Ansteuerung eines alpha-numerischen LC-Display

(mit PIC-Mikrocontroller)

Autor: Buchgeher Stefan

Letzte Bearbeitung: 10. Oktober 2004

Inhaltsverzeichnis

1. Grundlegendes zum LC-Display

Bild 1.1: LC-Disply

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.

LC-Displays werden von vielen Herstellern auf der ganzen Welt hergestellt. Wie viele Zeichen dargestellt werden können ist sehr unterschiedlich. Es gibt einzeilige, zweizeilige und vierzeilige Typen, die jeweils 8, 16, 20 oder 40 Zeichen je Zeile darstellen können. (Bild 1.1 zeigt ein zweizeiliges LC-Display mit je 16 Zeichen.) Die Ansteuerung dieser unter-schiedlichen Typen ist aber einheitlich und soll hier näher beschrieben werden.

Dass diese LC-Displays einheitlich angesteuert werden verdanken wir dem Hitachi-Controller HD44780. Dieser hat sich als Standard durchgesetzt und wird von allen LC-Display-Herstellern verwendet. Daneben gibt es aber noch Controller anderer Hersteller, die aber kompatibel zum HD44780 sind.

2. Hardware

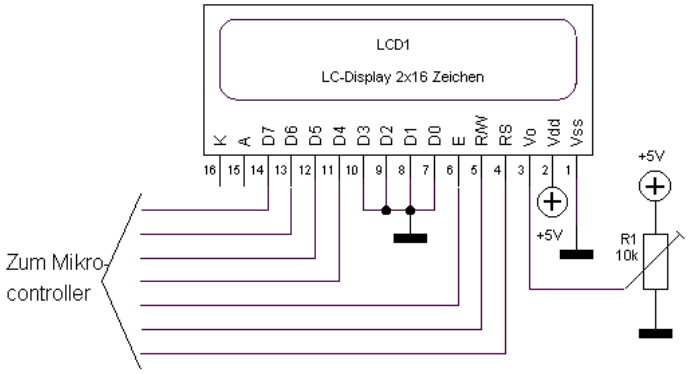

Bild 2.1: LCD-Ansteuerung im 4-Bit-Mode

Bild 2.1 zeigt die minimale Beschaltung des LC-Displays.

Ein LC-Display besitzt die folgenden Anschlusspins:

- Spannungsversorgung (Vdd und Vss, Pins 1 und 2): Die LC-Displays werden mit einer Betriebsspannung Vdd von +5V betrieben. Vss ist der Masseanschluss. Die Stromaufnahme beträgt ohne Beleuchtung meist unter 1mA, höchstens aber 5mA.
- Spannung zur Kontrasteinstellung (Vo, Pin 3): Die Kontrasteinstellung wird üblicherweise mit einem 10k-Trimmer realisiert
- 8-Bit-Datenbus (D0 bis D7, Pin 7 bis 14): Dieser Datenbus dient zur Übertragung der Daten zum/vom Display (schreiben oder lesen) und kann entweder 8-bittig oder nur 4-bittig erfolgen. Der Vorteil des 4-bittigen Datenbuses gegenüber dem 8-bittigen ist die geringere Anzahl an Anschlusspins zum Mikrocontroller. Dieser Vorteil wird aber mit einer etwas umfangreicheren Software erkauft. Beim 4-bittigen werden die nicht benötigten Datenpins D0 (7) bis D3 (10) mit Masse verbunden.
- Enableleitung (E, Pin 6): Dieser Anschluss kann das Display deaktiviert bzw. aktiviert werden. Nur wenn die Enableleitung auf High-Pegel liegt, kann mit das Display angesprochen werden.
- Read/Write-Leitung (R/W, Pin 5): Dieser Anschluss bestimmt, ob Daten zum Display geschrieben (R/W = 0) oder, ob Daten vom Display ausgelesen werden (R/W = 1). Anmerkung: Werden bei einer Anwendung **nur** Daten zum Display geschrieben, so

kann dieser Anschluss auch mit Masse verbunden werden. Der dadurch freiwerdende Pin des steuernden Mikrocontrollers kann dann für eine andere Aufgabe verwendet werden.

- Register-Select-Leitung (RS, Pin 4): Dieser Anschluss bestimmt, ob es sich bei den zum Display geschriebene Daten um Steuerbefehle für das Display (RS = 0), oder um am Display darstellbare Zeichen (RS = 1) handelt.
- **Optional:** Hintergrundbeleuchtung (A und K, Pin 15 und 16): Sofern diese beiden Anschlüsse vorhanden sind, kann damit das Display beleuchtet werden. Dabei muss der Abschluss A (Anode) mit der Betriebsspannung (+5V) während der Anschluss K (Kathode) mit der Masse verbunden werden.

3. Ansteuerung des LC-Displays

3.1. Befehle zur Ansteuerung des LC-Displays

Die folgende Tabelle zeigt die Anweisungen zur Kommunikation mit dem LC-Display.

Anweisung	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Anmerkung	Zeit
Display löschen	0	0	0	0	0	0	0	0	0	1	Löscht das Display, Cursor auf Adresse 0	1,64 ms
Cursor zum Anfang	0	0	0	0	0	0	0	0	1		Cursor auf Adresse 0, Display-Shift entfernen	1,64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	s	I/D: Cursor Laufrichtung, S: Shift ein/aus	40 µs
Display ein/aus	0	0	0	0	0	0	1	D	C	B	D: Display ein/aus C: Cursor ein/aus B: Blinkender Cursor ein/aus	40 µs
Cursor/Display-Shift	0	0	0	0	0	1	S/C	R/L			S/C: Display oder Cursor, R/L: nach rechts oder links schieben	40 µs
Function Set	0	0	0	0	1	DL	N	F			DL: Interface 8-/4-Bit, N: ein-/zweizeilig F:5x8/5x11 Zeichen	40 µs
CG RAM Address Set	0	0	0	1							CG RAM-Adresse einstellen	40 µs
DD RAM Address Set	0	0	1								DD RAM-Adresse einstellen	40 µs
Busy-Flag	0	1	BF								Busy-Flag, Adresszähler lesen	40 µs
CG RAM / DD RAM Daten schreiben	1	0									Daten zum Textpuffer oder CG RAM schreiben	40 µs
CG RAM / DD RAM Daten lesen	1	1									Daten zum Textpuffer oder CG RAM lesen	40 µs

Tabelle 3.1: Display-Befehle

Achtung:
Damit am Display Texte oder Werte angezeigt werden können muss es zuerst initialisiert werden. Beim Einschalten (oder beim Anlegen einer Betriebsspannung) führen die meisten Displays automatisch eine Power-On-Reset-Funktion aus. Das Display befindet sich dann meist im folgenden Zustand:

- Display aus (D=0), Cursor aus (C=0), Cursor blinkt nicht (B=0)
- Display-Shift aus (S=0)
- Cursor geht nach jedem Zeichen eine Position nach rechts (I/D=1)
- 8-Bit-Interface (DL=1)
- 1-zeiliges Display (N=0)

Diese Reset-Prozedur dauert ca. 15 ms. Während dieser Zeit kann es keine Anweisungen ausführen. Das Busy-Flag ist während dieser Zeit gesetzt.

Es ist also eine Initialisierung des Displays notwendig, damit es in den gewünschten Betriebsmode gebracht wird. Die einzelnen Parameter können nach der Initialisierung jederzeit geändert werden, falls dies erwünscht ist. Mehr zur Initialisierung siehe Abschnitt *4.3.1 Unterprogramm LCDINIT*.

3.2. Zeichensatz

Bild 3.1: Zeichensatz des LC-Displays

3.3. Zeitdiagramme

Daten bzw. Zeichen an das LC-Display schreiben:

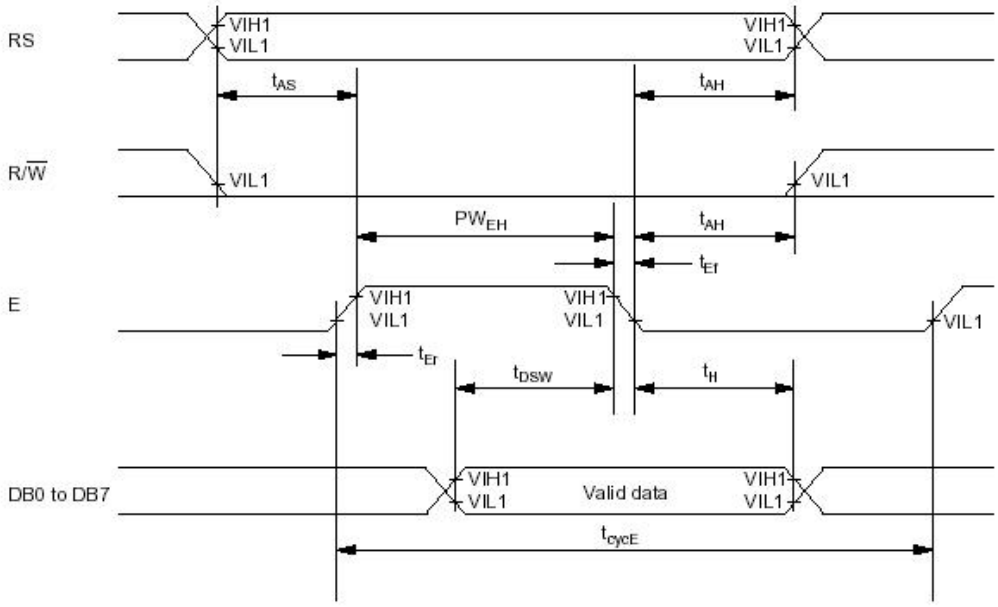

Bild 3.2. Zeitdiagramm (zum Display schreiben)

	Symbol	Min	Typ	Max	Einheit
Enable cycle time	t_{cycE}	500	—	—	ns
Enable pulse width (high level)	PW_{EH}	230	—	—	ns
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20	ns
Adress set-up time (RS, R/W to E)	t_{AS}	40	—	—	ns
Adress hold time	t_{AH}	10	—	—	ns
Data set-up time	t_{DSW}	80	—	—	ns
Data hold time	t_H	10	—	—	ns

Tabelle 3.2: Zeiten beim Schreiben zum Display

Daten bzw. Zeichen vom LC-Display lesen:

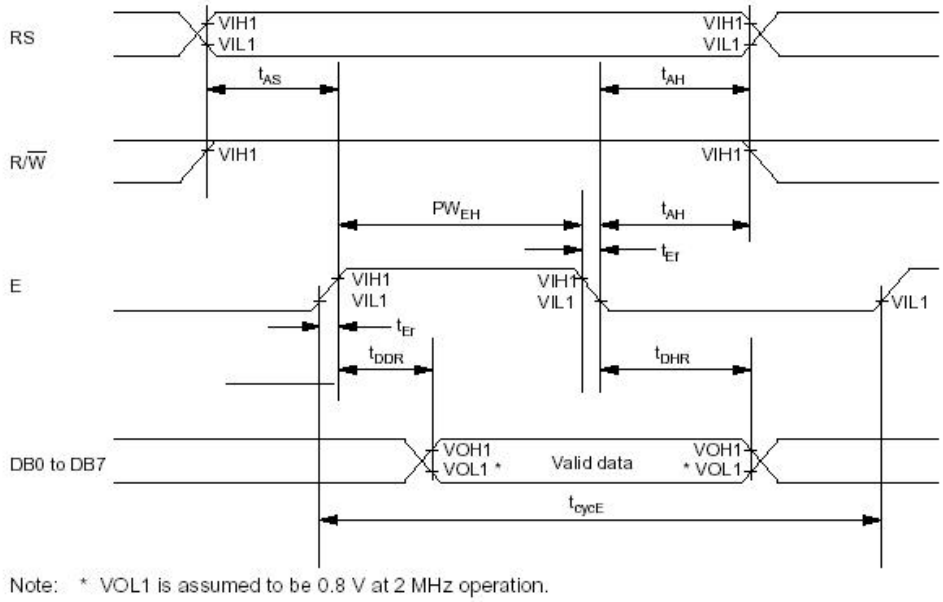

Bild 3.3. Zeitdiagramm (vom Display lesen)

	Symbol	Min	Typ	Max	Einheit
Enable cycle time	t_{cycE}	500	—	—	ns
Enable pulse width (high level)	PW_{EH}	230	—	—	ns
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20	ns
Adress set-up time (RS, R/W to E)	t_{AS}	40	—	—	ns
Adress hold time	t_{AH}	10	—	—	ns
Data delay time	t_{DDR}	—	—	160	ns
Data hold time	t_H	10	—	—	ns

Tabelle 3.3: Zeiten beim Lesen vom Display

4. Software (in Assembler)

4.1. Benötigte Register und Portdefinitionen

Register:
Für die softwaremäßige LC-Display-Ansteuerung werden neben einigen PIC internen Register (SFR, **S**pezielle **F**unktions-**R**egister) noch zwei Hilfsregister (TEMP1 und TEMP2) benötigt. Diese beiden Register sind so genannte temporäre Register. D.h. Sie werden nur kurzzeitig verwendet und können daher auch in beliebigen anderen Unterprogrammen verwendet werden.

