```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "lcd-routines.h"
#include <stdlib.h>
//#include "lcd-routines.c"


#include <util/delay.h>

int main()
{

Hauptprogramm


}
```

```
// Ansteuerung eines HD44780 kompatiblen LCD im 4-Bit-Interfacemodus
// http://www.mikrocontroller.net/articles/HD44780
// http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung
//
// Die Pinbelegung ist über defines in lcd-routines.h einstellbar

#include <avr/io.h>
#include "lcd-routines.h"
#include <util/delay.h>

/////////////////////////////////////////////////////////////////////////////
// Erzeugt einen Enable-Puls
static void lcd_enable( void )
{
   LCD_PORT |= (1<<LCD_EN);     // Enable auf 1 setzen
   _delay_us( LCD_ENABLE_US );  // kurze Pause
   LCD_PORT &= ~(1<<LCD_EN);    // Enable auf 0 setzen
}

/////////////////////////////////////////////////////////////////////////////
// Sendet eine 4-bit Ausgabeoperation an das LCD
static void lcd_out( uint8_t data )
{
   data &= 0xF0;                    // obere 4 Bit maskieren

   LCD_PORT &= ~(0xF0>>(4-LCD_DB));    // Maske löschen
   LCD_PORT |= (data>>(4-LCD_DB));     // Bits setzen
   lcd_enable();
}

/////////////////////////////////////////////////////////////////////////////
```

```
// Initialisierung: muss ganz am Anfang des Programms aufgerufen werden.
void lcd_init( void )
{
   // verwendete Pins auf Ausgang schalten
   uint8_t pins = (0x0F << LCD_DB) |          // 4 Datenleitungen
                  (1<<LCD_RS) |               // R/S Leitung
                  (1<<LCD_EN);                // Enable Leitung
   LCD_DDR |= pins;

   // initial alle Ausgänge auf Null
   LCD_PORT &= ~pins;

   // warten auf die Bereitschaft des LCD
   _delay_ms( LCD_BOOTUP_MS );

   // Soft-Reset muss 3mal hintereinander gesendet werden zur Initialisierung
   lcd_out( LCD_SOFT_RESET );
   _delay_ms( LCD_SOFT_RESET_MS1 );

   lcd_enable();
   _delay_ms( LCD_SOFT_RESET_MS2 );

   lcd_enable();
   _delay_ms( LCD_SOFT_RESET_MS3 );

   // 4-bit Modus aktivieren
   lcd_out( LCD_SET_FUNCTION |
         LCD_FUNCTION_4BIT );
   _delay_ms( LCD_SET_4BITMODE_MS );

   // 4-bit Modus / 2 Zeilen / 5x7
   lcd_command( LCD_SET_FUNCTION |
                LCD_FUNCTION_4BIT |
                LCD_FUNCTION_2LINE |
                LCD_FUNCTION_5X7 );

   // Display ein / Cursor aus / Blinken aus
   lcd_command( LCD_SET_DISPLAY |
                LCD_DISPLAY_ON |
                LCD_CURSOR_OFF |
                LCD_BLINKING_OFF);

   // Cursor inkrement / kein Scrollen
   lcd_command( LCD_SET_ENTRY |
                LCD_ENTRY_INCREASE |
                LCD_ENTRY_NOSHIFT );

   lcd_clear();
}

//////////////////////////////////////////////////////////////////////////////
// Sendet ein Datenbyte an das LCD
```

```
void lcd_data( uint8_t data )
{
   LCD_PORT |= (1<<LCD_RS);    // RS auf 1 setzen

   lcd_out( data );            // zuerst die oberen,
   lcd_out( data<<4 );         // dann die unteren 4 Bit senden

   _delay_us( LCD_WRITEDATA_US );
}

//////////////////////////////////////////////////////////////////////////
// Sendet einen Befehl an das LCD
void lcd_command( uint8_t data )
{
   LCD_PORT &= ~(1<<LCD_RS);    // RS auf 0 setzen

   lcd_out( data );            // zuerst die oberen,
   lcd_out( data<<4 );         // dann die unteren 4 Bit senden

   _delay_us( LCD_COMMAND_US );
}

//////////////////////////////////////////////////////////////////////////
// Sendet den Befehl zur Löschung des Displays
void lcd_clear( void )
{
   lcd_command( LCD_CLEAR_DISPLAY );
   _delay_ms( LCD_CLEAR_DISPLAY_MS );
}

//////////////////////////////////////////////////////////////////////////
