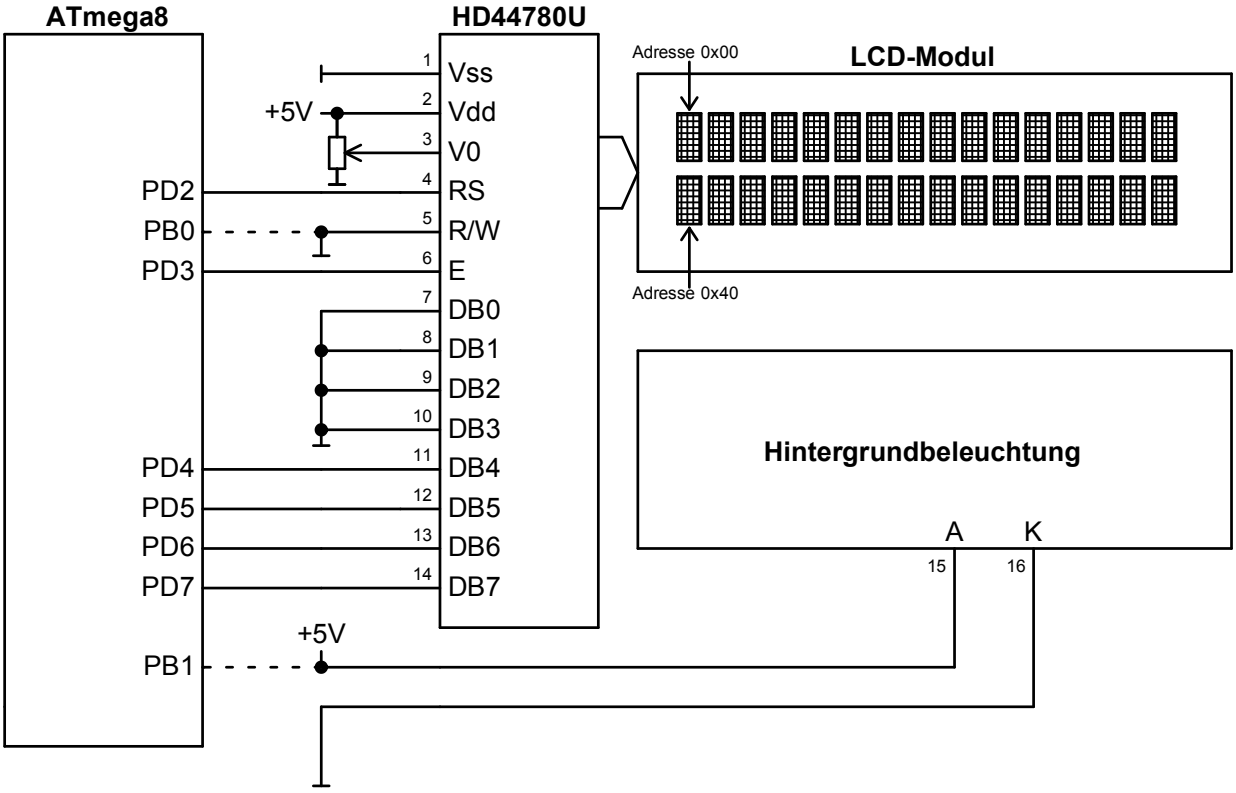
4.9. Eine LCD-Anzeige

Dieses Kapitel beschreibt die Anwendung einer LCD-Anzeige mit zwei Zeilen zu je 16 Zeichen.

Derartige Anzeigen sind meistens mit einem integrierten Controller HD44780U oder einem kompatiblen Typ ausgestattet, welcher von den binären Ausgängen eines Mikrocontrollers angesteuert werden kann. Das LCD-Modul besitzt insgesamt 16 Anschlüsse. Die Übertragung von Daten kann wahlweise über ein 8-Bit- oder 4-Bit-Interface erfolgen. Die folgende Bild zeigt die Anschlussbelegung an den ATmega8 mit einem 4-Bit-Datenbus.



