Zeichenorientierte LCD-Displays

- Eigenschaften von zeichenorientierten Displays
- Interface des Displays nach Industriestandard
- Die LiquidChristal Library
- Schaltung der Character-Display-Experimente
- Software der Experimente

Eigenschaften von zeichenorientierten Displays

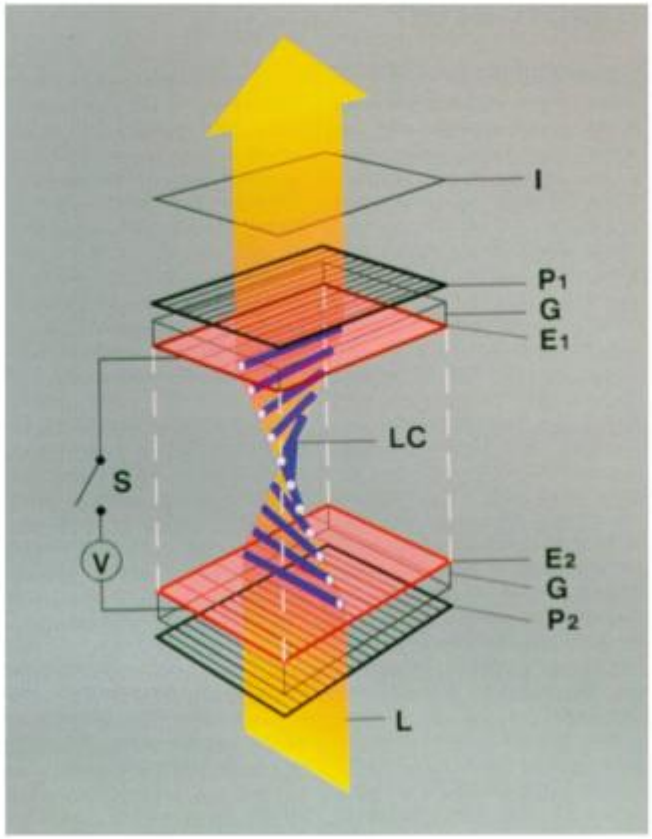
Für viele Anwendung ist die Darstellung von Texten und Worten in Form von Charactern (feste Zeichen) ausreichend. Die Character sind aus einer Pixel-Matrix aufgebaut. Character-orientierte Displays können mehrere Zeichensätze verwenden. Es sind ASCII-Zeichen für europäische / englische Spachen enthalten und weiter japanische oder chinesische Zeichensätze enthalten.

Der Vorteil von solchen Displays ist die Größe und der Kontrast der Zeichen. Sie sind in der Regel besser lesbar als auf Grafik-Displays. Auch die Anzeigegeschwindigkeit ist meist höher als die der durch Software generierten Punkten erstellten Zeichen auf Grafik-Displays.

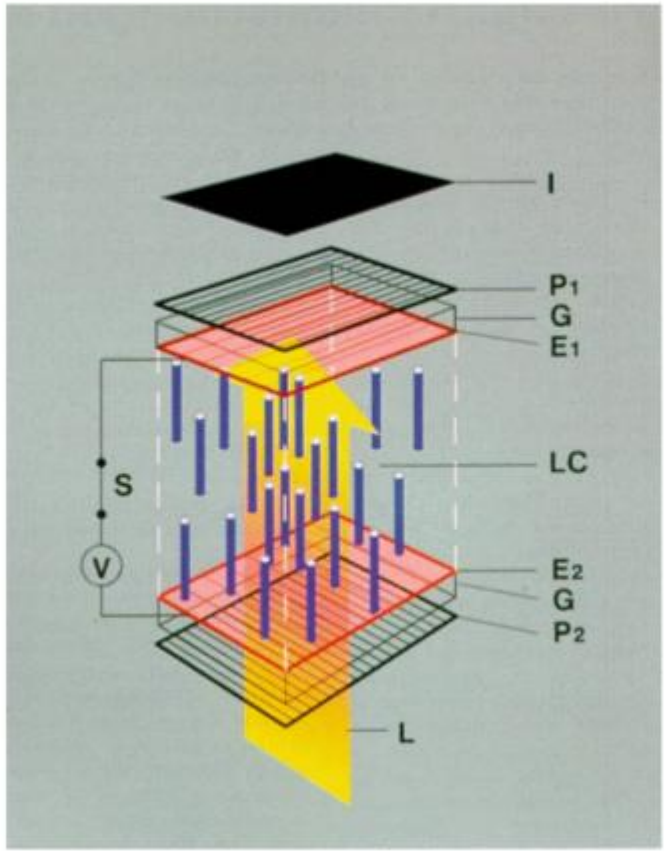
Die Character-Displays kann man in diversen Größen zwischen ca. 5 und 20€ bekommen. Die Anbieter schreiben gerne, dass die Displays kompatibel zum Industriestandard HD44780. Leider ließen sich nur 7 von 10 gekauften unterschiedlichen Displays erfolgreich ansteuern. Es scheint undokumentierte kleine abweichende Verhaltensweisen zu geben. Wenn man die Datenblätter der angeblich kompatibelen Anzeigecontroller durchliestm fallen einem gelegentlich Verständnisfehler der vom HD44780 übernommenen Texte auf, die konsequent fehlerhaft an mehreren Stellen des Dokumentes verwendet werden.

Wie funktioniert ein einfarbiges Pixel ?

Punkt ohne E-Feld:
LC-Material verdreht die Polarisation des durch-scheinenden Lichts so, dass es nicht durch den Polarisationsfilter am Ausgang kommt.

