

# Ansteuerung des Grafik-LC-Display NLC-122B032 (mit PIC-Mikrocontroller)

Buchgeher Stefan

16. September 2006

# Inhaltsverzeichnis

<b>1</b>	<b>Allgemeines</b>	<b>3</b>
<b>2</b>	<b>Hardware</b>	<b>4</b>
<b>3</b>	<b>Ansteuerung des Grafik-LC-Displays</b>	<b>6</b>
3.1	Befehle zur Ansteuerung des Grafik-LC-Displays . . . . .	6
3.2	Zeitdiagramm . . . . .	7
3.3	Zeichensatz . . . . .	8
<b>4</b>	<b>Software (in Assembler)</b>	<b>10</b>
4.1	Benötigte Register und Portdefinitionen . . . . .	10
4.2	Initialisierung (Unterprogramm INIT) . . . . .	11
4.3	Unterprogramme für die Grafik-LCD-Ansteuerung . . . . .	12
4.3.1	Unterprogramm GLCD_INIT . . . . .	12
4.3.2	Unterprogramm GLCD_COMMAND . . . . .	13
4.3.3	Unterprogramm GLCD_DATA1 . . . . .	13
4.3.4	Unterprogramm GLCD_DATA2 . . . . .	14
4.3.5	Unterprogramm GLCD_CLEAR . . . . .	14
4.3.6	Unterprogramm GLCD_SETCURSOR_ZEILE . . . . .	15
4.3.7	Unterprogramm GLCD_SETCURSOR_SPALTE . . . . .	15
<b>5</b>	<b>Software (in C mit CC5X)</b>	<b>17</b>
5.1	Portdefinitionen und externe Register . . . . .	17
5.2	Initialisierung (Unterprogramm INIT) . . . . .	18
5.3	Unterprogramme für die Grafik-LCD-Ansteuerung . . . . .	18
5.3.1	Unterprogramm GLCD_INIT . . . . .	20
5.3.2	Unterprogramm GLCD_COMMAND . . . . .	20
5.3.3	Unterprogramm GLCD_DATA1 . . . . .	20
5.3.4	Unterprogramm GLCD_DATA2 . . . . .	21
5.3.5	Unterprogramm GLCD_DATA_ZEICHEN . . . . .	21
5.3.6	Unterprogramm GLCD_CLEAR . . . . .	22
5.3.7	Unterprogramm GLCD_SETCURSOR_ZEILE . . . . .	23
5.3.8	Unterprogramm GLCD_SETCURSOR_SPALTE . . . . .	23
5.3.9	Unterprogramm GLCD_ZEICHEN_5x7 . . . . .	24
5.3.10	Unterprogramm GLCD_ZEICHEN_8x14 . . . . .	24
5.3.11	Unterprogramm GLCD_ZEICHEN_10x22 . . . . .	26

<b>6</b>	<b>Demonstrationsbeispiele</b>	<b>29</b>
6.1	Hardware . . . . .	29
6.2	Nachbauanleitung (der Hardware) . . . . .	31
6.3	Softwarebeispiel 1 (“HELLO WORLD“ in Assembler) . . . . .	35
6.3.1	Listing . . . . .	35
6.3.2	Anmerkungen zur Assembler-Software . . . . .	43
6.4	Softwarebeispiel 2 in C mit CC5X . . . . .	44
6.4.1	Listings . . . . .	44
6.4.2	Anmerkungen zur C-Software . . . . .	59
<b>A</b>	<b>Layout (Demonstrationsbeispiele)</b>	<b>61</b>
<b>B</b>	<b>Stückliste</b>	<b>62</b>
<b>C</b>	<b>Zeichensätze</b>	<b>63</b>
<b>D</b>	<b>Programmieren mit ICSP</b>	<b>81</b>
<b>E</b>	<b>Umgehen der 1k-Grenze (beim CC5X-Compiler)</b>	<b>84</b>

# Kapitel 1

## Allgemeines

Alphanumerische LC-Displays sind in den letzten Jahren auch für den Hobbyelektroniker preiswert geworden und eignen sich aufgrund der geringen zusätzlichen Hardware sehr gut für die Anzeige von Werten, Informationen oder zur Eingabe von Parametern. Der Nachteil an alphanumerischen LC-Displays ist aber, dass hier keine Grafiken oder Bilder angezeigt werden können, und auch der Zeichensatz ist fix vorgegeben.

Grafik-LC-Displays sind im Allgemeinen sehr teuer. Eine Ausnahme scheint aber das hier verwendete Grafik-Display vom Typ NLC-122B032 zu sein. Es bietet eine Auflösung von 122x32 Pixel.

Nachteilig an den grafikfähigen LC-Displays ist wohl, dass sich hier kein einheitlicher Controller (so wie bei den alphanumerischen LC-Displays) durchgesetzt hat. Dies macht die Ansteuerung nicht gerade einfacher. Das hier verwendete Grafik-LC-Display verwendet den Epson-Controller SED1520.

Dieses Dokument soll zeigen, wie relativ einfach die Ansteuerung des Grafik-LC-Display (NLC-122B032) mit einem PIC-Mikrocontroller ist.

Als Programmiersprachen für die PIC-Software wurden Assembler und C (mit dem CC5X-Compiler) verwendet.

# Kapitel 2

## Hardware

Abbildung 2.1 zeigt die minimale Beschaltung des Grafik-LC-Displays.

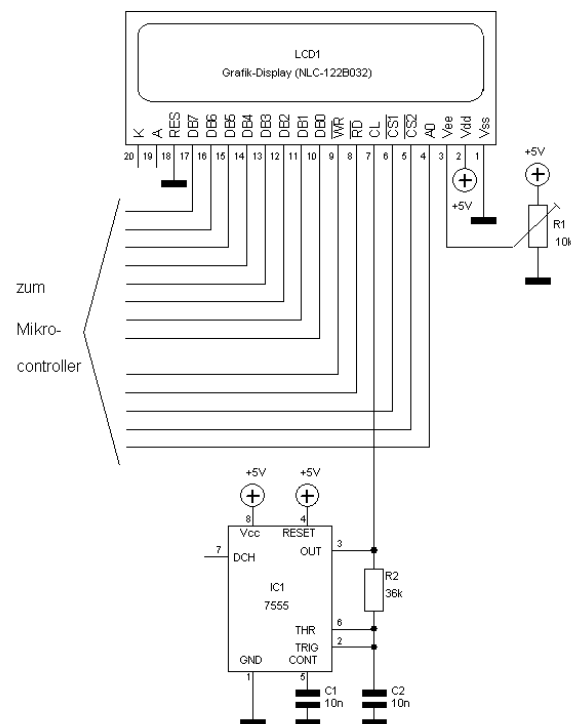


Abbildung 2.1: Grafik-LCD-Ansteuerung

Das hier verwendete Grafik-LC-Display (NLC-122B032) besitzt die folgenden Anschluss-pins:

- Spannungsversorgung ( $V_{ss}$  und  $V_{dd}$ , Pin 1 und 2): Das Grafik-LC-Display wird mit einer Betriebsspannung  $V_{dd}$  von +5V betrieben.  $V_{ss}$  ist der Masseanschluss.
- Spannung zur Kontrasteinstellung ( $V_{ee}$ , Pin 3): Die Kontrasteinstellung wird üblicherweise mit einem 10k-Trimмер realisiert.

- $A0$ , Pin 4: Dieser Anschluss bestimmt, ob es sich bei den zum Display geschriebenen Daten um Steuerbefehle für das Display ( $A0 = 0$ ), oder um am Display darstellbare Zeichen ( $A0 = 1$ ) handelt.
- Chipselekt ( $\overline{CS1}$  und  $\overline{CS2}$ , Pin 6 und 5): Mit  $\overline{CS1} = 0$  wird die rechte Displayhälfte und mit  $\overline{CS2} = 0$  die linke Displayhälfte ausgewählt.
- Taktleitung ( $CL$ , Pin 7): Mit dieser Leitung wird dem Display ein externer 2-kHz-Takt zugeführt. Dieser Takt wird mit einem Timer-IC (IC1) vom Typ 7555, einem Widerstand ( $R2$ ) und den Kondensatoren ( $C1$  und  $C2$ ) erzeugt.
- Read-Leitung ( $\overline{RD}$ , Pin 8): Dieser Anschluss bestimmt, ob Daten vom Display gelesen werden sollen.
- Write-Leitung ( $\overline{WR}$ , Pin 9): Dieser Anschluss bestimmt, ob Daten zum Display geschrieben werden.
- 8-Bit-Datenbus ( $DB0$  bis  $DB7$ , Pin 10 bis 17): Dieser Datenbus dient zur Übertragung der Daten zum/vom Display (schreiben oder lesen).
- Resetleitung ( $RES$ , Pin 18): Dieser Anschluss bestimmt, welches Interface verwendet wird. Grundsätzlich stehen zwei Interfaces zur Verfügung:
  - 80 Family MPU ( $RES = 0$ )
  - 68 Family MPU ( $RES = 1$ )

Hier wird die *80 Family MPU* verwendet. Daher kann der Resetpin auf Massepotential gelegt werden. **Wichtig:** Da sich die beiden Interfaces im Zeitverhalten unterscheiden, muss dies in der Software ebenfalls berücksichtigt werden! Da ich hier die *80 Family MPU* verwende gehe ich in dieser Dokumentation gar nicht näher auf die *68 Family MPU* ein.

- **Optional:** Hintergrundbeleuchtung ( $A$  und  $K$ , Pin 19 und 20): Sofern diese beiden Anschlüsse vorhanden sind, kann damit das Display beleuchtet werden. Dabei muss der Anschluss  $A$  (Anode) mit der Betriebsspannung (+5V) und der Anschluss  $K$  (Kathode) mit der Masse verbunden werden.

# Kapitel 3

## Ansteuerung des Grafik-LC-Displays

### 3.1 Befehle zur Ansteuerung des Grafik-LC-Displays

Tabelle 3.1 zeigt die Anweisungen zur Kommunikation mit dem Grafik-LC-Display.

	Command	Code												Function
		A0	RD	WR	D7	D6	D5	D4	D3	D2	D1	D0		
(1)	Display ON/OFF	0	1	0	1	0	1	0	1	1	1	0/1	Turns all display on or off, independently of display RAM data or internal status. 1: ON 0: OFF (Power-saving mode with static drive on)*	
(2)	Display start line	0	1	0	1	1	0	Display Start Address (0–31)					Specifies RAM line corresponding to uppermost line (COM0) of display.	
(3)	Set page address	0	1	0	1	0	1	1	1	0	Page (0–3)		Sets display RAM page in page address register.	
(4)	Set column (segment) address	0	1	0	0	Column Address (0–79)							Sets display RAM column address in column address register.	
(5)	Read status	0	0	1	Busy	ADC	ON/OFF	RESET	0	0	0	0	Reads the following status: BUSY 1: Internal operation, 0: Ready ADC 1: CW output (forward), 0: CCW output (reverse) ON/OFF 1: Display off, 0: Display on RESET 1: Being reset, 0: Normal	
(6)	Write display data	1	1	0	Write Data							Writes data from data bus into display RAM.	Display RAM location whose address has been preset is accessed. After access, the column address is incremented by 1.	
(7)	Read display data	1	0	1	Read Data							Reads data from display RAM onto data bus.		
(8)	Select ADC	0	1	0	1	0	1	0	0	0	0	0/1	Used to invert relationship of assignment between display RAM column addresses and segment driver outputs. 0: CW output (forward) 1: CCW output (reverse)	
(9)	Static drive ON/OFF	0	1	0	1	0	1	0	0	1	0	0/1	Selects normal display or static driving operation 1: Static drive (power-saving mode) 0: Normal driving	
(10)	Select duty	0	1	0	1	0	1	0	1	0	0	0/1	Selects LCD cell driving duty. 1: 1/32 0: 1/16	
(11)	Read modify write	0	1	0	1	1	1	0	0	0	0	0	Increments column address counter by 1 when display data is written. (This is not done when data is read.)	
(12)	End	0	1	0	1	1	1	0	1	1	1	0	Clears read modify write mode.	
(13)	Reset	0	1	0	1	1	1	0	0	0	1	0	Sets display start line register on the first line. Also sets column address counter and page address counter to 0.	

Tabelle 3.1: Display-Befehle

**Achtung:**

Beim Einschalten (oder beim Anlegen einer Betriebsspannung) befindet sich das Grafik-LC-Display im folgenden Zustand:

- Display off
- Display start line register: First line
- Static drive off
- Column address counter: Address 0
- Page address register: Page 0
- Select duty: 1/32
- Select ADC: Forward (ADC command D0="0", ADC status flag ="1")
- Read modify write off

Das Grafik-LC-Display befindet sich also schon in einem initialisierten Zustand. Trotzdem sollte das Grafik-LC-Display auch in der Software initialisiert werden, z.B. wenn bestimmte Parameter geändert werden sollen.

Mehr zur Initialisierung siehe Abschnitt 4.3.1 Unterprogramm GLCD\_INIT (für Assembler) bzw. Abschnitt 5.3.1 Unterprogramm GLCD\_INIT (für C).

## 3.2 Zeitdiagramm

Das Zeitdiagramm in Abbildung 3.1 bezieht sich auf die Verwendung der *80-Family-MPU*. Siehe auch Abschnitt 2 (Hardware).

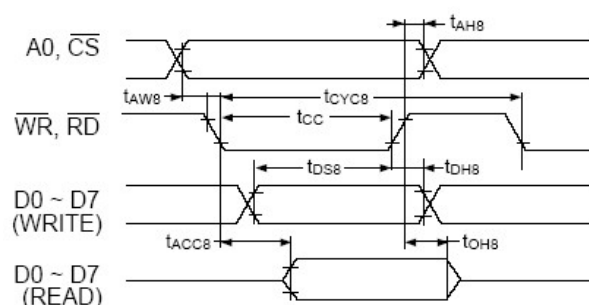


Abbildung 3.1: Zeitdiagramm



	Symbol	Min	Typ	Max	Einheit
Address setup time	$t_{AW8}$	20			ns
Address hold time	$t_{AH8}$	10			ns
System cycle time	$t_{CYC8}$	1000			ns
Control pulse width	$t_{CC}$	200			ns
Data setup time	$t_{DS8}$	80			ns
Data hold time	$t_{DH8}$	10			ns
Read access time	$t_{AW8}$			90	ns
Output disable time	$t_{AW8}$			90	ns

Tabelle 3.2: Zeiten

### 3.3 Zeichensatz

Das hier verwendete Grafik-LC-Display verfügt über **keinen** eigenen Zeichensatz. Alle benötigten Zeichen (Buchstaben, Ziffern, Sonderzeichen) müssen daher selbst erzeugt werden. Dieser Abschnitt zeigt wie man dazu prinzipiell vorgeht, während Demonstrationsbeispiel 1 (Abschnitt 6.3.) dies an einem praktischen Beispiel zeigt.

Beispiel 1: Buchstabe 'A'

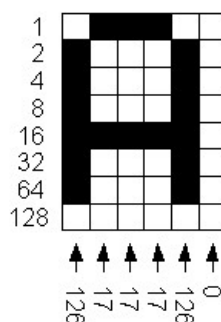


Abbildung 3.2: Zeichensatz (Beispiel 1)

Abbildung 3.2 zeigt wie der Buchstabe A aussehen soll. Nun müssen die 6 Spalten berechnet werden. Die Wertigkeit der Zeilen nimmt von oben nach unten zu. Zeile 1 besitzt die Wertigkeit 1 ( $=2^0$ ), Zeile 2 die Wertigkeit 2 ( $=2^1$ ), Zeile 3 die Wertigkeit 4 ( $=2^2$ ) usw. bis zur letzten ( $=8$ ten) Zeile mit der Wertigkeit 128 ( $=2^8$ ). Für jede Spalte müssen nun die Wertigkeiten der Zeilen welche aufleuchten sollen addiert werden. Für den Buchstaben A gemäß Abbildung 3.2 bedeutet dies:

1. Spalte:  $2 + 4 + 8 + 16 + 32 + 64 = 126$
2. Spalte:  $1 + 16 = 17$
3. Spalte:  $1 + 16 = 17$
4. Spalte:  $1 + 16 = 17$
5. Spalte:  $2 + 4 + 8 + 16 + 32 + 64 = 126$
6. Spalte: 0

Die Werte dieser sechs Spalten müssen nun nacheinander an das Grafik-LC-Display übertragen werden. Dazu ist die Anweisung *Write display data* gemäß Tabelle 3.1 zuständig.

### Beispiel 2: Sonderzeichen '%'

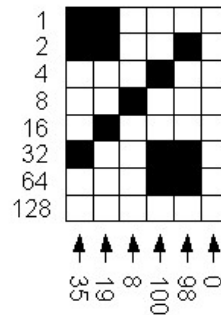


Abbildung 3.3: Zeichensatz (Beispiel 2)

Bei diesem Sonderzeichen ergeben sich für die Spalten folgende Werte (siehe auch Abbildung 3.3).

1. Spalte:  $1 + 2 + 32 = 35$
2. Spalte:  $1 + 2 + 16 = 19$
3. Spalte: 8
4. Spalte:  $4 + 32 + 64 = 100$
5. Spalte:  $2 + 32 + 64 = 98$
6. Spalte: 0

# Kapitel 4

## Software (in Assembler)

### 4.1 Benötigte Register und Portdefinitionen

#### Register:

Für die softwaremäßige Grafik-LC-Display-Ansteuerung werden neben einigen PIC internen Registern (SFR, **S**pezielle **F**unktions-**R**egister) noch folgende Register benötigt:

- **GLCD\_ZEILEN\_ZAEHLER**: Dieses Register beinhaltet ständig die aktuelle Zeile des “Cursors” (0 bis 3). **Achtung**: Mit Zeile ist hier das *Page address register* (gemäß Datenblatt) gemeint.
- **GLCD\_SPALTEN\_ZAEHLER**: Dieses Register beinhaltet ständig die aktuelle Spalte des “Cursors” (0 bis 121). **Achtung**: Mit Spalte ist hier **nicht** der *Column address counter* (gemäß Datenblatt) gemeint.
- Zwei Hilfsregister (**TEMP1** und **TEMP2**): Diese beiden Register sind so genannte temporäre Register. D.h. Sie werden nur kurzzeitig verwendet, entweder zum kurzzeitigen Zwischenspeichern von Werten, oder als Übergabeparameter beim Aufruf eines Unterprogramms. Sie können daher auch in beliebigen anderen Unterprogrammen verwendet werden.

#### Portdefinition:

Im Allgemeinen wird bei jeder Anwendung das Grafik-LC-Display an einem beliebigen Port angeschlossen. Damit dies in der Software nur an einer Stelle berücksichtigt werden muss befindet sich in der Software eine Portdefinition. Diese besteht aus den folgenden Parametern:

- **GLCD\_DATA**: Dieser Parameter gibt den Port für den 8-bit-Datenbus (DB0 bis DB7) an. (zum Beispiel Port A, Port B,...).
- **GLCD\_DATA\_TRIS**: Dieser Parameter ist für die Initialisierung des Ports für den Datenbus zuständig. Achtung: Wird für den Parameter **LCD\_DATA** der **PORTA** verwendet, so muss der Parameter **LCD\_DATA\_TRIS** den Parameter **TRISA** beinhalten!
- **GLCD\_CTRL**: Dieser Parameter gibt den Port für die Steuerleitungen an. (z.B. Port A, Port B,...).

- **GLCD\_CTRL\_TRIS**: Dieser Parameter ist für die Initialisierung des Ports für die Steuerleitungen zuständig. Achtung: Wird für den Parameter **LCD\_CTRL** der **PORTA** verwendet, so muss der Parameter **LCD\_CTRL\_TRIS** den Parameter **TRISA** beinhalten!
- **GLCD\_A0**: Dieser Parameter gibt den Portpin der Steuerleitung *A0* an.
- **GLCD\_CS1**: Dieser Parameter gibt den Portpin der Steuerleitung *Chipselect 1* ( $\overline{CS1}$ ) an.
- **GLCD\_CS2**: Dieser Parameter gibt den Portpin der Steuerleitung *Chipselect 2* ( $\overline{CS2}$ ) an.
- **GLCD\_RD**: Dieser Parameter gibt den Portpin der Steuerleitung *Read* ( $\overline{RD}$ ) an.
- **GLCD\_WR**: Dieser Parameter gibt den Portpin der Steuerleitung *Write* ( $\overline{WR}$ ) an.

Eine mögliche Portdefinition ist:

```
GLCD_DATA      equ    PORTD
GLCD_DATA_TRIS equ    TRISD
GLCD_CTRL      equ    PORTB
GLCD_CTRL_TRIS equ    TRISB

GLCD_A0        equ    5
GLCD_CS2       equ    4
GLCD_CS1       equ    3
GLCD_RD        equ    2
GLCD_WR        equ    1
```

Anmerkung: Für den Pin *RES* ist hier keine Portdefinition notwendig, da dieser Anschluss (Pin) direkt mit GND verbunden ist.

## 4.2 Initialisierung (Unterprogramm INIT)

Dieses Unterprogramm dient zur Initialisierung des Mikrocontrollers. Bei diesem Beispiel ist hier, für die Ansteuerung des Grafik-LC-Displays, nur die Definition der verwendeten Portpins als Ausgang notwendig.

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine für den PIC16F877. Das Grafik-LC-Display ist hier am Port B (Steuerleitungen) und Port D (Daten) angeschlossen.

```
INIT          bank1                ;Registerseite 1
              movlw    b'00000000' ;Port B und Port D als Ausgang
                                   ; definieren

              movwf    TRISB
              movwf    TRISD
              bank0                ;Registerseite 0

              return
```

Anmerkung: Bei den Anweisungen **bank1** bzw. **bank0** handelt es sich um so genannte Makros.

### 4.3 Unterprogramme für die Grafik-LCD-Ansteuerung

Zur Ansteuerung des Grafik-LC-Displays sind zumindest die folgenden Basis-Unterprogramme notwendig:

- Zuerst muss das Grafik-LC-Display initialisiert werden. Das Unterprogramm `GLCD_INIT` übernimmt diese Aufgabe.
- Befehle an das Grafik-LC-Display werden mit dem Unterprogramm `GLCD_COMMAND` erzeugt.
- Daten an das Grafik-LC-Display werden mit den Unterprogrammen `GLCD_DATA1` und `GLCD_DATA2` erzeugt. Da das Grafik-LC-Display in zwei Hälften unterteilt ist mit je einer eigenen Chip-Select-Steuerleitung, sind auch hier zwei Unterprogramme nötig. `GLCD_DATA1` schreibt Daten in die rechte Display-Hälfte, während `GLCD_DATA2` für die linke Displayhälfte zuständig ist.
- Ein weiteres Unterprogramm dient zum Löschen des Grafik-LC-Display (`GLCD_CLEAR`).
- Der Cursor kann mit den beiden Unterprogrammen `GLCD_SETCURSOR_ZEILE` und `GLCD_SETCURSOR_SPALTE` an eine beliebige Position gesetzt werden.

Nun aber zu den einzelnen Unterprogrammen im Detail:

#### 4.3.1 Unterprogramm `GLCD_INIT`

##### Aufgabe:

Dieses Unterprogramm initialisiert das Grafik-LC-Display (siehe auch Kapitel 3. Befehle zur Ansteuerung des Grafik-LC-Display).

##### Vorgehensweise:

Übergibt nacheinander die Befehle zur Initialisierung an das Grafik-LC-Display.

##### Hier das Unterprogramm:

```

GLCD_INIT      movlw    .175                ;Display on
                call     GLCD_COMMAND

                movlw    .164                ;static drive off
                call     GLCD_COMMAND

                movlw    .169                ;duty cyle: 1/32
                call     GLCD_COMMAND

                movlw    .160                ;ADC: CW output (forward)
                call     GLCD_COMMAND

                movlw    .238                ;read modify write off
                call     GLCD_COMMAND

                movlw    .192                ;line 0
                call     GLCD_COMMAND

                movlw    .184                ;1. Zeile
                call     GLCD_COMMAND

                movlw    .0                  ;1. Spalte
                call     GLCD_COMMAND

                return

```

### 4.3.2 Unterprogramm GLCD\_COMMAND

#### Aufgabe:

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden eines Befehls an das Grafik-LC-Display.

Anmerkung: A0 ist bei einem Befehl immer 0 (Low).

#### Übergabeparameter:

Das Arbeitsregister (w-Register) beinhaltet den Befehl.

#### Hier das Unterprogramm:

```
GLCD_COMMAND    bcf      GLCD_CTRL, GLCD_A0
                 bcf      GLCD_CTRL, GLCD_CS1
                 bcf      GLCD_CTRL, GLCD_CS2

                 bsf      GLCD_CTRL, GLCD_RD
                 bcf      GLCD_CTRL, GLCD_WR

                 movwf    GLCD_DATA          ;Befehl am Datenport ausgeben

                 bsf      GLCD_CTRL, GLCD_WR
                 bsf      GLCD_CTRL, GLCD_CS1
                 bsf      GLCD_CTRL, GLCD_CS2

                 return
```

### 4.3.3 Unterprogramm GLCD\_DATA1

#### Aufgabe:

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden von Daten an die erste Hälfte des Grafik-LC-Display.