Beschreibung der Anschlüsse:

Vss, Vdd	Spannungsversorgung, GND - +5 Volt
V0	Kontrastregelung
RS	Register Select; 0 = Instruction register (Schreiben) – Busy flag und Address counter (Lesen) 1 = Data register (Schreiben und Lesen)
R/W	Auswahl Lesen/Schreiben 0 = Schreiben 1 = Lesen
E	Lesen/Schreiben ausführen, Impulsdauer mind. 230 ns
DB0...DB3	Datenbytes; unteres Halbbyte (nicht benötigt bei 4-Bit-Interface)
DB4...DB7	Datenbytes; oberes Halbbyte bei 8 Bit-Interface; Zur Datenübertragung bei 4-Bit-Interface zuerst oberes und dann unteres Halbbyte; DB7 kann als Busy flag (BF) verwendet werden
A	Hintergrundbeleuchtung Anode (+ 5 Volt)
K	Hintergrundbeleuchtung Kathode (GND)

Der Mikrocontroller hat Zugriff auf zwei Register des HD44780U: Das Befehlsregister (IR) und das Datenregister (DR). Mit dem Signal R/W wird unterschieden, ob Daten geschrieben (R/W=0) oder gelesen (R/W=1) werden sollen. Wenn nur Daten zum Display gesendet werden sollen, kann dieser Anschluss fest auf Low geschaltet werden. Alternativ wird dieser Anschluss mit PB0 vom Mikrocontroller angesteuert.

Mit RS=0 erfolgt der Zugriff auf das Befehlsregister (IR). Zu den möglichen Befehlen gehören Display löschen, Bewegungsrichtung des Cursors festlegen, Zeilenanzahl festlegen (ein-/zweizeilig), Ausgabeadresse festlegen, Display ein-/ausschalten, Cursor ein-/ausschalten, Cursor Blinken ein-/ausschalten, u.a. .

Mit RS=1 erfolgt der Zugriff auf das Datenregister. Damit werden z.B. ASCII-Zeichen an festgelegter Adresse gesendet. Bei einem 4-Bit-Interface erfolgt der Zugriff nur über DB4 bis DB7. Dabei werden zuerst das höhere Halbbyte und dann das untere Halbbyte gesendet. Es ist zu beachten, dass die erste Stelle von Zeile 1 die Adresse 0x00 und die erste Stelle von Zeile 2 die Adresse 0x40 hat.

Ein Befehl wird ausgeführt durch einen Impuls von mindestens 230 ns Dauer am Anschluss E (Enable).

Auf die Möglichkeit eigene Charakterzeichen zu erzeugen wird an dieser Stelle nicht eingegangen (siehe Datenblatt!).

Die Hintergrundbeleuchtung kann über den Anschluss A (Anode) mit + 5 Volt eingeschaltet werden. Alternativ erfolgt das Einschalten über PB1 des Mikrocontrollers.

Die folgende Tabelle liefert eine Übersicht über die verfügbaren Kommandos.

Displaykommandos

Command	*RS*	*R/W*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*Execution time*	*Remark*
DISPLAY CLEAR	0	0	0	0	0	0	0	0	0	1	1,64 ms	
RETURN HOME	0	0	0	0	0	0	0	0	1	X	1,64 ms	Cursor move to first digit
ENTRY MODE SET	0	0	0	0	0	0	0	1	I/D	SH	42 µs	I/D: Set Cursor move direction I/D: 1 Increase; 0 Decrease SH: Specifies shift of display SH: 1 Display is shifted; 0 Display is not shifted
DISPLAY ON/OFF	0	0	0	0	0	0	1	D	C	B	42 µs	Display D: 1 Display on; 0 Display off Cursor C: 1 Cursor on; 0 Cursor off Blinking B: 1 Blinking on; 0 Blinking off
SHIFT	0	0	0	0	0	1	S/C	R/L	X	X	42 µs	S/C: 1 Display shift; 0 Cursor move R/L: 1 Right shift; 0 Left shift
SET FUNCTION	0	0	0	0	1	DL	N	F	X	X	42 µs	DL: 1 8 bits interface; 0 4 bits interface N: 1 2 line display; 0 1 line display F: 1 5 x 10 dots; 0 5 x 7 dots
SET CGRAM ADDRESS	0	0	0	1	CGRAM Adress (corresponds to cursor address)						42 µs	CG RAM Data is sent and received after this setting
SET DDRAM ADDRESS	0	0	1	DDRAM address							42 µs	DD RAM Data is sent and received after this setting
READ BUSY FLAG & ADDRESS	0	1	BF	Adress Counter used for both DD & CG RAM address							0 µs	BF: 1 Busy; 0 Ready -Reads BF indication internal operating is beeing performed -Reads address counter contents
WRITE DATA	1	0	Write Data								46 µs	Write data into DD or CG RAM
READ DATA	1	1	Read Data								46 µs	Read data from DD or CG RAM

Die Initialisierung der Anzeige:

Im Datenblatt des HD44780U ist beschreiben, wie die Anzeige nach dem Programmstart zu initialisieren ist. Dabei ist zu beachten, dass jedes LCD-Kommando eine gewisse Zeit erfordert. Die Ausführungszeiten sind in obenstehender Tabelle aufgeführt.