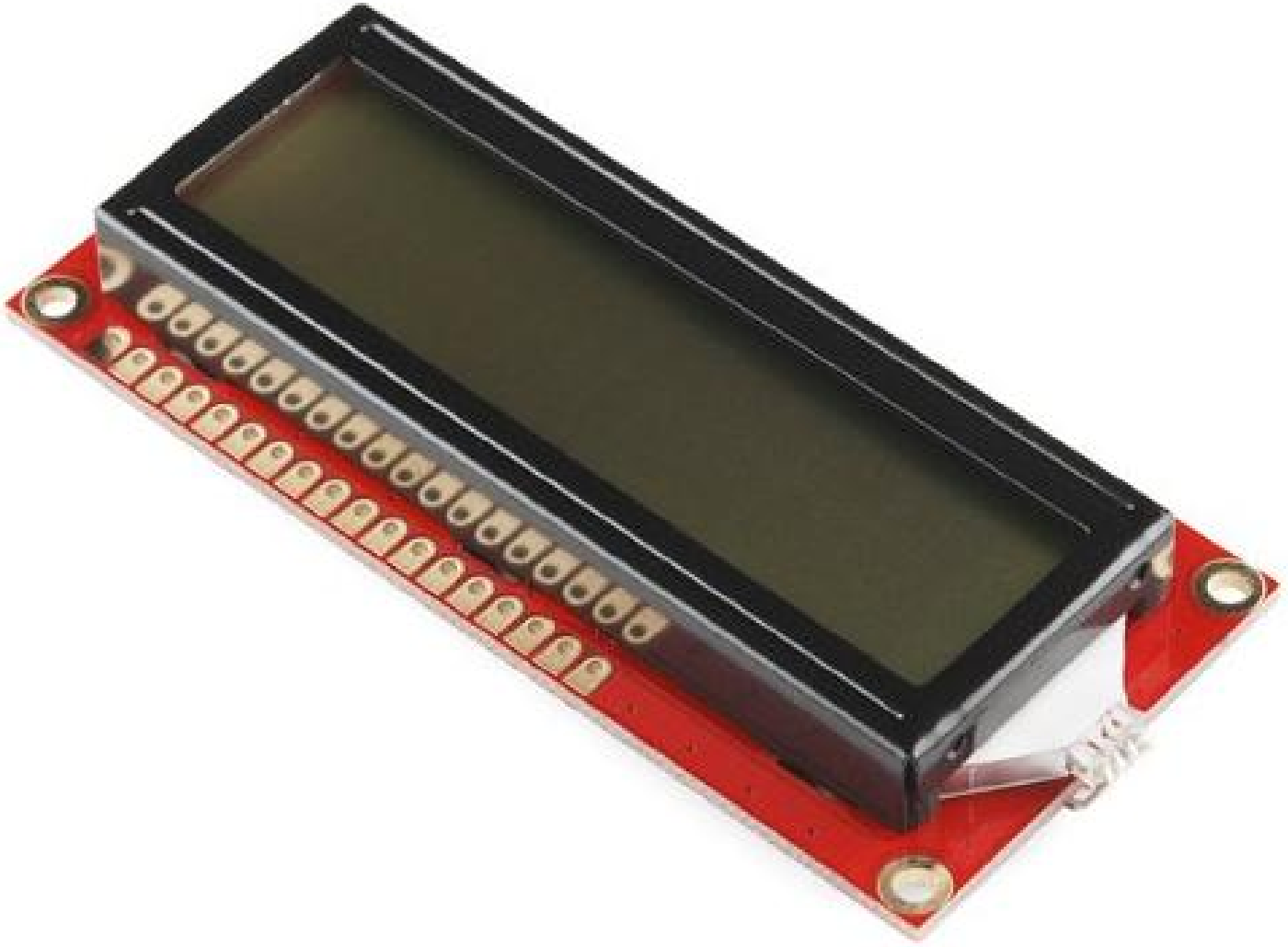


Punkt mit angelegtem E-Feld:
LC-Material verändert Polarisation nicht, Polarisiertes Licht kann durch den Polarisationsfilter am Ausgang kommen



Quelle: http://en.wikipedia.org/wiki/Twisted_nematic_field_effect#TN-effect

Beispiel einer LCD-Anzeige: **LCD-10862**



Quelle: http://www.lipoly.de

ADM1602K1 Neue Version, RGB

PIN ASSIGNMENT

	1	VSS
	2	VDD
	3	V0
	4	RS
	5	RW
18	6	E
	7	DB0
	8	DB1
	9	DB2
	10	DB3
	11	DB4
	12	DB5
	13	DB6
	14	DB7
	15	K
	16	A-RED
1	17	A-GREEN
	18	A-BLUE

Pin-Belegung

Alte Version
Einfarbig(weiss):

