







 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
0.1	Inhalt	Datum:

PC-Diagnose-Display

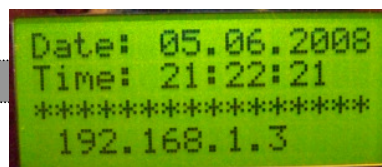
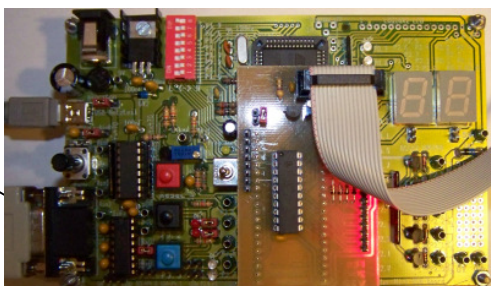
Projektbeschreibung
von Rolf Rahm

Inhaltsverzeichnis und Dateien

-  6_2_1_Bauteile_und_Bestueckung
-  6_2_2_GAL_Programmierung
-  6_2_3_Controller_Programmierung
-  6_2_4_Schaltplan_und_Layout_Target
-  6_2_5_ComMeldung
-  Anhang_Display_Programmierung
-  Anhang_Matrixtastatur_Programmierung
-  Anhang_Serielle_Schnittstelle_8051



V.24




Reichenbach, im Juni 2008

Tel.: 0711/3607-241

Fax: 0711/3607-102

Mail: rahm@fes-es.de

Web: <http://mikrocontroller.rahm-home.de>

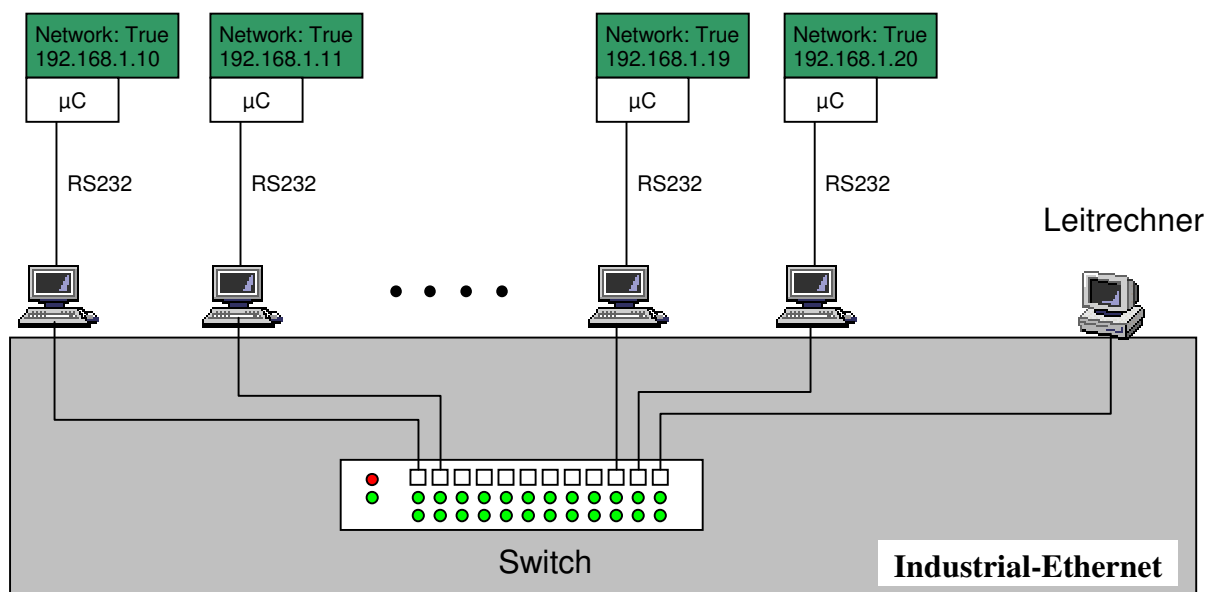
 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
6.2.3.1	Lastenheft	Datum:

In Deinem Betrieb wird zur Steuerung und Regelung verschiedenster Fertigungsprozesse eine Serie von Industrie-PCs eingesetzt. Die Steuersoftware läuft unter Windows XP und stellt den jeweiligen Teilprozess als Bildschirmfüllendes Prozessabbild dar. Die PCs sind über Ethernet mit dem Prozessleitreechner verbunden. Bisher wurde zur Kontrolle der korrekten Netzanbindung das Programm IPCONFIG auf jedem PC ausgeführt. Diese Vorgehensweise hat sich als zu umständlich erwiesen.

Daher soll die Anzeige der Netzwerkverbindung nun auf einem externen LC-Display erfolgen. Davon verspricht sich die Abteilungsleitung folgende Vorteile:

- Ständig aktuelle Infos zur Netzwerkverbindung
- Anzeige kann an leicht einsehbarem Ort angebracht sein
- Weitere Statusinformationen des PCs können angezeigt werden
- Kurze Reaktionszeiten bei Netzwerkproblemen und damit kürzere Anlagenstandzeiten
- Durch intelligente Displays (μ C) ist eine Alarmierung bei Netzwerkausfall unabhängig vom Steuerrechner möglich


Das Schaubild zeigt den prinzipiellen Aufbau des Systems:



Eure Aufgabe ist es auf Basis des Mikrocontroller-Miniboards ein Testsystem aufzubauen, zu programmieren und zu erproben, welches anschließend als Kleinserie aufgebaut werden soll. Dabei sind folgende Vorgaben zu berücksichtigen:

- LC-Display am externen Bus des μ C. Die Anschaltung wird über eine Aufsteckplatine für das Miniboard realisiert.
- Miniboard über RS232 mit Steuer-PC verbunden!
- Die Anzeigesoftware ComMeldung kann von der Hersteller-Website mikrocontroller.rahm-home.de/projekte.htm als Zip-Datei heruntergeladen werden!



