Projektlabor

LCD Ansteuerung

Contents

1 LCD

LCD steht für liquid crystal display, was im Deutschen
der Flüssigkristallbildschirm ist, dessen Flüssigkristallmoleküe, bei angelegter Spannung die Polarisationsrichtung des Lichts ändern. Die Flüssigkristallmoleküle sind so zwischen Zwei Elektroden und Polarisationsfiltern angeordnet, dass das LCD transparent erscheint. Durch das Anlegen einer Spannung kann für einzelne Pixel diese Ordnung gestört werden, wodurch sie dunkel bleiben und dadurch Zeichen dargestellt werden können.

Die Vorteile von LCDs sind die geringere Leistungsaufnahme, ein geringeres Magnetfeld und dass keine Röntgenstrahlung abgestrahlt wird. Hinzu haben sie ein flimmer- und verzerrungsfreies Bild und ein geringeres Gewicht. Nachteile sind der eingeschränkte Bereich der Betrachtungsrichtung mit konstantem Kontrast und bei der Produktion entstandene Pixelfehler. Anwendung finden LCDs in Digitaluhren, Taschenrechner, Mobiltelefonen und Notebooks, eben dort wo ein geringer Energieverbrauch vonnöten ist.

2 Hardware des Displays

2.1 Hardware

LCDs, die als Controllerchip den HD44780 verwenden, sind einzeilig, zweizeilig oder vierzeilig mit je acht bis vierzig Zeichen pro Zeile zu haben. Die Zeichen werden je mit einer 5x8 (bzw 5x10) Matrix dargestellt. Der HD44780 hat einen 80 Zeilen langen Textpuffer(DDRAM) ein paralleles Interface und einen LCD Treiber mit 16 Punkt-Zeilenleitungen und 40 Punkt-Spaltenleitungen. Die Displaypositionen(für unterschiedliche Displays) werden den Adressen des DDRAM wie in Figur1 zugeordnet. In einem weiteren Speicher, dem Character Generator RAM (CGRAM) können selbst definierte Zeichen (in Form 5x8 bzw 5x10 Punktmatrix) abgelegt werden.

Adressen der Displaypositionen im Textpuffer(hexadezimal, ohne Shift))

Displaytyp	1. Zeile	2. Zeile	3. Zeile	4. Zeile	Bemerkung
1x8	00-07	-	-	-	
1x16	00-0F	-	-	-	echtes einzeiliges Display mit Displaytreiber
1x16 (8+8)	00-07 40-47	-	-	-	linke Hälfte rechte Hälfte
1x20	00-13	-	-	-	
1x40	00-27		-	-	
2x8	00-07	40-47	-	-	
2x12	00-0B	40-4B	-	-	
2x16	00-0F	40-4F	-	-	
2x20	00-13	40-53	-	-	
2x24	00-17	40-57	-	-	
2x40	00-27	40-67	-	-	
4x16	00-0F	40-4F	10-1F	50-5F	
4x20	00-13	40-53	14-27	54-67	
4x40	00-27 -	40-67 -	- 00-27	- 40-67	1. Controller 2. Controller

Figure 1: DDRAM

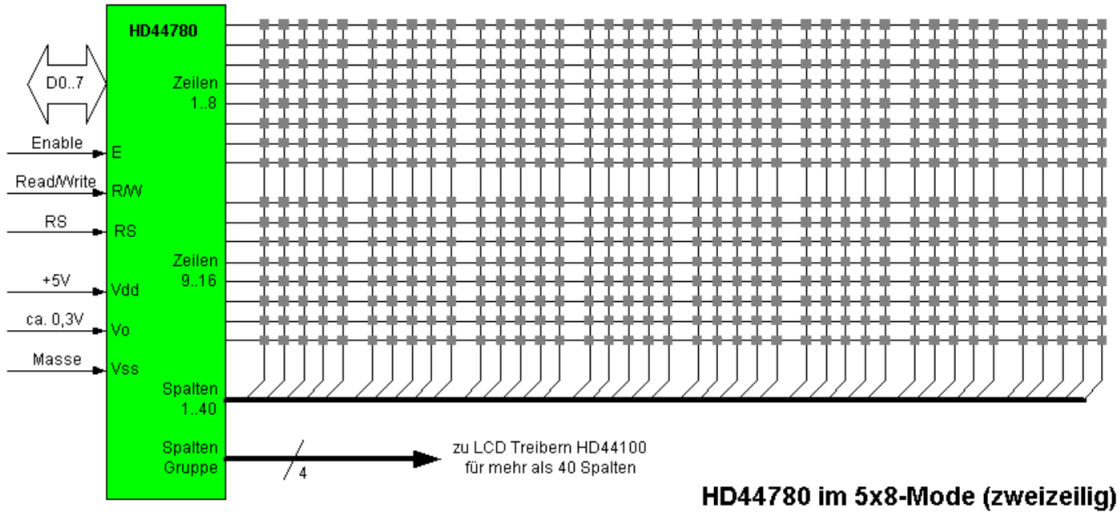

Figure 2: HD44780 im 5x8-Mode(zweizeilig)