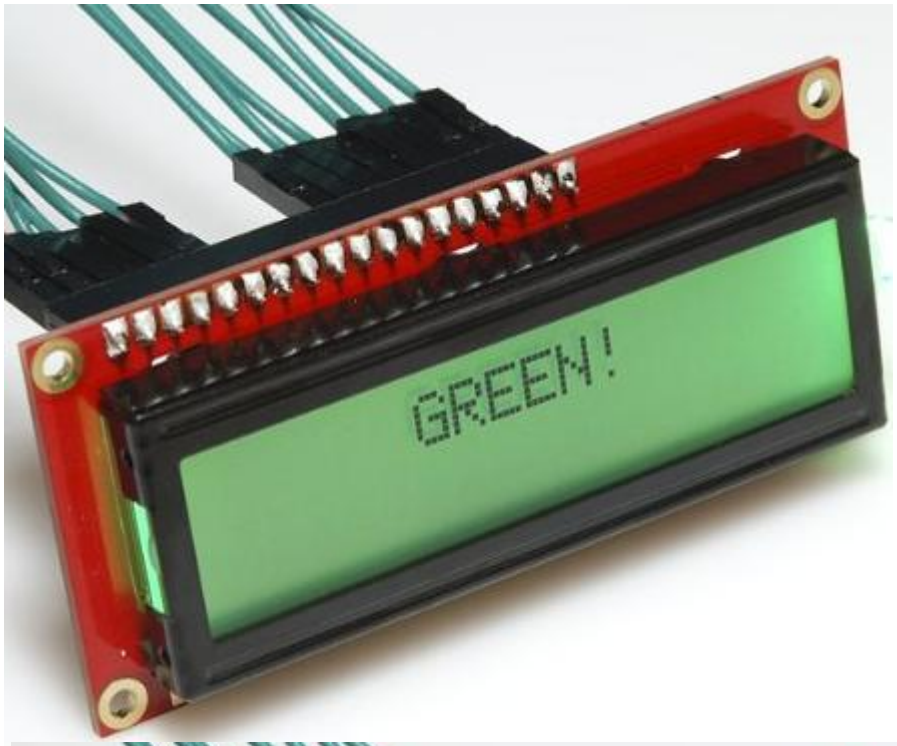
ADM1602K

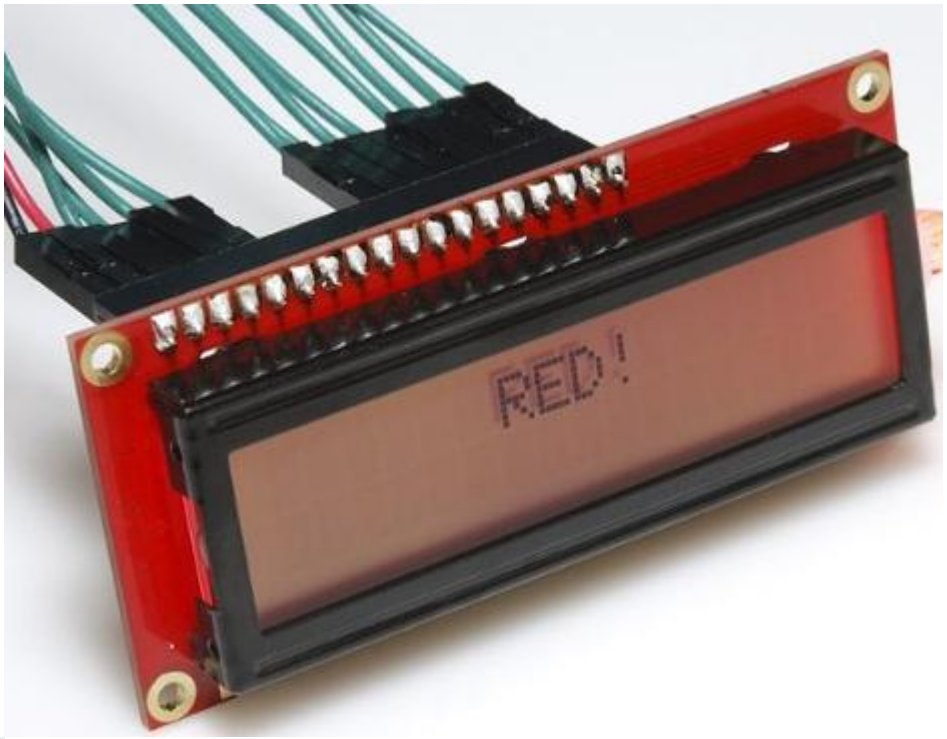
+5V WHITE

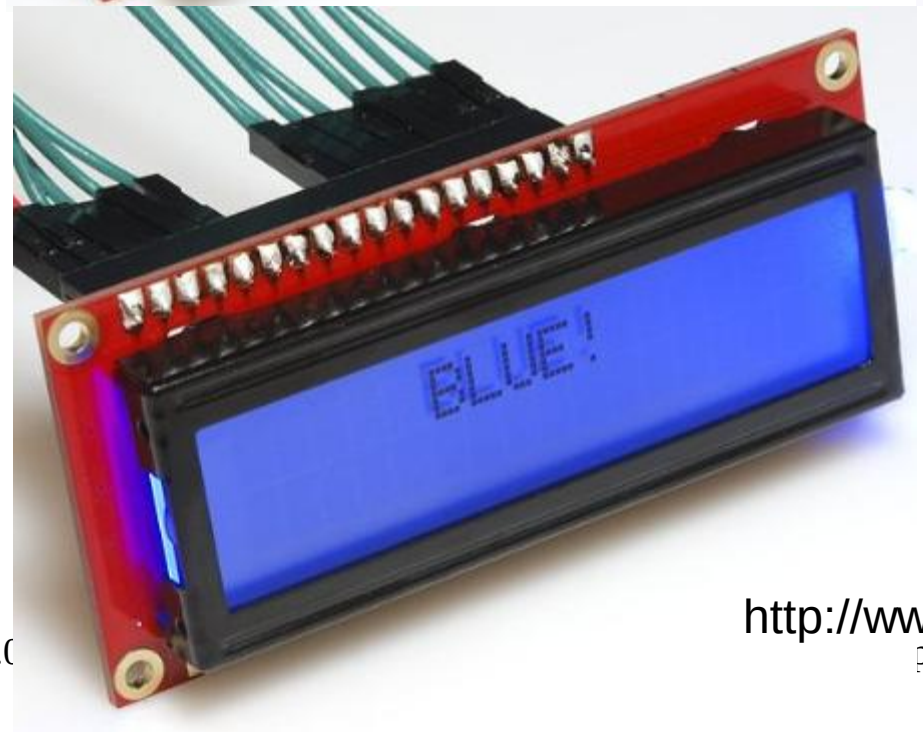
K

existiert nicht

existiert nicht


GREEN!


RED!


BLUE!

Technical Details LCD-10862 :

- **16 characters** wide, **2 rows**
- Black text on **multi-color background**
- Connection port is 0.1" pitch, single row for easy breadboarding and wiring
- Pins are documented on the back of the LCD to assist in wiring it up
- Single RGB LED backlight included can be dimmed easily with a resistor or PWM and uses much less power than LCD with EL (electroluminescent) backlights
- **Can be fully controlled with only 6 digital lines!** (Any analog/digital pins can be used) and 3 PWM pins for the backlight
- Built in **character set supports English/Japanese** text, see the **HD44780** datasheet for the full character set
- Up to **8 extra characters can be created** for custom glyphs or ’foreign’ language
- support

Interface des Displays nach Industriestandard

RS	H/L	H : Data, L : Instruction code
R/W	H/L	H : Read mode, L : Write mode
E	H, H → L	Chip enable signal
DB0	H/L	Data bit 0
DB1	H/L	Data bit 1
DB2	H/L	Data bit 2
DB3	H/L	Data bit 3
DB4	H/L	Data bit 4
DB5	H/L	Data bit 5
DB6	H/L	Data bit 6
DB7	H/L	Data bit 7