Anmerkung: A0 ist bei den Daten immer 1 (High).

#### Übergabeparameter:

Das Arbeitsregister (w-Register) beinhaltet die Daten.

#### Hier das Unterprogramm:

```
GLCD_DATA1      bsf      GLCD_CTRL, GLCD_A0
                 bsf      GLCD_CTRL, GLCD_CS1
                 bcf      GLCD_CTRL, GLCD_CS2

                 bsf      GLCD_CTRL, GLCD_RD
                 bcf      GLCD_CTRL, GLCD_WR

                 movwf    GLCD_DATA          ;Daten am Datenport ausgeben

                 bsf      GLCD_CTRL, GLCD_WR
                 bsf      GLCD_CTRL, GLCD_CS1
                 bsf      GLCD_CTRL, GLCD_CS2

                 return
```

### 4.3.4 Unterprogramm GLCD\_DATA2

#### Aufgabe:

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden von Daten an die zweite Hälfte des Grafik-LC-Display.

Anmerkung: A0 ist bei den Daten immer 1 (High).

#### Übergabeparameter:

Das Arbeitsregister (w-Register) beinhaltet die Daten.

#### Hier das Unterprogramm:

```
GLCD_DATA2    bsf      GLCD_CTRL, GLCD_A0
              bcf      GLCD_CTRL, GLCD_CS1
              bsf      GLCD_CTRL, GLCD_CS2

              bsf      GLCD_CTRL, GLCD_RD
              bcf      GLCD_CTRL, GLCD_WR

              movwf    GLCD_DATA          ;Daten am Datenport ausgeben

              bsf      GLCD_CTRL, GLCD_WR
              bsf      GLCD_CTRL, GLCD_CS1
              bsf      GLCD_CTRL, GLCD_CS2

              return
```

### 4.3.5 Unterprogramm GLCD\_CLEAR

#### Aufgabe:

Löschen des Grafik-LC-Displays.

#### Vorgehensweise:

Beginnend bei der letzten Zeile und letzten Spalte mit Hilfe zweier geschachtelten Schleifen Nullen zum Grafik-LC-Display (in beide Display-Hälften) schreiben. Dies bewirkt, dass am Display nichts angezeigt wird bzw. das Display wird somit gelöscht.

#### Hier das Unterprogramm:

```
GLCD_CLEAR    movlw    .4
              movwf    TEMP1

GLCD_CLR_SCHL1:
              decf     TEMP1,w
              addlw    .184          ;Befehl (184 + TEMP1 = Zeile )
              call     GLCD_COMMAND ; an das Grafik-LC-Display

              movlw    .61
              movwf    TEMP2

              movlw    .0            ;Befehl: 1. Spalte an das Grafik-LC-
              ; Display
              call     GLCD_COMMAND

GLCD_CLR_SCHL2:
              movlw    .0
              call     GLCD_DATA1
              call     GLCD_DATA2

              decfsz   TEMP2,f
              goto     GLCD_CLR_SCHL2
```

```

    decfsz    TEMP1,f
    goto     GLCD_CLR_SCHL1

    return

```

**Anmerkung:**

Die beiden temporären Register (TEMP1 und TEMP2) dienen hier nur als Schleifenzähler. Sie können daher auch in anderen Unterprogrammen verwendet werden.

- TEMP1...Zeilenzähler (3 bis 0)
- TEMP2...Spaltenzähler (61 bis 0)

**4.3.6 Unterprogramm GLCD\_SETCURSOR\_ZEILE****Aufgabe:**

Den Cursor an die übergebene Zeile (0 bis 3) setzen.

**Vorgehensweise:**

Den Übergabeparameter für die Zeile (im Arbeitsregister) im globalen Register GLCD\_ZEILEN\_ZAEHLER sichern und an das Grafik-LC-Display übergeben.

**Übergabeparameter:**

Das Arbeitsregister (w-Register) beinhaltet die Zeile.

**Hier das Unterprogramm:**

```

GLCD_SETCURSOR_ZEILE
    movwf    GLCD_ZEILEN_ZAEHLER ;GLCD_ZEILEN_ZAEHLER = w-Register
    addlw    .184
    call     GLCD_COMMAND        ;GLCD_BEFEHL(184 + GLCD_ZEILEN_ZAEHLER)

    return

```

**4.3.7 Unterprogramm GLCD\_SETCURSOR\_SPALTE****Aufgabe:**

Den Cursor an die übergebene Spalte (0 bis 121) setzen.

**Vorgehensweise:**

Den Übergabeparameter für die Spalte (im Arbeitsregister) im globalen Register GLCD\_SPALTEN\_ZAEHLER sichern und an das Grafik-LC-Display übergeben. Dabei ist zu beachten, dass das Grafik-LC-Display in zwei Bereiche aufgeteilt ist, wobei beide Bereiche mit 0 beginnen. Spalten größer als 61 befinden sich im zweiten Bereich. Da dieser wieder ab 0 beginnt muss bei Spalten die größer als 61 sind, 61 abgezogen werden.

**Übergabeparameter:**

Das Arbeitsregister (w-Register) beinhaltet die Spalte.



**Hier das Unterprogramm:**

```
GLCD_SETCURSOR_SPALTE
    movwf    GLCD_SPALTEN_ZAEHLER ;GLCD_SPALTEN_ZAEHLER = w-Register

    movlw    .61                    ;GLCD_SPALTEN_ZAEHLER < 61
    subwf    GLCD_SPALTEN_ZAEHLER,w
    btfsc    STAT,C
    goto     GLCD_SETCUR_W1         ;ja: GLCD_BEFEHL(GLCD_SPALTEN_ZAEHLER)
    movf     GLCD_SPALTEN_ZAEHLER,w
    call     GLCD_COMMAND
    goto     GLCD_SETCUR_W2

GLCD_SETCUR_W1
    movlw    .61                    ;nein: GLCD_BEFEHL(GLCD_SPALTEN_ZAEHLER
    ; -61)
    subwf    GLCD_SPALTEN_ZAEHLER,w
    call     GLCD_COMMAND

GLCD_SETCUR_W2
    return
```

# Kapitel 5

## Software (in C mit CC5X)

### 5.1 Portdefinitionen und externe Register

#### Portdefinition:

Im Allgemeinen wird bei jeder Anwendung das Grafik-LC-Display an einem beliebigen Port angeschlossen. Damit dies in der Software nur an einer Stelle berücksichtigt werden muss befindet sich in der Software eine Portdefinition. Diese besteht aus den folgenden Parametern:

- **GLCD\_DATA**: Dieser Parameter gibt den Port für den Datenbus an. (zum Beispiel Port A, Port B,...).
- **GLCD\_DATA\_TRIS**: Dieser Parameter ist für die Initialisierung des Ports für den Datenbus zuständig. Achtung: Wird für den Parameter **LCD\_DATA** der **PORTA** verwendet, so muss der Parameter **LCD\_DATA\_TRIS** den Parameter **TRISA** beinhalten!
- **GLCD\_CTRL**: Dieser Parameter gibt den Port für die Steuerleitungen an. (z.B. Port A, Port B,...).
- **GLCD\_CTRL\_TRIS**: Dieser Parameter ist für die Initialisierung des Ports für die Steuerleitungen zuständig. Achtung: Wird für den Parameter **LCD\_CTRL** der **PORTA** verwendet, so muss der Parameter **LCD\_CTRL\_TRIS** den Parameter **TRISA** beinhalten!
- **GLCD\_A0**: Dieser Parameter gibt den Portpin der Steuerleitung *A0* an.
- **GLCD\_CS1**: Dieser Parameter gibt den Portpin der Steuerleitung *Chipselect 1* ( $\overline{CS1}$ ) an.
- **GLCD\_CS2**: Dieser Parameter gibt den Portpin der Steuerleitung *Chipselect 2* ( $\overline{CS2}$ ) an.
- **GLCD\_RD**: Dieser Parameter gibt den Portpin der Steuerleitung *Read* ( $\overline{RD}$ ) an.
- **GLCD\_WR**: Dieser Parameter gibt den Portpin der Steuerleitung *Write* ( $\overline{WR}$ ) an.

Eine mögliche Portdefinition ist:

```
/* Port B */
#pragma char GLCD_CTRL      @ PORTB
#pragma char GLCD_CTRL_TRIS @ TRISB

#pragma bit GLCD_WR      @ PORTB.1
#pragma bit GLCD_RD      @ PORTB.2
#pragma bit GLCD_CS1     @ PORTB.3
#pragma bit GLCD_CS2     @ PORTB.4
#pragma bit GLCD_A0      @ PORTB.5

/* Port D */
#pragma char GLCD_DATA      @ PORTD
#pragma char GLCD_DATA_TRIS @ TRISD
```

Anmerkung: Für den Pin *RES* ist hier keine Portdefinition notwendig, da dieser Anschluss (Pin) direkt mit GND verbunden ist.

### Externe Register:

Die Cursorposition wird mit den beiden externen Registern `hlp_zeilen_zaeher` und `hlp_spalten_zaeher` gesichert und ist somit immer für jedes Unterprogramm verfügbar.

## 5.2 Initialisierung (Unterprogramm INIT)

Dieses Unterprogramm dient zur Initialisierung des Mikrocontrollers. Bei diesem Beispiel ist hier, für die Ansteuerung des Grafik-LC-Displays, nur die Definition der verwendeten Portpins als Ausgang notwendig.

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine für den PIC16F877. Das Grafik-LC-Display ist hier am Port B (Steuerleitungen) und Port D (Daten) angeschlossen.

```
void INIT(void)
{
    TRISB = 0;           // Port B als Ausgang definieren
    TRISD = 0;           // Port D als Ausgang definieren
}
```

## 5.3 Unterprogramme für die Grafik-LCD-Ansteuerung

Abbildung 5.1 zeigt wie die einzelnen Unterprogramme zur Ansteuerung des Grafik-LC-Displays miteinander verschachtelt sind.

Zur Ansteuerung des Grafik-LC-Displays sind zumindest die folgenden Basis-Unterprogramme notwendig:

- Zuerst muss das Grafik-LC-Display initialisiert werden. Das Unterprogramm `GLCD_INIT` übernimmt diese Aufgabe.
- Befehle an das Grafik-LC-Display werden mit dem Unterprogramm `GLCD_COMMAND` erzeugt.

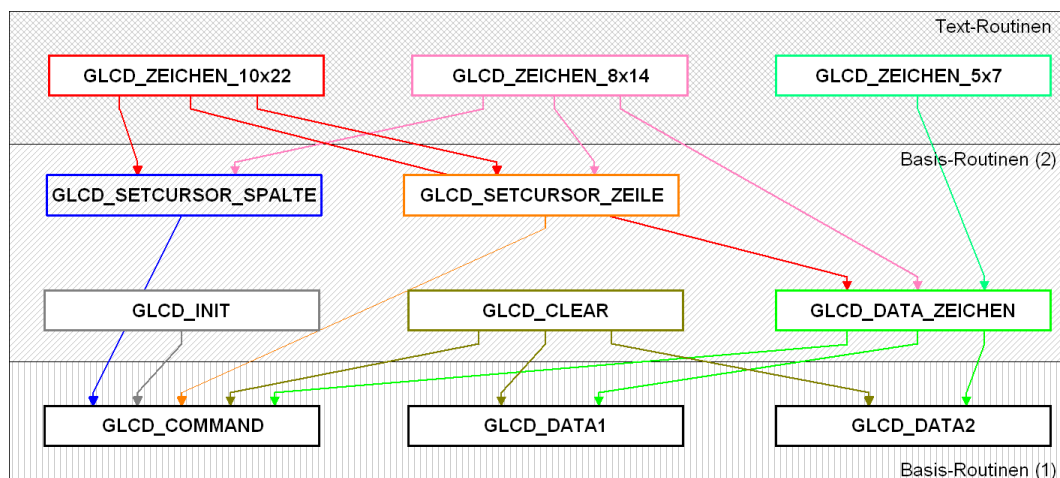


Abbildung 5.1: Softwarestruktur

- Daten an das Grafik-LC-Display werden mit den Unterprogrammen `GLCD_DATA1` und `GLCD_DATA2` erzeugt. Da das Grafik-LC-Display in zwei Hälften unterteilt ist, mit je einer eigenen Chip-Select-Steuerleitung, sind auch hier zwei Unterprogramme nötig. `GLCD_DATA1` schreibt Daten in die rechte Display-Hälfte, während `GLCD_DATA2` für die linke Displayhälfte zuständig ist. Das Unterprogramm `GLCD_DATA_ZEICHEN` bestimmt dabei, welches dieser beiden Unterprogramme verwendet werden soll.
- Ein weiteres Unterprogramm dient zum Löschen des Display (`GLCD_CLEAR`).
- Der Cursor kann mit den beiden Unterprogrammen `GLCD_SETCURSOR_ZEILE` und `GLCD_SETCURSOR_SPALTE` an eine beliebige Position gesetzt werden.

Für die Text- und Werteausgabe sind weitere Unterprogramme abhängig von der Schriftgröße notwendig:

- Das Unterprogramm `GLCD_ZEICHEN_5x7` für die Standardgröße
- Das Unterprogramm `GLCD_ZEICHEN_8x14` für die mittlere Größe
- Das Unterprogramm `GLCD_ZEICHEN_10x22` für die großen Zeichen

Diese Unterprogramme geben nur ein Zeichen (Charakter) aus.

Die Unterprogramme zur Grafikausgabe sind noch nicht ausgeführt!

Nun aber zu den einzelnen Unterprogrammen im Detail:

### 5.3.1 Unterprogramm GLCD\_INIT

#### Aufgabe:

Dieses Unterprogramm initialisiert das Grafik-LC-Display (siehe auch Kapitel 3. Befehle zur Ansteuerung des Grafik-LC-Display).

#### Vorgehensweise:

Übergibt nacheinander die Befehle zur Initialisierung an das Grafik-LC-Display.

#### Hier das Unterprogramm:

```
void GLCD_INIT(void)
{
    GLCD_COMMAND(175);           // Display on
    GLCD_COMMAND(164);           // static drive off
    GLCD_COMMAND(169);           // duty cyle: 1/32
    GLCD_COMMAND(160);           // ADC: CW output (forward)
    GLCD_COMMAND(238);           // read modify write off
    GLCD_COMMAND(192);           // line 0
    GLCD_COMMAND(184);           // 1. Zeile
    GLCD_COMMAND(0);             // 1. Spalte
}
```

### 5.3.2 Unterprogramm GLCD\_COMMAND

#### Aufgabe:

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden eines Befehls an das Grafik-LC-Display.

Anmerkung: A0 ist bei einem Befehl immer 0 (Low).

#### Hier das Unterprogramm:

```
void GLCD_COMMAND(unsigned char command)
{
    GLCD_A0 = 0;
    GLCD_CS1 = 0;
    GLCD_CS2 = 0;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = command;

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}
```

### 5.3.3 Unterprogramm GLCD\_DATA1

#### Aufgabe:

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden von Daten an die erste Hälfte des Grafik-LC-Display.

Anmerkung: A0 ist bei den Daten immer 1 (High).

**Hier das Unterprogramm:**

```
void GLCD_DATA1(unsigned char data)
{
    GLCD_A0 = 1;
    GLCD_CS1 = 1;
    GLCD_CS2 = 0;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = data;

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}
```

**5.3.4 Unterprogramm GLCD\_DATA2****Aufgabe:**

Umsetzung des Zeitdiagramms (gemäß Abschnitt 3.2) zum Senden von Daten an die zweite Hälfte des Grafik-LC-Display.

Anmerkung: A0 ist bei den Daten immer 1 (High).

**Hier das Unterprogramm:**

```
void GLCD_DATA2(unsigned char data)
{
    GLCD_A0 = 1;
    GLCD_CS1 = 0;
    GLCD_CS2 = 1;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = data;

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}
```

**5.3.5 Unterprogramm GLCD\_DATA\_ZEICHEN****Aufgabe:**

Je nach aktueller Position des Cursors den übergebenen Wert (*data*) an die erste oder zweite Displayhälfte übergeben.

**Vorgehensweise:**

- Befindet sich der Cursor an der 62. Spalte (`hlp_spalten.zaehler = 61`), die so genannte *Column Address* auf 0 setzen (gemäß Datenblatt). Dies bewirkt, dass der displayinterne Speicherzähler wieder bei 0 beginnt. (siehe Datenblatt)
- Befindet sich der Cursor an eine Spalte größer als 61 (also zweite Displayhälfte) den Übergabewert (*data*) mit dem Unterprogramm `GLCD_DATA2` an die zweite

Displayhälfte übergeben. Andernfalls den Übergabewert (data) mit dem Unterprogramm GLCD\_DATA1 an die erste Displayhälfte übergeben.

### Hier das Unterprogramm:

```
void GLCD_DATA_ZEICHEN(unsigned char data)
{
    if (hlp_spalten_zaehler == 61)      // Wenn Cursorspalte = 61
    {
        GLCD_COMMAND(0);                // ... "Column Address" auf 0 setzen
    }
    if (hlp_spalten_zaehler > 60)        // Cursorspalte > 60
    {
        GLCD_DATA2(data);               // ja: Uebergabewert (data) mit dem an
                                         // die zweite Displayhaelfte uebergeben
    }
    else
    {
        GLCD_DATA1(data);               // nein: Uebergabewert (data) mit dem an
                                         // die erste Displayhaelfte uebergeben
    }

    hlp_spalten_zaehler++;
}
```

### Anmerkung:

Das (externe) Hilfsregister `hlp_spalten_zaehler` gibt an, an welcher Spalte sich der Cursor befindet.

## 5.3.6 Unterprogramm GLCD\_CLEAR

### Aufgabe:

Löschen des Grafik-LC-Displays.

### Vorgehensweise:

Beginnend bei der ersten Zeile und ersten Spalte mit Hilfe zweier geschachtelten Schleifen Nullen zum Grafik-LC-Display (in beide Display-Hälften) schreiben. Dies bewirkt, dass am Display nichts angezeigt wird bzw. das Display wird somit gelöscht.

### Hier das Unterprogramm:

```
void GLCD_CLEAR(void)
{
    unsigned char Zeile, Spalte;

    for(Zeile = 0; Zeile < 4; Zeile++)
    {
        GLCD_COMMAND(184 + Zeile);      // Befehl: naechste Zeile

        GLCD_COMMAND(0);                // Befehl: 1.Spalte der Zeile

        for(Spalte = 0; Spalte < 61; Spalte++)
        {
            GLCD_DATA1(0);
            GLCD_DATA2(0);
        }
    }
}
```

### 5.3.7 Unterprogramm GLCD\_SETCURSORS\_ZEILE

**Aufgabe:**

Den Cursor an die übergebene Zeile (0 bis 3) setzen.

**Vorgehensweise:**

Übergabeparameter für die Zeile im globalen Register `hlp_zeilen_zaeher` sichern und an das Grafik-LC-Display übergeben.

**Übergabeparameter:**

`zeile`: 0 - 3

**Hier das Unterprogramm:**

```
void GLCD_SETCURSORS_ZEILE(unsigned char zeile)
{
    hlp_zeilen_zaeher = zeile;
    GLCD_COMMAND(184 + zeile);
}
```

### 5.3.8 Unterprogramm GLCD\_SETCURSORS\_SPALTE

**Aufgabe:**

Den Cursor an die übergebene Spalte (0 bis 121) setzen.

**Vorgehensweise:**

Übergabeparameter für die Spalte im globalen Register `hlp_spalten_zaeher` sichern und an das Grafik-LC-Display übergeben. Dabei ist zu beachten, dass das Grafik-LC-Display in zwei Bereiche aufgeteilt ist, wobei beide Bereiche mit 0 beginnen. Spalten größer als 61 befinden sich im zweiten Bereich. Da dieser wieder ab 0 beginnt, muss bei Spalten die größer als 61 sind, 61 abgezogen werden.

**Übergabeparameter:**

`spalte`: 0 - 121

**Hier das Unterprogramm:**

```
void GLCD_SETCURSORS_SPALTE(unsigned char spalte)
{
    hlp_spalten_zaeher = spalte;
    if (spalte < 61)
    {
        GLCD_COMMAND(spalte);
    }
    else
    {
        GLCD_COMMAND(spalte - 61);
    }
}
```



### 5.3.9 Unterprogramm GLCD\_ZEICHEN\_5x7

#### Aufgabe:

Ein Zeichen (Charakter im Format 5x7) am Grafik-LC-Display ausgeben.

#### Vorgehensweise:

Nacheinander die Spalten eines Zeichens an das Grafik-LC-Display mit dem Unterprogramm `GLCD_DATA_ZEICHEN` übergeben. Die Daten für die Spalten sind in Tabellen abgelegt, wobei für jede Spalte des Zeichens eine eigene Tabelle vorhanden ist. Diese Tabellen müssen an anderer Stelle hinzugefügt werden! Die letzte Spalte muss eine 0 enthalten, also eine "Leerspalte".

#### Übergabeparameter:

`zeichen` (im ASCII-Code)

#### Hier das Unterprogramm:

```
void GLCD_ZEICHEN_5x7(unsigned char zeichen)
{
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp1[zeichen]);

    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp2[zeichen]);

    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp3[zeichen]);

    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp4[zeichen]);

    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp5[zeichen]);

    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);
}
```

### 5.3.10 Unterprogramm GLCD\_ZEICHEN\_8x14

#### Aufgabe:

Ein Zeichen (Charakter im Format 8x14) am Grafik-LC-Display ausgeben.

#### Vorgehensweise:

- Die externen Hilfsregister `hlp_zeilen_zaeher` und `hlp_spalten_zaeher` sichern, da sie hier benötigt, aber gleichzeitig hier auch verändert werden.
- Nacheinander die Spalten für die erste Zeile des auszugebenden Zeichens an das Grafik-LC-Display mit dem Unterprogramm `GLCD_DATA_ZEICHEN` übergeben. Die Daten für die Spalten sind in Tabellen abgelegt, wobei für jede Spalte des Zeichens eine eigene Tabelle vorhanden ist. Diese Tabellen müssen an anderer Stelle hinzugefügt werden! Die letzte Spalte muss eine 0 enthalten, also eine "Leerspalte".
- Den Cursor an die nächste Zeile (`tmp_zeile + 1`) und an die erste Spalte des Zeichens (`tmp_spalte`) setzen.

- In gleicher Weise wie vorher die Spalten für die zweite Zeile des auszugebenden Zeichens an das Grafik-LC-Display mit dem Unterprogramm GLCD\_DATA\_ZEICHEN übergeben.
- Den Cursor an die erste Zeile des Zeichens (`tmp_zeile`) setzen.

### Übergabeparameter:

`zeichen`

### Hier das Unterprogramm:

```
void GLCD_ZEICHEN_8x14(unsigned char zeichen)
{
    char tmp_zeile = hlp_zeilen_zaeher;
    char tmp_spalte = hlp_spalten_zaeher;

    // 1. Zeile
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp6[zeichen]);
    // 7. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp7[zeichen]);
    // 8. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp8[zeichen]);
    // 9. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);

    // 2. Zeile und 1. Spalte des Zeichen
    GLCD_SETCURSOR_ZEILE(tmp_zeile + 1);
    GLCD_SETCURSOR_SPALTE(tmp_spalte);

    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp6[zeichen]);
    // 7. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp7[zeichen]);
    // 8. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp8[zeichen]);
    // 9. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);

    // 1. Zeile und Spalte vom Ende des Zeichen
    GLCD_SETCURSOR_ZEILE(tmp_zeile);
}
```

### 5.3.11 Unterprogramm GLCD\_ZEICHEN\_10x22

#### Aufgabe:

Ein Zeichen (Charakter im Format 10x22) am Grafik-LC-Display ausgeben.

#### Vorgehensweise:

- Die externen Hilfsregister `hlp_zeilen_zaeher` und `hlp_spalten_zaeher` sichern, da sie hier benötigt, aber gleichzeitig hier auch verändert werden.
- Nacheinander die Spalten für die erste Zeile des auszugebenden Zeichens an das Grafik-LC-Display mit dem Unterprogramm `GLCD_DATA_ZEICHEN` übergeben. Die Daten für die Spalten sind in Tabellen abgelegt, wobei für jede Spalte des Zeichens eine eigene Tabelle vorhanden ist. Diese Tabellen müssen an anderer Stelle hinzugefügt werden! Die letzte Spalte muss eine 0 enthalten, also eine "Leerspalte".
- Den Cursor an die nächste Zeile (`tmp_zeile + 1`) und an die erste Spalte des Zeichens (`tmp_spalte`) setzen.
- In gleicher Weise wie vorher die Spalten für die zweite Zeile des auszugebenden Zeichens an das Grafik-LC-Display mit dem Unterprogramm `GLCD_DATA_ZEICHEN` übergeben.
- Den Cursor an die nächste Zeile (`tmp_zeile + 2`) und an die erste Spalte des Zeichens (`tmp_spalte`) setzen.
- In gleicher Weise wie vorher die Spalten für die dritte Zeile des auszugebenden Zeichens an das Grafik-LC-Display mit dem Unterprogramm `GLCD_DATA_ZEICHEN` übergeben.
- Den Cursor an die erste Zeile des Zeichens (`tmp_zeile`) setzen.