Portdefinition:
Im Allgemeinen wird bei jeder Anwendung das 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:

- LCD_DATA: Dieser Parameter gibt den Port für den Datenbus an. (zum Beispiel Port A, Port B,..). Achtung: Es wird der 4-bit-Datenbus verwendet, welcher aus dem höheren Nibble des verwendeten Ports besteht.
- LCD_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!
- LCD_CTRL: Dieser Parameter gibt den Port für die Steuerleitungen an. (z.B. Port A, Port B,..)
- LCD_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!
- LCD_RS: Dieser Parameter gibt den Portpin der Steuerleitung *Register-Select* an.
- LCD_RW: Dieser Parameter gibt den Portpin der Steuerleitung *Read-Write* an.
- LCD_E: Dieser Parameter gibt den Portpin der Steuerleitung *Enable* an.

Eine mögliche Portdefinition ist:

```
LCD_DATA          equ   PORTB
LCD_DATA_TRIS     equ   TRISB
LCD_CTRL          equ   PORTB
LCD_CTRL_TRIS     equ   TRISB

LCD_RS            equ   1
LCD_RW            equ   2
LCD_E             equ   3
```

4.2. Initialisierung (Unterprogramm „INIT“)

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

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine für den PIC16F84. Das LC-Display ist hier am Port B angeschlossen, und wird hier im 4-bit-Mode betrieben.

```
INIT        bsf   STAT,RP0          ;Registerseite 1
            movlw b'00000000'       ;Port B als Ausgang definieren
            movwf TRISB
            bcf   STAT,RP0          ;Registerseite 0

            return
```

4.3. Unterprogramme für die LCD-Ansteuerung

Zur Ansteuerung eines alphanumerischen LC-Displays sind 5 Unterprogramme notwendig:

- Zuerst muss das LC-Display initialisiert werden. Das Unterprogramm *LCDINIT* übernimmt diese Aufgabe
- Befehle an das LC-Display werden mit dem Unterprogramm *LCDBEFEHL* erzeugt, während die am LC-Display anzuzeigenden Zeichen mit dem Unterprogramm *LCDZEICHEN* erzeugt werden
- Das LC-Display benötigt etwas Zeit zur Abarbeitung der Anweisungen und setzt dabei ein so genanntes Busy-Flag. Das Unterprogramm *LCDBUSY* überprüft dieses Flag, und wartet solange bis das LC-Display für eine Anweisung bereit ist
- Während der Initialisierung des LC-Displays benötigt das LC-Display verschiedene Zeitverzögerungen. Diese werden mit dem Unterprogramm *VERZ100US* erzeugt, wobei die Verzögerungszeit über einem Parameter eingestellt werden kann.

Nun aber zu den einzelnen Unterprogrammen im Detail:

4.3.1. Unterprogramm LCDINIT

Aufgabe:
Dieses Unterprogramm initialisiert das LC-Display (siehe auch Abschnitt 3*. Befehle zur Ansteuerung des LC-Display*)

Vorgehensweise:

- Die Steuerleitungen des LC-Displays (E, RS, R/W) auf Low legen
- 15 ms warten
- Befehl: 8-bit-Interface an LCD
- 4,1 ms warten
- Wiederholung des Befehls: 8-bit-Interface an LCD
- 0,1 ms warten
- Wiederholung des Befehls: 8-bit-Interface an LCD
- Warten, bis LCD für weitere Anweisungen bereit ist
- Befehl: 4-bit-Interface an LCD; Achtung: ab nun müssen Befehle an das LCD mit dem Unterprogramm LCDBEFEHL erfolgen
- Befehl: Display löschen und Cursor home
- Befehl: 4 Bit, 2 Zeilen, 5x7
- Befehl: Display aus, Cursor aus, Blinken aus
- Befehl: Shift aus
- Befehl: Display ein

Hier das Unterprogramm:

```
LCDINIT     bcf   LCD_CTRL,LCD_E    ;Alle Control-Lines = Low
            bcf   LCD_CTRL,LCD_RS
            bcf   LCD_CTRL,LCD_RW

            movlw .150              ;15 ms warten
            call  VERZ100US

            movlw 0x0F
            andwf LCD_DATA,f        ;Höherw. Nibble löschen
```

```
movlw 0x30              ;Befehl für 8-Bit-Interface
iorwf LCD_DATA,f        ;Befehl an LCD
bsf   LCD_CTRL,LCD_E    ;  (durch toggeln der LCD-Enable-Leitung)
bcf   LCD_CTRL,LCD_E

movlw .41               ;4.1 ms warten
call  VERZ100US

bsf   LCD_CTRL,LCD_E    ;Befehl (8-bit-Interface) wiederholen
bcf   LCD_CTRL,LCD_E    ;  (durch toggeln der LCD-Enable-Leitung)

movlw .1                ;100us warten
call  VERZ100US

bsf   LCD_CTRL,LCD_E    ;Befehl (8-bit-Interface) wiederholen
bcf   LCD_CTRL,LCD_E    ;  (durch toggeln der LCD-Enable-Leitung)

call  LCDBUSY           ;Warten bis LCD bereit

movlw 0x0F
andwf LCD_DATA,f        ;Höherw. Nibble löschen
movlw 0x20              ;Befehl für 4-Bit Interface
iorwf LCD_DATA,f        ;Befehl an LCD
bsf   LCD_CTRL,LCD_E    ;  (durch toggeln der LCD-Enable-Leitung)
bcf   LCD_CTRL,LCD_E

movlw 0x01              ;Display löschen und Cursor home
call  LCDBEFEHL         ;Befehl an LCD

movlw 0x28              ;4 Bit, 2 Zeilen, 5x7
call  LCDBEFEHL         ;Befehl an LCD

movlw 0x08              ;Display aus, Cursor aus, Blinken aus
call  LCDBEFEHL         ;Befehl an LCD

movlw 0x06              ;Shift aus
call  LCDBEFEHL         ;Befehl an LCD

movlw 0x0C              ;Display ein
call  LCDBEFEHL         ;Befehl an LCD

return
```

4.3.2. Unterprogramm LCDBEFEHL

Aufgabe:
Dieses Unterprogramm sendet einen Befehl nibbleweise an das LC-Display.

Vorgehensweise:

- Der Befehl, welcher an das LC-Display übertragen wird, befindet sich als 8-Bit-Wert im Arbeitsregister (w-Register). Diesen zunächst im temporären Register TEMP1 zwischenspeichern.
- Warten bis das LC-Display für einen Befehl bereit ist. (Dazu ist das Unterprogramm LCDBUSY zuständig).
- Für die Kommunikation mit dem LC-Display steht "nur" ein 4-Bit-Datenbus zur Verfügung. Der 8-bit-Befehl wird daher nibbleweise an das LC-Display übertragen. Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei dürfen

nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.

- Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung handelt.
- Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das höherwertigere Befehlsnibble ein)
- Nun das niederwertigere Befehlsnibble auf den 4-Bit-Datenbus legen. Auch hier dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das niederwertigere Befehlsnibble ein).

Hier das Unterprogramm:

```
LCDBEFEHL   movwf TEMP1             ;Auszugebendes Zeichen (befindet sich im
                                    ;  w-Register) in TEMP1 zwischenspeichern
            call  LCDBUSY           ;Warten bis LCD bereit ist
            movlw 0x0F
            andwf LCD_DATA,f        ;Höherwertiges Nibble löschen
            movf  TEMP1,w
            andlw 0xF0              ;Höherwertiges Nibble ausmaskieren
            iorwf LCD_DATA,f        ;High-Nibble an LCD übergeben
            bcf   LCD_CTRL,LCD_RW   ;LCD im Schreiben-Mode
            bcf   LCD_CTRL,LCD_RS   ;LCD im Befehl-Mode
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            movlw 0x0F
            andwf LCD_DATA,f        ;Höherwertiges Nibble löschen
            swapf TEMP1,w
            andlw 0xF0              ;Niederwert. Nibble ausmaskieren
            iorwf LCD_DATA,f        ;Low-Nibble an LCD übergeben
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            return
```

Anmerkungen:

- Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCDZEICHEN nur dadurch, dass hier die Register-Select-Steuerleitung gelöscht ist, da mit diesem Unterprogramm Befehle an das LC-Display übertragen werden. (Siehe Tabelle im Abschnitt *3. Befehle zur Ansteuerung des LC-Display*)
- Das temporäre Register TEMP1 dient hier nur als Hilfsregister. Es wird hier nur kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet werden.

4.3.3. Unterprogramm LCDZEICHEN

Aufgabe:
Dieses Unterprogramm sendet ein Zeichen, welches am LC-Display ausgegeben wird, nibbleweise an das LC-Display.

Vorgehensweise:

- Das Zeichen, welches am LC-Display ausgegeben werden soll, befindet sich als 8-Bit-Wert im Arbeitsregister (w-Register). Diesen Wert zunächst im temporären Register TEMP1 zwischenspeichern

- Warten bis das LC-Display für ein Zeichen bereit ist. (Dazu ist das Unterprogramm LCDBUSY zuständig)
- Für die Kommunikation mit dem LC-Display steht "nur" ein 4-bit-Datenbus zur Verfügung. Der 8-bit-Wert wird daher nibbleweise an das LC-Display übertragen
- Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung handelt.
- Register-Select-Steuerleitung setzen, da es sich hier um ein Zeichen handelt, welches am LC-Display ausgegeben werden soll.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das höherwertige Zeichennibble ein)
- Nun das niederwertigere Zeichennibble auf den 4-Bit-Datenbus legen. Auch hier dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das niederwertigere Zeichennibble ein)

Hier das Unterprogramm:

```
LCDZEICHEN  movwf TEMP1             ;Auszugebendes Zeichen (befindet sich im
                                    ;w-Register) in TEMP1 zwischenspeichern
            call  LCDBUSY           ;Warten bis LCD bereit ist
            movlw 0x0F
            andwf LCD_DATA,f        ;Höherwertiges Nibble löschen
            movf  TEMP1,w
            andlw 0xF0              ;Höherwertiges Nibble ausmaskieren
            iorwf LCD_DATA,f        ;High-Nibble an LCD übergeben
            bcf   LCD_CTRL,LCD_RW   ;LCD im Schreiben-Mode
            bsf   LCD_CTRL,LCD_RS   ;LCD im Daten-Mode
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            movlw 0x0F
            andwf LCD_DATA,f        ;Höherwertiges Nibble löschen
            swapf TEMP1,w
            andlw 0xF0              ;Niederwert. Nibble ausmaskieren
            iorwf LCD_DATA,f        ;Low-Nibble an LCD übergeben
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            return
```

Anmerkungen:

- Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCDBEFEHL nur dadurch, dass hier die Register-Select-Steuerleitung gesetzt ist, da mit diesem Unterprogramm am LC-Display auszugebende Zeichen an das LC-Display übertragen werden. (Siehe Tabelle im Abschnitt *3. Befehle zur Ansteuerung des LC-Display*)
- Das temporäre Register TEMP1 dient hier nur als Hilfsregister. Es wird hier nur kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet werden.