// Sendet den Befehl: Cursor Home
void lcd_home( void )
{
   lcd_command( LCD_CURSOR_HOME );
   _delay_ms( LCD_CURSOR_HOME_MS );
}

//////////////////////////////////////////////////////////////////////////
// Setzt den Cursor in Spalte x (0..15) Zeile y (1..4)

void lcd_setcursor( uint8_t x, uint8_t y )
{
   uint8_t data;

   switch (y)
   {
      case 1:    // 1. Zeile
         data = LCD_SET_DDADR + LCD_DDADR_LINE1 + x;
         break;

      case 2:    // 2. Zeile
```

```
            data = LCD_SET_DDADR + LCD_DDADR_LINE2 + x;
            break;

        case 3:    // 3. Zeile
            data = LCD_SET_DDADR + LCD_DDADR_LINE3 + x;
            break;

        case 4:    // 4. Zeile
            data = LCD_SET_DDADR + LCD_DDADR_LINE4 + x;
            break;

        default:
            return;                         // für den Fall einer falschen Zeile
    }

    lcd_command( data );
}

//////////////////////////////////////////////////////////////////////////////
// Schreibt einen String auf das LCD

void lcd_string( const char *data )
{
    while( *data != '\0' )
        lcd_data( *data++ );
}

//////////////////////////////////////////////////////////////////////////////
// Schreibt ein Zeichen in den Character Generator RAM

void lcd_generatechar( uint8_t startadresse, const uint8_t *data )
{
    // Startposition des Zeichens einstellen
    lcd_command( LCD_SET_CGADR | (startadresse<<3) ); //Startadressen: 0;1;2;3;4;5;6;7

    // Bitmuster übertragen
    for ( uint8_t i=0; i<8; i++ )
    {
        lcd_data( data[i] );
    }
    lcd_command(LCD_SET_DDADR); //DRAM auf 0 setzen
}

-----------------------------------------------------

// Ansteuerung eines HD44780 kompatiblen LCD im 4-Bit-Interfacemodus
// http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung
//

#ifndef LCD_ROUTINES_H
#define LCD_ROUTINES_H
```

```
////////////////////////////////////////////////////////////////////////////
// Hier die verwendete Taktfrequenz in Hz eintragen, wichtig!

#ifndef F_CPU
#define F_CPU 3686400UL
#endif

////////////////////////////////////////////////////////////////////////////
// Pinbelegung für das LCD, an verwendete Pins anpassen
// Alle LCD Pins müssen an einem Port angeschlossen sein und die 4
// Datenleitungen müssen auf aufeinanderfolgenden Pins liegen

//  LCD DB4-DB7 <-->  PORTD Bit PD0-PD3
#define LCD_PORT      PORTD
#define LCD_DDR       DDRD
#define LCD_DB        PD4

//  LCD RS      <-->  PORTD Bit PD2     (RS: 1=Data, 0=Command)
#define LCD_RS        PD2

//  LCD EN      <-->  PORTD Bit PD3     (EN: 1-Impuls für Daten)
#define LCD_EN        PD3

////////////////////////////////////////////////////////////////////////////
// LCD Ausführungszeiten (MS=Millisekunden, US=Mikrosekunden)

#define LCD_BOOTUP_MS          15
#define LCD_ENABLE_US          20
#define LCD_WRITEDATA_US       46
#define LCD_COMMAND_US         42

#define LCD_SOFT_RESET_MS1      5
#define LCD_SOFT_RESET_MS2      1
#define LCD_SOFT_RESET_MS3      1
#define LCD_SET_4BITMODE_MS     5

#define LCD_CLEAR_DISPLAY_MS    2
#define LCD_CURSOR_HOME_MS      2

////////////////////////////////////////////////////////////////////////////
// Zeilendefinitionen des verwendeten LCD
// Die Einträge hier sollten für ein LCD mit einer Zeilenlänge von 16 Zeichen passen
// Bei anderen Zeilenlängen müssen diese Einträge angepasst werden

#define LCD_DDADR_LINE1        0x00
#define LCD_DDADR_LINE2        0x40
#define LCD_DDADR_LINE3        0x10
#define LCD_DDADR_LINE4        0x50

////////////////////////////////////////////////////////////////////////////
// Initialisierung: muss ganz am Anfang des Programms aufgerufen werden.