#### Übergabeparameter:

`zeichen`

#### Hier das Unterprogramm:

```
void GLCD_ZEICHEN_10x22(unsigned char zeichen)
{
    char tmp_zeile = hlp_zeilen_zaeher;
    char tmp_spalte = hlp_spalten_zaeher;

    // 1. Zeile
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp6[zeichen]);
}
```

```

// 7. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp9[zeichen]);
// 10. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp10[zeichen]);
// 11. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);
// 12. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 2. Zeile und 1. Spalte des Zeichens
GLCD_SETCURSOR_ZEILE(tmp_zeile + 1);
GLCD_SETCURSOR_SPALTE(tmp_spalte);

// 1. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp1[zeichen]);
// 2. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp2[zeichen]);
// 3. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp3[zeichen]);
// 4. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp4[zeichen]);
// 5. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp5[zeichen]);
// 6. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp6[zeichen]);
// 7. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp9[zeichen]);
// 10. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp10[zeichen]);
// 11. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);
// 12. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 3. Zeile und 1. Spalte des Zeichens
GLCD_SETCURSOR_ZEILE(tmp_zeile + 2);
GLCD_SETCURSOR_SPALTE(tmp_spalte);

// 1. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp1[zeichen]);
// 2. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp2[zeichen]);
// 3. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp3[zeichen]);
// 4. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp4[zeichen]);
// 5. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp5[zeichen]);
// 6. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp6[zeichen]);
// 7. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp9[zeichen]);
// 10. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp10[zeichen]);
// 11. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);
// 12. Spalte des Zeichens

```

```
GLCD_DATA_ZEICHEN(0);

// 1. Zeile und Spalte vom Ende des Zeichen
GLCD_SETCURSOR_ZEILE(tmp_zeile);
}
```

# Kapitel 6

## Demonstrationsbeispiele

Die folgenden Beispiele dienen nur zur Demonstration. Sie zeigen eine mögliche Einbindung der zuvor beschriebenen Unterprogramme.

### 6.1 Hardware

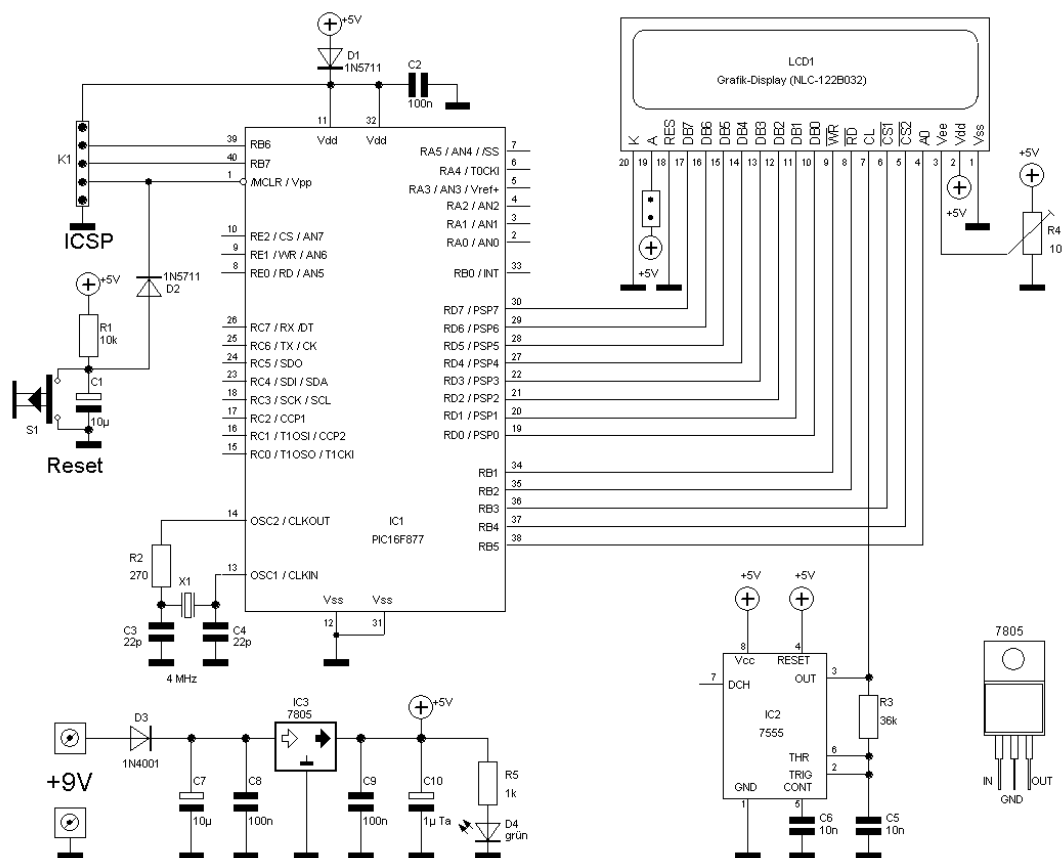


Abbildung 6.1: Schaltung zur Demonstration der Grafik-LCD-Routinen

Bei diesen Demonstrationsbeispielen soll ein Mikrocontroller (PIC16F877) beliebige Texte (z.B. Hello, World) und Daten (z.B. Uhrzeit und Datum) ausgeben. Später auch Grafi-

ken (Linien, Rechtecke, usw.). Die Beschaltung des Grafik-LC-Displays erfolgt gemäß Abschnitt 2.

Für die Takterzeugung dient eine Standardbeschaltung bestehend aus einem 4-MHz-Quarz (X1), zwei Kondensatoren (C3, C4) und einem Widerstand (R2).

Zur Erzeugung des Reset (für den Mikrocontroller) wurde ebenfalls eine einfache Standardlösung bestehend aus einem Widerstand (R1) und einem Elektrolyt-Kondensator (C1) gewählt. Da diese RC-Kombination nur beim Anlegen der Betriebsspannung (also beim Einschalten) einen Reset erzeugt, wurde zusätzlich der Taster S1 hinzugefügt. Mit diesem Taster kann nun jeder Zeit ein Reset ausgelöst werden. Da der Reseteingang des Mikrocontrollers (*MCLR*, Pin 1) auch gleichzeitig die Programmierspannung (ca. 13V) bei der ICSP-Programmierung ist, darf während einer Programmierung kein Reset ausgelöst werden. Durch die Diode D2 ist ein Reset durch das RC-Glied (R1 und C1) während einer Programmierung via ICSP nicht möglich.

Die Diode D1 ist notwendig, damit beim Programmieren des Mikrocontroller (IC1) das Programmiergerät nicht die gesamte Hardware versorgt, sondern nur den Mikrocontroller.

Der Kondensator C2 dient zur Entkoppelung der Betriebsspannung für den Mikrocontroller. Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser muss möglichst nahe an diesen IC angebracht werden.

Die ICSP-Schnittstelle (K1) dient zur Programmierung des Mikrocontrollers (IC1), wobei bei dieser Methode der Mikrocontroller nicht aus der Schaltung entfernt werden muss. (siehe auch Anhang D ab Seite 81)

Diese ICSP-Schnittstelle beinhaltet folgende Leitungen:

- Betriebsspannung (*V<sub>dd</sub>*, Pins 11 und 32): Damit beim Programmieren das Programmiergerät nicht die gesamte Hardware versorgen muss (könnte das Programmiergerät überlasten) wird mit Hilfe der Diode D1 nur der Mikrocontroller IC1 mit dem Programmiergerät versorgt.
- Taktleitung (*RB6*, Pin 39): Hier wird dieser Pin nur für die ICSP-Schnittstelle benötigt.
- Datenleitung (*RB7*, Pin 40): Hier wird dieser Pin nur für die ICSP-Schnittstelle benötigt.
- Programmierspannung (*MCLR/V<sub>pp</sub>*, Pin 1): Dieser Pin hat eine Doppelfunktion, er ist auch der Reset-Eingang, und muss deshalb mit einer Diode (D2) vor einem Reset durch das RC-Glied, bestehend aus R1 und C1, geschützt werden.
- Masse (*V<sub>ss</sub>*, Pins 12 und 31)

Die Stromversorgung besteht ebenfalls aus einer einfachen Standardlösung. Ein Festspannungsregler (IC3) vom Typ 7805 übernimmt mit den Kondensatoren C7 bis C10

die Spannungsregelung. Als Spannungsquelle dient beispielsweise ein unstabiliertes 9-V-Steckernetzteil.

Die Diode D3 dient hier als Verpolungsschutz, und die Leuchtdiode D4 dient zusammen mit dem Vorwiderstand R5 als Spannungskontrolle.

## 6.2 Nachbauanleitung (der Hardware)

Dieser Abschnitt beschreibt den Nachbau dieses Demonstrationsbeispiels. Das Herstellen der Platine wird hier nicht beschrieben, da hier jeder seine eigene Methode besitzt, außerdem werden industriell gefertigte Platinen (auch in der Einzelfertigung und auch für eine Privatperson) immer günstiger, so dass die Eigenfertigung immer mehr abnimmt.

### Schritt 1: Platine herstellen

Zu diesem Zweck ist im Anhang A (Seite 61) das Layout zu diesem Demonstrationsbeispiel abgedruckt, welches üblicherweise seitenverkehrt ist.

Selbstverständlich kann eine eigene Platine entworfen werden, die den eigenen Anforderungen entspricht.

### Schritt 2: Platine bestücken

Das Bestücken der Platine ist erst dann sinnvoll, wenn alle für diese Platine benötigten Bauteile vorhanden sind. Es sollten generell nur erstklassige und neuwertige Bauteile verwendet werden. Auf Bauteile aus ausgeschlachteten Geräten sollte grundsätzlich verzichtet werden, da ihre Funktionalität nicht gewährleistet ist, und eine unnötige Fehlersuche dadurch vermieden werden kann.

Weiters sollte ausreichend Platz und vor allem ausreichend Zeit für die Bestückung der Platine vorhanden sein.

Die Bauelemente entsprechend Abbildung 6.2 (Bestückungsplan), der folgenden Reihenfolge, der Stückliste (Anhang B) und dem Schaltplan (Abschnitt 6.1) bestücken. Die nach dem anlöten überstehenden Anschlüsse mit einem kleinen Seitenschneider entfernen.

Reihenfolge zur Bestückung der Platine:

- 1 Drahtbrücke (unter IC1)
- Dioden D1 und D2 (je 1N5711): **Achtung:** Polarität beachten!
- Widerstände R1 (10k), R2 (270 Ohm), R3 (36k), und R5 (1k): **Tipp:** Vor dem Einlöten des Widerstandes diesen überprüfen, auch wenn die Bauteile in einem Regal sortiert sind. (z.B. mit einem Multimeter). Die Praxis hat gezeigt, dass sich hin und wieder doch falsche Bauteilwerte in das Regal eingeschlichen haben. Dies gilt nicht nur für Widerstände, sondern auch für Dioden, Kondensatoren, Transistoren usw.
- Diode D3 (1N4001): **Achtung:** Polarität beachten!



- IC-Fassungen für IC1 (40polig) und IC2 (8polig): **Tipp 1:** Obwohl es bei den Fassungen elektrisch gesehen egal ist wie die Fassungen eingelötet sind, sollte man doch die Fassung so einlöten, dass die Kerbe in die richtige Richtung zeigt. Dies erleichtert das spätere Einsetzen der ICs bzw. erleichtert die Arbeit bei einem IC-Tausch. **Tipp 2:** Beim Einlöten der Fassungen sollte man wie folgt vorgehen: Fassung an der einzusetzenden Stelle lagerichtig einsetzen und zunächst nur einen beliebigen Eckpin anlöten. Fassung kontrollieren und eventuell Nachlöten. Sitzt die Fassung ordentlich, den gegenüberliegenden Pin anlöten. Fassung wieder kontrollieren und eventuell nachlöten. Erst wenn Sie mit der Lage der Fassung zufrieden sind, die restlichen Pins anlöten.
- Buchsenleiste für LCD1 (20polig, Grafik-LC-Display), ICSP-Schnittstelle (5polig) und optional für die hier nicht verwendeten Anschlüsse (=Ports) des Mikrocontrollers IC1: **Tipp:** Bei mehrpoligen Buchsenleisten zuerst den ersten Pin anlöten, Buchsenleiste anschließend ausrichten, eventuell nachlöten, dann den letzten Pin anlöten, Buchsenleiste eventuell ausrichten und/oder nachlöten. Erst danach die restlichen, dazwischen liegenden Pins anlöten.
- Keramikkondensatoren C3 und C4 (je 22pF)
- Tantal C10 ( $1\mu\text{F}/35\text{V}$ ): **Achtung:** Polarität beachten! (der längere Anschluss ist der +-Pol)
- Taster S1
- Keramikkondensatoren C2, C8, C9 (je 100nF) und C5, C6 (je 10nF)
- Leuchtdiode D4 (low current, 3mm, grün): **Achtung:** Der längere Anschluss ist die Anode (plus), der kürzere demnach die Kathode (minus).
- Trimmer R4 (10k)
- Stiftleiste für Jumper
- Elkos C1, C7 (je  $10\mu\text{F}/35\text{V}$ ): **Achtung:** Polarität beachten
- Anschlussklemme (2polig)
- Quarz X1 (4 MHz)
- Spannungsregler IC3 (7805): **Achtung:** Polarität beachten!

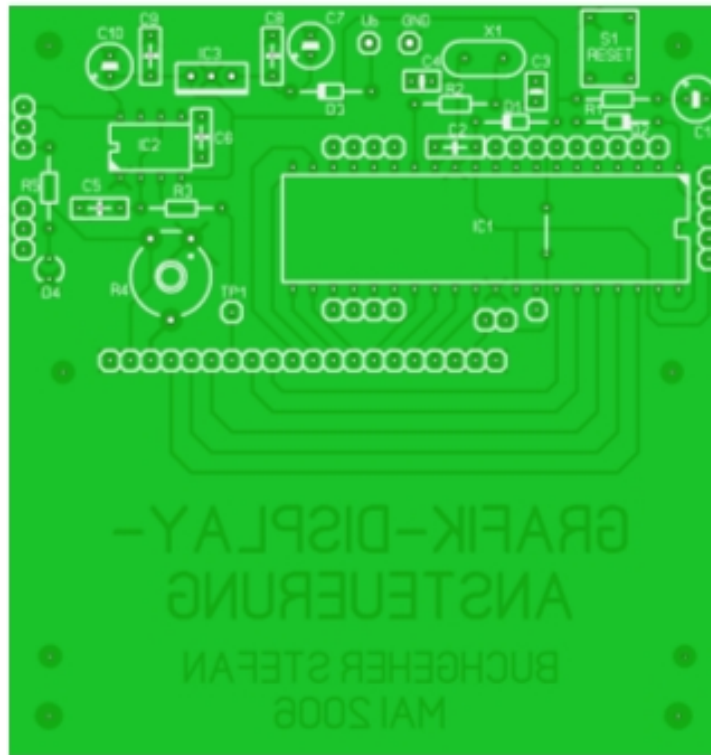


Abbildung 6.2: Bestückungsplan (Demonstrationsbeispiele)

Zum Schluss alle Lötstellen noch einmal sorgfältig auf Kurzschlüsse oder kalte Lötstellen überprüfen. Hier sollte man sich ausreichend Zeit nehmen!

### Schritt 3: Grafik-LC-Display vorbereiten

Auf der “Unterseite“ des Grafik-LC-Displays eine 20polige Kontaktleiste anlöten. Beim Löten ist dabei äußerste Vorsicht geboten. Kurzschlüsse zwischen Grafik-LC-Display und Kontaktleiste können nur sehr schwer gefunden und entfernt werden.

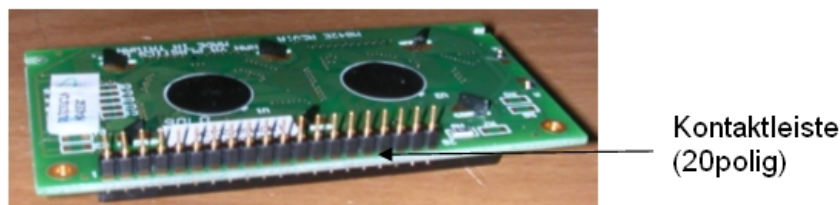


Abbildung 6.3: Grafik-LC-Display (Ansicht von unten)

### Schritt 4: Test

- IC1 (PIC16F877), IC2 (7555) und LCD1 (Grafik-LC-Display) noch **nicht** in die Fassungen einsetzen, und auch noch nicht die Betriebsspannung anschließen.
- Zuerst mit einem Multimeter prüfen, ob zwischen Betriebsspannung und Masse kein Kurzschluss herrscht. (Multimeter im Mode “Durchgangstester“ an der Anschlussklemme). Ist kein Kurzschluss feststellbar, als nächstes am z.B. IC-Sockel für IC1 an

den Pins 12 oder 31 ( $V_{ss}$  bzw.  $GND$ ) und 11 oder 32 ( $V_{cc}$  bzw.  $+5V$ ) messen. Diese Messung kann auch am IC-Sockel für IC2 an den Pins 1 ( $GND$ ) und 8 ( $V_{cc}$  bzw.  $+5V$ ) durchgeführt werden. Multimeter nach wie vor im Mode “Durchgangstester“. Eventuell festgestellte Kurzschlüsse müssen natürlich aufgespürt und entfernt werden!

- Eine Spannungsquelle mit einer Spannung zwischen 7V und 9V anschließen.
- Spannung an IC1 zwischen den Pins 12 oder 31 ( $V_{ss}$  bzw.  $GND$ ) und 11 oder 32 ( $V_{dd}$  bzw.  $+5V$ ) messen. Wird hier (bei angestecktem Netzteil) keine oder eine grob abweichende Spannung als 5V gemessen, so liegt ein Bestückungsfehler vor, welcher unbedingt aufgespürt und beseitigt werden muss.
- Ist die Spannung okay, diesen Vorgang bei IC2 zwischen den Pins 1 ( $GND$ ) und 8 ( $V_{cc}$  bzw.  $+5V$ ) wiederholen. Auch hier gilt: Wird hier (bei angestecktem Netzteil) keine oder eine grob abweichende Spannung als 5V gemessen, so liegt ein Bestückungs- und/oder Verdrahtungsfehler vor, welcher unbedingt aufgespürt und beseitigt werden muss.
- Ist die Spannung okay, diesen Vorgang bei LCD1 zwischen den Pins 1 ( $V_{ss}$  bzw.  $GND$ ) und 2 ( $V_{dd}$  bzw.  $+5V$ ) wiederholen. Auch hier gilt: Wird hier (bei angestecktem Netzteil) keine oder eine grob abweichende Spannung als 5V gemessen, so liegt ein Bestückungs- und/oder Verdrahtungsfehler vor, welcher unbedingt aufgespürt und beseitigt werden muss.
- Ist die Spannung okay, entweder die Stromversorgung abschalten, oder die Platine von der Stromversorgung trennen.
- Den Mikrocontroller PIC16F877 (IC1) sowie den Timer 7555 (IC2) im **ausgeschalteten** Zustand einsetzen. **Achtung:** Auf die Polarität achten!
- Stromversorgung wieder einschalten oder anschließen.
- Falls vorhanden: mit einem Oszilloskop oder mit einem Logic-Analyser am Pin 3 von IC2 (7555) messen: hier sollte eine rechteckförmige Spannung mit einer Frequenz von ca. 2kHz und eine Amplitude von 5V gemessen werden.

### Schritt 5: Zusammenbau

- An den 4 Ecken je eine Distanz M3x10 mit je einer Zahnscheibe und einer Mutter M3 gemäß Bild 6.4. anbringen.
- Zur Befestigung des Grafik-LC-Display 4 Distanzen M2,5x8 mit je einer Zahnscheibe und Zylinderkopfschraube M2,5x6 gemäß Abbildung 6.4 anbringen. Eventuell Grafik-LC-Display mit M2,5-Muttern befestigen.

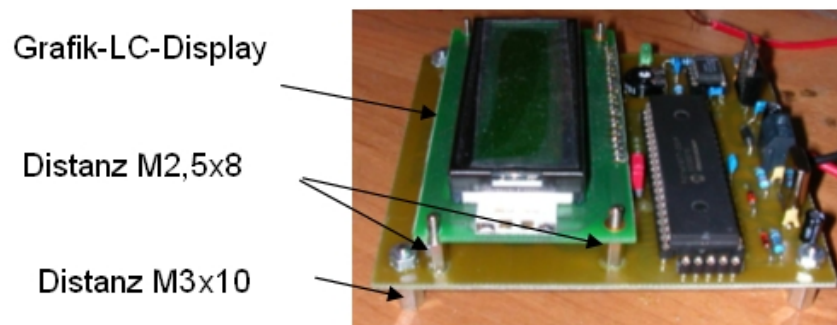


Abbildung 6.4: Zusammenbau

### 6.3 Softwarebeispiel 1 (“HELLO WORLD“ in Assembler)

Das erste Demonstrationsbeispiel soll mit Hilfe der Assembler-Basis-Routinen (aus Abschnitt 4.3) den Text “HELLO WORLD“ am Grafik-LC-Display ausgeben. (Abbildung 6.5)

Die Programmierung des Mikrocontrollers erfolgt mittels ICSP-Schnittstelle (siehe Anhang D ab Seite 81).