4.3.4. Unterprogramm LCDBUSY

Aufgabe:
Dieses Unterprogramm prüft das Busy-Flag des LC-Displays, und verlässt es erst wenn das Busy-Flag low ist, also wenn das LC-Display bereit für einen Befehl oder für ein auszugebendes Zeichen ist.

Vorgehensweise:
- Der Datenport muss als Eingang definiert werden, da nun der Zustand (insbesondere der Zustand das Busy-Flag) des LC-Display gelesen wird. Dazu muss aber in die Registerbank 1 des PIC gewechselt werden und anschließend wieder zurück zur Registerbank 0.
- Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.
- Read-Write-Steuerleitung setzen, da es sich hier um eine Leseanweisung handelt.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display gibt nun das höherwertigere Nibble auf den 4-Bit-Datenbus.)
- Den 4-Bit-Wert vom Datenbus einlesen und im Hilfsregister TEMP2 sichern.
- Eine weiter fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display gibt nun das niederwertigere Nibble auf den 4-Bit-Datenbus
- Den 4-Bit-Wert vom Datenbus einlesen und mit dem Hilfsregister TEMP2 zu einem 8-Bit-Wert verschmelzen.
- Das Busy-Flag (Bit 7 von TEMP2) überprüfen. Ist dieses Flag gesetzt, die soeben durchgeführten Schritte so oft wiederholen bis das LC-Display das Busy-Flag löscht.
- Ist das Busy-Flag low, (also gelöscht) die Read-Write-Steuerleitung wieder zurücksetzen und den Datenport wieder als Ausgang definieren. Dazu muss aber wieder in die Registerbank 1 des PIC gewechselt werden und anschließend wieder zurück zur Registerbank 0.

Hier das Unterprogramm:

```
LCDBUSY     bsf   STAT,RP0          ;Registerbank 1
            movlw 0xF0              ;Höherwertigeren HalbPort
            iorwf LCD_DATA_TRIS,w   ;  als Eingang definieren
            movwf LCD_DATA_TRIS
            bcf   STAT,RP0          ;Registerbank 0

            bcf   LCD_CTRL,LCD_RS   ;LCD im Befehls-Mode
            bsf   LCD_CTRL,LCD_RW   ;LCD im Lesen-Mode
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            movf  LCD_DATA,w
            andlw 0xF0              ;Niederw.Nibble ausmaskieren
            movwf TEMP2
            bsf   LCD_CTRL,LCD_E    ;Enable (LCD)
            bcf   LCD_CTRL,LCD_E    ;  toggeln
            swapf LCD_DATA,w        ;Niederwertigeres Nibble Busy-Flag
            andlw 0x0F              ;Höherwertiges Nibble ausmaskieren
            iorwf TEMP2,w           ;Nibbles verbinden
            btfsc TEMP2,7           ;Busy-Flag prüfen (High=busy)
            goto  LCDBUSY           ;Dieses UP so oft abarbeiten,bis das Busy-
                                    ;  Flag low ist

            bcf   LCD_CTRL,LCD_RW   ;Busy = low: LCD im Schreiben-Mode
            bsf   STAT,RP0          ;Registerbank 1
            movlw 0x0F              ;Höherwertigeren HalbPort
            andwf LCD_DATA,w        ;  als Ausgang definieren
            movwf LCD_DATA_TRIS
```

```
            bcf   STAT,RP0          ;Registerbank 0
            return
```

Anmerkung:
Das temporäre Register TEMP2 dient hier nur als Hilfsregister. Es wird hier nur kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet werden.

4.3.5. Unterprogramm VERZ100US

Aufgabe:
Dieses Unterprogramm erzeugt eine Zeitverzögerung. Der Übergabeparameter (im w-Register) gibt dabei an, wie oft eine Zeitverzögerung von 100µs erfolgen soll. Der mögliche Zeitverzögerungsbereich liegt hier daher im Bereich von 100µs und 25,5ms (=255 x 100us).

Hier das Unterprogramm:

```
VERZ100US
            movwf TEMP1       ;Übergabeparameter in TEMP1 sichern

VERZSCHL1
            movlw .25         ;Bei Verwendung eines 4-MHz-Taktes
;           movlw .75         ;Bei Verwendung eines 12-MHz-Taktes
;           movlw .125        ;Bei Verwendung eines 20-MHz-Taktes
            movwf TEMP2
VERZSCHL2   nop
            decfsz TEMP2,f
            goto  VERZSCHL2
            decfsz TEMP1,f
            goto  VERZSCHL1

            return
```

Anmerkung:

- Die temporären Register TEMP1 und TEMP2 dienen hier nur als Übergabe- bzw. als Hilfsregister. Diese Register werden hier nur kurzzeitig verwendet und können daher auch in anderen Unterprogrammen verwendet werden.
- Das Hilfsregister TEMP2 muss je nach verwendetem Takt mit unterschiedlichen Werten geladen werden:

Takt	Wert von TEMP2
4 MHz	25
12 MHz	75
20 MHz	125

5. Software (in C mit CC5X)

5.1. Benötigte Konstanten und Portdefinitionen

Konstanten:

```
#define     LCD_CLR                 0x01  // Display loeschen
#define     LCD_ON_C                0x0E  // Cursor home
#define     LCD_FUNCTION_SET        0x28  // 4 Bit, 2 Zeilen, 5x7
#define     LCD_DISP_ON             0x0C  // Display Ein
#define     LCD_DISP_OFF            0x08  // Display Aus
#define     LCD_ENTRY_INC           0x06
```

Portdefinition:
Im Allgemeinen wird bei jeder Anwendung das 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:

- LCD_DATA: Dieser Parameter gibt den Port für den Datenbus an. (zum Beispiel Port A, Port B,..). Achtung: Es wird der 4-bit-Datenbus verwendet, welcher aus dem höheren Nibble des verwendeten Ports besteht.
- LCD_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!
- LCD_CTRL: Dieser Parameter gibt den Port für die Steuerleitungen an. (z.B. Port A, Port B,..)
- LCD_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!
- LCD_RS: Dieser Parameter gibt den Portpin der Steuerleitung *Register-Select* an.
- LCD_RW: Dieser Parameter gibt den Portpin der Steuerleitung *Read-Write* an.
- LCD_E: Dieser Parameter gibt den Portpin der Steuerleitung *Enable* an.

Eine mögliche Portdefinition ist:

```
#pragma char LCD_DATA         @ PORTB
#pragma char LCD_DATA_TRIS    @ TRISB
#pragma char LCD_CTRL         @ PORTB
#pragma char LCD_CTRL_TRIS    @ TRISB

bit LCD_RS                    @ LCD_CTRL.1;
bit   LCD_RW                  @ LCD_CTRL.2;
bit LCD_E                     @ LCD_CTRL.3;
```

5.2. Initialisierung (Unterprogramm „INIT“)

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

Der folgende Programmausschnitt zeigt eine mögliche Initialisierungsroutine für den PIC16F84. Das LC-Display ist hier am Port B angeschlossen, und wird hier im 4-bit-Mode betrieben.

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

5.3. Unterprogramme für die LCD-Ansteuerung

Zur Ansteuerung eines alphanumerischen LC-Displays sind 5 Unterprogramme notwendig:

- Zuerst muss das LC-Display initialisiert werden. Das Unterprogramm *LCD_INIT* übernimmt diese Aufgabe
- Befehle an das LC-Display werden mit dem Unterprogramm *LCD_BEFEHL* erzeugt, während die am LC-Display anzuzeigenden Zeichen mit dem Unterprogramm *LCD_ZEICHEN* erzeugt werden
- Das LC-Display benötigt etwas Zeit zur Abarbeitung der Anweisungen und setzt dabei ein so genanntes Busy-Flag. Das Unterprogramm *LCD_BUSY* überprüft dieses Flag, und wartet solange bis das LC-Display für eine Anweisung bereit ist
- Während der Initialisierung des LC-Displays benötigt das LC-Display verschiedene Zeitverzögerungen. Diese werden mit dem Unterprogramm *VERZ100US* erzeugt, wobei die Verzögerungszeit über einem Parameter eingestellt werden kann.

Mit einem weiteren (optionalen) Unterprogramm kann eine ganze Zeichenkette am LC-Display ausgegeben werden. Dieses Unterprogramm heißt *LCD_STRING* und ist im Abschnitt 5.3.5 näher beschrieben.

Nun aber zu den einzelnen Unterprogrammen im Detail:

5.3.1. Unterprogramm LCD_INIT

Aufgabe:
Dieses Unterprogramm initialisiert das LC-Display (siehe auch Abschnitt 3*. Befehle zur Ansteuerung des LC-Display*)

Vorgehensweise:

- Die Steuerleitungen des LC-Displays (E, RS, R/W) auf Low legen
- 15 ms warten
- Befehl: 8-bit-Interface an LCD
- 4,1 ms warten
- Wiederholung des Befehls: 8-bit-Interface an LCD

- 0,1 ms warten
- Wiederholung des Befehls: 8-bit-Interface an LCD
- Warten, bis LCD für weitere Anweisungen bereit ist
- Befehl: 4-bit-Interface an LCD; Achtung: ab nun müssen Befehle an das LCD mit dem Unterprogramm LCDBEFEHL erfolgen
- Befehl: Display löschen und Cursor home
- Befehl: 4 Bit, 2 Zeilen, 5x7
- Befehl: Display aus, Cursor aus, Blinken aus
- Befehl: Shift aus
- Befehl: Display ein

Hier das Unterprogramm:

```
void LCD_INIT(void)
{
      LCD_RS = 0;                  // Alle LCD-Steuerleitungen LOW
      LCD_RW = 0;
      LCD_E = 0;
      VERZ100US(150);

      LCD_DATA = LCD_DATA & 0x0F;  // Hoeherwertigeres Nibble loeschen
      LCD_DATA = LCD_DATA | 0x30;  // Befehl fuer 8-Bit-Interface

      LCD_E = 1;                   // Befehl an LCD
      LCD_E = 0;                   //   (durch toggeln der LCD-Enable-
                                   //   Leitung)
      VERZ100US(41);               // 4.1 ms warten

      LCD_E = 1;                   // Befehl (8-Bit-Interface) wiederholen
      LCD_E = 0;                   //   (durch toggeln der LCD-Enable-
                                   //   Leitung)
      VERZ100US(1);                // 100 us warten

      LCD_E = 1;                   // Befehl (8-Bit-Interface) wiederholen
      LCD_E = 0;                   //   (durch toggeln der LCD-Enable-
                                   //Leitung)

      LCD_BUSY();                  // warten bis LCD bereit

      LCD_DATA = LCD_DATA & 0x0F;  // Hoeherwertigeres Nibble loeschen
      LCD_DATA = LCD_DATA | 0x20;  // Befehl fuer 4-Bit-Interface

      LCD_E = 1;                   // Befehl an LCD
      LCD_E = 0;                   //   (durch toggeln der LCD-Enable-
                                   //   Leitung)

      LCD_BEFEHL(LCD_CLR);         // Display loeschen und Cursor home
      LCD_BEFEHL(LCD_FUNCTION_SET); // 4 Bit, 2 Zeilem, 5x7
      LCD_BEFEHL(LCD_DISP_OFF);    // Display aus, Cursor aus, Blinken aus
      LCD_BEFEHL(LCD_ENTRY_INC);   // Shift aus
      LCD_BEFEHL(LCD_DISP_ON);     // Display ein
}
```

5.3.2. Unterprogramm LCD_BEFEHL

Aufgabe:
Dieses Unterprogramm sendet einen Befehl nibbleweise an das LC-Display.

Vorgehensweise:

- Warten bis das LC-Display für einen Befehl bereit ist. (Dazu ist das Unterprogramm LCD_BUSY zuständig).
- Für die Kommunikation mit dem LC-Display steht "nur" ein 4-Bit-Datenbus zur Verfügung. Der 8-bit-Befehl wird daher nibbleweise an das LC-Display übertragen. Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung handelt.
- Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das höherwertigere Befehlsnibble ein)
- Nun das niederwertigere Befehlsnibble auf den 4-Bit-Datenbus legen. Auch hier dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das niederwertigere Befehlsnibble ein).