 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
6.2.3.2	Anzeigesoftware ComMeldung	Datum:

- **Programmbeschreibung:**

ComMeldung ist eine in Visual Basic 2005/2008 Express erstellte Anwendung, die frei definierbare Texte oder Textbausteine mit Systeminformationen, wie Uhrzeit, Datum, Benutzer, Computername, IP-Adresse usw., über die serielle Schnittstelle überträgt. Die Daten können von einem Mikrocontroller empfangen und auf einem Standard Textdisplay angezeigt werden.

Ein eingebauter Timer sorgt dafür, dass die Anzeigedaten zyklisch aktualisiert werden.

- **Übertragungsformat:** ComMeldung sendet nacheinander folgende Daten:

1. **STX** (Start Of Text) = 02h
2. Mindestens 8 bis maximal 24 ASCII -Zeichen für die erste Display-Zeile
3. **LF** (Line Feed) = 0Ah
4. Zeile 2 + **LF** (wenn mind. 2 Zeilen eingestellt sind)
5. Zeile 3 + **LF** (wenn mind. 3 Zeilen eingestellt sind)
6. Zeile 4 + **LF** (wenn 4 Zeilen eingestellt sind)
7. **ETX** (End Of Text) = 03h (optional)

- **Installation:**

Die Anzeigesoftware **ComMeldung** kann von der Hersteller- Website

mikrocontroller.rahm-home.de/projekte.htm als Zip-Datei heruntergeladen werden!

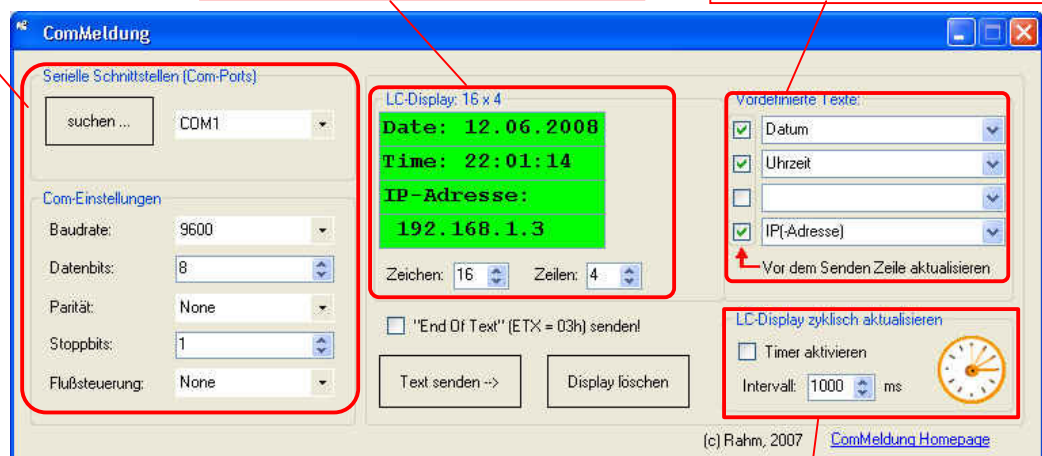
Entpacken Sie die Programm-Dateien und führen Sie **Setup.exe** aus.

Falls nicht bereits vorhanden, wird zunächst das **.NET-Framework 2.0/3.5** inklusive Language-Pack von der Microsoft-Website installiert

(ca.256MB). Sie sollten also unbedingt online sein. (Bei

Problemen mit dem automatischen Setup, kann das

.NET Framework auch direkt von www.microsoft.com



heruntergeladen und installiert werden.

Anschließend wird das Programm **ComMeldung**

installiert und gestartet. Danach können Sie die

Software jederzeit über das Startmenü neu Starten. Bei der Deinstallation verwenden Sie bitte die Windows Systemsteuerung.


Nach der Anzeige-Konfiguration kann der Timer aktiviert und

das Programm mit Doppelklick in das Icon auf der

Schnellstartleiste minimiert werden.