Abbildung 6.5: Demo 1 (alte Platinenversion)

#### 6.3.1 Listing

```

;*****
; ** Routinen zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) **
; **                                                                    **
; ** Demonstrationsbeispiel 1                                           **
; ** Ausgabe des Textes "HELLO WORLD" nur mit Hilfe der "Basisroutinen" **

```

```

; **
; ** Entwickler: Buchgeher Stefan
; ** Entwicklungsbeginn der Software: 22. Juli 2005
; ** Funktionsfaehig seit: 22. Juli 2005
; ** Letzte Bearbeitung: 8. September 2006
; ****

List p=PIC16F877

; **** Register (in Registerseite 0) ****
STAT      equ    3      ;Statusregister
PORTB     equ    6      ;Port B-Register
PORTD     equ    8      ;Port D-Register

; **** Register (in Registerseite 1) ****
TRISB     equ    6      ;Richtungsregister Port B
TRISD     equ    8      ;Richtungsregister Port D

; **** Eigene Register (in Registerbank 0) ****
TEMP1     equ    20     ;allgemeines Hilfsregister 1
TEMP2     equ    21     ;allgemeines Hilfsregister 2

; **** Bits in Registern ****
RPO       equ    5      ;Seitenauswahlbit 1 im Statuswort-Register
RP1       equ    6      ;Seitenauswahlbit 2 im Statuswort-Register

; **** Portbelegung ****
GLCD_DATA equ    PORTD
GLCD_DATA_TRIS equ TRISD
GLCD_CTRL equ    PORTB
GLCD_CTRL_TRIS equ TRISB

GLCD_A0   equ    5
GLCD_CS2  equ    4
GLCD_CS1  equ    3
GLCD_RD   equ    2
GLCD_WR   equ    1

; **** Konstanten ****
; keine Konstanten

; **** Ziele der Registeroperationen ****
w      equ    0
f      equ    1

; **** Makros ****
bank0   MACRO
        bcf    STAT,RPO
        bcf    STAT,RP1
        ENDM

bank1   MACRO
        bsf    STAT,RPO
        bcf    STAT,RP1
        ENDM

; **** Konfigurations-Bits ****
__config b'11110100111010'
;
;      +-+-----+ Bit 13-12, 5-4 (CP1:CP0): FLASH Memory Code Protection
;      |||||  |||  0 0 : 0000h to 1FFFh code protected
;      |||||  |||  0 1 : 1000h to 1FFFh code protected
;      |||||  |||  1 0 : 1F00h to 1FFFh code protected

```

```

;          |||||  |||  -> 1 1 : Code protection off
;          +----- Bit 11 (DEBUG): In-Circuit Debugger Mode
;          |||||  |||  0 : DEBUG on (= enabled)
;          |||||  |||  -> 1 : DEBUG off (= disabled)
;          +----- Bit 10 : Reserve
;          +----- Bit 9 (WRT): FLASH Programm Memory Write Enable
;          |||  |||  -> 0 : WRT off (= disabled)
;          |||  |||  1 : WRT on (= enabled)
;          +----- Bit 8 (CPD): Data EE Memory Code Protection
;          ||  |||  0 : CPD on (= enabled)
;          ||  |||  -> 1 : CPD off (= disabled)
;          +----- Bit 7 (LVP): Low Voltage ICSP Enable Bit
;          |  |||  -> 0 : LVP off (= disabled)
;          |  |||  1 : LVP on (= enabled)
;          +----- Bit 6 (BODEN): Brown-Out Detection
;          |||  -> 0 : BODEN on (= disabled)
;          |||  1 : BODEN off (= enabled)
;          +----- Bit 3 (PWRT): Power Up Timer
;          |||  0 : PWRT on (= enabled)
;          |||  -> 1 : PWRT off (= disabled)
;          +----- Bit 2 (WDTE): Watchdog Timer
;          ||  -> 0 : WDT off (= disabled)
;          ||  1 : WDT on (= enabled)
;          +-+ Bit 1-0 (FOSC2:FOSC0): Oszillator Selection
;          0 0 : LP Oszillator
;          0 1 : XT Oszillator
;          -> 1 0 : HS Oszillator
;          1 1 : RC Oszillator

ORG 0x000
goto Beginn
ORG 0x004
goto ISR

;***** ISR *****
ISR      retfie      ;keine ISR vorhanden

;***** Unterprogramme *****

;*****
;** Initialisierung des Prozessor: **
;** + Ports: Port A: unbenutzt **
;** Port B: Steuersignale fuer Grafik-LC-Display (Ausgaenge) **
;** Port C: unbenutzt **
;** Port D: 8-Bit-Daten fuer Grafik-LC-Display (Ausgaenge) **
;** Port E: unbenutzt **
;*****
INIT      bank1      ;Registerseite 1
          movlw b'00000000' ;Port B und Port D als Ausgang definieren
          movwf TRISB
          movwf TRISD
          bank0      ;Registerseite 0

          return

;***** Grafik-LCD Routinen *****

;*****
;** GLCD_INIT **
;** **
;** Aufgabe: **
;** Grafik-LC-Display initialisieren **
;** **
;** Vorgehensweise: **
;** Uebergibt nacheinander die Befehle zur Initialisierung an das Grafik-LC-Display. **
;*****
GLCD_INIT      movlw .175      ;Display on

```

```

        call    GLCD_COMMAND

        movlw   .164                ;static drive off
        call    GLCD_COMMAND

        movlw   .169                ;duty cyle: 1/32
        call    GLCD_COMMAND

        movlw   .160                ;ADC: CW output (forward)
        call    GLCD_COMMAND

        movlw   .238                ;read modify write off
        call    GLCD_COMMAND

        movlw   .192                ;line 0
        call    GLCD_COMMAND

        movlw   .184                ;1. Zeile
        call    GLCD_COMMAND

        movlw   .0                  ;1. Spalte
        call    GLCD_COMMAND

        return

;*****
;** GLCD_COMMAND
;**
;** Aufgabe:
;** Umsetzung des Zeitdiagramms zum Senden eines Befehls an das Grafik-LC-Display.
;**
;**
;**
;**      A0      -----X-----X-----
;**      nCS1,nCS2  -----X-----X-----
;**      nRD
;**      nWR      -----\-----/-----
;**      Data     -----X-----X-----
;**
;** Uebergabeparameter
;** Das Arbeitsregister (w-Register) beinhaltet den Befehl.
;*****
GLCD_COMMAND    bcf     GLCD_CTRL, GLCD_A0
                bcf     GLCD_CTRL, GLCD_CS1
                bcf     GLCD_CTRL, GLCD_CS2

                bsf     GLCD_CTRL, GLCD_RD
                bcf     GLCD_CTRL, GLCD_WR

                movwf   GLCD_DATA                ;Befehl am Datenport ausgeben

                bsf     GLCD_CTRL, GLCD_WR
                bsf     GLCD_CTRL, GLCD_CS1
                bsf     GLCD_CTRL, GLCD_CS2

        return

;*****
;** GLCD_DATA1
;**
;** Aufgabe:
;** Umsetzung des Zeitdiagramms zum Senden von Daten an die erste Haelfte des
;** Grafik-LC-Display.
;**
;**
;**
;**      -----

```

```

; **      A0, nCS1  ----X                      X-----
; **
; **      nCS2     ----X-----X-----
; **
; **      nRD
; **
; **      nWR      ----\-----/-----
; **
; **      Data     ----X-----X-----
; **
; **  Uebergabeparameter
; **      Das Arbeitsregister (w-Register) beinhaltet die Daten.
; ****
GLCD_DATA1    bsf    GLCD_CTRL, GLCD_A0
               bsf    GLCD_CTRL, GLCD_CS1
               bcf    GLCD_CTRL, GLCD_CS2

               bsf    GLCD_CTRL, GLCD_RD
               bcf    GLCD_CTRL, GLCD_WR

               movwf  GLCD_DATA           ;Daten am Datenport ausgeben

               bsf    GLCD_CTRL, GLCD_WR
               bsf    GLCD_CTRL, GLCD_CS1
               bsf    GLCD_CTRL, GLCD_CS2

               return

; ****
; ** GLCD_DATA2
; **
; ** Aufgabe:
; **      Umsetzung des Zeitdiagramms zum Senden von Daten an die zweite Haelfte des
; **      Grafik-LC-Display.
; **
; **
; **      A0, nCS2  ----X                      X-----
; **
; **      nCS1     ----X-----X-----
; **
; **      nRD
; **
; **      nWR      ----\-----/-----
; **
; **      Data     ----X-----X-----
; **
; **  Uebergabeparameter
; **      Das Arbeitsregister (w-Register) beinhaltet die Daten.
; ****
GLCD_DATA2    bsf    GLCD_CTRL, GLCD_A0
               bcf    GLCD_CTRL, GLCD_CS1
               bsf    GLCD_CTRL, GLCD_CS2

               bsf    GLCD_CTRL, GLCD_RD
               bcf    GLCD_CTRL, GLCD_WR

               movwf  GLCD_DATA           ;Daten am Datenport ausgeben

               bsf    GLCD_CTRL, GLCD_WR
               bsf    GLCD_CTRL, GLCD_CS1
               bsf    GLCD_CTRL, GLCD_CS2

               return

; ****
; ** GLCD_CLEAR
; **

```



```

** Aufgabe: **
** Loeschen des Grafik-LC-Displays. **
** **
** Vorgehensweise: **
** Beginnend bei der letzten Zeile und letzten Spalte mit Hilfe zweier geschachtelten **
** Schleifen Nullen zum Grafik-LC-Display (in beide Display-Haelften) schreiben. Dies **
** bewirkt, dass am Display nichts angezeigt wird bzw. das Display wird somit gelöscht. **
** **
** Anmerkung: **
** Die beiden temporären Register (TEMP1 und TEMP2) dienen hier nur als Schleifen- **
** zaehler. Sie koennen daher auch in anderen Unterprogrammen verwendet werden. **
** - TEMP1...Zeilenzaehler (3 bis 0) **
** - TEMP2...Spaltenzaehler (61 bis 0) **
*****
GLCD_CLEAR    movlw    .4
               movwf    TEMP1
GLCD_CLR_SCHL1: decf    TEMP1,w
               addlw    .184          ;Befehl (184 + TEMP1 = Zeile )
               call     GLCD_COMMAND ; an das Grafik-LC-Display

               movlw    .61
               movwf    TEMP2

               movlw    .0            ;Befehl: 1. Spalte an das Grafik-LC-Display
               call     GLCD_COMMAND

GLCD_CLR_SCHL2: movlw    .0
               call     GLCD_DATA1
               call     GLCD_DATA2

               decfsz    TEMP2,f
               goto     GLCD_CLR_SCHL2

               decfsz    TEMP1,f
               goto     GLCD_CLR_SCHL1

               return

***** Demobeispiel 1 *****

** AUSGABE **
** **
** Aufgabe: **
** Den Text "HELLO WORLD" am Grafik-LC-Display ausgeben. **
** **
** Vorgehensweise: **
** - Display loeschen (mit Hilfe des Unterprogramms GLCD_CLEAR). **
** - "Cursor" an die 1.Zeile und 1.Spalte setzen (mit Hilfe des Unterprogramms **
** GLCD_COMMAND). **
** - Hier nun den Text "HELLO" ausgeben. Dazu muessen nacheinander die Daten fuer die **
** einzelnen Buchstaben mit Hilfe des Unterprogramms GLCD_DATA1 an das Grafik-LC- **
** Display uebergeben werden. **
** 1 X X XXXXX X X XXX **
** 2 X X X X X X X X **
** 4 X X X X X X X X **
** 8 XXXXX XXXX X X X X **
** 16 X X X X X X X X **
** 32 X X X X X X X X **
** 64 X X XXXXX XXXXX XXXX XXX **
** 128 **
** ||||| **
** +----- 127 **
** +----- 8 **
** +----- 8 **
** +----- 8 **
** +----- 127 **
** +----- 0 **
** +----- 127 **
** +----- 73 **

```

```

; **                                     **
; **                                     **
; ** - "Cursor" an die 2.Zeile und 1.Spalte setzen (mit Hilfe des Unterprogramms   **
; ** GLCD_COMMAND).                                                              **
; ** - Hier nun den Text "WORLD" ausgeben. Dazu muessen nacheinander die Daten fuer die **
; ** einzelnen Buchstaben mit Hilfe des Unterprogramms GLCD_DATA1 an das Grafik-LC- **
; ** Display uebergeben werden.                                                 **
; **      1 X  X  XXX  XXXX  X      XXXX                                     **
; **      2 X  X X  X X  X X      X  X                                     **
; **      4 X  X X  X X  X X      X  X                                     **
; **      8 X X X X  X XXXX  X      X  X                                     **
; **     16 X X X X  X X X  X      X  X                                     **
; **     32 XX XX X  X X  X X      X  X                                     **
; **     64 X  X  XXX  X  X XXXXX XXXX                                     **
; **    128                                     **
; **      |||||||||||||||||||||||||                                     **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 127 **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 32  **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 24  **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 32  **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 127 **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 0   **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 62  **
; **      +-----+-----+-----+-----+-----+-----+-----+-----+ 65  **
; **                                     usw.                                     **
; ****
AUSGABE      call    GLCD_CLEAR              ;Display loeschen

              movlw   .184                    ;1. Zeile
              call    GLCD_COMMAND

              movlw   .0                      ;1. Spalte (der ersten Haelfte)
              call    GLCD_COMMAND

              ;Code fuer "HELLO"
              movlw   .127                    ;Code fuer "H"
              call    GLCD_DATA1
              movlw   .8
              call    GLCD_DATA1
              movlw   .8
              call    GLCD_DATA1
              movlw   .8
              call    GLCD_DATA1
              movlw   .127
              call    GLCD_DATA1
              movlw   .0
              call    GLCD_DATA1

              movlw   .127                    ;Code fuer "E"
              call    GLCD_DATA1
              movlw   .73
              call    GLCD_DATA1
              movlw   .73
              call    GLCD_DATA1
              movlw   .73
              call    GLCD_DATA1
              movlw   .65
              call    GLCD_DATA1
              movlw   .0
              call    GLCD_DATA1

              movlw   .127                    ;Code fuer "L"
              call    GLCD_DATA1
              movlw   .64
              call    GLCD_DATA1
              movlw   .64
              call    GLCD_DATA1
              movlw   .64
              call    GLCD_DATA1
              movlw   .64
              call    GLCD_DATA1
              movlw   .64
              call    GLCD_DATA1

```

```

movlw .0
call GLCD_DATA1

movlw .127                ;Code fuer "L"
call GLCD_DATA1
movlw .64
call GLCD_DATA1
movlw .64
call GLCD_DATA1
movlw .64
call GLCD_DATA1
movlw .64
call GLCD_DATA1
movlw .0
call GLCD_DATA1

movlw .62                ;Code fuer "0"
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .62
call GLCD_DATA1
movlw .0
call GLCD_DATA1

movlw .185                ;2. Zeile
call GLCD_COMMAND

clrw                      ;1. Spalte (der ersten Haelfte)
call GLCD_COMMAND

;Code fuer "WORLD"
movlw .127                ;Code fuer "W"
call GLCD_DATA1
movlw .32
call GLCD_DATA1
movlw .24
call GLCD_DATA1
movlw .32
call GLCD_DATA1
movlw .127
call GLCD_DATA1
movlw .0
call GLCD_DATA1

movlw .62                ;Code fuer "0"
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .65
call GLCD_DATA1
movlw .62
call GLCD_DATA1
movlw .0
call GLCD_DATA1

movlw .127                ;Code fuer "R"
call GLCD_DATA1
movlw .9
call GLCD_DATA1
movlw .25
call GLCD_DATA1
movlw .41
call GLCD_DATA1
movlw .70

```

```

        call    GLCD_DATA1
        movlw   .0
        call    GLCD_DATA1

        movlw   .127                ;Code fuer "L"
        call    GLCD_DATA1
        movlw   .64
        call    GLCD_DATA1
        movlw   .64
        call    GLCD_DATA1
        movlw   .64
        call    GLCD_DATA1
        movlw   .64
        call    GLCD_DATA1
        movlw   .0
        call    GLCD_DATA1

        movlw   .127                ;Code fuer "D"
        call    GLCD_DATA1
        movlw   .65
        call    GLCD_DATA1
        movlw   .65
        call    GLCD_DATA1
        movlw   .65
        call    GLCD_DATA1
        movlw   .62
        call    GLCD_DATA1
        movlw   .0
        call    GLCD_DATA1

        return

;***** Hauptprogramm *****
;*****
;** Aufgaben des Hauptprogramms: **
;** + Controller initialisieren (Unterprogramm INIT) **
;** + Grafik-LCD initialisieren (Unterprogramm GLCD_INIT) **
;** + Text zur Demonstration ausgeben (Unterprogramm AUSGABE) **
;** + Endlosschleife **
;*****
Beginn      call    INIT                ;Controller initialisieren
            call    GLCD_INIT            ;Grafik-Display initialisieren

            call    AUSGABE                ;Ausgabe eines Textes

Schleife    goto    Schleife

            end

```

### 6.3.2 Anmerkungen zur Assembler-Software

Die Software besteht im Wesentlichen aus einem kurzen Hauptprogramm, die im Abschnitt 4 beschriebenen Unterprogramme und einem weiteren Unterprogramm namens AUSGABE.

Das Hauptprogramm besteht nach der Initialisierung des Controllers (Unterprogramm INIT) und des Grafik-LC-Displays (Unterprogramm GLCD\_INIT) nur mehr aus einem Unterprogramm zur Ausgabe des Textes am Grafik-LC-Display (Unterprogramm AUSGABE) und aus einer Endlosschleife.

Das Unterprogramm INIT dient zur Initialisierung des Mikrocontrollers. In diesem Beispiel werden nur die Ports B und D als Ausgang konfiguriert. Das Unterprogramm GLCD\_INIT

initialisiert das Grafik-LC-Display (siehe Abschnitt 4.3.1).

Nach der Initialisierung des Grafik-LC-Displays (Unterprogramm GLCD\_INIT) ist das Grafik-LC-Display bereit Zeichen oder Werte anzuzeigen. Die Ausgabe der anzuzeigenden Zeichen übernimmt hier das Unterprogramm AUSGABE. Zuerst muss das Display mit dem Unterprogramm GLCD\_CLEAR gelöscht werden und der Cursor an die erste Zeile und erste Spalte gesetzt werden. Nun können beliebige Zeichen, oder Symbole an das Grafik-LC-Display gemäß Abschnitt 3.3. übertragen werden.

## 6.4 Softwarebeispiel 2 in C mit CC5X

Das zweite Demonstrationsbeispiel soll mit Hilfe der C-Routinen (aus Abschnitt 5.3.) die Informationen gemäß Abbildung 6.6 (Uhrzeit und Datum) am Grafik-LC-Display anzeigen. Die angezeigten Daten (hier Uhrzeit und Datum) sind in den Registern UHRZEIT\_STD, UHRZEIT\_MIN, UHRZEIT\_SEK, DATUM\_TAG und KALENDERWOCHE gespeichert. Die Texte für den Wochentag, Monat und das Jahr werden direkt als ASCII-Zeichen an das Grafik-LC-Display gesendet. Wie die Uhrzeit und das Datum zustande kommen ist bei diesem Demonstrationsbeispiel nicht von Bedeutung.

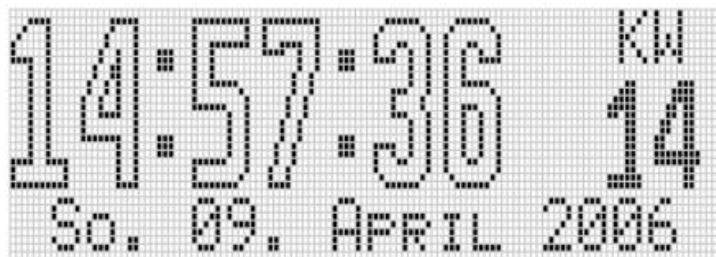


Abbildung 6.6: Demo 2

Aufgrund der 1k-Grenze des CC5X-Kompilers muss der C-Code in mehrere C-Dateien aufgeteilt werden. (siehe Anhang E ab Seite 84).

Die Programmierung des Mikrocontrollers erfolgt auch hier mittels ICSP-Schnittstelle (siehe Anhang D ab Seite 81).

### 6.4.1 Listings

Datei demo2.c

```

/*****
/* Demonstrationsbeispiel 2 zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) in */
/* C (CC5X)                                                                    */
/*                                                                            */
/* Entwickler: Buchgeher Stefan                                              */
/* Entwicklungsbeginn der Software: 3. Maerz 2006                          */
/* Funktionsfaehig seit: 2. April 2006                                       */
/* Letzte Bearbeitung: 8. September 2006                                    */
/*****/

/***** Include-Dateien *****/
#include "PROJEKT.H"

```

```

/***** Strukturen *****/
/* keine Strukturen verwendet */

/***** Externe Register *****/
unsigned char UHRZEIT_STD;      // beinhaltet die Stunden
unsigned char UHRZEIT_MIN;     // beinhaltet die Minuten
unsigned char UHRZEIT_SEK;     // beinhaltet die Sekunden
unsigned char DATUM_TAG;       // beinhaltet den Kalendertag
unsigned char KALENDERWOCHE;   // beinhaltet die Kalenderwoche

/***** Bits in den externen Registern *****/
/* keine externe Register */

/***** Konstanten *****/
/* keine Konstanten */

/***** Konfigurations-Bits *****/
#pragma config |= 0b.11110100111010
/*
    +-----+-----+ Bit 13-12, 5-4 (CP1:CP0): FLASH Memory Code Protection
    ||||| |||| 0 0 : 0000h to 1FFFh code protected
    ||||| |||| 0 1 : 1000h to 1FFFh code protected
    ||||| |||| 1 0 : 1F00h to 1FFFh code protected
    ||||| |||| -> 1 1 : Code protection off
    +-----+-----+ Bit 11 (DEBUG): In-Circuit Debugger Mode
    ||||| |||| 0 : DEBUG on (= enabled)
    ||||| |||| -> 1 : DEBUG off (= disabled)
    +-----+-----+ Bit 10 : Reserve
    +-----+-----+ Bit 9 (WRT): FLASH Programm Memory Write Enable
    ||| |||| -> 0 : WRT off (= disabled)
    ||| |||| 1 : WRT on (= enabled)
    +-----+-----+ Bit 8 (CPD): Data EE Memory Code Protection
    || |||| 0 : CPD on (= enabled)
    || |||| -> 1 : CPD off (= disabled)
    +-----+-----+ Bit 7 (LVP): Low Voltage ICSP Enable Bit
    | |||| -> 0 : LVP off (= disabled)
    | |||| 1 : LVP on (= enabled)
    +-----+-----+ Bit 6 (BODEN): Brown-Out Detection
    |||| -> 0 : BODEN on (= disabled)
    |||| 1 : BODEN off (= enabled)
    +-----+-----+ Bit 3 (PWRT): Power Up Timer
    ||| 0 : PWRT on (= enabled)
    ||| -> 1 : PWRT off (= disabled)
    +-----+-----+ Bit 2 (WDTE): Watchdog Timer
    || -> 0 : WDT off (= disabled)
    || 1 : WDT on (= enabled)
    +-----+-----+ Bit 1-0 (FOSC2:FOSC0): Oszillator Selection
    0 0 : LP Oszillator
    0 1 : XT Oszillator
    -> 1 0 : HS Oszillator
    1 1 : RC Oszillator
*/

/***** Funktionsprototypen *****/
/* Initialisierung des Mikrocontroller */
void INIT(void);

/* Unterprogramme zur Datenausgabe */
void AUSGABE(void);
void AUSGABEBEBCD_5x7(unsigned char wert);
void AUSGABEBEBCD_8x14(unsigned char wert);
void AUSGABEBEBCD_10x22(unsigned char wert);

/***** ISR - Timer0 *****/
/* keine ISR verwendet */

```

```

/***** Unterprogramme und Funktionen *****/

/***** externe Unterprogramme einbinden *****/
/* Mathematik-Routinen */
#include <MATH16.H>

/*****
/* INIT:
/*
/* Aufgabe:
/*   Initialisierung des Prozessor:
/*     + Port B und Port D als Ausgang definieren.
/*     + Die (externen) Register fuer die Uhrzeit und Datum mit (plausiblen) Werten
/*       laden (dient hier nur zur Demonstration!)
/*
/*
/* Uebergabeparameter:
/*   zeichen
/*
/* Rueckgabeparameter:
/*   keiner
*****/
void INIT(void)
{
    TRISB = 0;                // Port B als Ausgang definieren
    TRISD = 0;                // Port D als Ausgang definieren

    UHRZEIT_STD = 14;         // Die (externen) Register fuer die Uhrzeit
    UHRZEIT_MIN = 57;         //   und Datum mit (plausiblen) Werten laden
    UHRZEIT_SEK = 36;         //   (dient hier nur zur Demonstration!)
    DATUM_TAG = 9;
    KALENDERWOCHE = 14;
}

/*****
/* AUSGABE:
/*
/* Aufgabe:
/*   Uhrzeit und Datum am Grafik-LC-Display ausgeben.
/*
/*
/* Uebergabeparameter:
/*   keine
/*
/*
/* Rueckgabeparameter:
/*   keiner
/*
/*
/* Vorgehensweise:
/*   + Grafik-LC-Display loeschen (Unterprogramm GLCD_CLEAR)
/*   + Fuer jeden auszugebenden Wert oder Text gilt:
/*     + Cursor an die gewuenschte Position setzen (Unterprogramm GLCD_SETCURSOR_ZEILE
/*       fuer die Zeile und Unterprogramm GLCD_SETCURSOR_ZEILE fuer die Spalte).
/*     + Den Wert oder Text in der gewuenschten Groesse am Grafik-LC-Display ausgeben.
*****/
void AUSGABE(void)
{
    GLCD_CLEAR();            // Display loeschen

    // Uhrzeit (Stunde)
    GLCD_SETCURSOR_ZEILE(0); // Cursor an die 1.Zeile und 1. Spalte
    GLCD_SETCURSOR_SPALTE(0); //   setzen
    AUSGABEBCD_10x22(UHRZEIT_STD); // Stunde am Grafik-LC-Display ausgeben

    // Doppelpunkt
    GLCD_SETCURSOR_SPALTE(25);
    GLCD_DATA_ZEICHEN(224);
    GLCD_DATA_ZEICHEN(224);
    GLCD_DATA_ZEICHEN(224);

```

```

GLCD_SETCURSORS_ZEILE(2);
GLCD_SETCURSORS_SPALTE(25);
GLCD_DATA_ZEICHEN(7);
GLCD_DATA_ZEICHEN(7);
GLCD_DATA_ZEICHEN(7);

// Uhrzeit (Minute)
GLCD_SETCURSORS_ZEILE(0);           // Cursor an die 1.Zeile und 32. Spalte
GLCD_SETCURSORS_SPALTE(31);         // setzen
AUSGABEBCD_10x22(UHRZEIT_MIN);      // Minute am Grafik-LC-Display ausgeben

// Doppelpunkt
GLCD_SETCURSORS_SPALTE(56);
GLCD_DATA_ZEICHEN(224);
GLCD_DATA_ZEICHEN(224);
GLCD_DATA_ZEICHEN(224);
GLCD_SETCURSORS_ZEILE(2);
GLCD_SETCURSORS_SPALTE(56);
GLCD_DATA_ZEICHEN(7);
GLCD_DATA_ZEICHEN(7);
GLCD_DATA_ZEICHEN(7);

// Uhrzeit (Sekunde)
GLCD_SETCURSORS_ZEILE(0);           // Cursor an die 1.Zeile und 63. Spalte
GLCD_SETCURSORS_SPALTE(62);         // setzen
AUSGABEBCD_10x22(UHRZEIT_SEK);      // Sekunde am Grafik-LC-Display ausgeben

// Datum
GLCD_SETCURSORS_ZEILE(3);           // Cursor an die 4.Zeile und 8. Spalte
GLCD_SETCURSORS_SPALTE(7);          // setzen
GLCD_ZEICHEN_5x7('S');              // Datum am Grafik-LC-Display ausgeben
GLCD_ZEICHEN_5x7('o');
GLCD_ZEICHEN_5x7(' ');
GLCD_ZEICHEN_5x7(' ');
AUSGABEBCD_5x7(DATUM_TAG);
GLCD_ZEICHEN_5x7(' ');
GLCD_ZEICHEN_5x7(' ');
GLCD_ZEICHEN_5x7('A');
GLCD_ZEICHEN_5x7('p');
GLCD_ZEICHEN_5x7('r');
GLCD_ZEICHEN_5x7('i');
GLCD_ZEICHEN_5x7('l');
GLCD_ZEICHEN_5x7(' ');
GLCD_ZEICHEN_5x7('2');
GLCD_ZEICHEN_5x7('0');
GLCD_ZEICHEN_5x7('0');
GLCD_ZEICHEN_5x7('6');

// Kalenderwoche
GLCD_SETCURSORS_ZEILE(0);           // Cursor an die 1.Zeile und 105. Spalte
GLCD_SETCURSORS_SPALTE(104);        // setzen
GLCD_ZEICHEN_5x7('K');              // Text ('KW') am Grafik-LC-Display ausgeben
GLCD_ZEICHEN_5x7('W');

GLCD_SETCURSORS_ZEILE(1);           // Cursor an die 2.Zeile und 101. Spalte
GLCD_SETCURSORS_SPALTE(100);        // setzen
AUSGABEBCD_8x14(KALENDERWOCHE);    // Kalenderwoche am Grafik-LC-Display ausgeben
}

/*****
/* AUSGABEBCD_5x7:
/*
/* Aufgabe:
/*   Den uebergeben Wert in BCD-Form umwandeln und am Display ausgeben.
/*
/* Uebergabeparameter:
/*   wert
/*
*****/