void lcd_init( void );
```

```
////////////////////////////////////////////////////////////////////////////
// LCD löschen
void lcd_clear( void );

////////////////////////////////////////////////////////////////////////////
// Cursor in die 1. Zeile, 0-te Spalte
void lcd_home( void );

////////////////////////////////////////////////////////////////////////////
// Cursor an eine beliebige Position
void lcd_setcursor( uint8_t spalte, uint8_t zeile );

////////////////////////////////////////////////////////////////////////////
// Ausgabe eines einzelnen Zeichens an der aktuellen Cursorposition
void lcd_data( uint8_t data );

////////////////////////////////////////////////////////////////////////////
// Ausgabe eines Strings an der aktuellen Cursorposition
void lcd_string( const char *data );

////////////////////////////////////////////////////////////////////////////
// Definition eines benutzerdefinierten Sonderzeichens.
// data muss auf ein Array[8] mit den Zeilencodes des zu definierenden Zeichens
// zeigen. Später können siese mit lcd_data(0-7) aufgerufen werden
void lcd_generatechar( uint8_t startadresse, const uint8_t *data );

////////////////////////////////////////////////////////////////////////////
// Ausgabe eines Kommandos an das LCD.
void lcd_command( uint8_t data );


////////////////////////////////////////////////////////////////////////////
// LCD Befehle und Argumente.
// Zur Verwendung in lcd_command

// Clear Display ---------- 0b00000001
#define LCD_CLEAR_DISPLAY       0x01

// Cursor Home ---------- 0b0000001x
#define LCD_CURSOR_HOME         0x02

// Set Entry Mode ---------- 0b000001xx
#define LCD_SET_ENTRY           0x04

#define LCD_ENTRY_DECREASE      0x00
#define LCD_ENTRY_INCREASE      0x02
#define LCD_ENTRY_NOSHIFT       0x00
#define LCD_ENTRY_SHIFT         0x01

// Set Display ---------- 0b00001xxx
#define LCD_SET_DISPLAY         0x08
```

```
#define LCD_DISPLAY_OFF         0x00
#define LCD_DISPLAY_ON          0x04
#define LCD_CURSOR_OFF          0x00
#define LCD_CURSOR_ON           0x02
#define LCD_BLINKING_OFF        0x00
#define LCD_BLINKING_ON         0x01

// Set Shift ------------------ 0b0001xxxx
#define LCD_SET_SHIFT           0x10

#define LCD_CURSOR_MOVE         0x00
#define LCD_DISPLAY_SHIFT       0x08
#define LCD_SHIFT_LEFT          0x00
#define LCD_SHIFT_RIGHT         0x04

// Set Function ---------------- 0b001xxxxx
#define LCD_SET_FUNCTION        0x20

#define LCD_FUNCTION_4BIT       0x00
#define LCD_FUNCTION_8BIT       0x10
#define LCD_FUNCTION_1LINE      0x00
#define LCD_FUNCTION_2LINE      0x08
#define LCD_FUNCTION_5X7        0x00
#define LCD_FUNCTION_5X10       0x04

#define LCD_SOFT_RESET          0x30

// Set CG RAM Address --------- 0b01xxxxxx  (Character Generator RAM)
#define LCD_SET_CGADR           0x40

#define LCD_GC_CHAR0            0
#define LCD_GC_CHAR1            1
#define LCD_GC_CHAR2            2
#define LCD_GC_CHAR3            3
#define LCD_GC_CHAR4            4
#define LCD_GC_CHAR5            5
#define LCD_GC_CHAR6            6
#define LCD_GC_CHAR7            7

// Set DD RAM Address --------- 0b1xxxxxxx  (Display Data RAM)
#define LCD_SET_DDADR           0x80

#endif
```