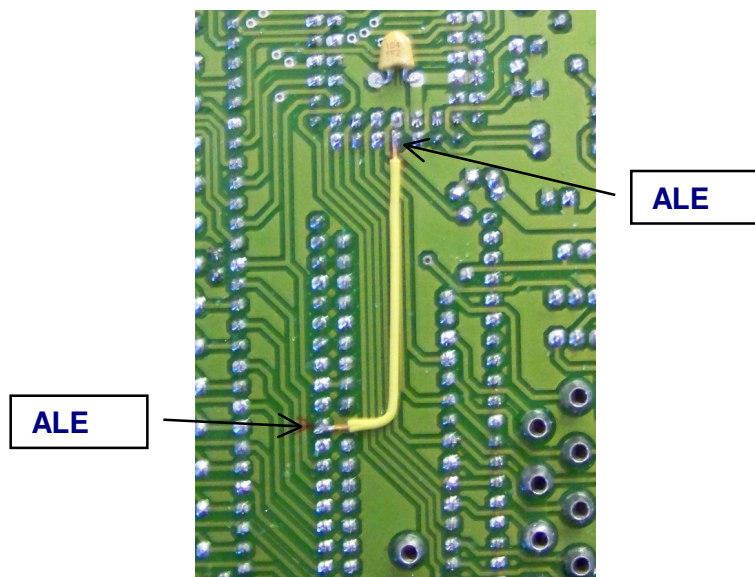
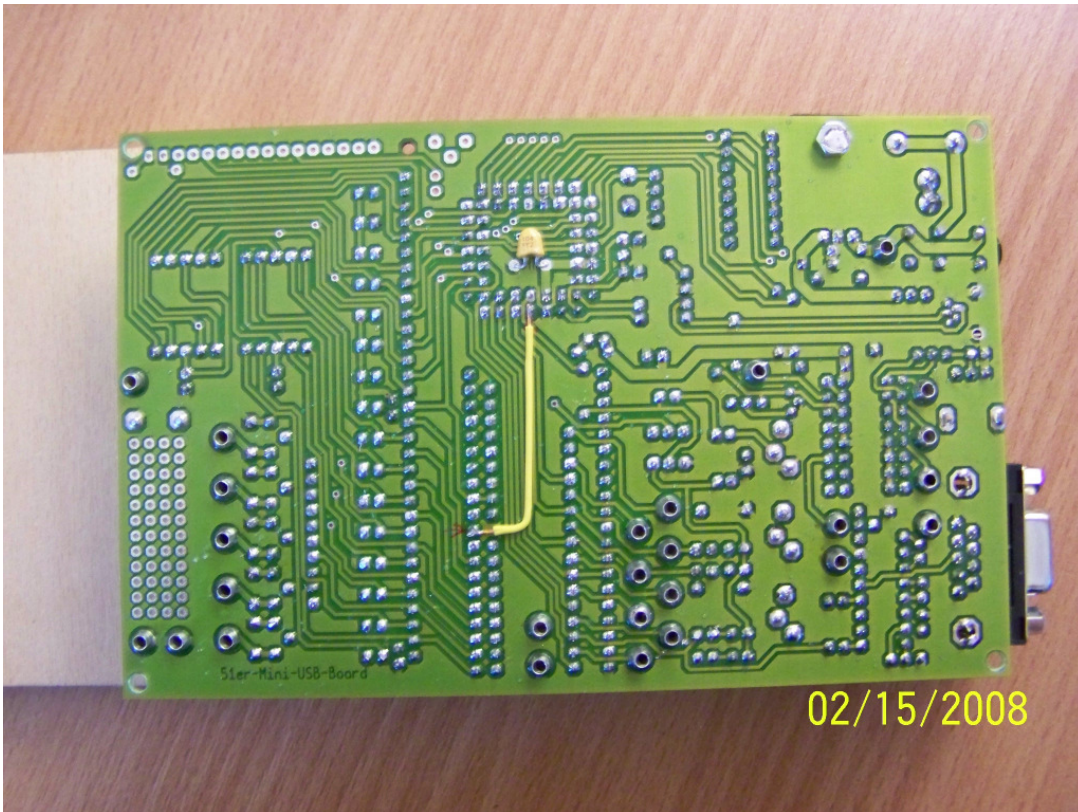



ComMeldung

 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
6.2.2	ALE-Signal auf den neuen USB-Miniboards!	Datum:

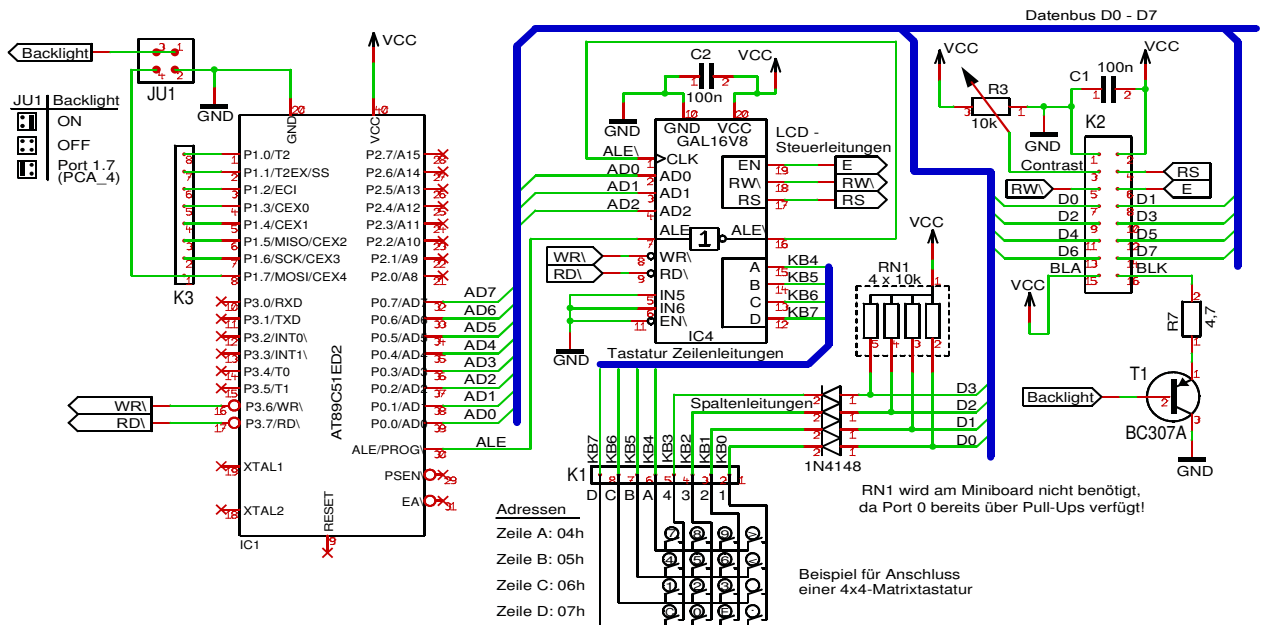
Auf den neuen USB-Miniboards sind die Signale in der Reihenfolge der Belegung eines „normalen“ 40-poligen DIP-Controllers auf zwei 20-polige Buchsenleisten geführt. Somit können alle Zusatzplatinen, also auch die Display-Platine weitergenutzt werden.

Leider fehlt als einziges das ALE-Signal, dass aber für die Funktion der Display-Platine unbedingt nötig ist. Das Foto zeigt wie mit einer Drahtbrücke das ALE-Problem gelöst werden kann. Die ALE Pins sind leicht zu identifizieren. Sie sind als einzige Pins nicht angeschlossen!