Nach Anlegen der Betriebsspannung ist eine Wartezeit von 50 ms einzuhalten. Danach ist an DB4...DB7 (verbunden mit PD4...PD7 des ATmega8) dreimal der Wert 0x30 mit entsprechenden Wartezeiten zu senden. Die Übertragung des Kommandos erfolgt durch einen Impuls von mindestens 230 ns Dauer an EN (verbunden mit PD3).

Durch Senden des Kommandos 0x20 wird danach in den 4-Bit-Modus geschaltet.

Jetzt ist die Anzeige bereit Kommandos im 4-Bit-Modus anzunehmen. Mit dem Kommando SET FUNCTION wird mit Bit 4 (DL) festgelegt ob ein 4-Bit- oder 8-Bit-Interface verwendet wird. Für 4 Bit ist DL = 0 zu setzen. Mit Bit 3 (N) wird die Anzahl der Zeilen festgelegt. Für eine zweizeilige Anzeige muss dieses Bit auf 1 gesetzt werden. Schließlich kann bei einzeiliger Ausgabe mit Bit 2 (F) noch festgelegt werden, ob die Ausgabe mit einer Matrix von 5 mal 7 oder 5 mal 10 Punkten erfolgen soll. Für F = 0 erfolgt die Anzeige mit 5 mal 7 Punkten. Aus diesen Vorgaben ergibt sich das Kommandowort zu 0b00101000 = 0x28.

Mit dem Kommando DISPLAY ON/OFF wird mit D = 1 das Display eingeschaltet. Mit C = 1 wird der Cursor eingeschaltet und mit B = 1 wird dieser zum Blinken veranlasst. Das Kommandowert ist also 0b00001111 = 0x0F.

Mit dem Kommando ENTRY MODE SET wird das Verhalten der Anzeige festgelegt. Mit I/D = 0 und SH = 0 wird festgelegt, dass der Cursor nach jeder Zeichenausgabe um eine Stelle nach rechts verschoben wird. Das Kommandowort ist 0b00000100 = 0x04.

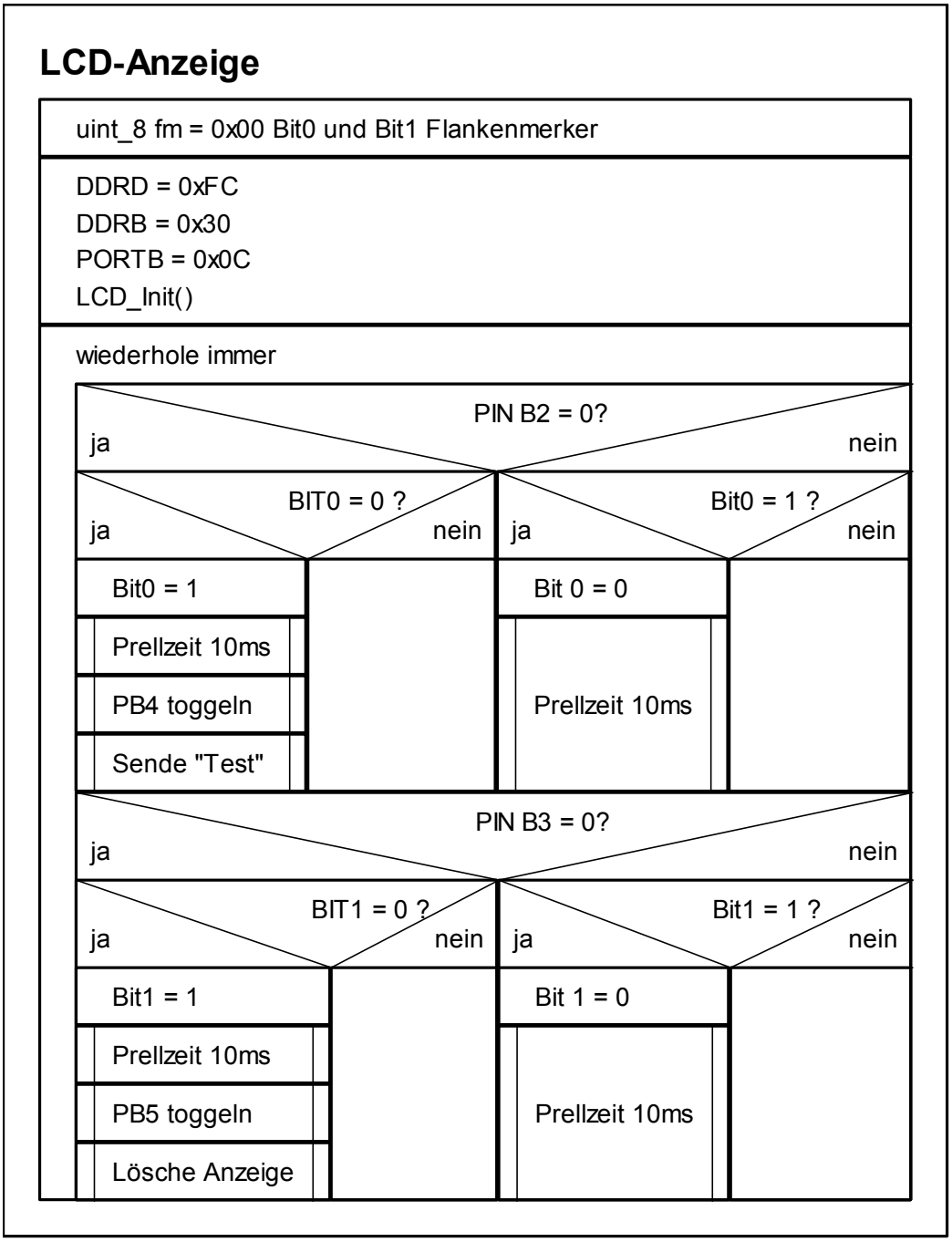
Das Kommando DISPLAY CLEAR (0X01) löscht die Anzeige und setzt den Cursor an die Startposition.