Hier das Unterprogramm:

```
void LCD_BEFEHL(unsigned char befehl)
{
      unsigned char temp;


      LCD_BUSY();                   // Warten bis LCD bereit ist

      LCD_DATA = LCD_DATA & 0x0F;   // Hoeherwertigeres Nibble loeschen
      temp = befehl & 0xF0;         // Hoeherwertiges Nibble ausmaskieren
      LCD_DATA = LCD_DATA | temp;   // High-Nibble an LCD uebergeben

      LCD_RW = 0;                   // LCD im Schreiben-Mode
      LCD_RS = 0;                   // LCD im Befehl-Mode
      LCD_E = 1;                    // Enable (LCD)
      LCD_E = 0;

      LCD_DATA = LCD_DATA & 0x0F;   // Hoeherwertigeres Nibble loeschen
      SWAPF(befehl,1);
      temp = befehl & 0xF0;         // Niederwert. Nibble ausmaskieren
      LCD_DATA = LCD_DATA | temp;   // High-Nibble an LCD uebergeben

      LCD_E = 1;                    // Enable (LCD)
      LCD_E = 0;                    //   toggeln
}
```

Anmerkung:

Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCD_ZEICHEN nur dadurch, dass hier die Register-Select-Steuerleitung gelöscht ist, da mit diesem Unterprogramm Befehle an das LC-Display übertragen werden. (Siehe Tabelle im Abschnitt *3. Befehle zur Ansteuerung des LC-Display*)

5.3.3. Unterprogramm LCD_ZEICHEN

Aufgabe:
Dieses Unterprogramm sendet ein Zeichen, welches am LC-Display ausgegeben wird, nibbleweise an das LC-Display.

Vorgehensweise:
- Warten bis das LC-Display für ein Zeichen bereit ist. (Dazu ist das Unterprogramm LCD_BUSY zuständig)
- Für die Kommunikation mit dem LC-Display steht "nur" ein 4-bit-Datenbus zur Verfügung. Der 8-bit-Wert wird daher nibbleweise an das LC-Display übertragen
- Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung handelt.
- Register-Select-Steuerleitung setzen, da es sich hier um ein Zeichen handelt, welches am LC-Display ausgegeben werden soll.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das höherwertige Zeichennibble ein)
- Nun das niederwertigere Zeichennibble auf den 4-Bit-Datenbus legen. Auch hier dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die restlichen vier Portpins dürfen nicht verändert werden.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display liest nun das niederwertigere Zeichennibble ein)

Hier das Unterprogramm:
```
void LCD_ZEICHEN(unsigned char zeichen)
{
      unsigned char temp;


      LCD_BUSY();                   // Warten bis LCD bereit ist

      LCD_DATA = LCD_DATA & 0x0F;   // Hoeherwertigeres Nibble loeschen
      temp = zeichen & 0xF0;        // Hoeherwertiges Nibble ausmaskieren
      LCD_DATA = LCD_DATA | temp;   // High-Nibble an LCD uebergeben

      LCD_RW = 0;                   // LCD im Schreiben-Mode
      LCD_RS = 1;                   // LCD im Daten-Mode
      LCD_E = 1;                    // Enable (LCD)
      LCD_E = 0;                    //   toggeln

      LCD_DATA = LCD_DATA & 0x0F;   // Hoeherwertigeres Nibble loeschen
      SWAPF(zeichen,1);
      temp = zeichen & 0xF0;        // Niederwert. Nibble ausmaskieren
      LCD_DATA = LCD_DATA | temp;   // High-Nibble an LCD uebergeben

      LCD_E = 1;                    // Enable (LCD)
      LCD_E = 0;                    //   toggeln
}
```

Anmerkungen:
Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCD_BEFEHL nur dadurch, dass hier die Register-Select-Steuerleitung gesetzt ist, da mit diesem

Unterprogramm am LC-Display auszugebende Zeichen an das LC-Display übertragen werden. (Siehe Tabelle im Abschnitt *3. Befehle zur Ansteuerung des LC-Display*)

5.3.4. Unterprogramm LCD_BUSY

Aufgabe:
Dieses Unterprogramm prüft das Busy-Flag des LC-Displays, und verlässt es erst wenn das Busy-Flag low ist, also wenn das LC-Display bereit für einen Befehl oder für ein auszugebendes Zeichen ist.

Vorgehensweise:
- Der Datenport muss als Eingang definiert werden, da nun der Zustand (insbesondere der Zustand das Busy-Flag) des LC-Display gelesen wird.
- Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.
- Read-Write-Steuerleitung setzen, da es sich hier um eine Leseanweisung handelt.
- Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display gibt nun das höherwertigere Nibble auf den 4-Bit-Datenbus.)
- Den 4-Bit-Wert vom Datenbus einlesen und im Hilfsregister temp1 sichern.
- Eine weiter fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-Display gibt nun das niederwertigere Nibble auf den 4-Bit-Datenbus)
- Den 4-Bit-Wert vom Datenbus einlesen und mit dem Hilfsregister temp2 zu einem 8-Bit-Wert verschmelzen.
- Das Busy-Flag (Bit 7 von temp2) überprüfen. Ist dieses Flag gesetzt, die soeben durchgeführten Schritte so oft wiederholen bis das LC-Display das Busy-Flag löscht.
- Ist das Busy-Flag low, (also gelöscht) die Read-Write-Steuerleitung wieder zurücksetzen und den Datenport wieder als Ausgang definieren.

Hier das Unterprogramm:

```
void LCD_BUSY(void)
{
      unsigned char temp1;
      unsigned char temp2;


      do
      {
            LCD_DATA_TRIS = LCD_DATA_TRIS | 0xF0; // Hoeherwertigen HalbPort
                                    // als Eingang definieren

            LCD_RS = 0;             // LCD im Befehls-Mode
            LCD_RW = 1;             // LCD im Lesen-Mode
            LCD_E = 1;              // Enable (LCD)
            LCD_E = 0;              //   toggeln

            temp1 = LCD_DATA;
            temp1 = temp1 & 0xF0;   // Niederw.Nibble ausmaskieren

            LCD_E = 1;              // Enable (LCD)
            LCD_E = 0;              //   toggeln

            temp2 = LCD_DATA;
            SWAPF(temp2,1);         // Niederwertiges Nibble Busy-Flag
            temp2 = temp2 & 0x0F;   // Hoeherwertiges Nibble ausmaskieren
            temp2 = temp1 | temp2;  // Nibbles verbinden
```

```
      } while (temp2.7);

      LCD_RW = 0;                    // Busy = low: LCD im Schreiben-Mode
      LCD_DATA_TRIS = LCD_DATA & 0x0F; // Höherwertigen HalbPort
}                                    // als Ausgang definieren
```

5.3.5. Unterprogramm LCD_STRING

Aufgabe:
Dieses Unterprogramm sendet einen String Zeichen für Zeichen an das LC-Display.

Vorgehensweise:

- Der Zeiger (engl. Pointer) *pString zeigt auf das erste Zeichen der übergebenen Zeichenkette. Den Inhalt dieses Zeigers in die Variable zeichen kopieren.
- Mit einer Schleife den Inhalt von zeichen an das LC-Display übergeben und die restliche Zeichenkette mit Hilfe des Zeigers Zeichen für Zeichen einlesen und am LC-Display ausgeben.

Hier das Unterprogramm:

```
void LCD_STRING(const char *pString)
{
      char zeichen;


      zeichen = *pString;
      while(zeichen > 0)
      {
            LCD_ZEICHEN(zeichen);   // zeichen am LC-Display ausgeben
            pString++;              // Zeiger um 1 erhoehen, er zeigt nun auf
                                    // das naechste Zeichen der auszugebenden
                                    // Zeichenkette
            zeichen = *pString;     // Den Inhalt diese Zeigers in die
      }                             //  Variable zeichen kopieren.
}
```

Anmerkung:
Eine Zeichenkette (engl. String) wird in C mit einem so genannten Nullbyte abgeschlossen.

5.3.6. Unterprogramm VERZ100US

Aufgabe:
Dieses Unterprogramm erzeugt eine Zeitverzögerung. Der Übergabeparameter VerzZeit gibt dabei an, wie oft eine Zeitverzögerung von 100us erfolgen soll. Der mögliche Zeitverzögerungsbereich liegt hier daher im Bereich von 100us und 25,5ms.

Hier das Unterprogramm:

```
void VERZ100US(unsigned char VerzZeit)
{
      unsigned char temp;


      #asm
      VERZSCHL1
```

```
            movlw .25               // Bei Verwendung eines 4-MHz-Taktes
//          movlw .75               // Bei Verwendung eines 12-MHz-Taktes
//          movlw .125              // Bei Verwendung eines 20-MHz-Taktes
            movwf temp

      VERZSCHL2   nop
            decfsz temp,f
            goto  VERZSCHL2
            decfsz VerzZeit,f
            goto  VERZSCHL1
      #endasm
}
```

6. Demonstrationsbeispiele

Das folgende Beispiel dient nur zur Demonstration. Es zeigt eine mögliche Einbindung der oben beschriebenen Unterprogramme.

6.1. Hardware

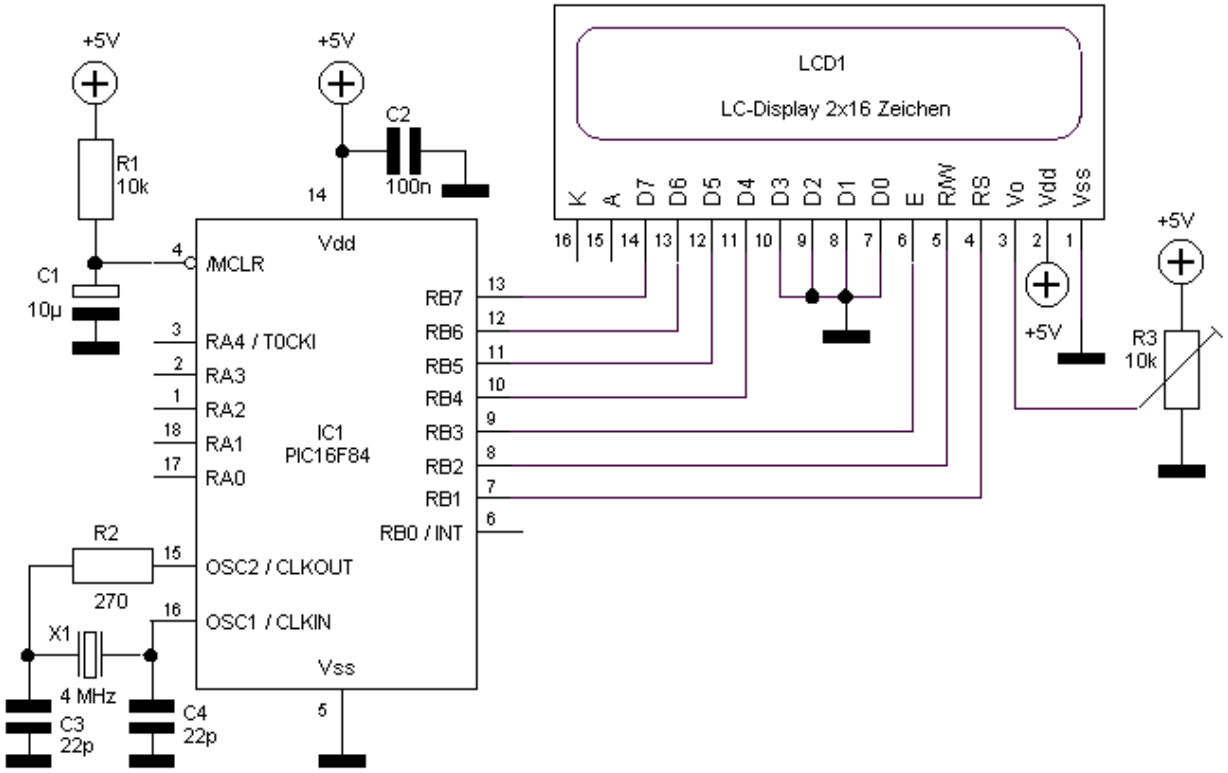

Bild 6.1: Schaltung zur Demonstration der LCD-Routinen

Bei diesem Demonstrationsbeispiel soll ein Mikrocontroller (PIC16F84) einen beliebigen Text (z.B. Hallo, wie geht's) am LC-Display ausgeben. Die Beschaltung des LC-Displays erfolgt gemäß Abschnitt 2.