 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
	6.2.2.1.1	GAL-Programmierung 1

Der GAL auf der Projektplatte erzeugt die Steuersignale für das LC-Display und für eine Matrixtastatur. Die Signale werden aus dem gemultiplexten Adress- und Datenbus (Port 0) des 8051-Controllers gewonnen. Der Mikrocontroller kann dann mit Schreib- und Lesebefehlen (movx) auf die entsprechenden externen Adressen zugreifen.



Arbeitsauftrag:

Programmiert die Steuersignale für LC-Display und Tastatur. Erstellt ein Projekt für den verwendeten GAL-Baustein (GAL-Aufdruck beachten) in DesignExpert und fügt die auf der folgenden Seite abgebildete ABEL-Datei ins Projekt ein. Bei ... muss fehlender Text eingefügt werden.

1. Programmiert das Signal **ALE/** als **equation**
 Die Steuersignale RS, RW, D, C, B, A müssen mit fallender Flanke von ALE **gespeichert** werden. Da die internen D-FFs des GAL auf steigende Flanke triggern, muss ALE zunächst invertiert werden. Anschließend wird ALE/ auf den CLK-Pin des GAL gelegt.
2. Programmiert die Signale **RS, RW, D, C, B, A** als **truth_table**
 Da nur 3 Adressbits benötigt werden, ergeben sich folgende Adresszuordnungen:
 Hinweis: Denke daran, dass diese Signale gespeichert sein müssen!!

Adressen			LCD		Tastatur					
hex	A2	A1	A0	RS	RW	D	C	B	A	
00h	0	0	0	0	0	1	1	1	1	Kommando schreiben Busy-Flag lesen Daten schreiben Daten lesen } LC-Display
01h	0	0	1	0	1	1	1	1		
02h	0	1	0	1	0	1	1	1		
03h	0	1	1	1	1	1	1	1		
04h	1	0	0	X	X	1	1	1	0	Zeile A Zeile B Zeile C Zeile D } Tastatur
05h	1	0	1	X	X	1	1	0	1	
06h	1	1	0	X	X	1	0	1	1	
07h	1	1	1	X	X	0	1	1	1	

3. Programmiert das Steuersignal **E** als **equation**
E darf nur bei Schreibzugriffen (WR/ = 0) auf die LCD-Adressen 00h und 01h, sowie bei Lesezugriffen (RD/ = 0) auf die Adressen 01h und 03h H-Pegel haben.

```

MODULE LCD_Tastatur_decoder
@dcset
declarations
    CLK_IN          pin 1;
    ALE             pin 7;
    ALE_not        pin 16          istype'buffer,com';

    AD2,AD1,AD0    pin 4,3,2;      // gemultiplexerter Adress-/Datenbus (P0.2..0.0)
    !WR,!RD        pin 8,9;       // Schreib-/Lese-Leitungen vom µC (Nullaktiv!!)

    E              pin 19          istype'buffer,com';    // LCD Freigabesignal
    RS,RW          pin 17,18       istype'buffer,reg';    // LCD Steuersignale

    A,B,C,D        pin 15,14,13,12 istype'buffer,reg';    // Tastatur-Zeilen-Signale

    LCD             = [RS, RW];
    Keyboard        = [A,B,C,D];

equations
    ...

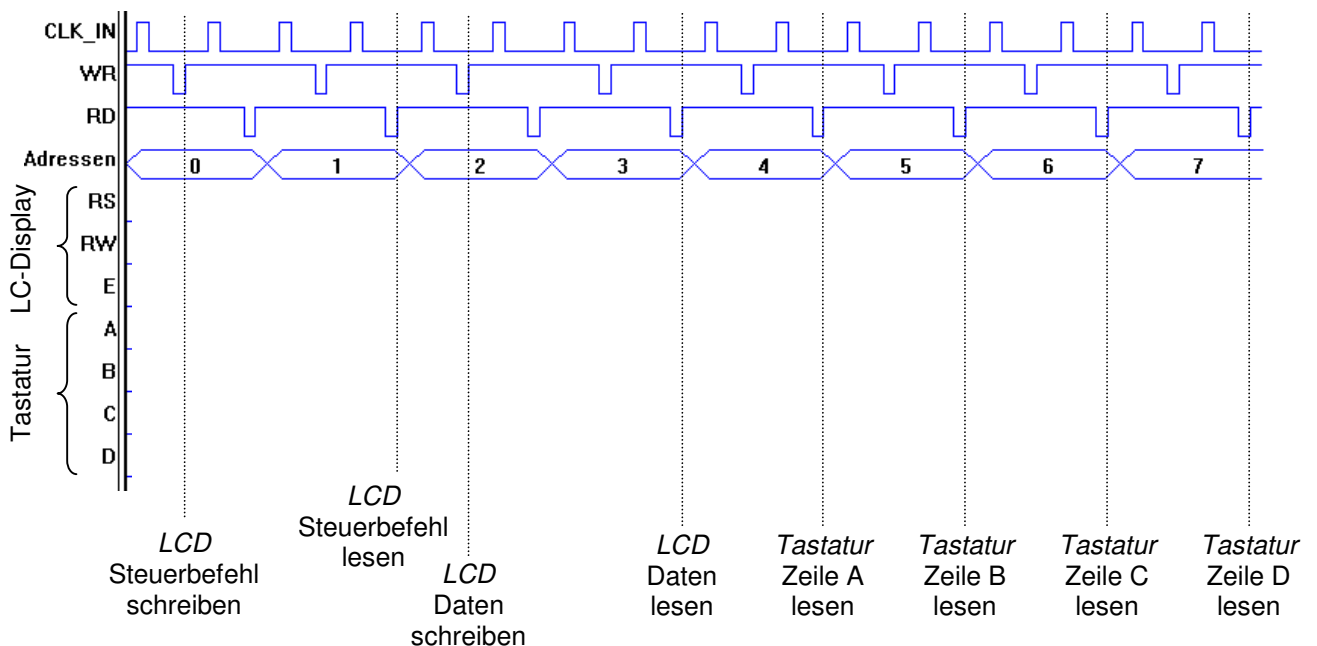
truth_table      // LCD- und Keyboard-Signale
    ...

test_vectors
    ([CLK_IN,!WR,!RD,[AD2,AD1,AD0]] -> [E, RS,RW,D,C,B,A]);


    @const a = 0;
    @repeat 8 {
        [.c., 1, 1, a] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.,.x.];
        [ 0, .k., 1, a] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.,.x.] // Schreiben
        [.c., 1, 1, a] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.,.x.];
        [ 0, 1, .k., a] -> [.x.,.x.,.x.,.x.,.x.,.x.,.x.,.x.] // Lesen
        @const a = a+1;
    }
END