Das Kommando SET DDRAM ADDRESS wird die Position festgelegt, wo das nächste Zeichen ausgegeben wird. Mit WRITE DATA werden dann die Zeichen (ASCII-Code) ausgegeben und der Cursor geht an die nächste Position.

Beispiel 4.9.1: LCD-Anzeige

Zusätzlich zur Anzeige werden zwei Taster an PB2 und PB3 sowie 2 LEDs an PB4 und PB5 angeschlossen. Wenn Taster 1 betätigt wird soll in der Anzeige der Text ‚Test' ausgegeben werden und die LED 1 umschalten. Wenn Taster 2 betätigt wird soll das Display gelöscht werden und die LED 2 umschalten.

Struktogramm des Hauptprogramms:



```
/*********************************************************************
Datei: Beispiel_4_9_1.c
Beispiel: LCD_Anzeige.c
Funktion:
Bei Betätigung von Taster1 schaltet LED1 um und 'Test' erscheint in der Anzeige
Bei Betätigung von Taster2 schaltet LED2 um und das Display wird gelöscht
*********************************************************************
Pinbelegung:
PD2 = RS (A)
PD3 = E  (A)
PD4 = DB4 (A)
PD5 = DB5 (A)
PD6 = DB6 (A)
PD7 = DB7 (A)
PB0 = R/W (A) (oder fest auf LOW)
PB1 = Hintergrundbeleuchtung (A) (Anode)
PB2 = Taster 1
PB3 = Taster 2
PB4 = LED1
PB5 = LED2
*****************************************/
#include <avr/io.h>
#define F_CPU 3686400UL     /* Quarz mit 3.6864 Mhz  */
#include <util/delay.h>

/********** EN-Impuls erzeugen *************/
void LCD_Enable()
{
      PORTD |= (1<<PD3);
      _delay_us(2);
      PORTD &= ~(1<<PD3);
}

/******** Kommando zur LCD-Anzeige senden **/
void LCD_Command(unsigned char val)
{
      PORTD &=~(1<<PD2);      //RS = Low
      PORTD &= 0x0f;          //oberes Nibble löschen
      PORTD |= val & 0xf0;    //oberes Nibble senden
      LCD_Enable();
      PORTD &= 0x0f;          //oberes Nibble löschen
      PORTD |= (val<<4)&0xf0; //unteres Nibble senden
      LCD_Enable();
      _delay_us(50);          //50 µs warten
}
```

```
/******** LCD-Anzeige initialisieren **************/
void LCD_Init()
{
      PORTD |= (1<<PD4)|(1<<PD5);   //PORTD = 0x30
      _delay_ms(50);                //50 ms warten
      LCD_Enable();
      _delay_ms(5);                 //5 ms warten
      LCD_Enable();
      _delay_us(100);               //100 µs warten
      LCD_Enable();
      _delay_ms(5);                 //5 ms warten
      PORTD &= ~(1<<PD4);           //PORTD = 0x20 - 4-Bit-Modus einschalten
      LCD_Enable();
      _delay_ms(5);                 //5 ms warten
      LCD_Command(0x28);            //4-Bit-Interface - 2 Zeilen
      LCD_Command(0x0f);            //Display ein - Cursor ein - Cursor blinken
      LCD_Command(0x04);            //Cursor Inkrement - kein Schieben
      LCD_Command(0x01);            //Display löschen
      _delay_ms(5);                 //warte 5 ms
}
/********** Datenbyte zur LCD-Anzeige senden **********/