```



```

/* Rueckgabeparameter:                                     */
/*   keiner                                                */
/*                                                         */
/* Vorgehensweise:                                         */
/*   + Hunderterstelle ermitteln (diese wird aber nicht Grafik-LC-Display ausgegeben!) */
/*   + Zehnerstelle ermitteln und mit dem Unterprogramm GLCD_ZEICHEN_5x7 am Grafik-LC- */
/*     Display ausgeben.                                   */
/*   + Die uebrigbleibende Einerstelle mit dem Unterprogramm GLCD_ZEICHEN_5x7 am   */
/*     Grafik-LC-Display ausgeben.                                   */
/* *****/
void AUSGABEBCD_5x7(unsigned char wert)
{
    unsigned char help;
    unsigned char help_modulo;

    // Hunderterstelle
    help = (char)(wert / 100);          // "Hunderterstelle" ermitteln
    // GLCD_ZEICHEN_5x7(help);
    help_modulo = wert % 100;

    // Zehnerstelle
    help = (char)(help_modulo / 10);    // "Zehnerstelle" ermitteln
    GLCD_ZEICHEN_5x7(help);             //   und am LC-Display ausgeben
    help_modulo = help_modulo % 10;

    // Einerstelle
    GLCD_ZEICHEN_5x7(help_modulo);      // "Einerstelle" am LC-Display ausgeben
}

/* *****/
/* AUSGABEBCD_8x14:                                       */
/*                                                         */
/* Aufgabe:                                               */
/*   Den uebergeben Wert in BCD-Form umwandeln und am Display ausgeben.           */
/*                                                         */
/* Uebergabeparameter:                                   */
/*   wert                                                */
/*                                                         */
/* Rueckgabeparameter:                                   */
/*   keiner                                              */
/*                                                         */
/* Vorgehensweise:                                         */
/*   + Hunderterstelle ermitteln (diese wird aber nicht Grafik-LC-Display ausgegeben!) */
/*   + Zehnerstelle ermitteln und mit dem Unterprogramm GLCD_ZEICHEN_8x14 am Grafik-LC- */
/*     Display ausgeben.                                   */
/*   + Die uebrigbleibende Einerstelle mit dem Unterprogramm GLCD_ZEICHEN_8x14 am   */
/*     Grafik-LC-Display ausgeben.                                   */
/* *****/
void AUSGABEBCD_8x14(unsigned char wert)
{
    unsigned char help;
    unsigned char help_modulo;

    // Hunderterstelle
    help = (char)(wert / 100);          // "Hunderterstelle" ermitteln
    // GLCD_ZEICHEN_8x14(help);
    help_modulo = wert % 100;

    // Zehnerstelle
    help = (char)(help_modulo / 10);    // "Zehnerstelle" ermitteln
    GLCD_ZEICHEN_8x14(help);            //   und am LC-Display ausgeben
    help_modulo = help_modulo % 10;

    // Einerstelle
    GLCD_ZEICHEN_8x14(help_modulo);      // "Einerstelle" am LC-Display ausgeben
}

/* *****/
/* AUSGABEBCD_10x22:                                     */

```

```

/*                                                    */
/* Aufgabe:                                                    */
/*   Den uebergeben Wert in BCD-Form umwandeln und am Display ausgeben.      */
/*                                                    */
/* Uebergabeparameter:                                                    */
/*   wert                                                    */
/*                                                    */
/* Rueckgabeparameter:                                                    */
/*   keiner                                                    */
/*                                                    */
/* Vorgehensweise:                                                    */
/*   + Hunderterstelle ermitteln (diese wird aber nicht Grafik-LC-Display ausgegeben!) */
/*   + Zehnerstelle ermitteln und mit dem Unterprogramm GLCD_ZEICHEN_10x22 am Grafik- */
/*   LC-Display ausgeben.                                                    */
/*   + Die uebrigbleibende Einerstelle mit dem Unterprogramm GLCD_ZEICHEN_10x22 am */
/*   Grafik-LC-Display ausgeben.                                                    */
/*                                                    */
/*****
void AUSGABEBCD_10x22(unsigned char wert)
{
    unsigned char help;
    unsigned char help_modulo;

    // Hunderterstelle
    help = (char)(wert / 100);          // "Hunderterstelle" ermitteln
// GLCD_ZEICHEN_10x22(help);
    help_modulo = wert % 100;

    // Zehnerstelle
    help = (char)(help_modulo / 10);    // "Zehnerstelle" ermitteln
    GLCD_ZEICHEN_10x22(help);          //   und am LC-Display ausgeben
    help_modulo = help_modulo % 10;

    // Einerstelle
    GLCD_ZEICHEN_10x22(help_modulo);    // "Einerstelle" am LC-Display ausgeben
}

/***** Hauptprogramm *****/
/*****
/* Aufgaben des Hauptprogramms:                                                    */
/*   + Controller initialisieren (Unterprogramm INIT)                            */
/*   + Grafik-LC-Display initialisieren (Unterprogramm GLCD_INIT)                */
/*   + Uhrzeit und Datum am Grafik-LC-Display ausgeben (Unterprogramm AUSGABE)   */
/*   + Endlosschleife                                                            */
/*****
void main(void)
{
    INIT();                            // Controller initialisieren
    GLCD_INIT();                       // Grafik-LC-Display initialisieren

    AUSGABE();                         // Text am Grafik-LC-Display ausgeben

    while(1);                          // Endlosschleife
}

```

## Datei GLCD\_BASIS\_ROUTINEN.C

```

/*****
/* Basis-Unterprogramme zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) */
/* in C (CC5X)                                                    */
/*                                                    */
/* Entwickler: Buchgeher Stefan                                                    */
/* Entwicklungsbeginn der Software: 5. Maerz 2006                            */
/* Funktionsfaehig seit: 2. April 2006                                          */
/* Letzte Bearbeitung: 8. September 2006                                       */
/*****

/***** Projekt-Header einbinden *****/
#include "PROJEKT.H"

```

```

/***** externe Register *****/
unsigned char hlp_zeilen_zaeher;
unsigned char hlp_spalten_zaeher;

/***** Basis-Unterprogramme zur Grafik-LCD-Ansteuerung *****/

/*****
/* GLCD_INIT
/*
/* Aufgabe:
/* Grafik-LC-Display initialisieren.
/*
/* Uebergabeparameter:
/* keine
/*
/* Rueckgabeparameter:
/* keiner
/*
/* Vorgehensweise:
/* Uebergibt nacheinander die Befehle zur Initialisierung an das Grafik-LC-Display.
*****/
void GLCD_INIT(void)
{
    GLCD_COMMAND(175);          // Display on
    GLCD_COMMAND(164);          // static drive off
    GLCD_COMMAND(169);          // duty cyle: 1/32
    GLCD_COMMAND(160);          // ADC: CW output (forward)
    GLCD_COMMAND(238);          // read modify write off
    GLCD_COMMAND(192);          // line 0
    GLCD_COMMAND(184);          // 1. Zeile
    GLCD_COMMAND(0);            // 1. Spalte ("Column Address" auf 0 setzen)
}

/*****
/* GLCD_COMMAND
/*
/* Aufgabe:
/* Umsetzung des Zeitdiagramms zum Senden eines Befehls an das Grafik-LC-Display.
/*
/*
/*          A0          -----X-----X-----
/*          nCS1,nCS2  -----X-----X-----
/*          nRD
/*          nWR          -----\-----/-----
/*          Data          -----X-----X-----
/*
/* Uebergabeparameter:
/* command
/*
/* Rueckgabeparameter:
/* keiner
*****/
void GLCD_COMMAND(unsigned char command)
{
    GLCD_A0 = 0;
    GLCD_CS1 = 0;
    GLCD_CS2 = 0;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = command;

```

```

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}

/*****
/* GLCD_DATA1
/*
/* Aufgabe:
/* Umsetzung des Zeitdiagramms zum Senden von Daten an die erste Haelfte des
/* Grafik-LC-Display.
/*
/*
/*          A0, nCS1  -----X-----X-----
/*          nCS2      -----X-----X-----
/*          nRD
/*          nWR        -----\-----/-----
/*          Data       -----X-----X-----
/*
/* Uebergabeparameter:
/* data
/*
/* Rueckgabeparameter:
/* keiner
*****/
void GLCD_DATA1(unsigned char data)
{
    GLCD_A0 = 1;
    GLCD_CS1 = 1;
    GLCD_CS2 = 0;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = data;

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}

/*****
/* GLCD_DATA2
/*
/* Aufgabe:
/* Umsetzung des Zeitdiagramms zum Senden von Daten an die zweite Haelfte des
/* Grafik-LC-Display.
/*
/*
/*          A0, nCS2  -----X-----X-----
/*          nCS1      -----X-----X-----
/*          nRD
/*          nWR        -----\-----/-----
/*          Data       -----X-----X-----
/*
/* Uebergabeparameter:
/* data
/*
/* Rueckgabeparameter:
*****/

```

```

/*  keiner                                                    */
/*****
void GLCD_DATA2(unsigned char data)
{
    GLCD_A0 = 1;
    GLCD_CS1 = 0;
    GLCD_CS2 = 1;

    GLCD_RD = 1;
    GLCD_WR = 0;

    GLCD_DATA = data;

    GLCD_WR = 1;

    GLCD_CS1 = 1;
    GLCD_CS2 = 1;
}

/*****
/* GLCD_DATA_ZEICHEN                                         */
/*                                                           */
/* Aufgabe:                                                  */
/*   Je nach aktueller Position des Cursors den uebergebenen Wert (data) an die erste */
/*   oder zweite Displayhaelfte uebergeben.                  */
/*                                                           */
/* Uebergabeparameter:                                       */
/*   data                                                    */
/*                                                           */
/* Rueckgabeparameter:                                       */
/*   keiner                                                  */
/*                                                           */
/* Vorgehensweise:                                           */
/*   + Befindet sich der Cursor an der 62. Spalte (hlp_spalten_zaehler = 61), die so */
/*     genannte "Column Address" auf 0 setzen (gemaess Datenblatt). Dies bewirkt, dass */
/*     der displayinterne Speicherzaehler wieder bei 0 beginnt. (siehe Datenblatt) */
/*   + Befindet sich der Cursor an eine Spalte groesser als 61 (also zweite Display- */
/*     haelfte) den Uebergabewert (data) mit dem Unterprogramm GLCD_DATA2 an die */
/*     zweite Displayhaelfte uebergeben. Andernfalls den Uebergabewert (data) mit dem */
/*     Unterprogramm GLCD_DATA1 an die erste Displayhaelfte uebergeben. */
/*                                                           */
/* Anmerkung:                                                */
/*   Das (externe) Hilfsregister hlp_spalten_zaehler gibt an, an welcher Spalte sich */
/*   der Cursor befindet.                                     */
/*****
void GLCD_DATA_ZEICHEN(unsigned char data)
{
    if (hlp_spalten_zaehler == 61)    // Wenn Cursorspalte = 61 ...
    {
        GLCD_COMMAND(0);              // ... "Column Address" auf 0 setzen
    }
    if (hlp_spalten_zaehler > 60)      // Cursorspalte > 60
    {
        GLCD_DATA2(data);             // ja: Uebergabewert (data) mit dem an
    }                                  // die zweite Displayhaelfte uebergeben.
    else
    {
        GLCD_DATA1(data);             // nein: Uebergabewert (data) mit dem an
    }                                  // die erste Displayhaelfte uebergeben.
    hlp_spalten_zaehler++;
}

/*****
/* GLCD_CLEAR                                                */
/*                                                           */
/* Aufgabe:                                                  */
/*   Loeschen des Grafik-LC-Displays.                      */
/*                                                           */
/* Uebergabeparameter:                                       */

```

```

/* keine */
/* */
/* Rueckgabeparameter: */
/* keiner */
/* */
/* Vorgehensweise: */
/* Beginnend bei der ersten Zeile und ersten Spalte mit Hilfe zweier geschachtelten */
/* Schleifen Nullen zum Grafik-LC-Display (in beide Display-Haelften) schreiben. Dies */
/* bewirkt, dass am Display nichts angezeigt wird bzw. das Display wird somit ge- */
/* loescht. */
/*****/
void GLCD_CLEAR(void)
{
    unsigned char Zeile, Spalte;

    for(Zeile = 0; Zeile < 4; Zeile++)
    {
        GLCD_COMMAND(184 + Zeile);    // Befehl: naechste Zeile
        GLCD_COMMAND(0);             // Befehl: 1.Spalte der Zeile ("Column
                                     // Address" auf 0 setzen)

        for(Spalte = 0; Spalte < 61; Spalte++)
        {
            GLCD_DATA1(0);
            GLCD_DATA2(0);
        }
    }
}

/*****/
/* GLCD_SETCURSOR_ZEILE */
/* */
/* Aufgabe: */
/* Den Cursor an die uebergabene Zeile (0 bis 3) setzen. */
/* */
/* Uebergabeparameter: */
/* zeile: 0 - 3 */
/* */
/* Rueckgabeparameter: */
/* keiner */
/* */
/* Vorgehensweise: */
/* Uebergabeparameter fuer die Zeile im globalen Register hlp_zeilen_zaeher sichern */
/* und an das Grafik-LC-Display uebergeben. */
/*****/
void GLCD_SETCURSOR_ZEILE(unsigned char zeile)
{
    hlp_zeilen_zaeher = zeile;
    GLCD_COMMAND(184 + zeile);
}

/*****/
/* GLCD_SETCURSOR_SPALTE */
/* */
/* Aufgabe: */
/* Den Cursor an die uebergabene Spalte (0 bis 121) setzen. */
/* */
/* Uebergabeparameter: */
/* spalte: 0 - 121 */
/* */
/* Rueckgabeparameter: */
/* keiner */
/* */
/* Vorgehensweise: */
/* Uebergabeparameter fuer die Spalte im globalen Register hlp_spalten_zaeher */
/* sichern und an das Grafik-LC-Display uebergeben. Dabei ist zu beachten, dass das */
/* Grafik-LC-Display in zwei Bereiche aufgeteilt ist, wobei beide Bereiche mit 0 */
/* beginnen. Spalten groesser als 61 befinden sich im zweiten Bereich. Da dieser */

```

```

/* wieder ab 0 beginnen, muss bei Spalten die groesser als 61 sind, 61 abgezogen */
/* werden. */
/*****
void GLCD_SETCURSOR_SPALTE(unsigned char spalte)
{
    hlp_spalten_zaeher = spalte;
    if (spalte < 61)
    {
        GLCD_COMMAND(spalte);
    }
    else
    {
        GLCD_COMMAND(spalte - 61);
    }
}
*/

```

### Datei GLCD\_FONT5X7.C

```

/*****
/* Unterprogramme zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) in C (CC5X) */
/* (Font 5x7) */
/* */
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 5. Maerz 2006 */
/* Funktionsfaehig seit: 2. April 2006 */
/* Letzte Bearbeitung: 8. September 2006 */
/*****

/***** Projekt-Header einbinden *****/
#include "PROJEKT.H"

/***** Unterprogramme zur Grafik-LCD-Ansteuerung *****/

/*****
/* GLCD_ZEICHEN_5x7 */
/* */
/* Aufgabe: */
/* Ein Zeichen (Character im Format 5x7) am Grafik-LC-Display ausgeben */
/* */
/* Uebergabeparameter: */
/* zeichen (im ASCII-Format) */
/* */
/* Rueckgabeparameter: */
/* keiner */
/* */
/* Vorgehensweise: */
/* Nacheinander die Spalten eines Zeichens an das Grafik-LC-Display mit dem Unter- */
/* programm GLCD_DATA_ZEICHEN uebergeben. Die Daten fuer die Spalten sind in */
/* Tabellen abgelegt, wobei fuer jede Spalte des Zeichens eine eigene Tabelle vor- */
/* handen ist. Diese Tabellen muessen an anderer Stelle hinzugefuegt werden! Die */
/* letzte Spalte muss eine 0 enthalten, also eine "Leerspalte". */
/*****
void GLCD_ZEICHEN_5x7(unsigned char zeichen)
{
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp1[zeichen]);

    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp2[zeichen]);

    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp3[zeichen]);

    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp4[zeichen]);

    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz5x7_sp5[zeichen]);
}

```

```

    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);
}

```

## Datei GLCD\_FONT8X14.C

```

/*****
/* Unterprogramme zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) in C (CC5X) */
/* (Font 8x14) */
/*
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 8. April 2006
/* Funktionsfaehig seit: 8. April 2006
/* Letzte Bearbeitung: 8. September 2006
*****/

/***** Projekt-Header einbinden *****/
#include "PROJEKT.H"

/***** Unterprogramme zur Grafik-LCD-Ansteuerung *****/

/*****
/* GLCD_ZEICHEN_8x14
/*
/* Aufgabe:
/* Ein Zeichen (Charakter im Format 8x14) am Grafik-LC-Display ausgeben.
/*
/* Uebergabeparameter:
/* zeichen
/*
/* Rueckgabeparameter:
/* keiner
/*
/* Vorgehensweise:
/* + Die externen Hilfsregister (hlp_zeilen_zaeher und hlp_spalten_zaeher) sichern,
/* da sie hier benoetigt, aber gleichzeitig hier auch veraendert werden.
/* + Nacheinander die Spalten fuer die erste Zeile des auszugebenden Zeichens an das
/* Grafik-LC-Display mit dem Unterprogramm GLCD_DATA_ZEICHEN uebergeben. Die Daten
/* fuer die Spalten sind in Tabellen abgelegt, wobei fuer jede Spalte des Zeichens
/* eine eigene Tabelle vorhanden ist. Diese Tabellen muessen an anderer Stelle
/* hinzugefuegt werden! Die letzte Spalte muss eine 0 enthalten, also eine "Leer-
/* spalte".
/* + Den Cursor an die naechste Zeile (tmp_zeile + 1) und an die erste Spalte des
/* Zeichens (tmp_spalte) setzen.
/* + In gleicher Weise wie vorher die Spalten fuer die zweite Zeile des auszugebenden
/* Zeichens an das Grafik-LC-Display mit dem Unterprogramm GLCD_DATA_ZEICHEN
/* uebergeben.
/* + Den Cursor an die erste Zeile des Zeichens (tmp_zeile) setzen.
*****/
void GLCD_ZEICHEN_8x14(unsigned char zeichen)
{
    char tmp_zeile = hlp_zeilen_zaeher;
    char tmp_spalte = hlp_spalten_zaeher;

    // 1. Zeile
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp6[zeichen]);
    // 7. Spalte des Zeichens

```



```

GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z1sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 2. Zeile und 1. Spalte des Zeichens
GLCD_SETCURSOR_ZEILE(tmp_zeile + 1);
GLCD_SETCURSOR_SPALTE(tmp_spalte);

// 1. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp1[zeichen]);
// 2. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp2[zeichen]);
// 3. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp3[zeichen]);
// 4. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp4[zeichen]);
// 5. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp5[zeichen]);
// 6. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp6[zeichen]);
// 7. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz8x14_z2sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 1. Zeile und Spalte vom Ende des Zeichens
GLCD_SETCURSOR_ZEILE(tmp_zeile);
}

```

## Datei GLCD\_FONT10X22.C

```

/*****
/* Unterprogramme zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) in C (CC5X) */
/* (Font 10x22) */
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 5. Maerz 2006
/* Funktionsfaehig seit: 2. April 2006
/* Letzte Bearbeitung: 8. September 2006
*****/

/***** Projekt-Header einbinden *****/
#include "PROJEKT.H"

/***** Unterprogramme zur Grafik-LCD-Ansteuerung *****/

/*****
/* GLCD_ZEICHEN_10x22
/*
/* Aufgabe:
/* Ein Zeichen (Charakter im Format 10x22) am Grafik-LC-Display ausgeben.
/*
/* Uebergabeparameter:
/* zeichen
/*
/* Rueckgabeparameter:
/* keiner
/*
/* Vorgehensweise:
/* + Die externen Hilfsregister (hlp_zeilen_zaeher und hlp_spalten_zaeher) sichern,
/* da sie hier benoetigt, aber gleichzeitig hier auch veraendert werden.
/* + Nacheinander die Spalten fuer die erste Zeile des auszugebenden Zeichens an das
/* Grafik-LC-Display mit dem Unterprogramm GLCD_DATA_ZEICHEN uebergeben. Die Daten
*/

```

```

/* fuer die Spalten sind in Tabellen abgelegt, wobei fuer jede Spalte des Zeichens */
/* eine eigene Tabelle vorhanden ist. Diese Tabellen muessen an anderer Stelle */
/* hinzugefuegt werden! Die letzten beiden Spalten muessen eine 0 enthalten, also */
/* eine "Leerspalte". */
/* + Den Cursor an die naechste Zeile (tmp_zeile + 1) und an die erste Spalte des */
/* Zeichens (tmp_spalte) setzen. */
/* + In gleicher Weise wie vorher die Spalten fuer die zweite Zeile des auszugebenden */
/* Zeichens an das Grafik-LC-Display mit dem Unterprogramm GLCD_DATA_ZEICHEN */
/* uerbergeben. */
/* + Den Cursor an die naechste Zeile (tmp_zeile + 2) und an die erste Spalte des */
/* Zeichens (tmp_spalte) setzen. */
/* + In gleicher Weise wie vorher die Spalten fuer die dritte Zeile des auszugebenden */
/* Zeichens an das Grafik-LC-Display mit dem Unterprogramm GLCD_DATA_ZEICHEN */
/* uerbergeben. */
/* + Den Cursor an die erste Zeile des Zeichens (tmp_zeile) setzen. */
/*****
void GLCD_ZEICHEN_10x22(unsigned char zeichen)
{
    char tmp_zeile = hlp_zeilen_zaeher;
    char tmp_spalte = hlp_spalten_zaeher;

    // 1. Zeile
    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp6[zeichen]);
    // 7. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp7[zeichen]);
    // 8. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp8[zeichen]);
    // 9. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp9[zeichen]);
    // 10. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z1sp10[zeichen]);
    // 11. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);
    // 12. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(0);

    // 2. Zeile und 1. Spalte des Zeichen
    GLCD_SETCURSOR_ZEILE(tmp_zeile + 1);
    GLCD_SETCURSOR_SPALTE(tmp_spalte);

    // 1. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp1[zeichen]);
    // 2. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp2[zeichen]);
    // 3. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp3[zeichen]);
    // 4. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp4[zeichen]);
    // 5. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp5[zeichen]);
    // 6. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp6[zeichen]);
    // 7. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp7[zeichen]);
    // 8. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp8[zeichen]);
    // 9. Spalte des Zeichens
    GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp9[zeichen]);

```

```

// 10. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z2sp10[zeichen]);
// 11. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);
// 12. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 3. Zeile und 1.Spalte des Zeichen
GLCD_SETCURSOR_ZEILE(tmp_zeile + 2);
GLCD_SETCURSOR_SPALTE(tmp_spalte);

// 1. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp1[zeichen]);
// 2. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp2[zeichen]);
// 3. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp3[zeichen]);
// 4. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp4[zeichen]);
// 5. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp5[zeichen]);
// 6. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp6[zeichen]);
// 7. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp7[zeichen]);
// 8. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp8[zeichen]);
// 9. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp9[zeichen]);
// 10. Spalte des Zeichens
GLCD_DATA_ZEICHEN(TabZeichensatz10x22_z3sp10[zeichen]);
// 11. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);
// 12. Spalte des Zeichens
GLCD_DATA_ZEICHEN(0);

// 1. Zeile und Spalte vom Ende des Zeichen
GLCD_SETCURSOR_ZEILE(tmp_zeile);
}