Für die Takterzeugung dient eine Standardbeschaltung bestehend aus einem 4-MHz-Quarz (X1), zwei Kondensatoren (C4, C5) und einem Widerstand (R2).

Das RC-Glied (R1, C1) erzeugt einen definierten Reset beim Anlegen der Betriebsspannung.

6.2. Softwarebeispiel (in Assembler)

6.2.1.Listing

```
;*************************************************************************************************
;** Demonstration zur Ansteuerung eines alphanumerischen LC-Display                             **
;**                                                                                             **
;** Pinbelegung: Port A: unbenutzt                                                              **
;**              Port B: RB0: unbenutzt (Reserviert für Hintergrundbeleuchtung)                 **
;**                      RB1: RS                                                                **
;**                      RB2: R/W                                                               **
;**                      RB3: E                                                                 **
;**                      RB4: D4                                                                **
;**                      RB5: D5                                                                **
;**                      RB6: D6                                                                **
;**                      RB7: D7                                                                **
;** Anmerkung: Die Bits sollten nach Möglichkeit nicht verändert werden, es kann jedoch         **
;**            ein anderer Port gewählt werden.                                                 **
;**                                                                                             **
;** Entwickler: Buchgeher Stefan                                                                **
;** Entwicklungsbeginn der Software: 21. Juli 2002                                              **
;** Funktionsfähig seit: 21. Juli 2002                                                          **
;** Letzte Bearbeitung: 26. Februar 2004                                                        **
;*************************************************************************************************

               List p=PIC16F84


;******************************** Register (in Registerseite 0) ********************************
STAT           equ    3              ;Statusregister
PORTB          equ    6              ;PortB-Register


;******************************** Register (in Registerseite 1) ********************************
TRISB          equ    6              ;Richtungsregister PortB


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


;******************************** Bits in Registern der Registerbank 1 *************************
RP0            equ    5              ;Seitenauswahlbit im Statuswort-Register


;******************************** Portbelegung *************************************************
;Port B
LCD_DATA       equ    PORTB
LCD_DATA_TRIS  equ    TRISB
LCD_CTRL       equ    PORTB
LCD_CTRL_TRIS  equ    TRISB

LCD_RS         equ    1
LCD_RW         equ    2
LCD_E          equ    3


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


;******************************** Konfigurations-Bits ******************************************
_lp_osc        equ    h'3FFC'
_xt_osc        equ    h'3FFD'
_hs_osc        equ    h'3FFE'
_rc_osc        equ    h'3FFF'

_wdt_off       equ    h'3FFB'
```

```
_wdt_on         equ     h'3FFF'

_pwrt_off       equ     h'3FFF'
_pwrt_on        equ     h'3FF7'

_cp_off         equ     h'3FFF'
_cp_on          equ     h'000F'


        __config        _hs_osc & _wdt_off & _pwrt_off & _cp_off


                ORG     0x000
                goto    Beginn
                ORG     0x004
                goto    ISR


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


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

;*************************************************************************************************
;** Initialisierung des Prozessor:                                                              **
;**     + Ports: Port A: (unbenutzt)                                                            **
;**              Port B: Ausgänge                                                               **
;*************************************************************************************************
INIT            bsf     STAT,RP0                ;Registerseite 1
                movlw   b'00000000'             ;Port B als Ausgang definieren
                movwf   TRISB
                bcf     STAT,RP0                ;Registerseite 0

                return


;******************************** LCD Routinen *******************************************

;*************************************************************************************************
;** LCDINIT                                                                                     **
;**                                                                                             **
;** Aufgabe:                                                                                    **
;**     Dieses Unterprogramm initialisiert das LC-Displays                                      **
;**                                                                                             **
;** Vorgehensweise:                                                                             **
;**     + Die Steuerleitungen des LC-Displays (E, RS, R/W) auf Low legen                        **
;**     + 15 ms warten                                                                          **
;**     + Befehl: 8-bit-Interface an LCD                                                        **
;**     + 4,1 ms warten                                                                         **
;**     + Wiederholung des Befehls: 8-bit-Interface an LCD                                      **
;**     + 0,1 ms warten                                                                         **
;**     + Wiederholung des Befehls: 8-bit-Interface an LCD                                      **
;**     + Warten, bis LCD für weitere Anweisungen bereit ist                                    **
;**     + Befehl: 4-bit-Interface an LCD                                                        **
;**       Achtung: ab nun müssen Befehle an das LCD mit dem Unterprogramm LCDBEFEHL erfolgen    **
;**     + Befehl: Display löschen und Cursor home                                               **
;**     + Befehl: 4 Bit, 2 Zeilen, 5x7                                                          **
;**     + Befehl: Display aus, Cursor aus, Blinken aus                                          **
;**     + Befehl: Shift aus                                                                     **
;**     + Befehl: Display ein                                                                   **
;*************************************************************************************************
LCDINIT         bcf     LCD_CTRL,LCD_E          ;Alle Control-Lines = Low
                bcf     LCD_CTRL,LCD_RS
                bcf     LCD_CTRL,LCD_RW

                movlw   .150                    ;15 ms warten
                call    VERZ100US

                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherw. Nibble löschen
                movlw   0x30                    ;Befehl für 8-Bit-Interface
                iorwf   LCD_DATA,f              ;Befehl an LCD
                bsf     LCD_CTRL,LCD_E          ;  (durch toggeln der LCD-Enable-Leitung)
                bcf     LCD_CTRL,LCD_E

                movlw   .41                     ;4.1 ms warten
                call    VERZ100US
```

```
                bsf     LCD_CTRL,LCD_E          ;Befehl (8-bit-Interface) wiederholen
                bcf     LCD_CTRL,LCD_E          ;  (durch toggeln der LCD-Enable-Leitung)

                movlw   .1                      ;100us warten
                call    VERZ100US

                bsf     LCD_CTRL,LCD_E          ;Befehl (8-bit-Interface) wiederholen
                bcf     LCD_CTRL,LCD_E          ;  (durch toggeln der LCD-Enable-Leitung)

                call    LCDBUSY                 ;Warten bis LCD bereit

                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherw. Nibble löschen
                movlw   0x20                    ;Befehl für 4-Bit Interface
                iorwf   LCD_DATA,f              ;Befehl an LCD
                bsf     LCD_CTRL,LCD_E          ;  (durch toggeln der LCD-Enable-Leitung)
                bcf     LCD_CTRL,LCD_E

                movlw   0x01                    ;Display löschen und Cursor home
                call    LCDBEFEHL               ;Befehl an LCD

                movlw   0x28                    ;4 Bit, 2 Zeilen, 5x7
                call    LCDBEFEHL               ;Befehl an LCD

                movlw   0x08                    ;Display aus, Cursor aus, Blinken aus
                call    LCDBEFEHL               ;Befehl an LCD

                movlw   0x06                    ;Shift aus
                call    LCDBEFEHL               ;Befehl an LCD

                movlw   0x0C                    ;Display ein
                call    LCDBEFEHL               ;Befehl an LCD

                return


;*************************************************************************************************
;** LCDBEFEHL                                                                                   **
;**                                                                                             **
;** Aufgabe:                                                                                    **
;**     Dieses Unterprogramm sendet einen Befehl nibbleweise an das LC-Display.                 **
;**                                                                                             **
;** Vorgehensweise:                                                                             **
;**     + Der Befehl, welcher an das LC-Display übertragen wird, befindet sich als 8-Bit-       **
;**       Wert im Arbeitsregister (w-Register). Diesen zunächst im temporären Register          **
;**       TEMP1 zwischenspeichern                                                               **
;**     + Warten bis das LC-Display für eine Befehl bereit ist. (Dazu ist das Unterprogramm     **
;**       LCDBUSY zuständig)                                                                    **
;**     + für die Kommunikation mit dem LC-Display steht "nur" ein 4-Bit-Datenbus zur Ver-      **
;**       fügung. Der 8-bit-Befehl wird daher nibbleweise an das LC-Display übertragen.         **
;**       Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei            **
;**       dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die           **
;**       restlichen vier Portpins dürfen nicht verändert werden.                               **
;**     + Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung            **
;**       handelt.                                                                              **
;**     + Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.       **
;**     + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-    **
;**       Display liest nun das höherwertigere Befehlsnibble ein)                               **
;**     + Nun das niederwertigere Befehlsnibble auf den 4-Bit-Datenbus legen. Auch hier         **
;**       dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die           **
;**       restlichen vier Portpins dürfen nicht verändert werden.                               **
;**     + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC-    **
;**       Display liest nun das niederwertigere Befehlsnibble ein)                              **
;**                                                                                             **
;** Anmerkungen:                                                                                **
;**     + Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCDZEICHEN nur dadurch,      **
;**       dass hier die Register-Select-Steuerleitung gelöscht ist, da mit diesem Unter-        **
;**       programm Befehle an das LC-Display übertragen werden.                                 **
;**     + Das temporäre Register TEMP1 dient hier nur als Hilfsregister. Es wird hier nur       **
;**       kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet        **
;**       werden.                                                                               **
;*************************************************************************************************
LCDBEFEHL       movwf   TEMP1                   ;Auszugebendes Zeichen (befindet sich im w-
                                                ;  Register) in TEMP1 zwischenspeichern
                call    LCDBUSY                 ;Warten bis LCD bereit ist
                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherwertiges Nibble löschen
                movf    TEMP1,w
                andlw   0xF0                    ;Höherwertiges Nibble ausmaskieren
```

```
                iorwf   LCD_DATA,f              ;High-Nibble an LCD übergeben
                bcf     LCD_CTRL,LCD_RW         ;LCD im Schreiben-Mode
                bcf     LCD_CTRL,LCD_RS         ;LCD im Befehl-Mode
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherwertiges Nibble löschen
                swapf   TEMP1,w
                andlw   0xF0                    ;Niederwert. Nibble ausmaskieren
                iorwf   LCD_DATA,f              ;Low-Nibble an LCD übergeben
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                return


;*****************************************************************************************
;** LCDZEICHEN                                                                           **
;**                                                                                      **
;** Aufgabe:                                                                             **
;**    Diese Unterprogramm sendet ein Zeichen, welches am LC-Display ausgegeben wird,    **
;**    nibbleweise an das LC-Display.                                                    **
;**                                                                                      **
;** Vorgehensweise:                                                                      **
;**    + Das Zeichen, welches am LC-Display ausgegeben werden soll, befindet sich als 8-Bit- **
;**      Wert im Arbeitsregister (w-Register). Diesen Wert zunächst im temporären      **
;**      Register TEMP1 zwischenspeichern                                              **
;**    + Warten bis das LC-Display für ein Zeichen bereit ist. (Dazu ist das Unterprogramm **
;**      LCDBUSY zuständig)                                                            **
;**    + für die Kommunikation mit dem LC-Display steht "nur" ein 4-bit-Datenbus zur ver- **
;**      fügung. Der 8-bit-Wert wird daher nibbleweise an das LC-Display übertragen.   **
;**      Zuerst wird das höherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei    **
;**      dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die   **
;**      restlichen vier Portpins dürfen nicht verändert werden.                       **
;**    + Read-Write-Steuerleitung löschen, da es sich hier um eine Schreibanweisung    **
;**      handelt.                                                                      **
;**    + Register-Select-Steuerleitung setzen, da es sich hier um eine Zeichen handelt, **
;**      welches am LC-Display ausgegeben werden soll.                                 **
;**    + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC- **
;**      Display liest nun das höherwertigere Zeichennibble ein)                       **
;**    + Nun das niederwertigere Zeichennibble auf den 4-Bit-Datenbus legen. Auch hier **
;**      dürfen nur die für den Datenbus definierten Leitungen verändert werden. Die   **
;**      restlichen vier Portpins dürfen nicht verändert werden.                       **
;**    + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC- **
;**      Display liest nun das niederwertigere Zeichennibble ein).                     **
;**                                                                                      **
;** Anmerkung:                                                                           **
;**    + Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCDBEFEHL nur dadurch, **
;**      dass hier die Register-Select-Steuerleitung gesetzt ist, da mit diesem Unter- **
;**      programm am LC-Display auszugebende Zeichen an das LC-Display übertragen werden. **
;**    + Das temporäre Register TEMP1 dient hier nur als Hilfsregister. Es wird hier nur **
;**      kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet **
;**      werden.                                                                       **
;*****************************************************************************************
LCDZEICHEN      movwf   TEMP1                   ;Auszugebendes Zeichen (befindet sich im
                                                ;w-Register) in TEMP1 zwischenspeichern
                call    LCDBUSY                 ;Warten bis LCD bereit ist
                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherwertiges Nibble löschen
                movf    TEMP1,w
                andlw   0xF0                    ;Höherwertiges Nibble ausmaskieren
                iorwf   LCD_DATA,f              ;High-Nibble an LCD übergeben
                bcf     LCD_CTRL,LCD_RW         ;LCD im Schreiben-Mode
                bsf     LCD_CTRL,LCD_RS         ;LCD im Daten-Mode
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                movlw   0x0F
                andwf   LCD_DATA,f              ;Höherwertiges Nibble löschen
                swapf   TEMP1,w
                andlw   0xF0                    ;Niederwert. Nibble ausmaskieren
                iorwf   LCD_DATA,f              ;Low-Nibble an LCD übergeben
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                return


;*****************************************************************************************
;** LCDBUSY                                                                              **
;**                                                                                      **
;** Aufgabe:                                                                             **
```

```
;**    Diese Unterprogramm prüft das Busy-Flag des LC-Display, und verlässt es erst wenn     **
;**    das Busy-Flag low ist, also wenn das LC-Display bereit für eine Befehl oder für      **
;**    ein auszugebendes Zeichen ist.                                                       **
;**                                                                                         **
;** Vorgehensweise:                                                                         **
;**    + Der Datenport muss als Eingang definiert werden, da nun der Zustand (insbesondere  **
;**      der Zustand das Busy-Flag) des LC-Display gelesen wird. Dazu muss aber in die      **
;**      Registerbank 1 des PIC gewechselt werden und anschließend wieder zurück zur        **
;**      Registerbank 0.                                                                    **
;**    + Register-Select-Steuerleitung löschen, da es sich hier um einen Befehl handelt.     **
;**    + Read-Write-Steuerleitung setzen, da es sich hier um eine Leseanweisung handelt.    **
;**    + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das LC- **
;**      Display gibt nun das höherwertigere Nibble auf den 4-Bit-Datenbus.)                **
;**    + Den 4-Bit-Wert vom Datenbus einlesen und im Hilfsregister TEMP2 sichern.           **
;**    + Eine weiter fallende Flanke auf der Enable-Steuerleitung durch Toggeln er-         **
;**      zeugen. (Das LC-Display gibt nun das niederwertigere Nibble auf den 4-Bit-Datenbus) **
;**    + Den 4-Bit-Wert vom Datenbus einlesen und mit dem Hilfsregister TEMP2 zu einem 8-   **
;**      Bit-Wert verschmelzen.                                                             **
;**    + Das Busy-Flag (Bit 7 von TEMP2) überprüfen. Ist dieses Flag gesetzt, die soeben    **
;**      durchgeführten Schritte so oft wiederholen bis das LC-Display das Busy-Flag        **
;**      löscht.                                                                            **
;**    + Ist das Busy-Flag low, (also gelöscht) die Read-Write-Steuerleitung wieder         **
;**      zurücksetzen und den Datenport wieder als Ausgang definieren. Dazu muss aber       **
;**      wieder in die Registerbank 1 des PIC gewechselt werden und anschließend wieder     **
;**      zurück zur Registerbank 0.                                                         **
;**                                                                                         **
;** Anmerkung:                                                                              **
;**    Das temporäre Register TEMP2 dient hier nur als Hilfsregister. Es wird hier nur      **
;**    kurzzeitig verwendet und kann daher auch in anderen Unterprogrammen verwendet werden **
;*********************************************************************************************
LCDBUSY         bsf     STAT,RP0                ;Registerbank 1
                movlw   0xF0                    ;Höherwertigen HalbPort
                iorwf   LCD_DATA_TRIS,w         ;  als Eingang definieren
                movwf   LCD_DATA_TRIS
                bcf     STAT,RP0                ;Registerbank 0

                bcf     LCD_CTRL,LCD_RS         ;LCD im Befehls-Mode
                bsf     LCD_CTRL,LCD_RW         ;LCD im Lesen-Mode
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                movf    LCD_DATA,w
                andlw   0xF0                    ;Niederw.Nibble ausmaskieren
                movwf   TEMP2
                bsf     LCD_CTRL,LCD_E          ;Enable (LCD)
                bcf     LCD_CTRL,LCD_E          ;  toggeln
                swapf   LCD_DATA,w              ;Niederwertiges Nibble Busy-Flag
                andlw   0x0F                    ;Höherwertiges Nibble ausmaskieren
                iorwf   TEMP2,w                 ;Nibbles verbinden
                btfsc   TEMP2,7                 ;Busy-Flag prüfen (High=busy)
                goto    LCDBUSY                 ;Dieses UP so oft abarbeiten, bis das Busy-Flag
                                                ;  low ist

                bcf     LCD_CTRL,LCD_RW         ;Busy = low: LCD im Schreiben-Mode
                bsf     STAT,RP0                ;Registerbank 1
                movlw   0x0F                    ;Höherwertigen HalbPort
                andwf   LCD_DATA,w              ;  als Ausgang definieren
                movwf   LCD_DATA_TRIS
                bcf     STAT,RP0                ;Registerbank 0
                return


;*********************************************************************************************
;** Warteschleifen                                                                          **
;**                                                                                         **
;** Aufgabe:                                                                                **
;**    Dieses Unterprogramm erzeugt eine Zeitverzögerung. Der Übergabeparameter (im w-      **
;**    Register) gibt dabei an, wie oft eine Zeitverzögerung von 100us erfolgen soll. Der   **
;**    mögliche Zeitverzögerungsbereich liegt hier daher im Bereich von 100us und 25,5ms    **
;**    (=255 x 100us).                                                                      **
;**                                                                                         **
;** Anmerkungen:                                                                            **
;**    + Die temporären Register TEMP1 und TEMP2 dienen hier nur als Übergabe- bzw. als     **
;**      Hilfsregister. Diese Register werden hier nur kurzzeitig verwendet und können      **
;**      daher auch in anderen Unterprogrammen verwendet werden.                            **
;**    + Das Hilfsregister TEMP2 muss je nach verwendetem Takt mit unterschiedlichen Werten **
;**      geladen werden:                                                                    **
;**                      Takt        Wert von TEMP2                                       **
;**                    ------------------------------                                       **
;**                      4 MHz       25                                                   **
```

```
;**                    12 MHz        75                                                  **
;**                    20 MHz       125                                                  **
;*****************************************************************************************
VERZ100US
              movwf   TEMP1          ;Übergabeparameter in TEMP1 sichern

VERZSCHL1
              movlw   .25            ;Bei Verwendung eines 4-MHz-Taktes
;             movlw   .75            ;Bei Verwendung eines 12-MHz-Taktes
;             movlw   .125           ;Bei Verwendung eines 20-MHz-Taktes
              movwf   TEMP2
VERZSCHL2     nop
              decfsz  TEMP2,f
              goto    VERZSCHL2
              decfsz  TEMP1,f
              goto    VERZSCHL1

              return


;*****************************************************************************************
;** AUSGABE                                                                             **
;**                                                                                     **
;** Aufgabe:                                                                            **
;**    Dieses Unterprogramm gibt in der ersten Zeile den Text "Hallo..." und in der zweiten  **
;**    Zeile den Text "...wie geht's" am LC-Display aus.                                **
;**                                                                                     **
;** Vorgehensweise:                                                                     **
;**    + Display löschen (dabei wird auch gleichzeitig der Cursor auf die erste Stelle am    **
;**      LC-Display gesetzt)                                                            **
;**    + Den Text der ersten Zeile Zeichen für Zeichen mit dem Unterprogramm LCDZEICHEN am   **
;**      LC-Display ausgeben.                                                           **
;**    + Den Cursor auf die zweite Zeile setzen. Dies erfolgt mit dem Befehl 'C0'            **
;**    + Den Text der zweiten Zeile Zeichen für Zeichen mit dem Unterprogramm LCDZEICHEN     **
;**      am LC-Display ausgeben.                                                        **
;*****************************************************************************************
AUSGABE       movlw   0x01                ;Display löschen
              call    LCDBEFEHL

              movlw   'H'
              call    LCDZEICHEN
              movlw   'a'
              call    LCDZEICHEN
              movlw   'l'
              call    LCDZEICHEN
              movlw   'l'
              call    LCDZEICHEN
              movlw   'o'
              call    LCDZEICHEN
              movlw   '.'
              call    LCDZEICHEN
              movlw   '.'
              call    LCDZEICHEN
              movlw   '.'
              call    LCDZEICHEN

              movlw   0xC0                ;2.Zeile
              call    LCDBEFEHL

              movlw   '.'
              call    LCDZEICHEN
              movlw   '.'
              call    LCDZEICHEN
              movlw   '.'
              call    LCDZEICHEN
              movlw   'w'
              call    LCDZEICHEN
              movlw   'i'
              call    LCDZEICHEN
              movlw   'e'
              call    LCDZEICHEN
              movlw   20                  ;Leerzeichen
              call    LCDZEICHEN
              movlw   'g'
              call    LCDZEICHEN
              movlw   'e'
              call    LCDZEICHEN
              movlw   'h'
              call    LCDZEICHEN
```

```
                movlw   't'
                call    LCDZEICHEN
                movlw   27                      ;Hochkomma (')
                call    LCDZEICHEN
                movlw   's'
                call    LCDZEICHEN

                return


;****************************** Hauptprogramm ***********************************************

;*************************************************************************************************
;** Aufgaben des Hauptprogramms:                                                                 **
;** + Controller initialisieren (Unterprogramm INIT)                                             **
;** + LCD initialisieren (Unterprogramm LCDINIT)                                                 **
;** + In einer Endlosschleife ununterbrochen einen beliebigen Text am LC-Display ausgeben    **
;**   und 15ms warten                                                                            **
;*************************************************************************************************
Beginn          call    INIT                    ;Controller initialisieren
                call    LCDINIT                 ;Display initialisieren

Schleife
                call    AUSGABE                 ;Einen Text am LC-Display ausgeben

                movlw   .150                    ;  und anschließend 15ms warten
                call    VERZ100US

                goto    Schleife

                end
```