void LCD_SendData(uint8_t val)
{
      PORTD |=(1<<PD2);             //RS = High
      PORTD &= 0x0f;                //oberes Nibble löschen
      PORTD |= val & 0xf0;          //oberes Nibble senden
      LCD_Enable();
      PORTD &= 0x0f;                //oberes Nibble löschen
      PORTD |= (val<<4)&0xf0;       //unteres Nibble senden
      LCD_Enable();
      PORTD &=~(1<<PD2);            //RS = Low
      _delay_us(50);                //50 µs warten
}
/******** Zeichen an Adresse der LCD-Anzeige senden ********/
void LCD_SendDataAddr(uint8_t addr, uint8_t data)
{
      LCD_Command(0x80|addr);       //setze DDRAM-Adresse
      LCD_SendData(data);           //Daten senden
}

/******** String an Adresse ausgeben ***************/
void LCD_SendStringAddr(uint8_t addr,char* str)
{
      uint8_t i=0;
      LCD_Command(0x80|addr);       //setze DDRAM-Adresse
      while(str[i]) LCD_SendData(str[i++]);   //sende Daten bis Zeichen 0
}
/*********** Display löschen ***************/
void LCD_Clear()
{
      LCD_Command(0x01);            //Display löschen
      _delay_ms(5);                 //warte 5 ms
}
```

```
/************ Hauptprogramm ************/
int main(void)
{
	uint8_t fm=0x00;		//Flankenmerker
	DDRD = 0xfc;			//PD2...PD7 zur Ausgabe
	DDRB |= (1<<PB4)|(1<<PB5);	//PB4...PB5 zur Ausgabe
	PORTB |= (1<<PB2)|(1<<PB3);	//Pullup für PB2...PB3
	LCD_Init();
/********** Main-Schleife ******************/
	while(1)
	{
	if(!(PINB & 0x04))		//PINB2 = 0?
		{
		if(!(fm & 0x01))	//Flankenmerker = 0?
			{
			fm |= (1<<0);	//Flankenmerker setzen
			_delay_ms(10);	//Prellzeit
			PORTB ^=(1<<PB4); //PB4 toggeln
			LCD_SendStringAddr(0x06,"Test");   //String ausgeben
			}
		}
	else				//wenn PINB2 = 1
		{
		if(fm & 0x01)		//Flankenmerker = 1?
			{
			fm &=~(1<<0);	//Flankenmerker zurück setzen
			_delay_ms(10);	//Prellzeit
			}
		}
if(!(PINB & 0x08))			//PINB3 = 0?
		{
		if(!(fm & 0x02))	//Flankenmerker = 0?
			{
			fm |= (1<<1);	//Flankenmerker setzen
			_delay_ms(10);	//Prellzeit
			PORTB ^=(1<<PB5); //PB5 toggeln
			LCD_Clear();	//Display löschen
			}
		}
	else				//wenn PINB2 = 1
		{
		if(fm & 0x02)		//Flankenmerker = 1?
			{
			fm &=~(1<<1);	//Flankenmerker zurück setzen
			_delay_ms(10);	//Prellzeit
			}
		}
	}
/*********** Ende Main-Schleife ****************/
return(0);
}
```

Übung 4.9.1:

Erweitern Sie bitte obiges Programm derart, dass nach dem Starten des Programms und nach jeder Tastenbetätigung eine Meldung über den Zustand der LEDs ausgegeben wird! In Zeile 1 des Displays soll ‚LED 1 ist an|aus' und in Zeile 2 ‚LED 2 ist an|aus' erscheinen.

Übung 4.9.2:

Am PC0 des ATmega8 wird ein Analogwert eingelesen, welcher im Klartext auf einer LCD-Anzeige von 0,00 Volt bis 5,10 Volt dargestellt werden soll (8 Bit Auflösung)