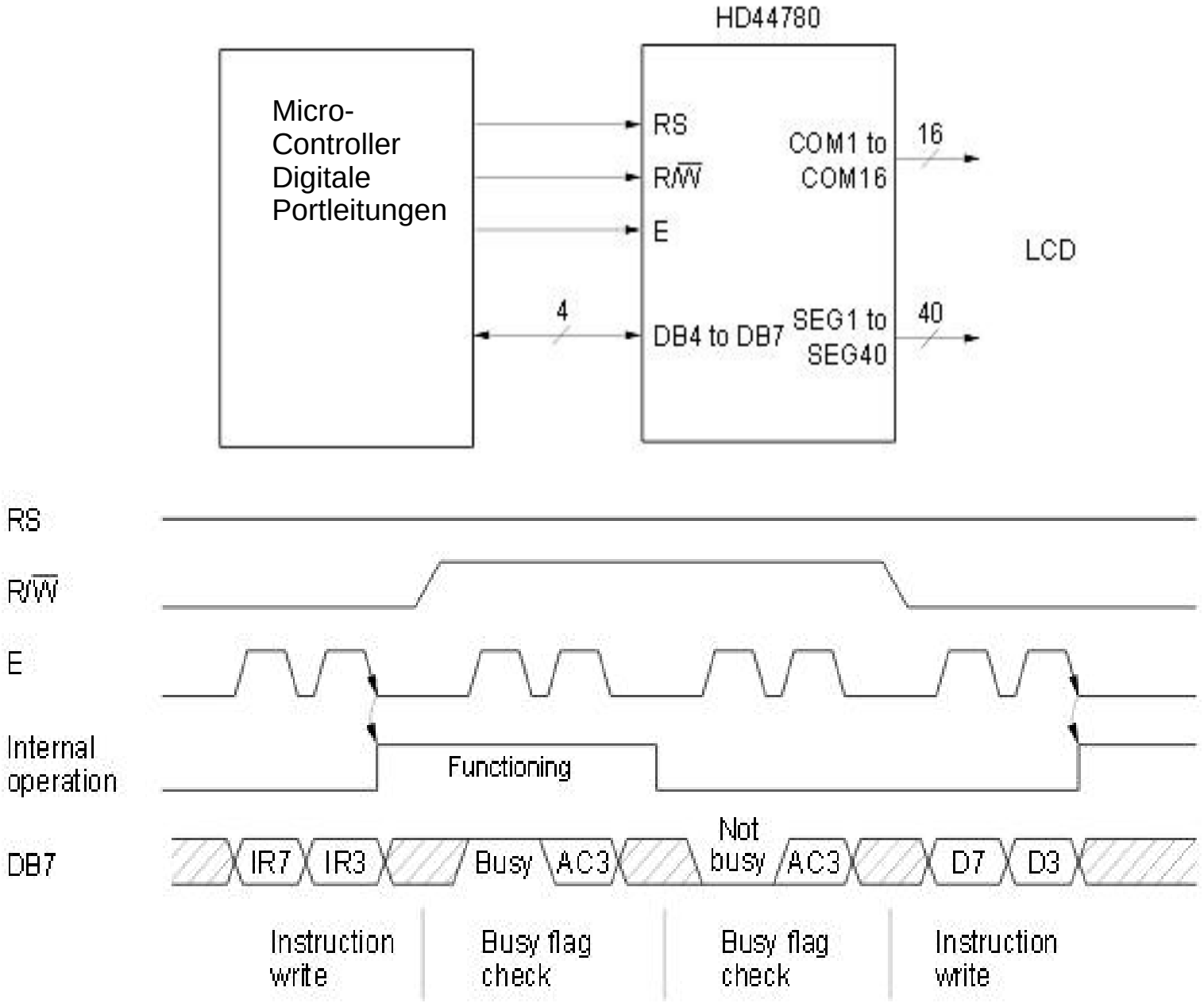
Table 1 Register Selection

RS	R/$\overline{W}$	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB6)
1	0	DR write as an internal operation (DR to DDRAM or CGRAM)
1	1	DR read as an internal operation (DDRAM or CGRAM to DR)

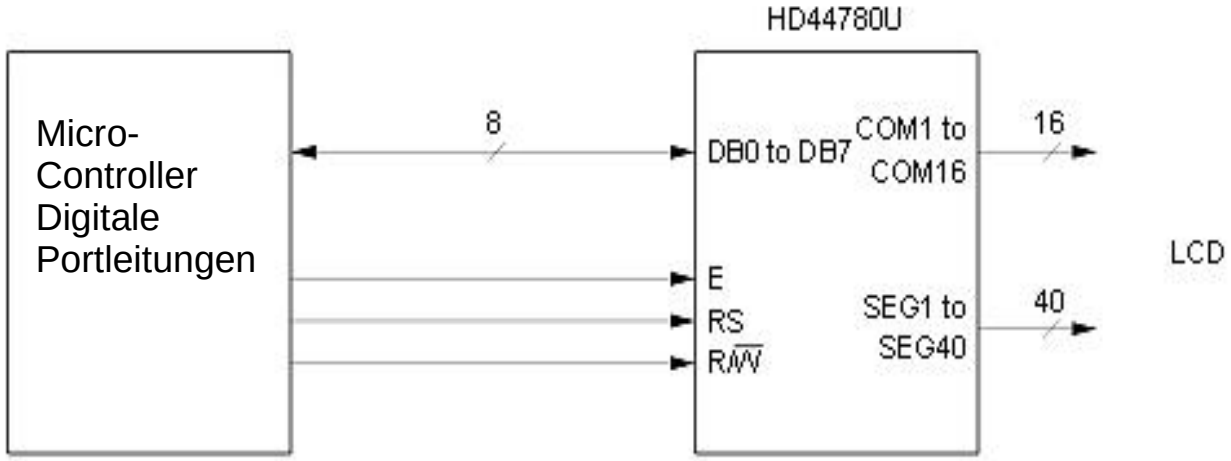
Schreiben in Kommandoregister

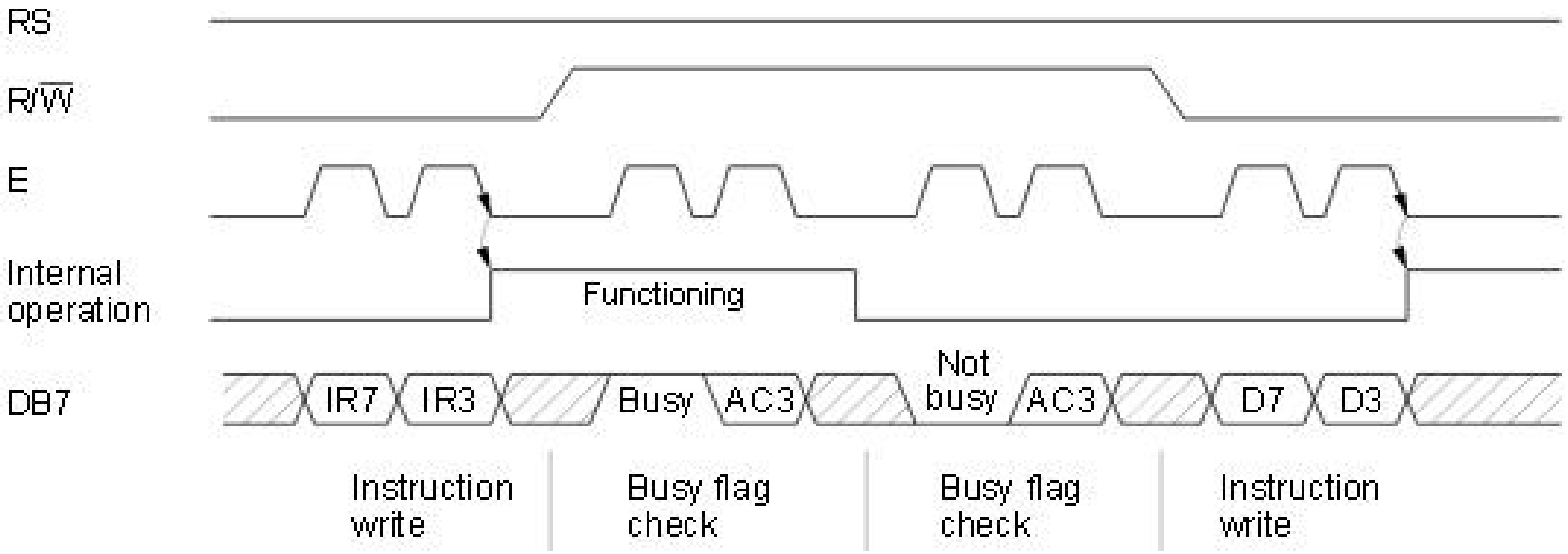
Initialisierungsdaten oder Grafik Daten schreiben