6.2.2.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 LC-Displays (Unterprogramm LCDINIT) nur mehr aus einer Endlosschleife. In dieser Endlosschleife wird ständig das Unterprogramm AUSGABE aufgerufen, und anschließend eine Zeitverzögerung von 15 ms.

Das Unterprogramm INIT dient zur Initialisierung des Controllers. Hier wird, in diesem Beispiel, nur der Port B als Ausgang konfiguriert.
Das Unterprogramm LCDINIT initialisiert das LC-Display (siehe Abschnitt 4.3.1).

Nach der Initialisierung des LC-Displays (Unterprogramm LCDINIT) ist das LC-Display bereit Zeichen oder Werte anzuzeigen. Die Ausgabe der anzuzeigenden Zeichen übernimmt hier das Unterprogramm AUSGABE. Zuerst wird das Display gelöscht. Dazu wird die Anweisung 0x01 mit dem Unterprogramm LCDBEFEHL an das Display übergeben. (0x01 bedeutet Display löschen, siehe auch *Tabelle 1 Display-Befehle* im *Abschnitt 3.1 Befehle zur Ansteuerung des LC-Display*). Durch das Löschen des Displays befindet sich auch gleichzeitig der Cursor an der ersten Stelle der ersten Zeile. Nun können die am Display darzustellenden Zeichen mit dem Unterprogramm LCDZEICHEN nacheinander an das Display gesendet werden. Für die Anzeige des Buchstaben H auf dem Display sind die beiden Anweisungen

```
movlw     'H'
call      LCDZEICHEN
```

notwendig.