```

4. Testet die Programmierung mit dem Testvektor mit einer **Functional Simulation!** Der Testvektor erzeugt die folgende Eingangssignale. Dokumentiert die Ausgangssignale:



5. Programmiert das JEDEC-File mit dem GALEP IV-Programmiergerät auf den GAL und testet die Programmierung mit dem Programm **LCD_test.hex** auf dem Miniboard!

 Friedrich-Ebert-Schule Esslingen	Projekt: PC-Diagnose-Display	Name:
	6.2.3.4	Kommunikation mit ComMeldung

Für die Kommunikation zwischen Miniboard und der PC-Software (ComMeldung) über die serielle Schnittstelle ist das Assemblerprogramm zu schreiben. ComMeldung sendet die Anzeigedaten in folgender Reihenfolge:

- ➔ **STX** („Start Of Text“) ASCII = 02h
- ➔ **Zeile 1**: 16 ASCII-Zeichen
- ➔ **LF** („Line Feed“) ASCII = 0Ah
- ➔ **Zeile 2 + LF**
- ➔ **Zeile 3 + LF**
- ➔ **Zeile 4 + LF**

Parameter für die RS232-Kommunikation:

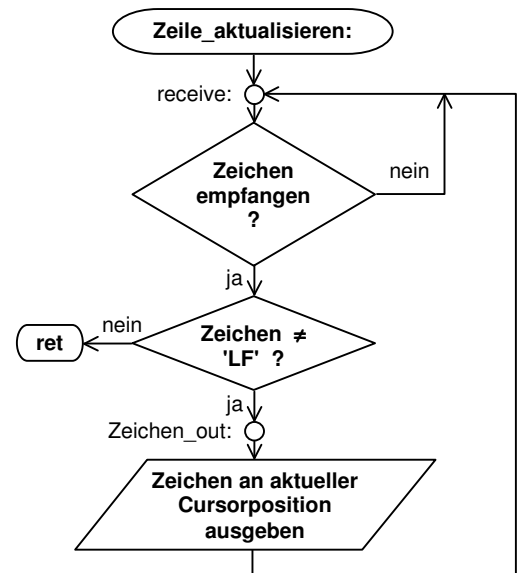
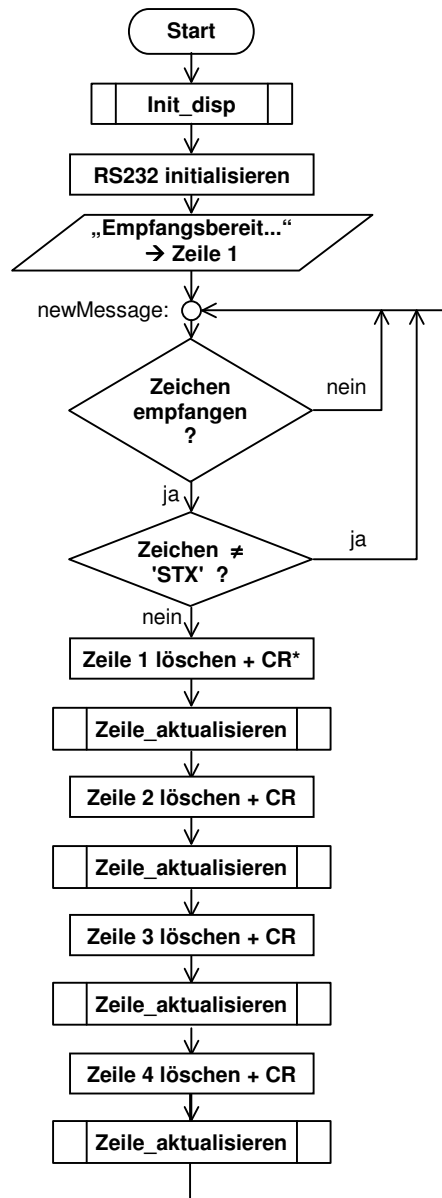
- 9600 Baud,
- 8 Datenbit,
- kein Paritätsbit,
- 1 Stopbit
- keine Flusskontrolle

Aufgabe:

Erstellen Sie in ihrer Arbeitsgruppe das Assemblerprogramm. Es muss die Daten von der seriellen Schnittstelle entgegennehmen und auf dem Display darstellen. Sie können sich dabei an dem abgebildete PAP orientieren.

Vorgehensweise


1. Assemblerdatei erstellen und ins Projekt ed2.aof einfügen.
2. Display-Bibliothek ins Projekt einbinden.
3. Hauptprogramm nach PAP erstellen.
4. Unterprogramm **Zeile aktualisieren** nach PAP erstellen.
5. Programm auf den Mikrocontroller laden.
6. ComMeldung starten und Kommunikation testen.
7. Programm gut **dokumentieren**, ausdrucken und abgeben!