4-Bit-Schnittstelle



8-Bit-Schnittstelle

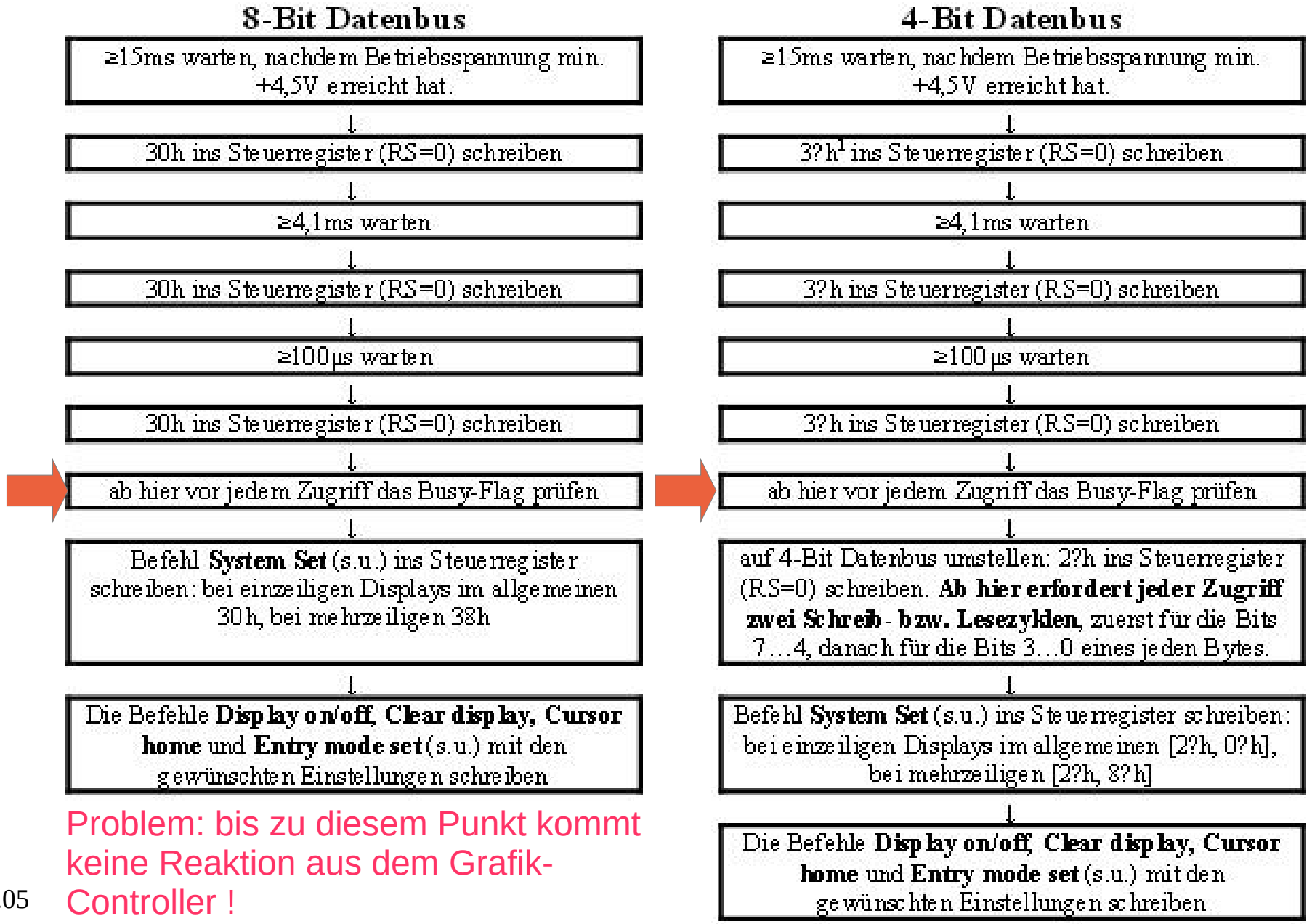




Kommandos an den Grafik-Controller

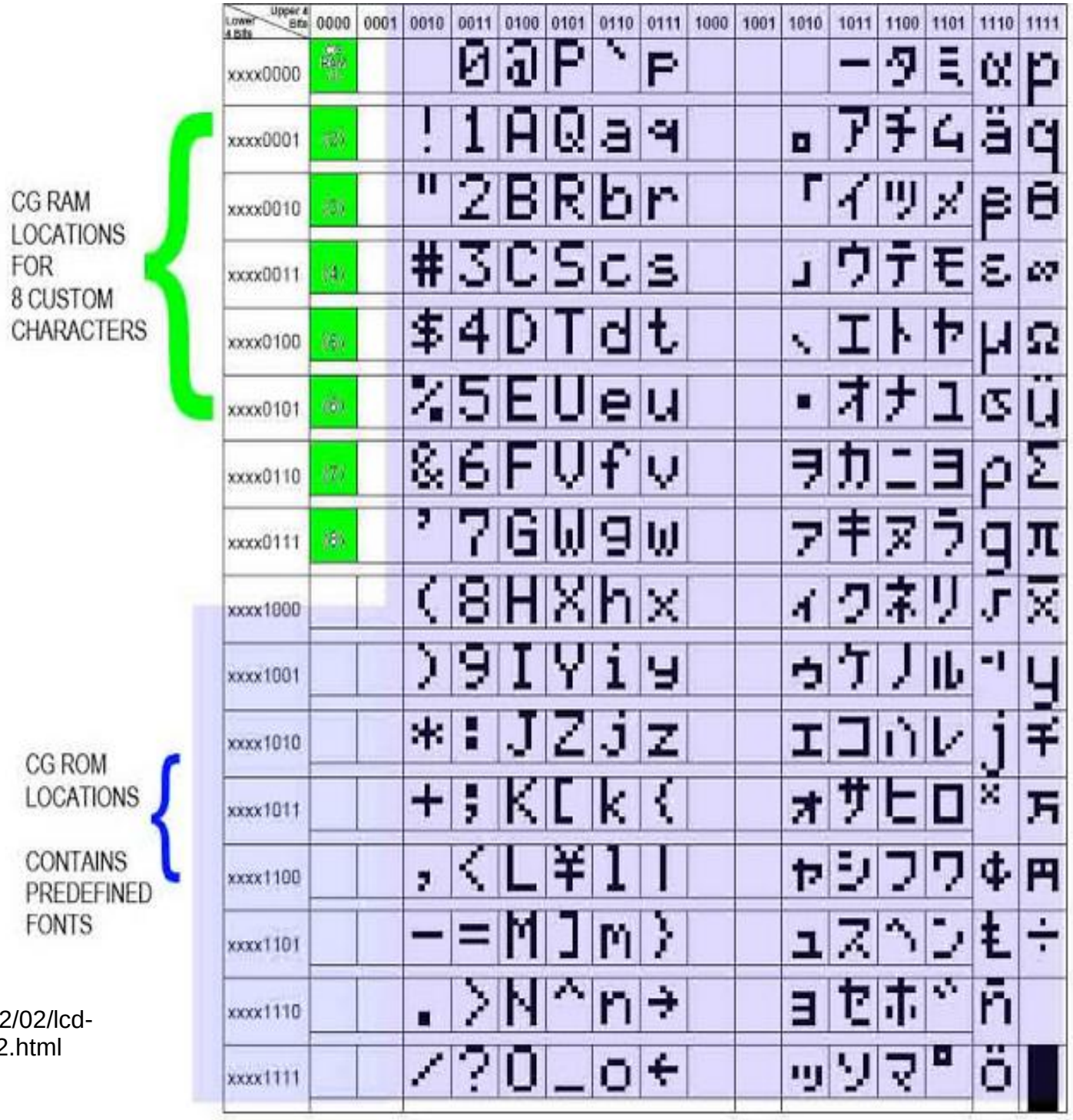
Befehl	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Funktion
Clear display	0	0	0	0	0	0	0	0	0	1	Anzeige löschen
Cursor home	0	0	0	0	0	0	0	0	1	*	Platziert den Cursor an DD-RAM-Adresse 0
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	I/D=1: vorwärts/inkrementieren/rechts I/D=0: rückwärts/dekrementieren/links S=1: Die Anzeige wird nach dem Schreiben eines Zeichens jeweils um eine Stelle entsprechend I/D verschoben S=0: Der Cursor wird nach dem Schreiben eines Zeichens jeweils um eine Stelle entsprechend I/D verschoben
Display on/off	0	0	0	0	0	0	1	D	C	B	D=1/0: Display ein/aus C=1/0: Unterstrich-Cursor ein/aus B=1/0: Blinkender Cursor ein/aus
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Verschiebt die Anzeige (S/C=1) oder den Cursor (S/C=0) um eine Stelle nach rechts (R/L=1) oder nach links (R/L=0)
System set	0	0	0	0	1	DL	N	F	*	*	DL=0: 4-Bit Ansteuerung über D4…D7 DL=1: 8-Bit Ansteuerung N=1: 2 oder 4 Displayzeilen N=0: 1 Displayzeile F=1: 5×10-Zeichenbox F=0: 5×7-Zeichenbox
Set CG-RAM address	0	0	0	1	A5	A4	A3	A2	A1	A0	Stellt die Schreibadresse (0…63) ins Zeichengenerator(CG)-RAM ein. Die nachfolgenden Zugriffe auf das Datenregister greifen auf das CG-RAM zu.
Set DD-RAM address	0	0	1	A6	A5	A4	A3	A2	A1	A0	Stellt die Schreibadresse (0…39, 64…103) ins Display(DD)-RAM ein. Die nachfolgenden Zugriffe auf das Datenregister greifen auf das DD-RAM zu.
Read busy flag/address counter	0	1	BF	A6	A5	A4	A3	A2	A1	A0	BF=1: das Display ist beschäftigt/kein Schreib-/Lesezugriff möglich BF=0: Display bereit/Zugriff möglich A6…A0: aktuelle Adresse im CG- oder DD-RAM
Write Data	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Schreibt Daten in CG- oder DD-RAM
Read Data	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Liest Daten aus dem CG- oder DD-RAM

Initialisierung
4/8-Bit-Schnittstelle