Mit der Befehlsfolge

```
movlw     0xC0
call      LCDBEFEHL
```

gelangt der Cursor auf die erste Position der zweiten Zeile. Nun können die Zeichen an das Display übergeben werden welche in der zweiten Zeile erscheinen sollen.

6.3. Softwarebeispiel (in C mit CC5x)

6.3.1.Listing

```
/*******************************************************************************/
/* Demonstrationsbeispiel zur Ansteuerung eines alphanumerischen LC-Display in C (CC5X)  */
/*                                                                             */
/* Entwickler: Buchgeher Stefan                                                */
/* Entwicklungsbeginn der Software: 12. Juli 2004                              */
/* Funktionsfähig seit: 13. Juli 2004                                          */
/* Letzte Bearbeitung: 9. September 2004                                       */
/*******************************************************************************/

/****************** Include-Dateien ****************************************************/
#include <INLINE.H>                 // Fuer Assembler-Anweisungen


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


/****************** Externe Register ***************************************************/


/****************** Bits in den externen Registern *************************************/


/****************** Portbelegung *******************************************************/
/* Port B */
#pragma char LCD_DATA         @ PORTB
#pragma char LCD_DATA_TRIS    @ TRISB
#pragma char LCD_CTRL         @ PORTB
#pragma char LCD_CTRL_TRIS    @ TRISB

bit LCD_RS                    @ LCD_CTRL.1;
bit    LCD_RW                 @ LCD_CTRL.2;
bit LCD_E                     @ LCD_CTRL.3;


/****************** Konstanten *********************************************************/
#define        LCD_CLR               0x01    // Display loeschen
#define        LCD_ON_C              0x0E    // Cursor home
#define        LCD_FUNCTION_SET      0x28    // 4 Bit, 2 Zeilen, 5x7
#define        LCD_DISP_ON           0x0C    // Display Ein
#define        LCD_DISP_OFF          0x08    // Display Aus
#define        LCD_ENTRY_INC         0x06


/****************** Konfigurations *****************************************************/
#pragma config |= 0b.11111111111010
/*                    ++++++-+++------- Bit 13-8, 6-4: (CP) Code Protection
                             |   ||||     ->   1 : code protection off
                             |   ||||          0 : Programm memory is code protected
                             +---------- Bit 7 (DP) Data Memory Code Protection
                                  ||||     ->  1 : code protection off
                                  ||||          0: Programm memory is code protected
                                  +------ Bit 3 (PWRTE): 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 (FOSC1:FOSC0): Oszillator Selection
                                           00 : LP Oszillator
                                           01 : XT Oszillator
                                     ->    10 : HS Oszillator
                                           11 : RC Oszillator
*/


/****************** Funktionsprototypen ************************************************/
void INIT(void);
void AUSGABE(void);

void LCD_INIT(void);
void LCD_ZEICHEN(unsigned char zeichen);
void LCD_BEFEHL(unsigned char befehl);
void LCD_BUSY(void);
void LCD_STRING(const char *pString);
void VERZ100US(unsigned char VerzZeit);


/****************** Hauptprogramm ******************************************************/

/***************************************************************************************/
/* Aufgaben des Hauptprogramms:                                                        */
/*      + Controller initialisieren (Unterprogramm INIT)                               */
/*      + LC-Display initialisieren (Unterprogramm LCD_INIT)                           */
/*      + Alle 15ms das Unterprogramm AUSGABE aufrufen                                 */
/***************************************************************************************/
void main(void)
{
       INIT();                          // PIC-Controller initialisieren
       LCD_INIT();                      // LC-Display initialisieren

       while(1)
       {
              AUSGABE();                // Text (und/oder Werte) am Display ausgeben
              VERZ100US(150);           // 15 ms warten
       }
}


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

/***************************************************************************************/
/* Initialisierung des Prozessor:                                                      */
/*   + Port B als Ausgang definieren                                                   */
/***************************************************************************************/
void INIT(void)
{
       TRISB = 0;                       // Port B als Ausgang definieren
}


/***************************************************************************************/
/* AUSGABE                                                                             */
/*                                                                                     */
/* Aufgabe:                                                                            */
/*   Dieses Unterprogramm gibt in der ersten Zeile den Text "Hallo ..." und in der     */
/*   zweiten Zeile den Text "...wie geht's?" am LC-Display aus.                        */
/*                                                                                     */
/* Vorgehensweise:                                                                     */
/*   + Display loeschen (dabei wird auch gleichzeitig der Cursor auf die erste Stelle am */
/*     LC-Display gesetzt)                                                             */
/*   + Den Text der ersten Zeile Zeichen für Zeichen mit dem Unterprogramm LCD_ZEICHEN  */
/*     am LC-Display ausgeben.                                                         */
/*   + Den Cursor auf die zweite Zeile setzen. Dies erfolgt mit dem Befehl 'C0'        */
/*   + Den Text der zweiten Zeile mit dem Unterprogramm LCD_STRING am LC-Display       */
/*     ausgeben.                                                                       */
/***************************************************************************************/
void AUSGABE(void)
{
       LCD_BEFEHL(LCD_CLR);             // Display loeschen

       LCD_ZEICHEN('H');
       LCD_ZEICHEN('a');
       LCD_ZEICHEN('l');
       LCD_ZEICHEN('l');
```

```
        LCD_ZEICHEN('o');
        LCD_ZEICHEN(0x20);                  // Leerzeichen
        LCD_ZEICHEN('.');
        LCD_ZEICHEN('.');
        LCD_ZEICHEN('.');

        LCD_BEFEHL(0xC0);                   // 2.Zeile

        LCD_STRING("... wie geht's?");
}


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

/*****************************************************************************************/
/* LCD_INIT                                                                              */
/*                                                                                       */
/* Aufgabe:                                                                              */
/*   Dieses Unterprogramm initialisiert das LC-Displays                                  */
/*                                                                                       */
/* Vorgehensweise:                                                                       */
/*   + Die Steuerleitungen des LC-Displays (E, RS, R/W) auf Low legen                    */
/*   + 15 ms warten                                                                      */
/*   + Befehl: 8-bit-Interface an LCD                                                    */
/*   + 4,1 ms warten                                                                     */
/*   + Wiederholung des Befehls: 8-bit-Interface an LCD                                  */
/*   + 0,1 ms warten                                                                     */
/*   + Wiederholung des Befehls: 8-bit-Interface an LCD                                  */
/*   + Warten, bis LCD fuer weitere Anweisungen bereit ist                               */
/*   + Befehl: 4-bit-Interface an LCD                                                    */
/*     Achtung: ab nun muessen Befehle an das LCD mit dem Unterprogramm LCD_BEFEHL       */
/*     erfolgen                                                                          */
/*   + Befehl: Display loeschen und Cursor home                                          */
/*   + Befehl: 4 Bit, 2 Zeilen, 5x7                                                      */
/*   + Befehl: Display aus, Cursor aus, Blinken aus                                      */
/*   + Befehl: Shift aus                                                                 */
/*   + Befehl: Display ein                                                               */
/*****************************************************************************************/
void LCD_INIT(void)
{
        LCD_RS = 0;                         // Alle LCD-Steuerleitungen LOW
        LCD_RW = 0;
        LCD_E = 0;
        VERZ100US(150);

        LCD_DATA = LCD_DATA & 0x0F;         // Hoeherwertigeres Nibble loeschen
        LCD_DATA = LCD_DATA | 0x30;         // Befehl fuer 8-Bit-Interface

        LCD_E = 1;                          // Befehl an LCD
        LCD_E = 0;                          //   (durch toggeln der LCD-Enable-Leitung)
        VERZ100US(41);                      // 4.1 ms warten

        LCD_E = 1;                          // Befehl (8-Bit-Interface) wiederholen
        LCD_E = 0;                          //   (durch toggeln der LCD-Enable-Leitung)
        VERZ100US(1);                       // 100 us warten

        LCD_E = 1;                          // Befehl (8-Bit-Interface) wiederholen
        LCD_E = 0;                          //   (durch toggeln der LCD-Enable-Leitung)

        LCD_BUSY();                         // warten bis LCD bereit

        LCD_DATA = LCD_DATA & 0x0F;         // Hoeherwertigeres Nibble loeschen
        LCD_DATA = LCD_DATA | 0x20;         // Befehl fuer 4-Bit-Interface

        LCD_E = 1;                          // Befehl an LCD
        LCD_E = 0;                          //   (durch toggeln der LCD-Enable-Leitung)

        LCD_BEFEHL(LCD_CLR);                // Display loeschen und Cursor home
        LCD_BEFEHL(LCD_FUNCTION_SET);       // 4 Bit, 2 Zeilem, 5x7
        LCD_BEFEHL(LCD_DISP_OFF);           // Display aus, Cursor aus, Blinken aus
        LCD_BEFEHL(LCD_ENTRY_INC);          // Shift aus
        LCD_BEFEHL(LCD_DISP_ON);            // Display ein
}


/*****************************************************************************************/
/* LCD_BEFEHL:                                                                           */
/*                                                                                       */
/* Aufgabe:                                                                              */
```

```
/*   Dieses Unterprogramm sendet einen Befehl nibbleweise an das LC-Display.             */
/*                                                                                       */
/* Vorgehensweise:                                                                       */
/*   + Warten bis das LC-Display für eine Befehl bereit ist. (Dazu ist das Unterprogramm */
/*     LCD_BUSY zustaendig)                                                              */
/*   + Fuer die Kommunikation mit dem LC-Display steht "nur" ein 4-Bit-Datenbus zur Ver- */
/*     fuegung. Der 8-bit-Befehl wird daher nibbleweise an das LC-Display uebertragen.   */
/*     Zuerst wird das hoeherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei       */
/*     duerfen nur die fuer den Datenbus definierten Leitungen veraendert werden. Die    */
/*     restlichen vier Portpins duerfen nicht veraendert werden.                         */
/*   + Read-Write-Steuerleitung loeschen, da es sich hier um eine Schreibanweisung       */
/*     handelt.                                                                          */
/*   + Register-Select-Steuerleitung loeschen, da es sich hier um einen Befehl handelt.  */
/*   + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das    */
/*     LC-Display liest nun das hoeherwertigere Befehlsnibble ein)                       */
/*   + Nun das niederwertigere Befehlsnibble auf den 4-Bit-Datenbus legen. Auch hier     */
/*     duerfen nur die für den Datenbus definierten Leitungen veraendert werden. Die     */
/*     restlichen vier Portpins duerfen nicht veraendert werden.                         */
/*   + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das    */
/*     LC-Display liest nun das niederwertigere Befehlsnibble ein)                       */
/*                                                                                       */
/* Anmerkung:                                                                            */
/*   Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCD_ZEICHEN nur dadurch,  */
/*   dass hier die Register-Select-Steuerleitung geloescht ist, da mit diesem Unter-     */
/*   programm Befehle an das LC-Display uebertragen werden.                              */
/*****************************************************************************************/
void LCD_BEFEHL(unsigned char befehl)
{
       unsigned char temp;


       LCD_BUSY();                          // Warten bis LCD bereit ist

       LCD_DATA = LCD_DATA & 0x0F;          // Hoeherwertigeres Nibble loeschen
       temp = befehl & 0xF0;                // Hoeherwertiges Nibble ausmaskieren
       LCD_DATA = LCD_DATA | temp;          // High-Nibble an LCD uebergeben

       LCD_RW = 0;                          // LCD im Schreiben-Mode
       LCD_RS = 0;                          // LCD im Befehl-Mode
       LCD_E = 1;                           // Enable (LCD)
       LCD_E = 0;

       LCD_DATA = LCD_DATA & 0x0F;          // Hoeherwertigeres Nibble loeschen
       SWAPF(befehl,1);
       temp = befehl & 0xF0;                // Niederwert. Nibble ausmaskieren
       LCD_DATA = LCD_DATA | temp;          // High-Nibble an LCD uebergeben

       LCD_E = 1;                           // Enable (LCD)
       LCD_E = 0;                           //   toggeln
}


/*****************************************************************************************/
/* LCD_ZEICHEN:                                                                          */
/*                                                                                       */
/* Aufgabe:                                                                              */
/*   Dieses Unterprogramm sendet ein Zeichen, welches am LC-Display ausgegeben wird,     */
/*   nibbleweise an das LC-Display.                                                      */
/*                                                                                       */
/* Vorgehensweise:                                                                       */
/*   + Warten bis das LC-Display für ein Zeichen bereit ist. (Dazu ist das Unterprogramm */
/*     LCDBUSY zustaendig)                                                               */
/*   + Fuer die Kommunikation mit dem LC-Display steht "nur" ein 4-bit-Datenbus zur ver- */
/*     fuegung. Der 8-bit-Wert wird daher nibbleweise an das LC-Display uebertragen.     */
/*     Zuerst wird das hoeherwertigere Nibble auf den 4-Bit-Datenbus gelegt. Dabei       */
/*     duerfen nur die fuer den Datenbus definierten Leitungen veraendert werden. Die    */
/*     restlichen vier Portpins duerfen nicht veraendert werden.                         */
/*   + Read-Write-Steuerleitung loeschen, da es sich hier um eine Schreibanweisung       */
/*     handelt.                                                                          */
/*   + Register-Select-Steuerleitung setzen, da es sich hier um eine Zeichen handelt,    */
/*     welches am LC-Display ausgegeben werden soll.                                     */
/*   + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das    */
/*     LC-Display liest nun das hoeherwertigere Zeichennibble ein)                       */
/*   + Nun das niederwertigere Zeichennibble auf den 4-Bit-Datenbus legen. Auch hier     */
/*     duerfen nur die fuer den Datenbus definierten Leitungen veraendert werden. Die    */
/*     restlichen vier Portpins duerfen nicht veraendert werden.                         */
/*   + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das    */
/*     LC-Display liest nun das niederwertigere Zeichennibble ein).                      */
/*                                                                                       */
```

```
/* Anmerkung:                                                                    */
/*   Dieses Unterprogramm unterscheidet sich zum Unterprogramm LCD_BEFEHL nur dadurch,   */
/*   dass hier die Register-Select-Steuerleitung gesetzt ist, da mit diesem Unter-      */
/*   programm am LC-Display auszugebende Zeichen an das LC-Display uebertragen werden.   */
/*********************************************************************************/
void LCD_ZEICHEN(unsigned char zeichen)
{
       unsigned char temp;


       LCD_BUSY();                          // Warten bis LCD bereit ist

       LCD_DATA = LCD_DATA & 0x0F;          // Hoeherwertigeres Nibble loeschen
       temp = zeichen & 0xF0;               // Hoeherwertiges Nibble ausmaskieren
       LCD_DATA = LCD_DATA | temp;          // High-Nibble an LCD uebergeben

       LCD_RW = 0;                          // LCD im Schreiben-Mode
       LCD_RS = 1;                          // LCD im Daten-Mode
       LCD_E = 1;                           // Enable (LCD)
       LCD_E = 0;                           //   toggeln

       LCD_DATA = LCD_DATA & 0x0F;          // Hoeherwertigeres Nibble loeschen
       SWAPF(zeichen,1);
       temp = zeichen & 0xF0;               // Niederwert. Nibble ausmaskieren
       LCD_DATA = LCD_DATA | temp;          // High-Nibble an LCD uebergeben

       LCD_E = 1;                           // Enable (LCD)
       LCD_E = 0;                           //   toggeln
}


/*********************************************************************************/
/* LCD_BUSY:                                                                     */
/*                                                                               */
/* Aufgabe:                                                                      */
/*   Dieses Unterprogramm prueft das Busy-Flag des LC-Display, und verlaesst es erst     */
/*   wenn das Busy-Flag low ist, also wenn das LC-Display bereit fuer eine Befehl oder   */
/*   fuer ein auszugebendes Zeichen ist.                                         */
/*                                                                               */
/* Vorgehensweise:                                                               */
/*   + Der Datenport muss als Eingang definiert werden, da nun der Zustand (insbesondere */
/*     der Zustand das Busy-Flag) des LC-Display gelesen wird.                   */
/*   + Register-Select-Steuerleitung loeschen, da es sich hier um einen Befehl handelt.  */
/*   + Read-Write-Steuerleitung setzen, da es sich hier um eine Leseanweisung handelt.   */
/*   + Eine fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen. (Das    */
/*     LC-Display gibt nun das hoeherwertigere Nibble auf den 4-Bit-Datenbus.)   */
/*   + Den 4-Bit-Wert vom Datenbus einlesen und im Hilfsregister temp1 sichern.          */
/*   + Eine weiter fallende Flanke auf der Enable-Steuerleitung durch Toggeln erzeugen.  */
/*     (Das LC-Display gibt nun das niederwertigere Nibble auf den 4-Bit-Datenbus)       */
/*   + Den 4-Bit-Wert vom Datenbus einlesen und mit dem Hilfsregister temp2 zu einem 8-  */
/*     Bit-Wert verschmelzen.                                                    */
/*   + Das Busy-Flag (Bit 7 von temp2) ueberpruefen. Ist dieses Flag gesetzt, die soeben */
/*     durchgefuehrten Schritte so oft wiederholen bis das LC-Display das Busy-Flag     */
/*     loescht.                                                                  */
/*   + Ist das Busy-Flag low, (also geloescht) die Read-Write-Steuerleitung wieder       */
/*     zuruecksetzen und den Datenport wieder als Ausgang definieren.            */
/*********************************************************************************/
void LCD_BUSY(void)
{
       unsigned char temp1;
       unsigned char temp2;


       do
       {
              LCD_DATA_TRIS = LCD_DATA_TRIS | 0xF0; // Hoeherwertigen HalbPort
                                           // als Eingang definieren

              LCD_RS = 0;                  // LCD im Befehls-Mode
              LCD_RW = 1;                  // LCD im Lesen-Mode
              LCD_E = 1;                   // Enable (LCD)
              LCD_E = 0;                   //   toggeln

              temp1 = LCD_DATA;
              temp1 = temp1 & 0xF0;        // Niederw.Nibble ausmaskieren

              LCD_E = 1;                   // Enable (LCD)
              LCD_E = 0;                   //   toggeln
```

```
                temp2 = LCD_DATA;
                SWAPF(temp2,1);                 // Niederwertiges Nibble Busy-Flag
                temp2 = temp2 & 0x0F;           // Hoeherwertiges Nibble ausmaskieren
                temp2 = temp1 | temp2;          // Nibbles verbinden

        } while (temp2.7);

        LCD_RW = 0;                             // Busy = low: LCD im Schreiben-Mode
        LCD_DATA_TRIS = LCD_DATA & 0x0F;        // Höherwertigen HalbPort
}                                               // als Ausgang definieren


/*********************************************************************************************/
/* LCD_STRING:                                                                               */
/*                                                                                           */
/* Aufgabe:                                                                                  */
/*   Dieses Unterprogramm sendet einen String Zeichen fuer Zeichen an das LC-Display.        */
/*                                                                                           */
/* Vorgehensweise:                                                                           */
/*  + Der Zeiger (engl. Pointer) *pString zeigt auf das erste Zeichen der uebergebenen       */
/*    Zeichenkette. Den Inhalt diesee Zeigers in die Variable zeichen kopieren.              */
/*  + Mit einer Schleife den Inhalt von zeichen an das LC-Display uebergeben und die         */
/*    restliche Zeichenkette mit Hilfe des Zeigers Zeichen fuer Zeichen einlesen und am      */
/*    LC-Display ausgeben.                                                                   */
/*                                                                                           */
/* Anmerkung:                                                                                */
/*   Eine Zeichenkette (engl. String) wird in C mit einem so genannten Nullbyte abge-        */
/*   schlossen.                                                                              */
/*********************************************************************************************/
void LCD_STRING(const char *pString)
{
        char zeichen;


        zeichen = *pString;
        while(zeichen > 0)
        {
                LCD_ZEICHEN(zeichen);           // zeichen am LC-Display ausgeben
                pString++;                      // Zeiger um 1 erhoehen, er zeigt nun auf das
                                                // naechste Zeichen der auszugebenden Zeichen-
                                                // kette
                zeichen = *pString;             // Den Inhalt diese Zeigers in die Variable
        }                                       //  zeichen kopieren.
}


/*********************************************************************************************/
/* Warteschleife:                                                                            */
/*                                                                                           */
/* Aufgabe:                                                                                  */
/*   Dieses Unterprogramm erzeugt eine Zeitverzoegerung. Der Uebergabeparameter VerzZeit     */
/*   gibt dabei an, wie oft eine Zeitverzoegerung von 100us erfolgen soll. Der moegliche     */
/*   Zeitverzoegerungsbereich liegt hier daher im Bereich von 100us und 25,5ms.              */
/*********************************************************************************************/
void VERZ100US(unsigned char VerzZeit)
{
        unsigned char temp;


        #asm
        VERZSCHL1
                movlw   .25                     // Bei Verwendung eines 4-MHz-Taktes
//              movlw   .75                     // Bei Verwendung eines 12-MHz-Taktes
//              movlw   .125                    // Bei Verwendung eines 20-MHz-Taktes
                movwf   temp

        VERZSCHL2       nop
                decfsz  temp,f
                goto    VERZSCHL2
                decfsz  VerzZeit,f
                goto    VERZSCHL1
        #endasm
}
```