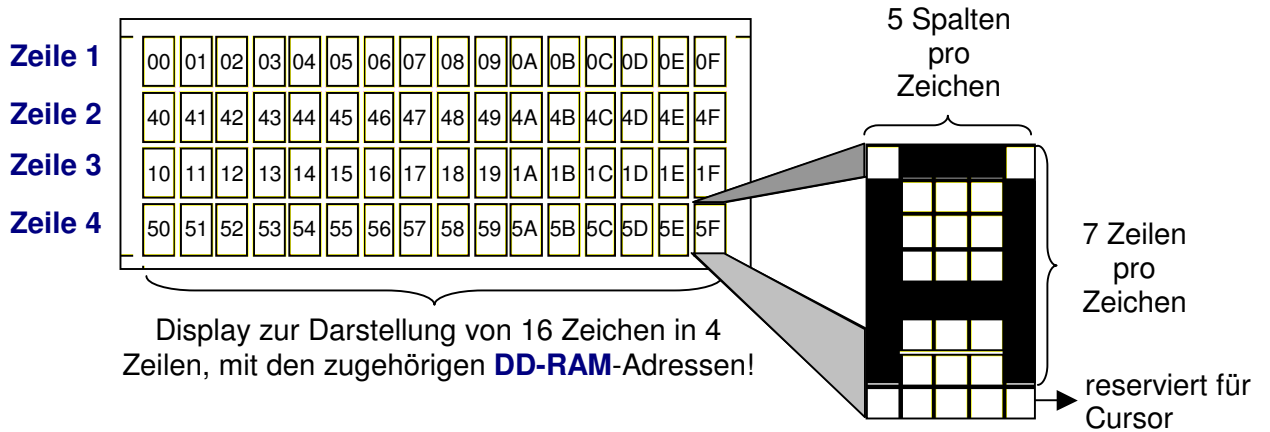
* **CR** (Carriage Return): Cursor auf den Zeilenanfang setzen.

Zusatzaufgabe:

- In ComMeldung Version 1.0.0.7 wird nach dem Ausgeben der letzten Zeile optional das Steuerzeichen ETX gesendet. Wie müsste die Auswertung dieses Zeichens geschehen. Ändern Sie den PAP entsprechend ab und schreiben Sie das Programm.


 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
	4.6.2.1	Aufbau eines LC-Textdisplay

In einem LC-Textdisplay mit HD44780-Controller werden Textzeichen in einer 5x7 dot Punktmatrix dargestellt. Ein ASCII-ähnlicher Zeichensatz ist im **Character-Generator-ROM** (CG-ROM) des Controllers fest abgelegt. Zusätzlich können im 64Byte großen **Character-Generator-RAM** (CG-RAM) bis zu 8 Zeichen (5x7) frei programmiert werden. Zur Darstellung der Zeichen muß der entsprechende Zeichencode auf die gewünschte Adresse im **Display-Data-RAM** (DD-RAM) geschrieben werden.



Zeichenmuster (Pattern) und zugehörige Zeichencodes

Lower 4 bit \ Upper 4 bit	0000 (\$0x)	0010 (\$2x)	0011 (\$3x)	0100 (\$4x)	0101 (\$5x)	0110 (\$6x)	0111 (\$7x)	1010 (\$Ax)	1011 (\$Bx)	1100 (\$Cx)	1101 (\$Dx)	1110 (\$Ex)	1111 (\$Fx)
xxxx0000 (\$x0)	CG RAM (0)	0	@	P	`	P		-	9	3	α	p	
xxxx0001 (\$x1)	(1)	!	1	A	Q	a	q	◻	7	4	ä	q	
xxxx0010 (\$x2)	(2)	"	2	B	R	b	r	┌	ι	×	β	θ	
xxxx0011 (\$x3)	(3)	#	3	C	S	c	s	└	υ	ε	ε	∞	
xxxx0100 (\$x4)	(4)	\$	4	D	T	d	t	√	I	†	μ	Ω	
xxxx0101 (\$x5)	(5)	%	5	E	U	e	u	•	♠	‡	σ	ü	
xxxx0110 (\$x6)	(6)	&	6	F	V	f	v	☞	カ	ニ	ρ	Σ	
xxxx0111 (\$x7)	(7)	'	7	G	W	g	w	ア	キ	ヌ	ラ	g	π
xxxx1000 (\$x8)	CG RAM (0)	<	8	H	X	h	x	イ	ク	ネ	リ	ル	̄
xxxx1001 (\$x9)	(1))	9	I	Y	i	y	ウ	ケ	ノ	ル	'	y
xxxx1010 (\$xA)	(2)	*	:	J	Z	j	z	エ	コ	ハ	レ	j	〒
xxxx1011 (\$xB)	(3)	+	;	K	[k	[オ	サ	ヒ	ロ	*	斤
xxxx1100 (\$xC)	(4)	,	<	L	¥	l	l	ヤ	シ	フ	ワ	φ	円
xxxx1101 (\$xD)	(5)	-	=	M]	m]	ユ	ズ	ヘ	ン	も	÷
xxxx1110 (\$xE)	(6)	.	>	N	^	n	→	ヨ	セ	ホ	ッ	ñ	
xxxx1111 (\$xF)	(7)	/	?	O	_	o	←	ッ	リ	マ	□	ö	■

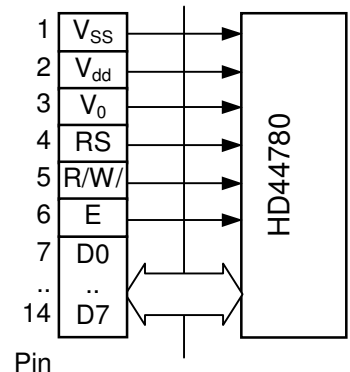
 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.6.2.2	Aufbau eines LC-Textdisplay	Datum:

Die Ansteuerung des Display-Controllers durch den Mikrocontroller erfolgt über 8 Datenleitungen D0..D7 und 3 Steuerleitungen. Dies sind die Signale:

- RS** (Register Select): 0: Kommando oder Busy-Flag + Adresszähler
1: Datenregister
- RW/** (read/write): 0: schreiben
1: lesen
- E** (Enable): 1: Schreib- bzw. Lesefreigabe


Für RS und RW/ ergeben sich vier Befehlsgruppen mit unterschiedlicher Wirkung:

RS	RW/	Befehl (Display-Operation)
0	0	Kommando schreiben
0	1	Busy-Flag und Adresszähler lesen
1	0	Daten schreiben
1	1	Daten lesen



Die Tabelle zeigt alle Kommandos und Befehle des HD44780:

Befehl	Code										Beschreibung
	RS A1	RW/ A0	D7	D6	D5	D4	D3	D2	D1	D0	
Clear display	0	0	0	0	0	0	0	0	0	1	Löscht das Display und setzt den Cursor auf die Adresse 00h
Cursor home	0	0	0	0	0	0	0	0	1	x	Cursor auf Adresse 00h setzen
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	I/D: 0 = Adress-Zähler dekrementieren 1 = Adress-Zähler inkrementieren S: 0 = Display Shift AUS 1 = Display Shift EIN
Display On/Off control	0	0	0	0	0	0	1	D	C	B	D: 0 = Display AUS 1 = Display EIN C: 0 = Cursor AUS 1 = Cursor EIN B: 0 = Blinken AUS 1 = Blinken EIN
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	x	x	S/C: 0 = Cursorshift 1 = Displayshift R/L: 0 = Links schieben 1 = Rechts schieben
Function set	0	0	0	0	1	DL	N	F	x	x	DL: 0 = 4 Bit Mode 1 = 8 Bit Mode N: 0 = Display Einzeilig 1 = Display Zweizeilig F: 0 = 5x7dots Zeichensatz 1 = 5x10dots Zeichensatz
Set CGRAM address	0	0	0	1	CGRAM Adresse 6 Bit					Nach diesem Kommando werden die nächsten Lese- und Schreiboperationen im Zeichengenerator-RAM durchgeführt.	
Set DDRAM address	0	0	1	DDRAM Adresse 8 Bit					Nach diesem Kommando werden die nächsten Lese- und Schreiboperationen im Display-RAM durchgeführt.		
Read busy-flag and address counter	0	1	BF	Address counter					BF: 0 = Display kann Daten empfangen (ready) 1 = Display ist beschäftigt (busy) Address counter: Z.B. Abfragen der aktuellen Cursorposition		
Write Data to CGRAM or DDRAM	1	0	write data					Schreibt Daten in die aktuelle Adresse und erhöht den Adress-Zähler			
Read Data from CGRAM or DDRAM	1	1	read data					Liest Daten von der aktuellen Adresse und erhöht den Adress-Zähler			

 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.6.4.1	Verwenden der Display-Assemblerbibliothek	Datum:

Zur Nutzung der Bibliotheksfunktionen muss die Datei **lcd_library.a51** ins Projekt eingebunden werden. Weiterhin ist ins Projektverzeichnis (z.B. C:\controller\ed2) die Datei **LCD.inc** zu kopieren. Dort werden globale Symbole zur Displaysteuerung definiert. In jedem Assemblermodul das diese benötigt, muss die Datei inkludiert werden.

LCD.inc

```

;*****
; ** Symboldefinitionen für LC-Display, R.Rahm, 13.12.06
; ** Vor Inklusion dieser Datei muss die Umgebungsvariable LCD16 oder LCD17 gesetzt werden
;*****
; $Set (LCD20) ; Hier die Umgebungsvariable für das verwendete Display
$Set (LCD16) ; setzen! Unbedingt vor Inclusion von LCD.inc

backlight bit p1.7 ; backlight = 0 --> Hintergrundbeleuchtung ein!
busyFlag bit acc.7 ; Busy-Flag wird in Akkubit 7 zurücklesen!

WriteLCDCommand equ 00h ; 8 Bit Schreib- und Leseadressen für Display
ReadBusy equ 01h ; Lesen mit MOVX a,@Rn
WriteLCDData equ 02h ; Schreiben mit MOVX @Rn,a
ReadLCDData equ 03h ; !!! Keinen Datenpointer verwenden !!!
; (16 Bit Adressierung benötigt P2)

; Steuerbefehle
DisplayClear equ 00000001b ; LCD-Befehle
ReturnHome equ 00000010b
CursorOn equ 00001110b
CursorOff equ 00001100b
CursorShiftRight equ 00010100b
CursorShiftLeft equ 00010000b
DisplayShiftRight equ 00011100b
DisplayShiftLeft equ 00011000b

Z1_Start_Address equ 0h ; DDRAM-Adressen für Zeilenanfänge
Z2_Start_Address equ 40h
$if defined (LCD20)
Z3_Start_Address equ 14h ; Abhängig von der Displaygröße
Z4_Start_Address equ 54h
$elseif defined (LCD16)
Z3_Start_Address equ 10h
Z4_Start_Address equ 50h
$endif

```

Die Bibliotheksfunktionen der Datei **lcd_library.a51** sind als Unterprogramme definiert. Dabei ist immer der Akku oder der Datenpointer das Übergaberegister:

Displaysteuerung


<i>init_disp</i>	Initialisierung muss vor jeder Nutzung des Displays durchgeführt werden!
<i>define_char</i>	Definition von eigenen Sonderzeichen
<i>WaitWhileBusy</i>	Das Busy-Flag sollte vor jedem Schreib-/Lesevorgang abgefragt werden!
<i>clear_disp</i>	Gesamtes Display löschen
<i>clear_zeile1</i>	Zeile 1 löschen
<i>clear_zeile2</i>	Zeile 2 löschen
<i>clear_zeile3</i>	Zeile 3 löschen
<i>clear_zeile4</i>	Zeile 4 löschen

Cursorsteuerung

<i>cursor_home</i>	Cursor auf DDRAM Adresse 00h
<i>cursor_rechts</i>	Cursor eine Position nach rechts
<i>cursor_links</i>	Cursor eine Position nach links
<i>set_cursor</i>	Cursor auf beliebige Adresse setzen
<i>cursor_on</i>	Cursor Einschalten
<i>cursor_off</i>	Cursor Ausschalten