2.2 Verbindung

Pins
Die folgende Abbildung zeigt wie die Pins des LCD mit denen des Mikrocontrollers verbunden werden(4-Bit-Modus). Vss liegt auf Masse, Vcc ist an der Versorgungsspannung und über Vee kann der Kontrast des Displays reguliert werden. Dazu wird zwischen Vss und Vcc ein 10kOhmPotentiometer gesetzt mit dem Schleifer an Vee. Bei einer geringeren Betriebsspannung kann an Vee auch eine negative Spannung bis zu -500mA angelegt werden um den Kontrast des Displays zu erhalten. Dieser Wert sollte aber nicht unterschritten werden, da sonst der Stromfluss zu groß ist und der Controller Schaden nimmt (Bei Hochtemperaturdisplays Kontrastspannung auch -5V). Ohne Hintergrundbeleuchtung liegt

die Stromaufnahme unter 1mA, möglich sind aber Maximalwerte von 5mA.

Pin #-LCD	Bezeichnung-LCD	Pin-µC
1	Vss	GND
2	Vcc	5V
3	Vee	GND oder Poti (siehe oben)
4	RS	PD4 am AVR
5	RW	GND
6	E	PD5 am AVR
7	DB0	offen (unbenutzt)
8	DB1	
9	DB2	
10	DB3	
11	DB4	PD0 am AVR
12	DB5	PD1 am AVR
13	DB6	PD2 am AVR
14	DB7	PD3 am AVR

Figure 3: Pinbelegung

Ansteuerung
Über die drei Leitungen RS, RW und E (bzw. EN) wird die Datenübertragung gesteuert. Die Übertragung der Daten geschieht über die Pins DB0-DB7. Dies kann in Form des 8-Bit-Modus oder des 4-Bit-Modus umgesetzt werden. Beim 8-Bit-Modus werden bei jedem Zugriff die 8Bit übertragen, wohingegen beim 4-Bit-Modus zweimal zugegriffen werden muss für die Übertragung von 8Bit, was vier Pins am Mikrocontroller spart. Hierbei werden mittels DB4-DB7 zuerst die vier höherwertigen Bits und nachfolgend die niederwertigeren übertragen. Die Pins DB0-DB3 werden mit Masse verbunden. Mit RS (RegisterSelect) wird angegeben, ob das an den Datenleitungen anliegende Signal als Befehl behandelt oder auf dem Display ausgegeben werden soll. Ist RS Low, handelt es sich um einen Befehl, bei High um die Displayausgabe. RW (Read/Write) ist High wenn gelesen und Low wenn geschrieben werden soll. E (Enable) gibt dem LCD zu erkennen, dass die Signale von den Datenleitungen übernommen werden können. Zwei weitere Pins (PIN 15-16) können für eine Hintergrundbeleuchtung des Displays genutzt werden.

Für das Ausführen von Befehlen braucht das LCD ein paar Mikrosekunden, somit sollte zwischen den Befehlen etwas abgewartet werden. Die benötigte Zeit schwankt von Lcd zu Lcd, daher kann anstatt anzuwarten auch das Busy-Flag abgefragt werden, welches an DB7 liegt und High ist, solange das LCD beschäftigt ist. Zum Abfragen des Busy-Flags wird RW auf High und RS auf Low gesetzt. Falls das Busy Flag nicht benötigt und einfach abgewartet wird zwischen den

Befehlen, kann RW auch mit Masse verbunden werden, wenn nicht vom Display gelesen werden soll. Dies spart einen weiteren Pin am Mikrocontroller.Die Pins des LCDs müssen am selben Port des Mikrocontrollers angeschlossen sein, mit aufeinanderfolgenden Datenpins.

3 Softwareansteuerung

Beim Anlegen der Betriebsspannung wird das Display initialisiert. Figur 4 zeigt wie die Zustände des Displays zum Beispiel im 4Bit Modus gesetzt werden.

Power On										
mindestens 15 ms warten										auf das Ende des internen Reset warten
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Interface auf 8-Bit setzen
0	0	0	0	1	1	-	-	-	-	
mindestens 4,1 ms warten										
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Interface auf 8-Bit setzen
0	0	0	0	1	1	-	-	-	-	
wenigstens 100 µs warten										
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Interface auf 8-Bit setzen
0	0	0	0	1	1	-	-	-	-	
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Interface auf 4-Bit setzen
0	0	0	0	1	0	-	-	-	-	
RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	2-zeilig, 5x8-Punkt-Matrix
0	0	0	0	1	0					
0	0	N(1)	F(0)	-	-					
0	0	0	0	0	0					Display aus
0	0	1	0	0	0					
0	0	0	0	0	0					Display löschen
0	0	0	0	0	1					
0	0	0	0	0	0					Kursor nach rechts wandernd, kein Display shif
0	0	0	1	I/D(1)	S(0)					
0	0	0	0	0	0					Display ein
0	0	1	1	0	0					
fertig										