```

## Datei PROJEKT.H

```

/*****
/* Header zum Demoprojekt 2 zur Ansteuerung des Grafik-LCD (mit SED1520-Controller) */
/* in C (CC5X) */
/*
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 5. Maerz 2006 */
/* Funktionsfaehig seit: 2. April 2006 */
/* Letzte Bearbeitung: 8. April 2006 */
*****/

#ifndef __PROJEKT
#define __PROJEKT

/***** Pragma-Anweisungen *****/
#pragma chip PIC16F877 // PICmicro Device
#pragma library 1 //.. library functions that are deleted if unused

/***** Externe Strukturen *****/
/* keine externen Strukturen verwendet */

/***** Externe Register *****/
extern bank0 unsigned char hlp_zeilen_zaeahler;

```

```

extern bank0 unsigned char hlp_spalten_zaehler;

/***** Bits in den externen Registern *****/
/* keine externen Register verwendet */

/***** Konstanten *****/
/* keine Konstanten */

/***** Zeichensatz einbinden *****/
#include "zeichensatz5x7_b.inc" // Zeichensatz (5x7) einbinden
#include "zeichensatz8x14_a.inc" // Zeichensatz (8x14) einbinden
#include "zeichensatz10x22_b.inc" // Zeichensatz (10x22) einbinden

/***** Portbelegung *****/
/* Port B */
#pragma char GLCD_CTRL @ PORTB
#pragma char GLCD_CTRL_TRIS @ TRISB

#pragma bit GLCD_WR @ PORTB.1
#pragma bit GLCD_RD @ PORTB.2
#pragma bit GLCD_CS1 @ PORTB.3
#pragma bit GLCD_CS2 @ PORTB.4
#pragma bit GLCD_A0 @ PORTB.5

/* Port D */
#pragma char GLCD_DATA @ PORTD
#pragma char GLCD_DATA_TRIS @ TRISD

/***** Funktionsprototypen *****/
/* Unterprogramme fuer Grafik-LC-Display */
extern page0 void GLCD_INIT(void);

extern page0 void GLCD_COMMAND(bank0 unsigned char command);
extern page0 void GLCD_DATA1(bank0 unsigned char data);
extern page0 void GLCD_DATA2(bank0 unsigned char data);
extern page0 void GLCD_DATA_ZEICHEN(bank0 unsigned char data);

extern page0 void GLCD_CLEAR(void);

extern page0 void GLCD_SETCURSOR_ZEILE(bank0 unsigned char zeile);
extern page0 void GLCD_SETCURSOR_SPALTE(bank0 unsigned char spalte);

extern page0 void GLCD_ZEICHEN_5x7(bank0 unsigned char zeichen);
extern page0 void GLCD_ZEICHEN_8x14(bank0 unsigned char zeichen);
extern page0 void GLCD_ZEICHEN_10x22(bank0 unsigned char zeichen);

#endif

```

### Datei glcd2.bat

```

C:\Programme\bknd\CC5X\CC5X.EXE GLCD_BASIS_ROUTINEN.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT5X7.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT8X14.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT10X22.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE demo2.c -IC:\Programme\bknd\CC5X\ -u -r -a -r2

```

## 6.4.2 Anmerkungen zur C-Software

Die Software besteht auch hier im Wesentlichen aus einem sehr kurzen Hauptprogramm (main in der Datei demo2.c), die im Abschnitt 5 beschriebenen Unterprogramme (in den Dateien GLCD\_BASIS\_ROUTINE.C, GLCD\_FONT5X7.C, GLCD\_FONT8X14.C und

GLCD\_FONT10X22.C) einem weiteren Unterprogramm namens **AUSGABE** (in der Datei **demo2.c**) und je einem Unterprogramm zur Umwandlung in BCD-Darstellung der 8-Bit-Zahlenwerte für jede Schriftgröße (auch in der Datei **demo2.c**).

Das Hauptprogramm besteht nach der Initialisierung des Controllers (Unterprogramm **INIT**) und des Grafik-LC-Displays (Unterprogramm **GLCD\_INIT**) nur mehr aus dem Unterprogramm zur Ausgabe der Daten gemäß Abbildung 6.6 am Grafik-LC-Display (Unterprogramm **AUSGABE**).

Das Unterprogramm **INIT** dient zur Initialisierung des Controllers. In diesem Beispiel werden nur die Ports B und D als Ausgang konfiguriert und die Datenregister für Uhrzeit und Datum mit Demonstrationsinhalten geladen.

Das Unterprogramm **GLCD\_INIT** initialisiert das LC-Display (siehe Abschnitt 5.3.1).

Nach der Initialisierung des Grafik-LC-Displays (Unterprogramm **GLCD\_INIT**) ist das Grafik-LC-Display bereit Zeichen oder Werte anzuzeigen. Die Ausgabe der anzuzeigenden Zeichen übernimmt hier das Unterprogramm **AUSGABE**. Zuerst muss das Display mit dem Unterprogramm **GLCD\_CLEAR** gelöscht werden. Anschließend den Cursor mit den beiden Unterprogrammen **GLCD\_SETCURSOR\_ZEILE** und **GLCD\_SETCURSOR\_SPALTE** an die gewünschte Position am Grafik-LC-Display (Zeile und Spalte) setzen. Nun können beliebige Daten (z.B. mit dem Unterprogramm **AUSGABEBCD\_5x7**) oder Zeichen und Symbole (z.B. mit dem Unterprogramm **GLCD\_ZEICHEN\_5x7**) an das Grafik-LC-Display übergeben werden.

# Anhang A

## Layout (Demonstrationsbeispiele)



# Anhang B

## Stückliste

Nr.	Bezeichnung	St.	Lieferant	Bestell-Nr.	E-Preis	Bemerkungen
R2	Widerstand 270 Ohm	1	Conrad	418188	0,14	
R5	Widerstand 1k	1	Conrad	418250	0,14	
R1	Widerstand 10k	1	Conrad	418374	0,14	
R3	Widerstand 36k	1	Conrad	420921	0,14	
R4	Trimmer 10 k	1	Conrad	430862	0,40	
C3, C4	Keramikkondensator 22pF	2	Conrad	457167	0,14	
C5, C6	Keramikkondensator 10nF	2	Conrad	453323	0,23	
C2, C8, C9	Keramikkondensator 100nF	3	Conrad	453358	0,25	
C10	Tantal 1 $\mu$ F/35V	1	Conrad	481670	0,62	
C1, C7	Mini-Elko 10 $\mu$ F/25V	2	Conrad	460524	0,14	
D1, D2	Diode 1N5711	2	Conrad	155499	0,30	
D3	Diode 1N4001	1	Conrad	162213	0,08	
D4	Low-current-LED 3mm grün	1	Conrad	145971	0,21	
IC1	PIC16F877	1	Farnell	413-7358	10,50	
IC2	Timer 7555	1	Conrad	175200	0,78	
IC3	Spannungsregler 7805	1	Conrad	179205	0,54	
X1	Quarz 4 MHz	1	Conrad	182087	1,15	
LCD1	Grafik-LC-Display 122x32 Pixel	1	Conrad	187399	28,99	
	IC-Präzisions-Sockel 8polig	1	Conrad	189600	0,35	für IC2
	IC-Präzisions-Sockel 40polig	1	Conrad	189677	1,15	für IC1
S1	Taster	1	Conrad	700460	0,43	
JP1	Stiftleiste 2polig	1	Conrad	732478	0,79	Preis für 36polig
	Buchsenleiste 16polig	1	Conrad	740438	1,15	Preis für 32polig
	Kontaktstreifen	1	Conrad	739073	3,57	Preis für 32polig
	Schraubklemme 2polig	1	Conrad	729949	0,42	
	Platine-Basismaterial	1	Conrad	529249	2,71	Preis für Euroformat
	Distanz M3x10	4				optional
	Mutter M3	4				optional
	Distanz M2,5x8	4				optional
	Zylinderkopfschraube M2,5x6	4				optional
	Zahnscheibe M3	4				optional

# Anhang C

## Zeichensätze

Zeichensatz 1a (5x7):

MSB \ LSB	x000	x001	x010	x011	x100	x101	x110	x111
0000	0A		0aP	Y	P			
0001	1a	!	1A	Q	a	a		
0010	2a	"	2B	R	b	r		
0011	3a	#	3C	S	c	s		
0100	4B	\$	4D	T	d	t		
0101	5	%	5E	U	e	u		
0110	6	&	6F	V	f	v		
0111	7	'	7G	W	g	w		
1000	8E	(	8H	X	h	x		
1001	9#	)	9I	Y	i	y		
1010	A€	*	A	Z	j	z		
1011	B"	+	B	[	k	[		
1100	Cv	,	C	\	l	\		
1101	D±	-	D	]n	?			
1110	E↑	.	E	^	n	÷		
1111	F↓	/	F	o	€			



## Includedatei in C (für CC5X) zeichensatz5x7\_a.inc

```

/*****
/* zeichensatz5x7_a.inc
/* ASCII-Zeichensatz fuer 5x7 grosse Zeichen
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 5. Maerz 2006
/* Funktionsfaehig seit: 5. Maerz 2006
/* Letzte Bearbeitung: 12. Juli 2006
*****/

const char TabZeichensatz5x7_sp1[128] =
{
    62,0,66,33,24,39,60,1,54,6,126,127,62,127,127,127,
    121,32,56,60,126,0,0,0,72,21,20,0,124,72,4,16,
    0,0,0,20,36,35,54,0,0,0,20,8,0,8,0,32,
    62,0,66,33,24,39,60,1,54,6,0,0,8,20,65,2,
    50,126,127,62,127,127,127,62,127,0,32,127,127,127,62,
    127,62,127,38,1,63,7,127,99,7,97,0,2,0,4,64,
    0,32,127,56,56,56,8,12,127,0,32,0,0,120,124,56,
    124,8,124,72,2,60,28,60,68,12,68,8,0,0,8,8
};

const char TabZeichensatz5x7_sp2[128] =
{
    81,66,97,65,20,69,74,1,73,73,17,73,65,65,73,9,
    36,85,69,65,1,0,0,0,126,22,62,7,16,72,2,32,
    0,0,7,127,42,19,73,5,28,65,8,8,80,8,96,16,
    81,66,97,65,20,69,74,1,73,73,54,86,20,20,34,1,
    73,17,73,65,65,73,9,65,8,65,64,8,64,2,4,65,
    9,65,9,73,1,64,24,32,20,8,81,127,4,65,2,64,
    1,84,72,68,68,84,126,82,8,72,64,127,65,4,8,68,
    20,20,8,84,63,64,32,64,40,80,100,54,0,65,8,28
};

const char TabZeichensatz5x7_sp3[128] =
{
    73,127,81,69,18,69,73,121,73,73,17,73,65,65,73,9,
    34,84,68,64,37,0,0,0,73,124,85,5,32,126,127,127,
    0,95,0,20,127,8,85,3,34,34,62,62,48,8,96,8,
    73,127,81,69,18,69,73,121,73,73,54,54,34,20,20,81,
    121,17,73,65,65,73,9,73,8,127,65,20,64,12,8,65,
    9,81,25,73,127,64,96,24,8,112,73,65,8,65,1,64,
    2,84,68,68,68,84,9,82,4,122,68,16,127,24,4,68,
    20,20,4,84,66,64,64,48,16,80,84,65,127,65,42,42
};

const char TabZeichensatz5x7_sp4[128] =
{
    69,64,73,75,127,69,73,5,73,41,17,73,65,34,73,9,
    36,85,69,65,37,0,0,0,65,22,85,7,32,72,2,32,
    0,0,7,127,42,100,34,0,65,28,8,8,0,8,0,4,
    69,64,73,75,127,69,73,5,73,41,0,0,65,20,8,9,
    65,17,73,65,34,73,9,73,8,65,63,34,64,2,16,65,
    9,33,41,73,1,64,24,32,20,8,69,65,16,127,2,64,
    4,84,68,68,72,84,1,82,4,64,61,40,64,4,4,68,
    20,20,4,84,64,32,32,64,40,80,76,65,0,54,28,8
};

const char TabZeichensatz5x7_sp5[128] =
{
    62,0,70,49,16,57,48,3,54,30,126,54,34,28,65,1,
    121,120,56,60,26,0,0,0,66,21,85,0,28,72,4,16,
    0,0,0,20,18,98,80,0,0,0,20,8,0,8,0,2,
    62,0,70,49,16,57,48,3,54,30,0,0,0,20,0,6,
    62,126,54,34,28,65,1,58,127,0,1,65,64,127,127,62,
    6,94,70,50,1,63,7,127,99,7,67,0,32,0,4,64,
    0,120,56,40,127,24,2,62,120,0,0,68,0,120,120,56,
    8,124,8,32,32,124,28,60,68,60,68,0,0,8,8,8
};

```

## Zeichensatz 1b (5x7):

MSB \ LSB	x000	x001	x010	x011	x100	x101	x110	x111
0000	0A		0B	P	P			
0001	1A	!	1A	Q	Q			
0010	2B	"	2B	R	R			
0011	3D	#	3D	S	S			
0100	4B	\$	4D	T	T			
0101	5	%	5E	U	U			
0110	6	&	6F	V	V			
0111	7	'	7G	W	W			
1000	8E	(	8H	X	X			
1001	9#	)	9I	Y	Y			
1010	A€	*	JZ	z	z			
1011	B"	+	K	[	k	[		
1100	Cv	,	L	\	L			
1101	D±	-	M	]m	]m	}		
1110	E†	.	>	N	^	N	÷	
1111	F↓	/	?D	o	o	€		

## Includedatei in C (für CC5X) zeichensatz5x7\_b.inc

```

/*****
/* zeichensatz5x7_b.inc
/* ASCII-Zeichensatz fuer 5x7 grosse Zeichen
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 5. Maerz 2006
/* Funktionsfaehig seit: 5. Maerz 2006
/* Letzte Bearbeitung: 12. Juli 2006
*****/

const char TabZeichensatz5x7_sp1[128] =
{
    62,0,66,33,24,39,60,1,54,6,126,127,62,127,127,127,
    121,120,56,60,126,0,0,0,72,21,20,0,124,72,4,16,
    0,0,0,20,36,35,54,0,0,0,20,8,0,8,0,32,
    62,0,66,33,24,39,60,1,54,6,0,0,8,20,65,2,
    50,126,127,62,127,127,127,62,127,0,32,127,127,127,62,
    127,62,127,38,1,63,7,127,99,7,97,0,2,0,4,64,
    0,120,124,56,124,124,124,56,124,0,32,124,124,124,124,56,
    124,56,124,72,4,60,28,60,68,4,68,8,0,0,8,8
};

const char TabZeichensatz5x7_sp2[128] =
{
    81,66,97,65,20,69,74,1,73,73,17,73,65,65,73,9,
    36,37,69,65,1,0,0,0,126,22,62,7,16,72,2,32,
    0,0,7,127,42,19,73,5,28,65,8,8,80,8,96,16,
    81,66,97,65,20,69,74,1,73,73,54,86,20,20,34,1,
    73,17,73,65,65,73,9,65,8,65,64,8,64,2,4,65,
    9,65,9,73,1,64,24,32,20,8,81,127,4,65,2,64,
    1,36,84,68,68,84,20,68,16,68,68,16,64,8,8,68,
    20,68,20,84,4,64,32,64,40,8,100,54,0,65,8,28
};

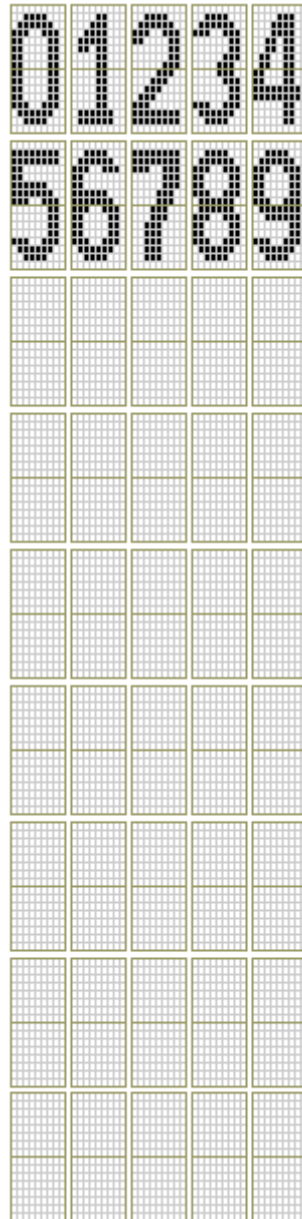
const char TabZeichensatz5x7_sp3[128] =
{
    73,127,81,69,18,69,73,121,73,73,17,73,65,65,73,9,
    34,36,68,64,37,0,0,0,73,124,85,5,32,126,127,127,
    0,95,0,20,127,8,85,3,34,34,62,62,48,8,96,8,
    73,127,81,69,18,69,73,121,73,73,54,54,34,20,20,81,
    121,17,73,65,65,73,9,73,8,127,65,20,64,12,8,65,
    9,81,25,73,127,64,96,24,8,112,73,65,8,65,1,64,
    2,36,84,68,68,84,20,68,16,124,60,40,64,16,16,68,
    20,84,52,84,124,64,64,32,16,112,84,65,127,65,42,42
};

const char TabZeichensatz5x7_sp4[128] =
{
    69,64,73,75,127,69,73,5,73,41,17,73,65,34,73,9,
    36,37,69,65,37,0,0,0,65,22,85,7,32,72,2,32,
    0,0,7,127,42,100,34,0,65,28,8,8,0,8,0,4,
    69,64,73,75,127,69,73,5,73,41,0,0,65,20,8,9,
    65,17,73,65,34,73,9,73,8,65,63,34,64,2,16,65,
    9,33,41,73,1,64,24,32,20,8,69,65,16,127,2,64,
    4,36,84,68,68,84,20,84,16,68,4,68,64,8,32,68,
    20,36,84,84,4,64,32,64,40,8,76,65,0,54,28,8
};

const char TabZeichensatz5x7_sp5[128] =
{
    62,0,70,49,16,57,48,3,54,30,126,54,34,28,65,1,
    121,120,56,60,26,0,0,0,66,21,85,0,28,72,4,16,
    0,0,0,20,18,98,80,0,0,0,20,8,0,8,0,2,
    62,0,70,49,16,57,48,3,54,30,0,0,0,20,0,6,
    62,126,54,34,28,65,1,58,127,0,1,65,64,127,127,62,
    6,94,70,50,1,63,7,127,99,7,67,0,32,0,4,64,
    0,120,40,40,56,68,4,48,124,0,0,0,64,124,124,56,
    8,88,8,32,4,60,28,60,68,4,68,0,0,8,8,8
};

```

Zeichensatz 3a (8x14):



## Includedatei in C (für CC5X) zeichensatz8x14.a.inc

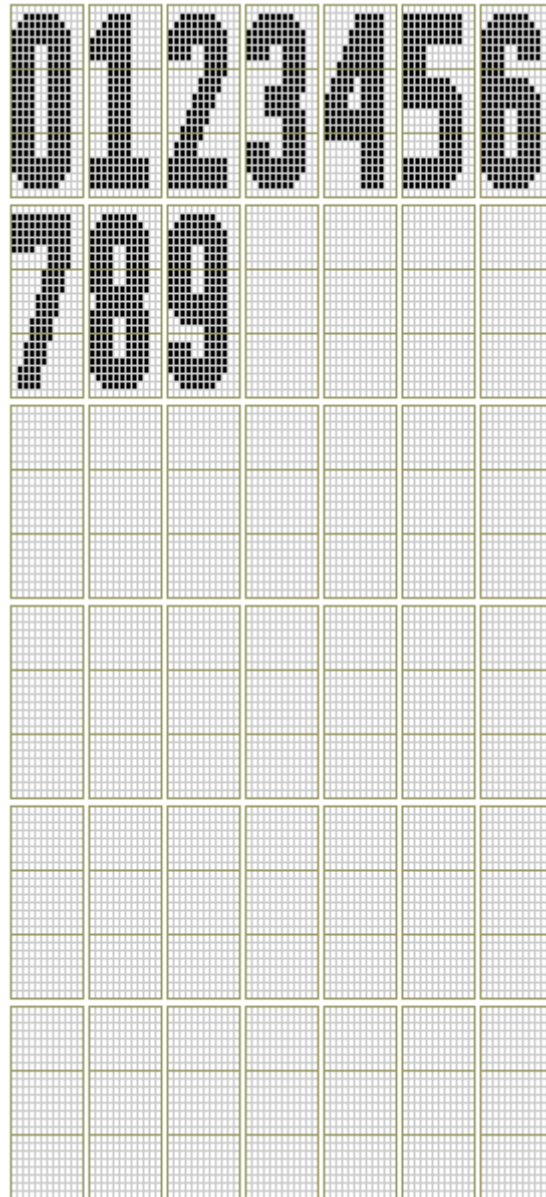
```

/*****
/* zeichensatz8x14.a.inc */
/* Zeichensatz fuer 8x14 grosse Zeichen (Ziffern 0 bis 9) */
/* */
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 8. April 2006 */
/* Funktionsfaehig seit: 8. April 2006 */
/* Letzte Bearbeitung: 12. Juli 2006 */
*****/

const char TabZeichensatz8x14_z1sp1[10] = { 248,0,24,8,0,254,248,14,56,248 };
const char TabZeichensatz8x14_z1sp2[10] = { 252,48,28,12,192,254,252,14,124,252 };
const char TabZeichensatz8x14_z1sp3[10] = { 14,56,14,14,240,230,206,14,238,142 };
const char TabZeichensatz8x14_z1sp4[10] = { 6,28,6,6,60,102,198,6,198,6 };
const char TabZeichensatz8x14_z1sp5[10] = { 6,254,6,134,14,102,198,198,198,6 };
const char TabZeichensatz8x14_z1sp6[10] = { 14,254,142,206,254,230,206,254,238,142 };
const char TabZeichensatz8x14_z1sp7[10] = { 252,0,252,252,254,198,140,254,124,252 };
const char TabZeichensatz8x14_z1sp8[10] = { 248,0,248,120,0,134,8,62,56,248 };
const char TabZeichensatz8x14_z2sp1[10] = { 31,0,112,16,15,16,31,0,30,16 };
const char TabZeichensatz8x14_z2sp2[10] = { 63,96,120,48,15,48,63,0,63,49 };
const char TabZeichensatz8x14_z2sp3[10] = { 112,96,124,112,13,112,113,124,115,115 };
const char TabZeichensatz8x14_z2sp4[10] = { 96,96,110,96,12,96,96,127,97,99 };
const char TabZeichensatz8x14_z2sp5[10] = { 96,127,103,97,12,96,96,127,97,99 };
const char TabZeichensatz8x14_z2sp6[10] = { 112,127,99,115,127,112,113,3,115,115 };
const char TabZeichensatz8x14_z2sp7[10] = { 63,96,97,63,127,63,63,0,63,63 };
const char TabZeichensatz8x14_z2sp8[10] = { 31,0,96,30,12,31,31,0,30,31 };

```

Zeichensatz 4a (10x22):