6.3.2. Anmerkungen zur C-Software

Die Software besteht im Wesentlichen aus einem kurzen Hauptprogramm, die im Abschnitt 5 beschriebenen Unterprogramme und einem weiteren Unterprogramm namens AUSGABE.

Das Hauptprogramm besteht nach der Initialisierung des Controllers (Unterprogramm INIT) und des LC-Displays (Unterprogramm LCD_INIT) nur mehr aus einer Endlosschleife. In dieser Endlosschleife wird ständig das Unterprogramm AUSGABE aufgerufen, und anschließend eine Zeitverzögerung von 15 ms.

Das Unterprogramm INIT dient zur Initialisierung des Controllers. Hier wird, in diesem Beispiel, nur der Port B als Ausgang konfiguriert.
Das Unterprogramm LCD_INIT initialisiert das LC-Display (siehe Abschnitt 5.3.1).

Nach der Initialisierung des LC-Displays (Unterprogramm LCD_INIT) ist das LC-Display bereit Zeichen oder Werte anzuzeigen. Die Ausgabe der anzuzeigenden Zeichen übernimmt hier das Unterprogramm AUSGABE. Zuerst wird das Display gelöscht. Dazu wird die Anweisung 0x01 mit dem Unterprogramm LCD_BEFEHL an das Display übergeben. (0x01 bedeutet Display löschen, siehe auch *Tabelle 1 Display-Befehle* im *Abschnitt 3.1 Befehle zur Ansteuerung des LC-Display*). Durch das Löschen des Displays befindet sich auch gleichzeitig der Cursor an der ersten Stelle der ersten Zeile. Nun können die am Display darzustellenden Zeichen mit dem Unterprogramm LCD_ZEICHEN nacheinander an das Display gesendet werden.

7. Eigene Zeichen am LC-Display definieren

Noch nicht ausgeführt!

8. Fallen und Sonstiges

Achtung, bei Verwendung des Timer –Interrupt:
Das Flag RBPU (Bit 7 von OPTION) muss beim laden des OPTION-Register gesetzt werden/bleiben, wenn das LC-Display am Port B betrieben wird. (d.h. die internen Pull-Up-Widerstände am Port B werden/bleiben deaktiviert)

9. Quellen

- Datenblatt des Hitachi-Controller HD44780.
- http://www.sprut.de/electronic/lcd/index.htm
- http://home.wtal.de/Mischka/LCD.html