Figure 4: 4-Bit-Modus

HD44780 Befehlssatz

Befehl	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
Bildschirminhalt löschen	0	0	0	0	0	0	0	0	0	1
Cursor auf Startpos	0	0	0	0	0	0	0	0	1	X
Modus festlegen	0	0	0	0	0	0	0	1	I/D	S
Display/Cursor	0	0	0	0	0	0	1	D	C	B
Cursor/Display schieben	0	0	0	0	0	1	S/C	R/L	X	X
Funktionen	0	0	0	0	1	DL	N	F	X	X
CGRAM Adresse setzen	0	0	0	1	CGRAM-Adresse					
DDRAM Adresse setzen	0	0	1	DDRAM-Adresse						
Adresse/Status lesen	0	1	BF	CG-/DDRAM-Adresse						
Daten in DDRAM/CGRAM schreiben	1	0	Daten							
Daten aus DDRAM/CGRAM lesen	1	1	Daten							

Figure 5: Befehlssatz HD44780

Steuerbefehle werden vom Display durch die Steuerleitung RS erkannt, die in diesem Fall 0 gesetzt ist. Damit über die Ansteuerungsfunktionen für das LCD verfügt werden kann, muss in das Projekt die Datei lcd-routines.c eingefügt werden. Sowohl in der Datei lcd-routines.c als auch in dem eigentlichen Programm muss die Includedatei lcd-routines.h eingebunden sein. In dieser müssen die Frequenz, die Port bzw. Pinbelegung und die Zeilenlänge des Displays angepasst werden.

```
// lcd-routines.h

#ifndef F_CPU                   // Taktfrequenz anpassen in Hz
#define F_CPU 8000000           // hier z.B 8MHz
#endif
 //Port und pins am Mikrocontroller wählen
#define LCD_PORT    PORTD  // für LCD Port wird Port D am Mikrocontroller
#define LCD_DDR     DDRD   // für LCD Datenrichtungsregister wird DDRD am Mikrocontroller
#define LCD_DB      PD0    // Pins der Datenleitungen

 // hier Zeilenlänge 16 Zeichen
#define LCD_DDADR_LINE1         0x00
#define LCD_DDADR_LINE2         0x40
#define LCD_DDADR_LINE3         0x10
#define LCD_DDADR_LINE4         0x50
```

Figure 6: LCD-Routine

Am Anfang des Hauptprogramms muss die Funktion lcd_init aufgerufen werden zur Initialisierung des Displays. Das folgende Codebeispiel zeigt die lcd_init Funktion, sowie die Funktionen lcd_out,lcd_data und lcd_command, welche in der lcd_init Funktion verwendet werden .Die Pins fuer die 4 Datenleitungen(da der 4Bit Modus initialisiert werden soll), RS und EN werden in LCD_DDR 1 und damit als Ausgang gesetzt. Diese Ausgänge werden dann mit 0 initilisiert. Mit Aufruf von lcd _out mit LCD_SOFT_RESET wird der Cursor des Displays auf die Startposition gesetzt. Weiter wird hier der 4 Bit Modus aktiviert und Zeilen sowie Zeilenlänge gewählt. Figur 7 zeigt eine Übersicht der zuübergebenen Befehle für die Ansteuerung.

```
void lcd_init(void){
        uint8_t pins = (0x0F << LCD_DB) | (1<<LCD_RS) | (1<<LCD_EN);
 LCD_DDR |= pins;

    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 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 loeschen
    LCD_PORT |= (data>>(4-LCD_DB));        // Bits setzen
    lcd_enable();
}

// 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 );
```

}

HD44780 Befehlssatz

Befehl	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
Bildschirminhalt löschen	0	0	0	0	0	0	0	0	0	1
Cursor auf Startpos	0	0	0	0	0	0	0	0	1	X
Modus festlegen	0	0	0	0	0	0	0	1	I/D	S
Display/Cursor	0	0	0	0	0	0	1	D	C	B
Cursor/Display schieben	0	0	0	0	0	1	S/C	R/L	X	X
Funktionen	0	0	0	0	1	DL	N	F	X	X
CGRAM Adresse setzen	0	0	0	1	CGRAM-Adresse					
DDRAM Adresse setzen	0	0	1	DDRAM-Adresse						
Adresse/Status lesen	0	1	BF	CG-/DDRAM-Adresse						
Daten in DDRAM/CGRAM schreiben	1	0	Daten							
Daten aus DDRAM/CGRAM lesen	1	1	Daten							

Figure 7: Befehlssatz HD44780

4 Quellen

http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung
http://www.mikrocontroller.net/articles/HD44780
http://www.sprut.de/electronic/lcd/
http://de.wikipedia.org/wiki/Flüssigkristallanzeige