## Includedatei in C (für CC5X) zeichensatz10x22\_a.inc

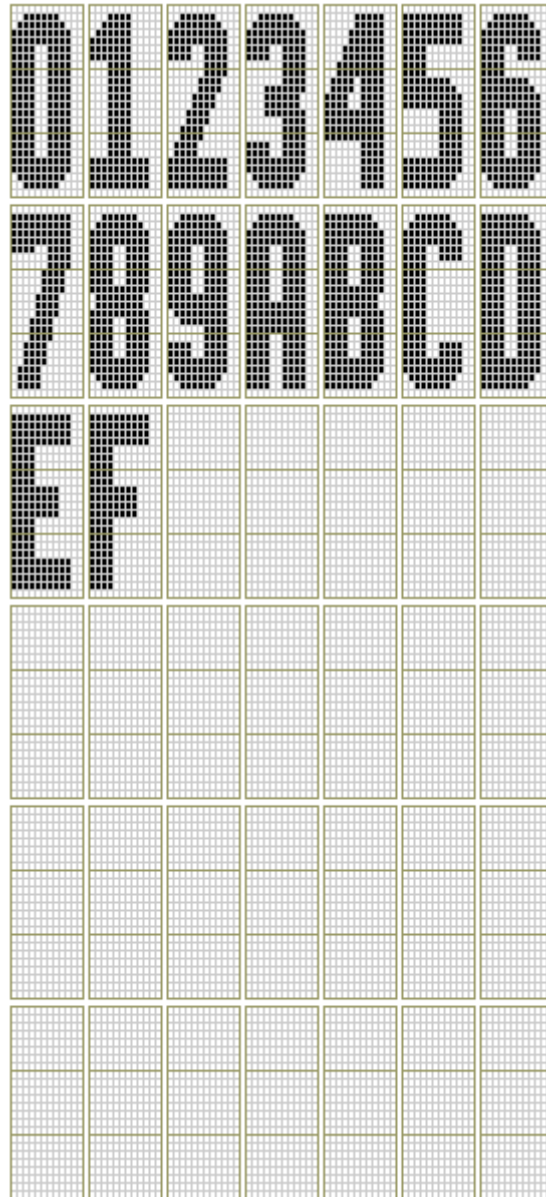
```

/*****
/* zeichensatz10x22_a.inc */
/* Zeichensatz fuer 10x22 grosse Zeichen (Ziffern 0 bis 9) */
/* */
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 13. April 2006 */
/* Funktionsfaehig seit: 13. April 2006 */
/* Letzte Bearbeitung: 12. Juli 2006 */
*****/

const char TabZeichensatz10x22_z1sp1[10] = { 248,112,120,120,0,254,248,62,248,248 };
const char TabZeichensatz10x22_z1sp2[10] = { 252,120,124,124,0,254,252,62,252,252 };
const char TabZeichensatz10x22_z1sp3[10] = { 254,124,126,126,128,254,254,62,254,254 };
const char TabZeichensatz10x22_z1sp4[10] = { 254,254,126,126,224,254,254,62,254,254 };
const char TabZeichensatz10x22_z1sp5[10] = { 30,254,30,30,248,30,30,30,30,30 };
const char TabZeichensatz10x22_z1sp6[10] = { 30,254,30,30,254,30,30,30,30,30 };
const char TabZeichensatz10x22_z1sp7[10] = { 254,254,254,254,254,30,126,254,254,254 };
const char TabZeichensatz10x22_z1sp8[10] = { 254,0,254,254,254,30,126,254,254,254 };
const char TabZeichensatz10x22_z1sp9[10] = { 252,0,252,252,254,30,124,254,252,252 };
const char TabZeichensatz10x22_z1sp10[10] = { 248,0,248,248,254,30,120,254,248,248 };
const char TabZeichensatz10x22_z2sp1[10] = { 255,0,0,0,248,31,255,0,231,31 };
const char TabZeichensatz10x22_z2sp2[10] = { 255,0,0,0,254,31,255,0,255,63 };
const char TabZeichensatz10x22_z2sp3[10] = { 255,0,128,0,255,31,255,0,255,127 };
const char TabZeichensatz10x22_z2sp4[10] = { 255,255,224,60,255,31,255,192,255,127 };
const char TabZeichensatz10x22_z2sp5[10] = { 0,255,248,60,231,30,30,248,60,120 };
const char TabZeichensatz10x22_z2sp6[10] = { 0,255,254,60,225,30,30,255,60,120 };
const char TabZeichensatz10x22_z2sp7[10] = { 255,255,127,255,255,254,254,255,255,255 };
const char TabZeichensatz10x22_z2sp8[10] = { 255,0,31,255,255,254,254,63,255,255 };
const char TabZeichensatz10x22_z2sp9[10] = { 255,0,7,255,255,252,252,7,255,255 };
const char TabZeichensatz10x22_z2sp10[10] = { 255,0,1,231,255,248,248,0,231,255 };
const char TabZeichensatz10x22_z3sp1[10] = { 31,120,120,30,1,120,31,0,31,30 };
const char TabZeichensatz10x22_z3sp2[10] = { 63,120,126,62,1,120,63,112,63,62 };
const char TabZeichensatz10x22_z3sp3[10] = { 127,120,127,126,1,120,127,126,127,126 };
const char TabZeichensatz10x22_z3sp4[10] = { 127,127,127,126,1,120,127,127,127,126 };
const char TabZeichensatz10x22_z3sp5[10] = { 120,127,127,120,1,120,120,127,120,120 };
const char TabZeichensatz10x22_z3sp6[10] = { 120,127,121,120,1,120,120,15,120,120 };
const char TabZeichensatz10x22_z3sp7[10] = { 127,127,120,127,127,127,127,1,127,127 };
const char TabZeichensatz10x22_z3sp8[10] = { 127,120,120,127,127,127,127,0,127,127 };
const char TabZeichensatz10x22_z3sp9[10] = { 63,120,120,63,127,63,63,0,63,63 };
const char TabZeichensatz10x22_z3sp10[10] = { 31,120,120,31,127,31,31,0,31,31 };

```

Zeichensatz 4a (hex) (10x22):





## Includedatei in C (für CC5X) zeichensatz10x22\_a\_hex.inc

```

/*****
/* zeichensatz10x22_a_hex.inc
/* Zeichensatz fuer 10x22 grosse Zeichen (Ziffern 0 bis 9 und A bis F)
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 13. April 2006
/* Funktionsfaehig seit: 13. April 2006
/* Letzte Bearbeitung: 16. September 2006
*****/

// 1. Zeile
const char TabZeichensatz10x22_z1sp1[16] =
{
    248,112,120,120,0,254,248,62,248,248,248,254,248,254,254,254
};
const char TabZeichensatz10x22_z1sp2[16] =
{
    252,120,124,124,0,254,252,62,252,252,252,254,252,254,254,254
};
const char TabZeichensatz10x22_z1sp3[16] =
{
    254,124,126,126,128,254,254,62,254,254,254,254,254,254,254,254
};
const char TabZeichensatz10x22_z1sp4[16] =
{
    254,254,126,126,224,254,254,62,254,254,254,254,254,254,254,254
};
const char TabZeichensatz10x22_z1sp5[16] =
{
    30,254,30,30,248,30,30,30,30,30,30,30,30,30,30,30
};
const char TabZeichensatz10x22_z1sp6[16] =
{
    30,254,30,30,254,30,30,30,30,30,30,30,30,30,30,30
};
const char TabZeichensatz10x22_z1sp7[16] =
{
    254,254,254,254,254,30,126,254,254,254,254,126,254,30,30
};
const char TabZeichensatz10x22_z1sp8[16] =
{
    254,0,254,254,254,30,126,254,254,254,254,126,254,30,30
};
const char TabZeichensatz10x22_z1sp9[16] =
{
    252,0,252,252,254,30,124,254,252,252,252,252,124,252,30,30
};
const char TabZeichensatz10x22_z1sp10[16] =
{
    248,0,248,248,254,30,120,254,248,248,248,248,120,248,30,30
};

// 2. Zeile
const char TabZeichensatz10x22_z2sp1[16] =
{
    255,0,0,0,248,31,255,0,231,31,255,255,255,255,255,255
};
const char TabZeichensatz10x22_z2sp2[16] =
{
    255,0,0,0,254,31,255,0,255,63,255,255,255,255,255,255
};
const char TabZeichensatz10x22_z2sp3[16] =
{
    255,0,128,0,255,31,255,0,255,127,255,255,255,255,255,255
};
const char TabZeichensatz10x22_z2sp4[16] =
{
    255,255,224,60,255,31,255,192,255,127,255,255,255,255,255,255
};

```

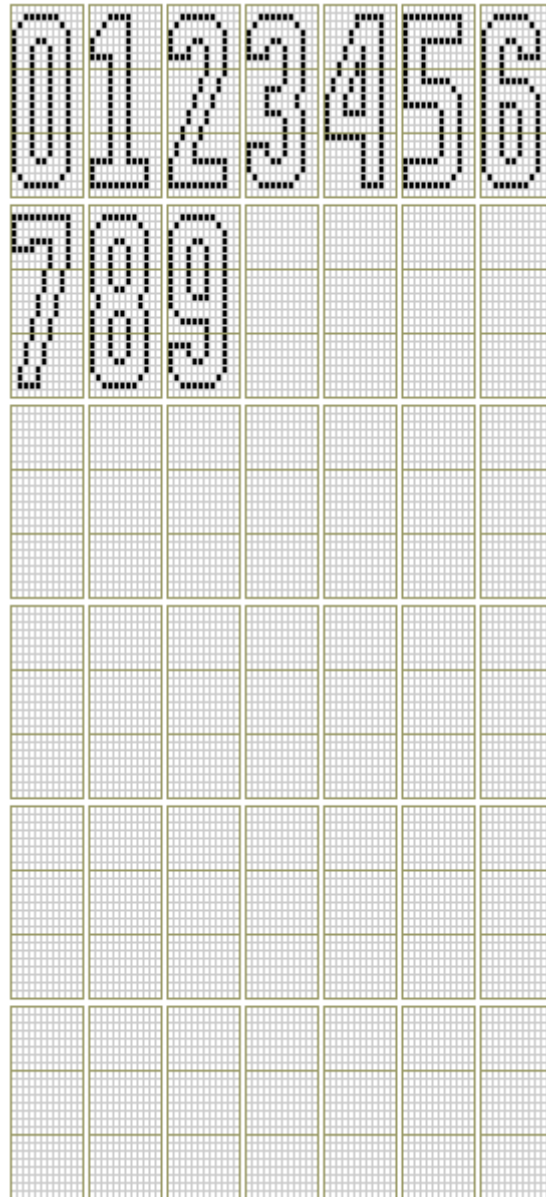
```

const char TabZeichensatz10x22_z2sp5[16] =
{
    0,255,248,60,231,30,30,248,60,120,240,60,0,0,60,60
};
const char TabZeichensatz10x22_z2sp6[16] =
{
    0,255,254,60,225,30,30,255,60,120,240,60,0,0,60,60
};
const char TabZeichensatz10x22_z2sp7[16] =
{
    255,255,127,255,255,254,254,255,255,255,255,0,255,60,60
};
const char TabZeichensatz10x22_z2sp8[16] =
{
    255,0,31,255,255,254,254,63,255,255,255,255,0,255,60,60
};
const char TabZeichensatz10x22_z2sp9[16] =
{
    255,0,7,255,255,252,252,7,255,255,255,255,0,255,0,0
};
const char TabZeichensatz10x22_z2sp10[16] =
{
    255,0,1,231,255,248,248,0,231,255,255,231,0,255,0,0
};

// 3. Zeile
const char TabZeichensatz10x22_z3sp1[16] =
{
    31,120,120,30,1,120,31,0,31,30,127,127,31,127,127,127
};
const char TabZeichensatz10x22_z3sp2[16] =
{
    63,120,126,62,1,120,63,112,63,62,127,127,63,127,127,127
};
const char TabZeichensatz10x22_z3sp3[16] =
{
    127,120,127,126,1,120,127,126,127,126,127,127,127,127,127,127
};
const char TabZeichensatz10x22_z3sp4[16] =
{
    127,127,127,126,1,120,127,127,127,126,127,127,127,127,127,127
};
const char TabZeichensatz10x22_z3sp5[16] =
{
    120,127,127,120,1,120,120,127,120,120,0,120,120,120,120,0
};
const char TabZeichensatz10x22_z3sp6[16] =
{
    120,127,121,120,1,120,120,15,120,120,0,120,120,120,120,0
};
const char TabZeichensatz10x22_z3sp7[16] =
{
    127,127,120,127,127,127,127,1,127,127,127,127,126,127,120,0
};
const char TabZeichensatz10x22_z3sp8[16] =
{
    127,120,120,127,127,127,127,0,127,127,127,127,126,127,120,0
};
const char TabZeichensatz10x22_z3sp9[16] =
{
    63,120,120,63,127,63,63,0,63,63,127,63,62,63,120,0
};
const char TabZeichensatz10x22_z3sp10[16] =
{
    31,120,120,31,127,31,31,0,31,31,127,31,30,31,120,0
};

```

### Zeichensatz 4b (10x22):



## Includedatei in C (für CC5X) zeichensatz10x22\_b.inc

```

/*****
/* zeichensatz10x22_b.inc */
/* Zeichensatz fuer 10x22 grosse Zeichen (Ziffern 0 bis 9) */
/* */
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 13. Maerz 2006 */
/* Funktionsfaehig seit: 13. Maerz 2006 */
/* Letzte Bearbeitung: 12. Juli 2006 */
*****/

const char TabZeichensatz10x22_z1sp1[10] = { 248,112,120,120,0,254,248,62,248,248 };
const char TabZeichensatz10x22_z1sp2[10] = { 4,72,68,68,0,2,4,34,4,4 };
const char TabZeichensatz10x22_z1sp3[10] = { 2,68,66,66,128,2,2,34,2,2 };
const char TabZeichensatz10x22_z1sp4[10] = { 226,194,98,98,96,242,226,50,226,226 };
const char TabZeichensatz10x22_z1sp5[10] = { 18,2,18,18,24,18,18,18,18,18 };
const char TabZeichensatz10x22_z1sp6[10] = { 18,2,18,18,134,18,18,18,18,18 };
const char TabZeichensatz10x22_z1sp7[10] = { 226,254,226,226,226,18,98,242,226,226 };
const char TabZeichensatz10x22_z1sp8[10] = { 2,0,2,2,2,18,66,2,2,2 };
const char TabZeichensatz10x22_z1sp9[10] = { 4,0,4,4,2,18,68,2,4,4 };
const char TabZeichensatz10x22_z1sp10[10] = { 248,0,248,248,254,30,120,254,248,248 };
const char TabZeichensatz10x22_z2sp1[10] = { 255,0,0,0,248,31,255,0,231,31 };
const char TabZeichensatz10x22_z2sp2[10] = { 0,0,0,0,6,16,0,0,24,32 };
const char TabZeichensatz10x22_z2sp3[10] = { 0,0,128,0,1,16,0,0,0,64 };
const char TabZeichensatz10x22_z2sp4[10] = { 255,255,96,60,56,19,243,192,195,71 };
const char TabZeichensatz10x22_z2sp5[10] = { 0,0,24,36,38,18,18,56,36,72 };
const char TabZeichensatz10x22_z2sp6[10] = { 0,0,134,36,33,18,18,7,36,72 };
const char TabZeichensatz10x22_z2sp7[10] = { 255,255,97,195,63,226,226,192,195,207 };
const char TabZeichensatz10x22_z2sp8[10] = { 0,0,24,0,0,2,2,56,0,0 };
const char TabZeichensatz10x22_z2sp9[10] = { 0,0,6,24,0,4,4,7,24,0 };
const char TabZeichensatz10x22_z2sp10[10] = { 255,0,1,231,255,248,248,0,231,255 };
const char TabZeichensatz10x22_z3sp1[10] = { 31,120,120,30,1,120,31,0,31,30 };
const char TabZeichensatz10x22_z3sp2[10] = { 32,72,70,34,1,72,32,112,32,34 };
const char TabZeichensatz10x22_z3sp3[10] = { 64,72,65,66,1,72,64,78,64,66 };
const char TabZeichensatz10x22_z3sp4[10] = { 71,79,64,70,1,72,71,65,71,70 };
const char TabZeichensatz10x22_z3sp5[10] = { 72,64,78,72,1,72,72,112,72,72 };
const char TabZeichensatz10x22_z3sp6[10] = { 72,64,73,72,1,72,72,14,72,72 };
const char TabZeichensatz10x22_z3sp7[10] = { 71,79,72,71,127,71,71,1,71,71 };
const char TabZeichensatz10x22_z3sp8[10] = { 64,72,72,64,64,64,64,0,64,64 };
const char TabZeichensatz10x22_z3sp9[10] = { 32,72,72,32,64,32,32,0,32,32 };
const char TabZeichensatz10x22_z3sp10[10] = { 31,120,120,31,127,31,31,0,31,31 };

```



## Includedatei in C (für CC5X) zeichensatz10x22\_a\_hex.inc

```

/*****
/* zeichensatz10x22_b.inc
/* Zeichensatz fuer 10x22 grosse Zeichen (Ziffern 0 bis 9 und A bis F)
/*
/* Entwickler: Buchgeher Stefan
/* Entwicklungsbeginn der Software: 13. Maerz 2006
/* Funktionsfaehig seit: 13. Maerz 2006
/* Letzte Bearbeitung: 16. September 2006
*****/

// 1. Zeile
const char TabZeichensatz10x22_z1sp1[16] =
{
    248,112,120,120,0,254,248,62,248,248,248,254,248,254,254,254
};
const char TabZeichensatz10x22_z1sp2[16] =
{
    4,72,68,68,0,2,4,34,4,4,4,2,4,2,2,2
};
const char TabZeichensatz10x22_z1sp3[16] =
{
    2,68,66,66,128,2,2,34,2,2,2,2,2,2,2,2
};
const char TabZeichensatz10x22_z1sp4[16] =
{
    226,194,98,98,96,242,226,50,226,226,226,242,226,242,242,242
};
const char TabZeichensatz10x22_z1sp5[16] =
{
    18,2,18,18,24,18,18,18,18,18,18,18,18,18,18,18
};
const char TabZeichensatz10x22_z1sp6[16] =
{
    18,2,18,18,134,18,18,18,18,18,18,18,18,18,18,18
};
const char TabZeichensatz10x22_z1sp7[16] =
{
    226,254,226,226,226,18,98,242,226,226,226,226,98,226,18,18
};
const char TabZeichensatz10x22_z1sp8[16] =
{
    2,0,2,2,2,18,66,2,2,2,2,2,66,2,18,18
};
const char TabZeichensatz10x22_z1sp9[16] =
{
    4,0,4,4,2,18,68,2,4,4,4,4,68,4,18,18
};
const char TabZeichensatz10x22_z1sp10[16] =
{
    248,0,248,248,254,30,120,254,248,248,248,248,120,248,30,30
};

// 2. Zeile
const char TabZeichensatz10x22_z2sp1[16] =
{
    255,0,0,0,248,31,255,0,231,31,255,255,255,255,255,255
};
const char TabZeichensatz10x22_z2sp2[16] =
{
    0,0,0,0,6,16,0,0,24,32,0,0,0,0,0,0
};
const char TabZeichensatz10x22_z2sp3[16] =
{
    0,0,128,0,1,16,0,0,0,64,0,0,0,0,0,0
};
const char TabZeichensatz10x22_z2sp4[16] =
{
    255,255,96,60,56,19,243,192,195,71,159,231,255,255,231,231
};

```

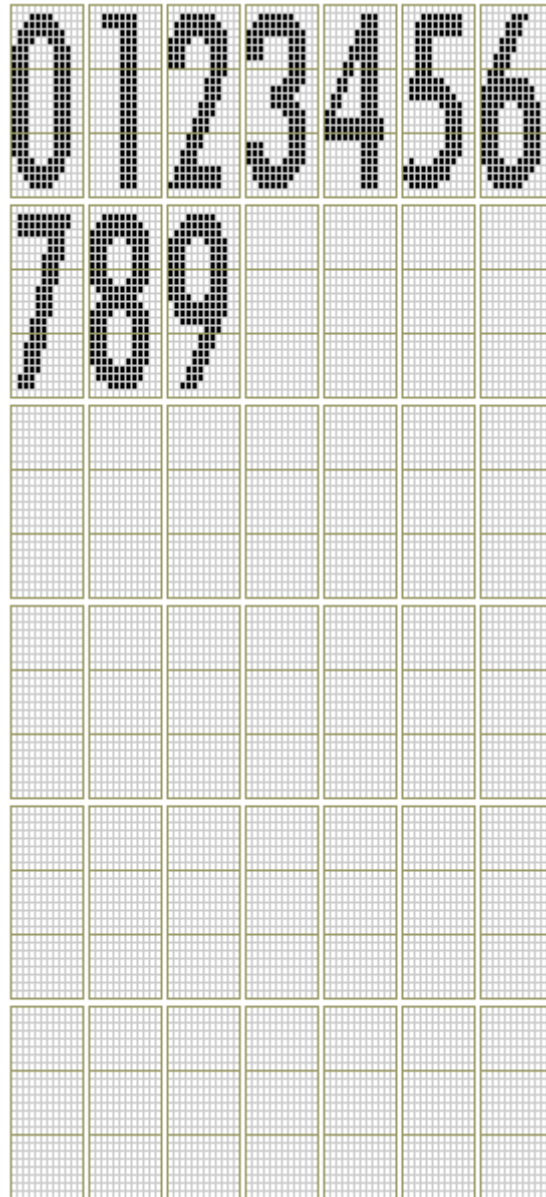
```

const char TabZeichensatz10x22_z2sp5[16] =
{
    0,0,24,36,38,18,18,56,36,72,144,36,0,0,36,36
};
const char TabZeichensatz10x22_z2sp6[16] =
{
    0,0,134,36,33,18,18,7,36,72,144,36,0,0,36,36
};
const char TabZeichensatz10x22_z2sp7[16] =
{
    255,255,97,195,63,226,226,192,195,207,159,231,0,255,36,36
};
const char TabZeichensatz10x22_z2sp8[16] =
{
    0,0,24,0,0,2,2,56,0,0,0,0,0,60,60
};
const char TabZeichensatz10x22_z2sp9[16] =
{
    0,0,6,24,0,4,4,7,24,0,0,24,0,0,0,0
};
const char TabZeichensatz10x22_z2sp10[16] =
{
    255,0,1,231,255,248,248,0,231,255,255,231,0,255,0,0
};

// 3. Zeile
const char TabZeichensatz10x22_z3sp1[16] =
{
    31,120,120,30,1,120,31,0,31,30,127,127,31,127,127,127
};
const char TabZeichensatz10x22_z3sp2[16] =
{
    32,72,70,34,1,72,32,112,32,34,64,64,32,64,64,64
};
const char TabZeichensatz10x22_z3sp3[16] =
{
    64,72,65,66,1,72,64,78,64,66,64,64,64,64,64,64
};
const char TabZeichensatz10x22_z3sp4[16] =
{
    71,79,64,70,1,72,71,65,71,70,127,79,71,79,79,127
};
const char TabZeichensatz10x22_z3sp5[16] =
{
    72,64,78,72,1,72,72,112,72,72,0,72,72,72,72,0
};
const char TabZeichensatz10x22_z3sp6[16] =
{
    72,64,73,72,1,72,72,14,72,72,0,72,72,72,72,0
};
const char TabZeichensatz10x22_z3sp7[16] =
{
    71,79,72,71,127,71,71,1,71,71,127,79,70,71,72,0
};
const char TabZeichensatz10x22_z3sp8[16] =
{
    64,72,72,64,64,64,64,0,64,64,64,64,66,64,72,0
};
const char TabZeichensatz10x22_z3sp9[16] =
{
    32,72,72,32,64,32,32,0,32,32,64,32,34,32,72,0
};
const char TabZeichensatz10x22_z3sp10[16] =
{
    31,120,120,31,127,31,31,0,31,31,127,31,30,31,120,0
};

```

Zeichensatz 4c (10x22):





## Includedatei in C (für CC5X) zeichensatz10x22.c.inc

```

/*****
/* zeichensatz10x22.c.inc */
/* Zeichensatz fuer 10x22 grosse Zeichen (Ziffern 0 bis 9) */
/*
/* Entwickler: Buchgeher Stefan */
/* Entwicklungsbeginn der Software: 8. April 2006 */
/* Funktionsfaehig seit: 8. April 2006 */
/* Letzte Bearbeitung: 12. Juli 2006 */
*****/

const char TabZeichensatz10x22_z1sp1[10] = { 224,0,240,0,0,0,0,248,240 };
const char TabZeichensatz10x22_z1sp2[10] = { 248,0,252,120,0,0,0,14,252,248 };
const char TabZeichensatz10x22_z1sp3[10] = { 252,6,254,124,0,252,192,14,254,252 };
const char TabZeichensatz10x22_z1sp4[10] = { 30,6,30,126,128,254,240,14,30,30 };
const char TabZeichensatz10x22_z1sp5[10] = { 14,254,14,14,224,254,252,14,14,14 };
const char TabZeichensatz10x22_z1sp6[10] = { 14,254,14,14,120,14,62,206,14,14 };
const char TabZeichensatz10x22_z1sp7[10] = { 30,254,30,14,254,14,14,254,30,30 };
const char TabZeichensatz10x22_z1sp8[10] = { 252,0,254,254,254,14,2,254,254,254 };
const char TabZeichensatz10x22_z1sp9[10] = { 248,0,252,252,254,14,0,126,252,252 };
const char TabZeichensatz10x22_z1sp10[10] = { 224,0,248,248,0,14,0,6,248,240 };
const char TabZeichensatz10x22_z2sp1[10] = { 255,0,0,0,224,0,252,0,227,15 };
const char TabZeichensatz10x22_z2sp2[10] = { 255,0,0,0,248,14,255,0,247,63 };
const char TabZeichensatz10x22_z2sp3[10] = { 255,0,0,0,222,15,255,0,255,127 };
const char TabZeichensatz10x22_z2sp4[10] = { 0,0,0,0,199,15,31,192,62,120 };
const char TabZeichensatz10x22_z2sp5[10] = { 0,255,192,28,193,15,14,252,28,112 };
const char TabZeichensatz10x22_z2sp6[10] = { 0,255,240,28,192,14,14,255,28,112 };
const char TabZeichensatz10x22_z2sp7[10] = { 0,255,254,62,255,30,30,63,62,248 };
const char TabZeichensatz10x22_z2sp8[10] = { 255,0,63,255,255,252,254,3,255,255 };
const char TabZeichensatz10x22_z2sp9[10] = { 255,0,7,247,255,248,252,0,247,255 };
const char TabZeichensatz10x22_z2sp10[10] = { 255,0,1,227,192,240,240,0,227,63 };
const char TabZeichensatz10x22_z3sp1[10] = { 7,0,64,14,1,48,15,0,15,0 };
const char TabZeichensatz10x22_z3sp2[10] = { 31,0,112,62,1,120,63,96,63,0 };
const char TabZeichensatz10x22_z3sp3[10] = { 63,0,124,126,1,112,127,126,63,64 };
const char TabZeichensatz10x22_z3sp4[10] = { 120,0,127,120,1,112,120,127,120,112 };
const char TabZeichensatz10x22_z3sp5[10] = { 112,127,127,112,1,112,112,31,112,120 };
const char TabZeichensatz10x22_z3sp6[10] = { 112,127,115,112,1,120,112,3,112,62 };
const char TabZeichensatz10x22_z3sp7[10] = { 120,127,112,120,127,60,120,0,120,15 };
const char TabZeichensatz10x22_z3sp8[10] = { 63,0,112,63,127,63,63,0,127,3 };
const char TabZeichensatz10x22_z3sp9[10] = { 31,0,112,31,127,15,31,0,63,0 };
const char TabZeichensatz10x22_z3sp10[10] = { 7,0,0,15,1,7,15,0,15,0 };

```

# Anhang D

## Programmieren mit ICSP

### Allgemeines:

Unter Programmieren mit ICSP versteht man das Programmieren eines (programmierbaren) Bauteils direkt in der Schaltung. Der zu programmierende Bauteil (hier der PIC) muss bei einem Software-Update oder bei der Erstprogrammierung nicht aus der Schaltung genommen werden. Je nach Bauteilfamilie sind dazu unterschiedlich viele Leitungen notwendig. Bei der PIC-Familie sind für die ICSP-Programmierung folgende 5 Leitungen notwendig:

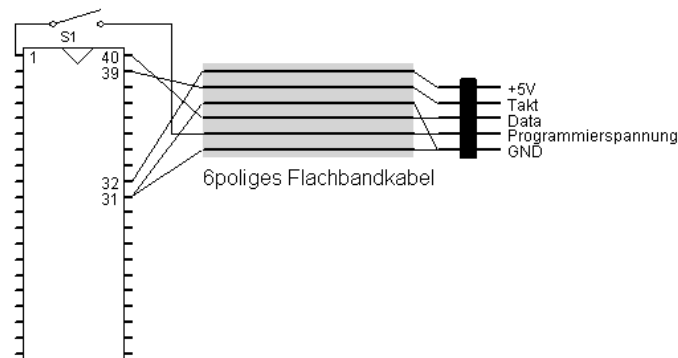
- Betriebsspannung ( $V_{dd}$ , Pins 11 und 32)
- Taktleitung ( $RB6$ , Pin 39)
- Datenleitung ( $RB7$ , Pin 40)
- Programmierspannung ( $MCLR/V_{pp}$ , Pin 1)
- Masse ( $V_{ss}$ , Pins 12 und 31)

Weiters ist bei der PIC-Familie für die ICSP-Programmierung eine entsprechende Interfaceschaltung notwendig. Siehe dazu Abschnitt 6.1 ab Seite 29.