Problem: bis zu diesem Punkt kommt keine Reaktion aus dem Grafik-Controller !

Characterset HD44780



Quelle:
http://www.circuitvalley.com/2012/02/lcd-custom-character-hd44780-16x2.html

Beispiel für die Anwendung kundenspezifischer Symbole zur Batterie-Überwachung

Quelle: http://www.circuitvalley.com/2012/02/lcd-custom-character-hd44780-16x2.html

Online-Generatoren zur Umwandlung eines Pixelmusters in Datenwerte

http://www.geocities.com/dinceraydin/lcd/gfxcalc88.htm

http://www.quinapalus.com/hd44780udg.html

http://omerk.github.io/lcdchargen/

Die LiquidChristal Lib

Man kann durch Port-Operationen an einzelnen Port-Bits bzw. kompletten 8-Bit-Port die beschriebenen Signale anlegen. Es gibt für viele Mikrorechner und Programmier-sprachen jedoch Libraries, die mit fertigen Routinen hier wesentliche Unterstützung bieten.

Bei Arduino ist es die ***LyquidChristal***-Library :

LyquidChristal (1)

Funktion	**Bedeutung**	**Syntax**
LiquidCrystal()	Definition des Interface zum Mikrorechner	*LiquidCrystal(rs, enable, d4, d5, d6, d7)* *LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)* *LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)* *LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)*
begin()	Definition der Anzeigegröße	*lcd.begin(cols, rows)*
clear()	Anzeige löschen und Cursor auf Position links oben setzen	*lcd.clear()*
home()	Cursor auf Position links oben setzen	*lcd.home()*

Funktion	**Bedeutung**	**Syntax**
setCursor()	Cursor auf eine Position setzen	*lcd.setCursor(col, row)*
write()	Character auf Cursor-Position schreiben	*number = lcd.write(data)*
print()	Ausdruck von Text und Zahlen	*number = lcd.print(data)* *number =lcd.print(data, BASE)* *data: (char, byte, int, long, or string)* *BASE: BIN binary (base 2), DEC decimal (base 10), OCT octal (base 8), HEX hexadecimal (base 16).*
cursor()	Cursor-Markierung einschalten (blinkender Unterstrich)	*lcd.cursor()*
noCursor()	Cursor-Markierung ausschalten	*lcd.nocursor()*

