, 0b00001101);
                            break;
                  case '2': fenster_sende( 0b10101000, 0b00110000, 0b00000111);
                            delay(25);
                            fenster_sende( 0b10101000, 0b00110000, 0b00000111);
                            break;
                  case '1': fenster_sende( 0b10100100, 0b00110000, 0b00000010);
                                                        delay(25);
                                                        fenster_sende(
0b10100100, 0b00110000, 0b00000010);
                            break;
                  default:  break;
                }
                write_lcd( 0, 0b10000000);
                                write_lcd( 1, 'S');
                write_lcd( 1, 't');
                write_lcd( 1, 'o');
                write_lcd( 1, 'p');
                write_lcd( 1, 'p');
                write_lcd( 1, ' ');
                write_lcd( 1, kanal);
                write_lcd( 1, ' ');
                break;
    case ZU:    switch( kanal) {
                  case '3': fenster_sende( 0b01110000, 0b00110000, 0b00000010);
                            delay(25);
                            fenster_sende( 0b01110000, 0b00110000, 0b00000010);
                            break;
                  case '2': fenster_sende( 0b01101000, 0b00110000, 0b00001000);
                            delay(25);
                                                        fenster_sende(
0b01101000, 0b00110000, 0b00001000);
                                                        break;
                  case '1': fenster_sende( 0b01100100, 0b00110000, 0b00001101);
                                                        delay(25);
                                                        fenster_sende(
0b01100100, 0b00110000, 0b00001101);
                                                        break;
                  default:  break;
                }
                write_lcd( 0, 0b10000000);
                                write_lcd( 1, 'Z');
                write_lcd( 1, 'u');
                write_lcd( 1, ' ');
                write_lcd( 1, kanal);
                write_lcd( 1, ' ');
                write_lcd( 1, ' ');
                write_lcd( 1, ' ');
                write_lcd( 1, ' ');
        break;
  }
}

main() {
 unsigned int temprichtung;
 unsigned char z;

 init_led();
```

```
init_rs232();
init_pwm();
richtung = (unsigned int)(0.123 * PWM1MS) + ( (unsigned int)(0.45 * PWM1MS));
init_lcd();
init_termometer();
init_adc();
init_fenster();
init_schrittmotor();
schrittmotor_referenzfahrt();
delay( 500);
schrittmotor_fahre(175);
delay( 200);

ei();  //Globale Interrupts erlaubt
PEIE = 1; //Perepherie Interrupts erlaubt

 while(1) {
   switch( getch()) {
     case 'a':
     case 'A': z = getch(); //A1,A2,A3 Ziffer=Motorkanal
                            fenster(AUF,z);
               sende_rs232('A');
                            sende_rs232(z);
               break;
     case 's':
     case 'S': z = getch(); //S1,S2,S3 Ziffer=Motorkanal
                            fenster(STOPP,z);
               sende_rs232('S');
                            sende_rs232(z);
               break;
     case 'z':
     case 'Z': z = getch(); //Z1,Z2,Z3 Ziffer=Motorkanal
               fenster(ZU,z);
               sende_rs232('Z');
                            sende_rs232(z);
               break;
     case 'u':
     case 'U': if( richtung > 50) richtung -= 50;
               zeige_richtung(richtung);
               break;
     case 'd':
     case 'D': if( richtung < (PWM1MS - 50)) richtung += 50;
               zeige_richtung(richtung);
               break;
     case '-': if( schritt > 0) {
                 schrittmotor_fahre( schritt - 1);
                 schrittmotor_zeige_richtung(schritt);
               }
               break;
     case '+': if( schritt < 351) {
                 schrittmotor_fahre( schritt + 1);
                 schrittmotor_zeige_richtung(schritt);
               }
               break;
     case 'n':
     case 'N': temprichtung = schritt;
               schrittmotor_referenzfahrt();
               delay( 200);
               schrittmotor_fahre(temprichtung);
               schrittmotor_zeige_richtung(schritt);
               break;
     case 't':
     case 'T': temperatur = lese_temperatur();
               zeige_temperatur( temperatur);
               sende_rs232( 'T');
               sende_rs232( (char)(temperatur / 10.0) + '0');
               sende_rs232( ((char)(temperatur) % 10) + '0');
               sende_rs232( (char)(temperatur * (double)10.0) % 10 + '0');
               break;
     case 'l':
     case 'L': licht = lese_licht();
               sende_rs232( 'L');
               sende_rs232( (char)((licht%10000) / 1000) + '0');
```

```
                sende_rs232( (char)((licht%1000) / 100) + '0');
                sende_rs232( (char)((licht%100) / 10) + '0');
                sende_rs232( (char)(licht%10) + '0');
                zeige_licht(licht);
                break;
      case 'h':
      case 'H': z = getch(); //Lese Richtung in Grad
                if( z >= '0' && z <= '9') {
                   temprichtung = z - '0';
                   temprichtung *= 10;
                   z = getch();
                   if( z >= '0' && z <= '9') {
                     temprichtung += z - '0';
                     temprichtung *= 10;
                     z = getch();
                     if( z >= '0' && z <= '9') {
                       temprichtung += z - '0';
                       if( temprichtung <= 315) { //Max 0...315 Grad
                         schrittmotor_fahre( (unsigned int)((double)temprichtung
* 1.111111) + 1);
                         schrittmotor_zeige_richtung(schritt);
                       }
                     }
                   }
                }
                break;
      case 'v':
      case 'V': z = getch(); //Lese Elevation in Grad
                if( z >= '0' && z <= '9') {
                   temprichtung = z - '0';
                   temprichtung *= 10;
                   z = getch();
                   if( z >= '0' && z <= '9') {
                     temprichtung += z - '0';
                     temprichtung *= 10;
                     z = getch();
                     if( z >= '0' && z <= '9') {
                       temprichtung += z - '0';
                       if( temprichtung <= 90) { //Max 0...90 Grad
                         temprichtung = 90 - temprichtung;
                         richtung = (unsigned int)(0.123 * PWM1MS) + ( (unsigned
int)((double)temprichtung * PWM1MS / 100));
                         zeige_richtung(richtung);
                       }
                     }
                   }
                }
                break;
      default:  sende_rs232('?');
                break;
    }
   sende_rs232( 10);

 }
}
```