### Downloadkabel:

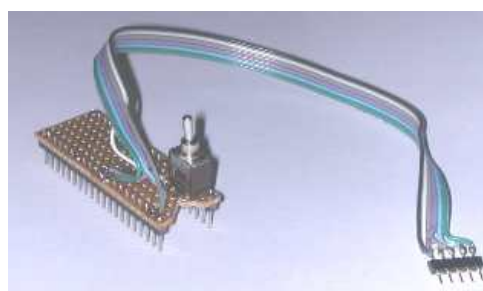
Als Downloadkabel verwende ich ein 6poliges Flachbandkabel mit einer Länge von ca. 20 cm. Warum 6polig? Die Taktleitung ist von störenden Einstreuungen sehr empfindlich. Sie befindet sich bei mir daher zwischen zwei Leitungen mit fixem Potential (GND und +5V).

Das folgende Bild zeigt die Schaltung des Downloadkabels.



Pin 1 des PIC ist beim 40poligen Gehäuse im “normalen Betrieb“ die MCLR-Leitung. Bei der Programmierung via ICSP hat diese Leitung allerdings eine andere Bedeutung. Hier liefert dieser Pin die Programmierspannung. Damit nach der Programmierung das Downloadkabel nicht von der Schaltung entfernt werden muss, kann die Programmierspannung mit dem Schalter S1 abgeschaltet werden. Somit ist ein “normaler Betrieb“ der Applikation möglich. Dies ist vor allem während der Programmentwicklung praktisch, da hier schnell die geänderte Software überprüft werden kann, ohne dass das Downloadkabel ständig angeschlossen bzw. entfernt werden muss.

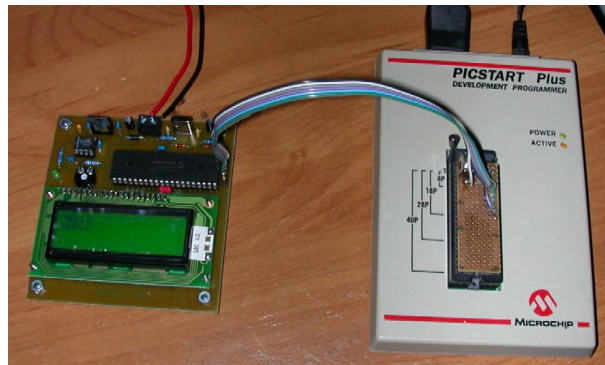
Das folgende Bild zeigt einen möglichen Aufbau des Downloadkabels. Eine etwa 19x52mm große Lochrasterplatine (7x20 Löcher) und zwei 20polige Stiftleisten bilden die Basis. Daran wird das 6polige Flachbandkabel entsprechend dem oberen Schaltplan angelötet. Am anderen Ende des Flachbandkabels befindet sich eine 5polige Stiftleiste (**Achtung:** Diese Stiftleiste muss natürlich in die entsprechende Buchsenleiste der Anwendung passen. Siehe dazu Abschnitt 6.1 ab Seite 29).



### Programmiergerät:

Zur Programmierung der PIC-Mikrocontroller verwende ich PICSTART Plus. Dieser Programmer verwendet intern auch das ICSP-Prinzip, so dass es für den Programmer egal ist, ob nun ein PIC in der Fassung sitzt oder das hier beschriebene Downloadkabel. Es ist aber zu beachten dass die Anschlüsse für die ICSP-Programmierung von PIC zu PIC unterschiedlich sind. Das hier vorgestellte Downloadkabel lässt sich zum Beispiel nicht für einen 18poligen PIC (PIC16F84 oder PIC16F62x) verwenden, auch wenn in der Entwicklungsumgebung ein PIC16F84 oder ein PIC16F62x ausgewählt wurde. Das hier beschriebene Downloadkabel eignet sich aber für alle 40poligen und 28poligen PICs.

Das folgende Bild zeigt das Downloadkabel im Einsatz.



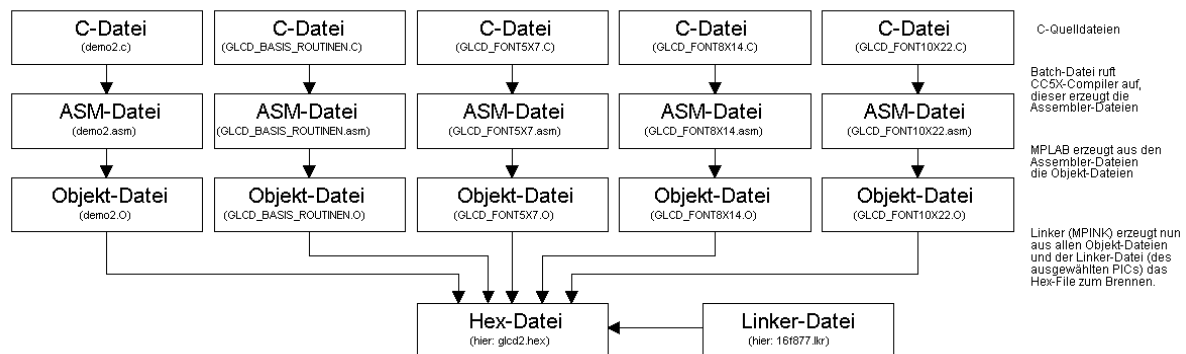
# Anhang E

## Umgehen der 1k-Grenze (beim CC5X-Compiler)

Zum umgehen der 1k-Grenze sind folgende drei Schritte notwendig:

- Schritt 1: Aufteilen des Projekts in mehrere C-Dateien
- Schritt 2: Kompilieren mit einer .BAT-Datei
- Schritt 3: Zusammenfügen der einzelnen .ASM-Dateien

Das folgende Bild zeigt diesen prinzipiellen Vorgang.



### Schritt 1: Aufteilen des Projekts in mehrere C-Dateien

Wird also eine C-Datei so groß, dass es beim Kompilieren die 1k-Grenze überschreitet, so muss es in mehrere C-Dateien aufgeteilt werden, wobei auch hier gilt: Jede C-Datei darf beim kompilieren die 1k-Grenze nicht überschreiten!

Die Aufteilung, welches Unterprogramm sich in welcher C-Datei befindet ist grundsätzlich beliebig. Hier erfolgt die Aufteilung gemäß Abschnitt 6.4.

### Schritt 2: Kompilieren mit einer .BAT-Datei

Im nächsten Schritt wird nun jede C-Datei (mit Hilfe des CC5X-Compilers) in eine ASM-Datei kompiliert. Diese Aufgabe übernimmt eine .BAT-Datei. Dazu geht man wie folgt vor:

1. Im Projektordner eine neue Textdatei erzeugen. Der Name der Datei ist egal, wichtig ist nur die Endung `.bat` (Also z.B. `glcd2.bat`). Es entsteht also eine Datei vom Typ *Stapelverarbeitungsdatei für MS-DOS*.
2. Diese Datei (`glcd2.bat`) mit der rechten Maustaste anklicken und im Pop-Up-Menü *bearbeiten* auswählen. Im Editor nun die folgenden Einträge einfügen:

```
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_BASIS_ROUTINEN.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT5X7.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT8X14.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE GLCD_FONT10X22.C -IC:\Programme\bknd\CC5X\ -u -r -a -r2
C:\Programme\bknd\CC5X\CC5X.EXE demo2.c -IC:\Programme\bknd\CC5X\ -u -r -a -r2
```

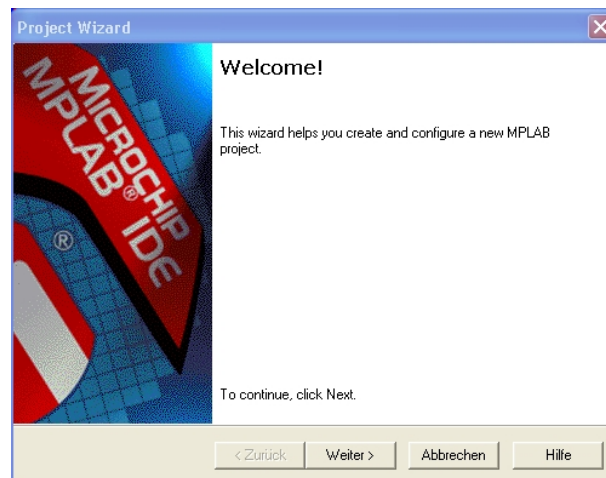
Hinweis:

- Für jede C-Datei muss eine entsprechende Zeile eingefügt werden
3. Editor schließen (Datei natürlich vorher speichern)
  4. Batch-Datei (hier `glcd2.bat`) mit einem Doppelklick ausführen. Wenn in den C-Dateien alles richtig ist (also die Syntax) werden die dazugehörigen `.ASM`-Dateien erzeugt (hier `GLCD_BASIS_ROUTINEN.asm`, `GLCD_FONT5X7.asm`, `GLCD_FONT8X14.asm`, `GLCD_FONT10X22.asm` und `demo2.ASM`). Bei einem Syntaxfehler (in der C-Datei) wird die eventuell schon vorhandene `.ASM`-Datei gelöscht, und die `.OCC`-Datei gibt Auskunft über die aufgetretenen Syntaxfehler.

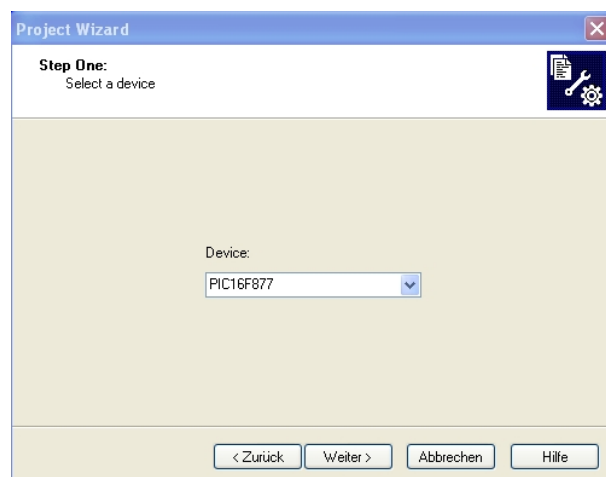
### Schritt 3: Zusammenfügen der einzelnen `.ASM`-Dateien

Die im vorhergehenden Schritt erzeugten Assembler-Dateien müssen nun zu einem gesamten Projekt zusammengefügt werden. Und schließlich soll daraus die zum Programmieren des PIC notwendige `.HEX`-Datei erzeugt werden. Dazu geht man wie folgt vor:

*MPLAB starten → Project → Project Wizard ...*

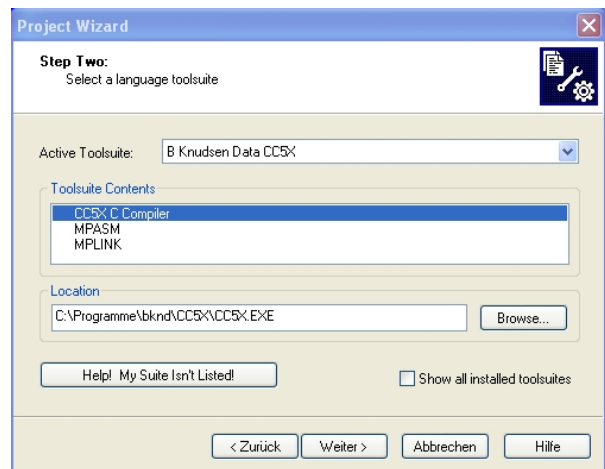


Taste *Weiter* >



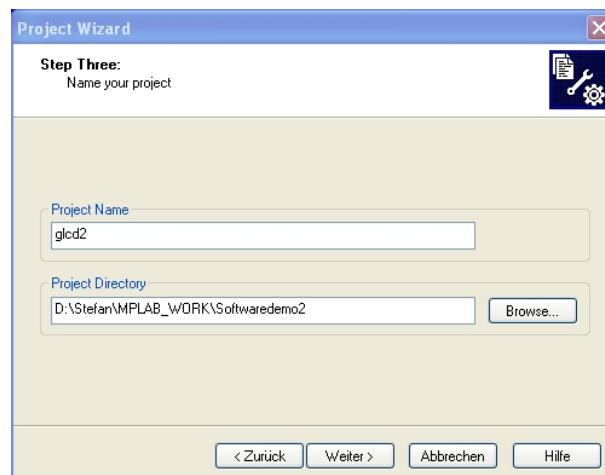
PIC auswählen (hier den PIC16F877)

Taste *Weiter* >



Active Toolsuite: *B Knudsen Data CC5X* auswählen

Taste *Weiter* >

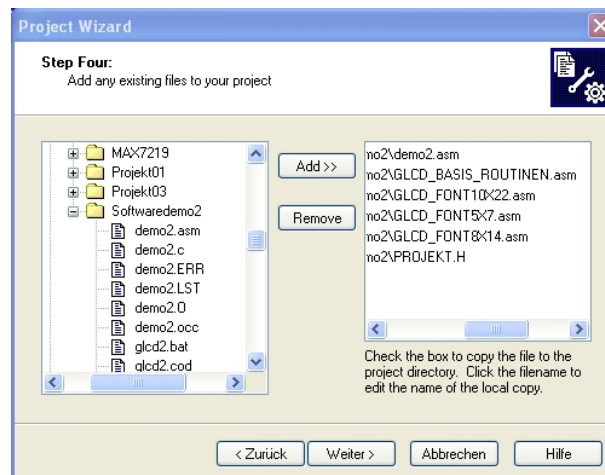


Den Projektordner auswählen, wo dieses Beispiel-Projekt abgelegt werden soll. Dieser Ordner muss schon existieren!

Projektname eingeben (hier: `glcd2`). Anmerkung: der Projektname muss **nicht** identisch mit dem Ordnername sein!

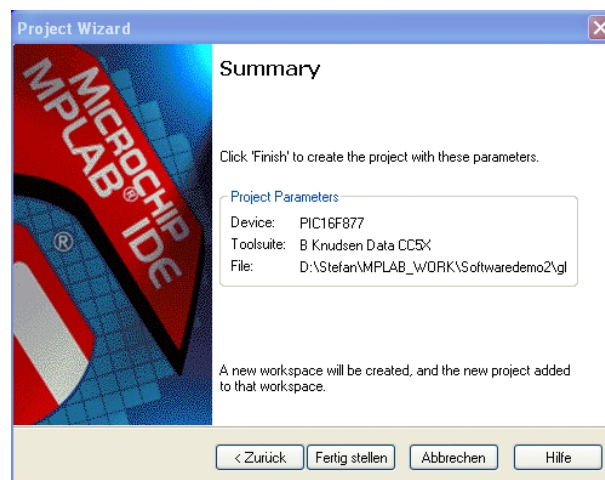
Taste *Weiter* >



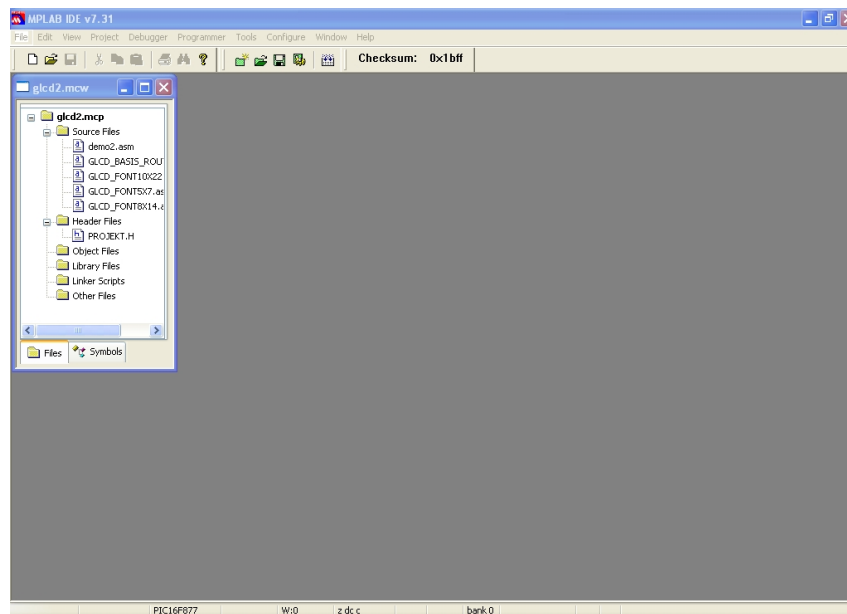


Zunächst in der rechten Spalte die Datei `demo2.asm` auswählen und die Taste *Add >>* anklicken. Die ausgewählte Datei erscheint nun in der rechten Spalte. Diesen Vorgang für alle Assembler-Dateien (hier also `GLCD_BASIS_ROUTINEN.asm`, `GLCD_FONT5X7.asm`, `GLCD_FONT8X14.asm` und `GLCD_FONT10X22.asm`) und die Header-Datei `PROJEKT.H` wiederholen.

Taste *Weiter >*

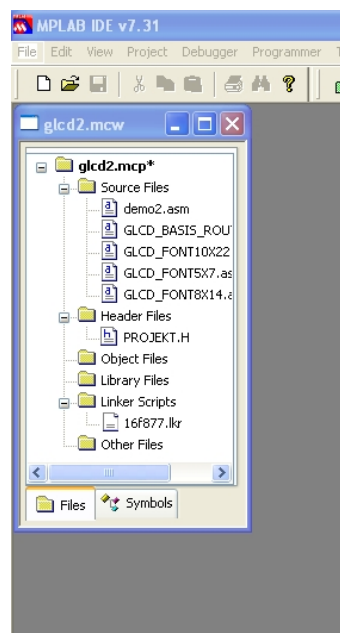


Taste *Fertig stellen*

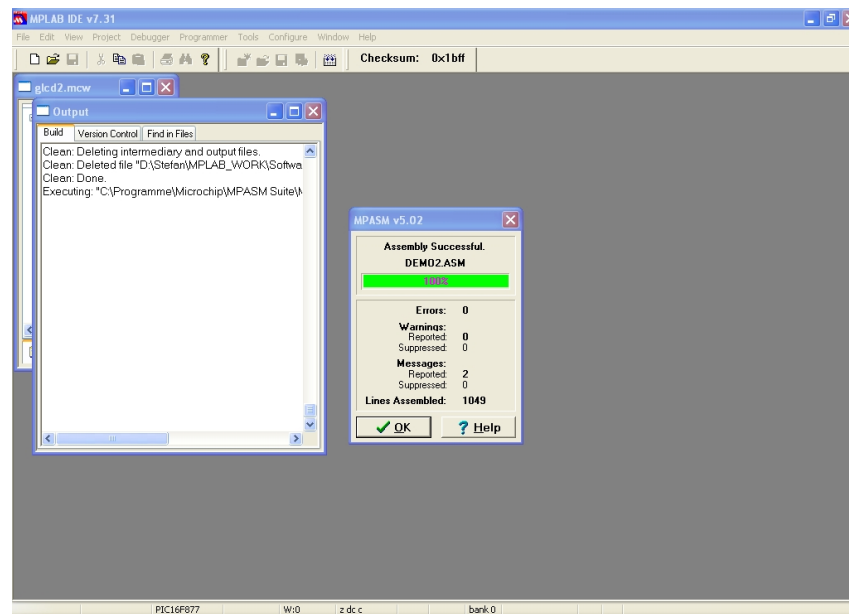


*Linker Script* → *Add Files ...*

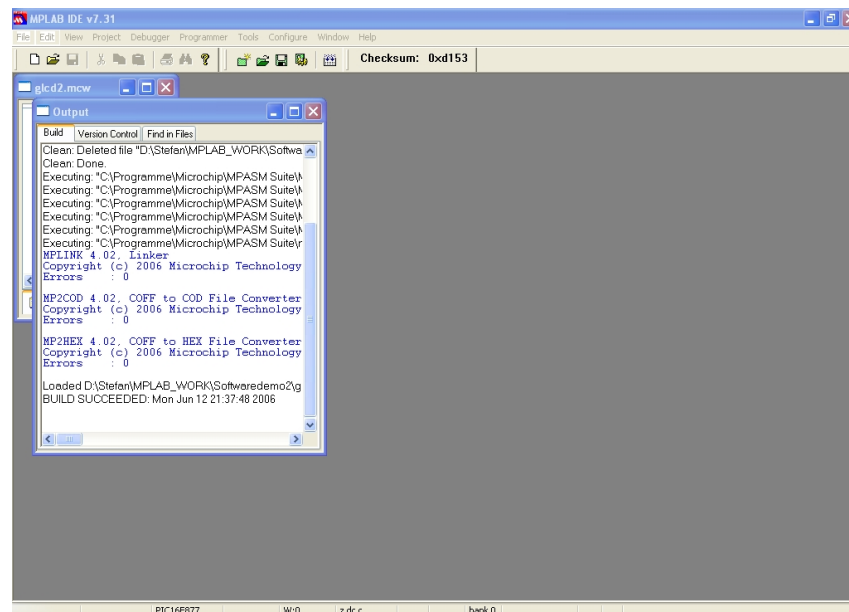
Linker-Datei für den PIC16F877 (16f877.lkr), dieser befindet sich unter C:\\Programme\\Microchip\\MPASM Suite\\LKR\\\\ hinzufügen:



*Project* → *Build*



Für jede Assembler-Datei sollte nun nacheinander die Box *Assembly Successful* erscheinen. Diese muss jedes Mal mit der OK-Taste bestätigt werden. Wichtig ist die Meldung *BUILD SUCCEEDED* am Ende der Assemblierung.



# Literaturverzeichnis

- [1] Datenblatt des Epson-Controller SED1520
- [2] Datenblatt des Grafik-LC-Display NLC-122B032
- [3] Homepage des Mikrocontroller-Hersteller [www.microchip.com](http://www.microchip.com)
- [4] CC5X Versions 3.2 User's Manual
- [5] [www.cc5x.de](http://www.cc5x.de)
- [6] [www.stefan-buchgeher.info/elektronik/cc5x/cc5x.html](http://www.stefan-buchgeher.info/elektronik/cc5x/cc5x.html)