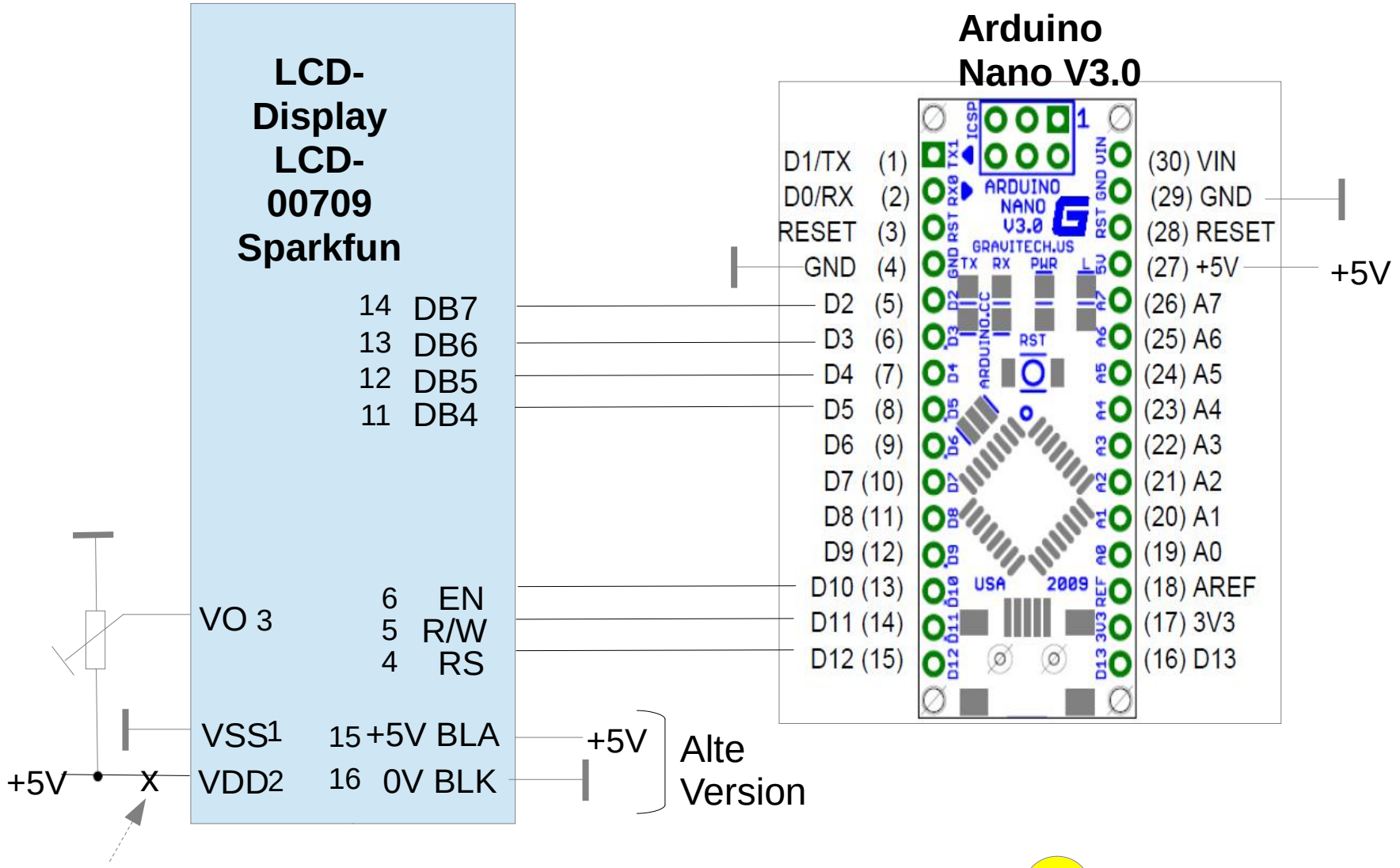
LyquidChristal (3)

Funktion	**Bedeutung**	**Syntax**
blink()	Cursor-Markierung blinken einschalten	*lcd.blink()*
noBlink()	*Cursor-Markierung blinken ausschalten*	*lcd.noBlink()*
display()	Display einschalten	*lcd.display()*
noDisplay()	Display ausschalten	*lcd.noDisplay()*
scrollDisplayLeft()	Textinhalt eine Position nach links scrollen (Beide Zeilen !)	*lcd.scrollDisplayLeft()*
scrollDisplayRight()	Textinhalt eine Position nach rechts scrollen (Beide Zeilen !)	*lcd.scrollDisplayRightt(()*
autoscroll()	Automatisches Scrollen bei Eingabe eines neuen Zeichens nach links (!) ein-schalten,	*lcd.autoscroll()*

Funktion	**Bedeutung**	**Syntax**
noAutoscroll()	Automatisches Scrollen ausschalten	*Lcd.noAutoscroll()*
leftToRight()	Schreib-Richtung link → rechts einstellen	*lcd.leftToRight()*
rightToLeft()	Schreib-Richtung rechts → links einstellen	*lcd.rightToLeft()*
createChar()	Anwenderspezif. Character definieren	lcd.createChar(num, data)

Diese Library lässt sich direkt aus der Entwicklungsumgebung unter Reference/Libraries nachschlagen.

Schaltung der Character-Display-Experimente



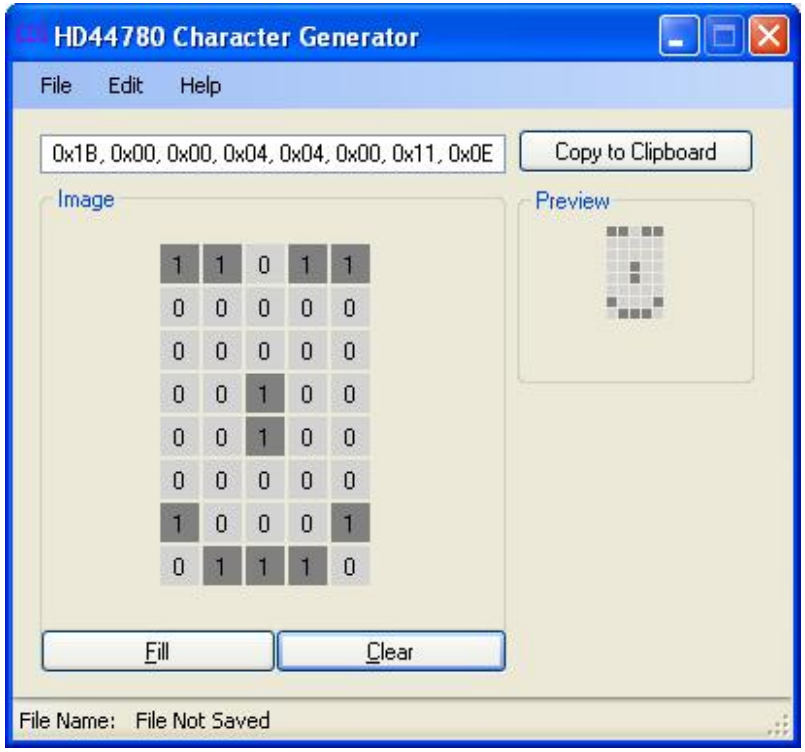
Software der Experimente

Displ3Char1

Diese Programm zeigt die typische Anwendung beim Umgang mit Texten auf dem Display. Man kann nach Definition der Interface-Pins und der Einstellung des Zeichenformats (z.B. 20x2) eine Cursorposition setzen und von dort aus schreiben. Beim Schreiben wird der Cursor nach jedem Zeichen weitergesetzt. Der Cursor kann sichtbar gemacht werden.
Wenn man Teile des Textes (z.B. Messwerte) aktualisieren will, dann muss man sie an der entsprechenden Position überschreiben. Löschen ohne neuen Inhalt erfolgt durch Überschreiben mit Leerzeichen.
Die LiquidChristal-Library bietet Scrolling-Funktionen, die den geschriebenen Text ohne Neuschreiben verschieben können. Leider haben diese Funktionen doch einige Begrenzungen:
- Wenn Zeichen über den Rand des Displays geschoben werden, dann werden sie in den insgesamt 30 Character/Zeile grossen Anzeigespeicher geschoben. Von dort können sie auch zurückgeholt werden. Wird über die Grenze 30 hinaus geschoben, ***erst dann erscheint dieser Text in der nächsten Zeile (!).***
- Das gilt leider auch für Auto-Scrolling.
- Steht in beiden Zeilen bereits Text, so werden beim Scrolling BEIDE Zeilen gleichmäßig verschoben. Ein Fließtext ähnlich wie bei einem Datenterminal wird von der Library nicht unterstützt.

Displ3Char1

Dieses Programm zeigt den Umgang mit kundenspezifischen Symbolen bei Controllern, die zum HD44870 kompatibel sind.
Ein Teil des ROMs, in dem die Character als Punktmuster abgelegt sind, ist als RAM ausgelegt. Es sind maximal 8 Zeichen vom Anwender dort abzulegen. Die Adressen (Zeichenwerte) dürfen nur im Bereich 1...8 liegen !
Bei der Erstellung der Muster helfen im Internet Online arbeitende oder als Freeware ladbare Mustergeneratoren.



Diese Mustergeneratoren erzeugen aus den gezeichneten Punktmustern den Inhalt für Datenarrays, die diese Zeichen vorhalten. Dio Übernahme erfolgt über das Clipboard.
Diese zeichen werden wie normal Zeichen aus dem ROM mit lcd.write() an den gewünschten Cursorstellen dargestellt.

Displ3Char1 (1)

```
/*  Displ3Char1
  LCD-Anzeige mit 16x2 Zeichen
  ============================

  Anschluss der LCD-Anschlüsse an den Arduino:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 10
 * LCD D4 pin to digital pin 9
 * LCD D5 pin to digital pin 8
 * LCD D6 pin to digital pin 7
 * LCD D7 pin to digital pin 6
 * LCD R/W pin an GND oder pin 11
 * Kontrast: Poti zwischen +5V
 * und GND an LCD VO pin (pin 3)
 */

#include <LiquidCrystal.h>
//Maximale Anzahl Zeichen pro Zeile
#define COL_MAX 16
```

Displ3Char1 (2)

```
// Initialisierung der Library und
// Zuweisung der verwendeten Pins
LiquidCrystal lcd(12, 10, 9, 8, 7, 6);
byte n = 0; // Zeichenzähler

void setup()
{
  // Pin 11 wird als Ausgang für das R/W-
  // Signal verwendet
  pinMode(11, OUTPUT);
  //Festlegung der Zeichenanzahl und der Zeilenanzahl
  lcd.begin(16, 2);
  // R/W auf LOW legen (nur Schreiben moeglich)
  digitalWrite(11,LOW);
```

Displ3Char1 (3)

```
  //Text auf das Display schreiben.
  //Zeichen 0 in Zeile 0
  lcd.print("hello, world!");
  //Cursor auf zeichen 0 Zeile 1 setzen
  lcd.setCursor(0, 1) ;
  //Weiteren Text in Zeile 2 schreiben
  lcd.print("CQ DE DK4AQ");
  n = 0;
}
```

Displ3Char1 (4)

```
void loop()
{
  delay(1000);
  // Beide Zeilen(!) um 1 Stelle nach rechts scrollen
  lcd.scrollDisplayRight();
  n++;
  //Nach COL_MAX Schiebvorgängen Disolay loeschen
  //und neu schreiben
  if( n > COL_MAX)
  {
    n = 0;
    //DisolaY loeschen, Cursor auf Zeichen 0
    //und Zeile 0 setzen
    lcd.clear();
    lcd.print("hello, world!");
    lcd.setCursor(0, 1) ;
    lcd.print("CQ DE DK4AQ");
  }
}
```

Displ3Char2 (1)

```
/* Displ3Char2
 Nutzung von selbstdefinierten Zeichen
 ======================================
 Alle Pins sind wie im letzten Beispiel definiert
*/
#include <LiquidCrystal.h>

// Initialisierung der Library und
// Zuweisung der verwendeten Pins
LiquidCrystal lcd(12, 10, 9, 8, 7, 6);
int v = 0; // Anzeige Wert

byte EIN_BALKEN[8]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F};
byte ZWEI_BALKEN[8]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x1F};
byte DREI_BALKEN[8]= {0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x1F, 0x1F};
byte VIER_BALKEN[8]= {0x00, 0x00, 0x00, 0x00, 0x1F, 0x1F, 0x1F, 0x1F};
byte FUENF_BALKEN[8]= {0x00, 0x00, 0x00, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F};
byte SECHS_BALKEN[8]= {0x00, 0x00, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F};
byte SIEBEN_BALKEN[8]= {0x00, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F};
byte ACHT_BALKEN[8]= {0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F, 0x1F};
```

Displ3Char2 (2)

```
byte SMILEY[8] = {0x00, 0x1B, 0x00, 0x04, 0x04, 0x11, 0x0E, 0x00};
byte ANGRY[8] = {0x1B, 0x00, 0x04, 0x04, 0x00, 0x0E, 0x11, 0x00};
byte BAH[8] = {0x1B, 0x00, 0x04, 0x04, 0x00, 0x1F, 0x11, 0x0E};

void setup()
{
  pinMode(11, OUTPUT);
  digitalWrite(11,LOW);
  Serial.begin(9600);
  lcd.clear();
  lcd.begin(16,2);

  // Maximal 8 kundenspez. Caharacter moeglich !
  lcd.createChar(1, EIN_BALKEN);
  lcd.createChar(2, ZWEI_BALKEN);
  lcd.createChar(3, DREI_BALKEN);
  lcd.createChar(4, VIER_BALKEN);
  lcd.createChar(5, FUENF_BALKEN);
  lcd.createChar(6, SECHS_BALKEN);
  lcd.createChar(7, SIEBEN_BALKEN);
  lcd.createChar(8, ACHT_BALKEN);
```

Displ3Char2 (3)

```
  lcd.clear();
  lcd.print("Bargrafen ");

  v = 1;
  //Ausgabe der Symbolstufen nebeneinander
  lcd.setCursor(0,1);
  //Aufsteigend
  while(v<8)
  {
    lcd.write(v);
    v++;
    delay(300);
  }
  //Absteigend
  while(v>0)
  {
    lcd.write((byte)v);
    v--;
    delay(300);
  }
}
```