Zeichenausgabe

<i>out_char</i>	Ausgabe eines Zeichens auf der aktuellen Cursorposition
<i>out_disp</i>	Ausgabe einer mit 0 terminierten konstanten Zeichenkette
<i>out_var</i>	Ausgabe einer mit 0 terminierten variablen Zeichenkette
<i>text_zeile1</i>	Ausgabe einer 0 terminierten Zeichenkette in Zeile 1
<i>text_zeile2</i>	Ausgabe einer 0 terminierten Zeichenkette in Zeile 2
<i>text_zeile3</i>	Ausgabe einer 0 terminierten Zeichenkette in Zeile 3
<i>text_zeile4</i>	Ausgabe einer 0 terminierten Zeichenkette in Zeile 4

 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.6.4.2	Verwenden der Display-Assemblerbibliothek	Datum:

Vor dem Aufruf der Bibliotheksfunktionen müssen die benötigten Unterprogramme als extern deklariert werden:

```
extern code init_disp
extern code define_char, clear_disp, clear_zeile1, clear_zeile2, clear_zeile3, clear_zeile4
extern code cursor_home, cursor_rechts, cursor_links, set_cursor, cursor_on, cursor_off
extern code out_char, out_disp, out_var, text_zeile1, text_zeile2, text_zeile3, text_zeile4
```

Funktionsaufrufe

- Beispiele für parameterlose Funktionen

```
lcall  init_disp
```

```
lcall  clear_disp
```

```
lcall  cursor_home
```

- Beispiele für Funktionen mit Übergabeparameter im Akku

```
mov    a, #'A'
lcall  out_char
```

```
mov    a, #47h
lcall  set_cursor
```

- Beispiel für Funktion mit Übergabeparameter im Datenpointer

```
mov    dptr, #text1
lcall  text_zeile2
```

...

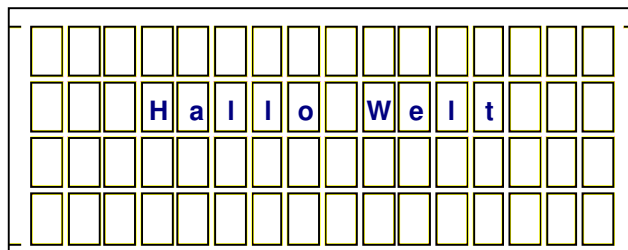
```
text1: db "Hallo", 0
```

- Beispiel für Funktion mit Übergabeparameter in Akku und Datenpointer

```
mov    dptr, #zeichen1
mov    a, #001000b
lcall  define_char
```

Übungen:

Schreiben Sie ein Assemblerprogramm, welches den Text „**Hallo Welt**“ auf Zeile 2 des LC-Displays ausgibt:



- Verwenden Sie nur die LCD-Funktionen **init_disp** und **Wait_While_Busy**
- Lösen Sie die Aufgabe mit beliebigen Bibliotheksfunktionen

 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.6.7	LCD-Displaybefehle mit C	Datum:

Für die Ansteuerung des LC-Displays sind die folgenden Funktionsprototypen in **LCD8.h** definiert:

```

//... Displaysteuerung
extern void InitDisp(void); //Display initialisieren
//Eigene Zeichen definieren
extern void DefineChar(unsigned char *PixelTabelle, unsigned char ZeichenNr);
extern void ClearDisp(void); //Display Löschen
extern void ClearZeile(unsigned char Zeilennummer); //Zeile löschen (1,2,3,4)

//... Cursorsteuerung
extern void CursorHome(void);
extern void CursorRechts(void);
extern void CursorLinks(void);
extern void SetCursor(unsigned char DDRamAddress);
extern void CursorAnfangZeile(unsigned char Zeilennummer);
extern void CursorEin(void);
extern void CursorAus(void);

//... Zeichenausgabe
extern void OutChar(unsigned char Zeichen);
extern void OutDisp(unsigned char *pChar); // Ausgabestring muss /0-terminiert sein
// Ausgabe ab aktueller Cursorposition
extern void TextZeile(unsigned char *pChar, unsigned char Zeilennummer);

```

Die Datei **LCD8.c** muß dem Projekt hinzugefügt werden, oder in die Bibliothek **rc51atms.lib** integriert werden (Anleitung siehe Bubbers). Eine fertige rc51atms.lib befindet sich auf der DVD.

LCD Programmbeispiel:

```

#include <5131.h> // AT89C5131
#include <lcd8.h> // Headerdatei für LC-Display
#include <stdio.h> // wird benötigt für sprintf

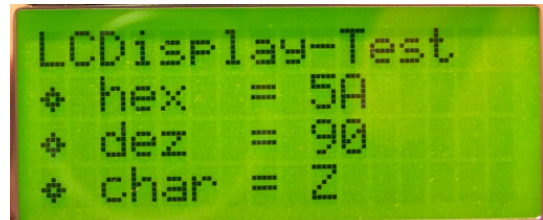
code unsigned char Zeichenl[] = { // Tabelle für selbstdefiniertes
    0b00000, // Zeichen 5 x 8 -Matrix
    0b00000,
    0b00100,
    0b01110,
    0b11011,
    0b01110,
    0b00100,
    0b00000
};


void main (void)
{
    unsigned char a;
    unsigned char buffer[16]; // Puffer für Textzeilen

    InitDisp(); // Display initialisieren
    DefineChar(Zeichenl, 0x01); // Erlaubte CGRam-Adressen: 0x01..0x08
    TextZeile("LCDisplay-Test", 1); // konstante Zeichenkette anzeigen

    while (1) // Endlosschleife
    {
        a = P1; // Port 1 einlesen
        sprintf(buffer, "\x01 hex = %X ", a); // Anzeigestring in buffer[]
        // Escape Sequenz \x01 = Adresse 0x01
        TextZeile(buffer, 2); // Ausgabe auf Zeile 2
        sprintf(buffer, "\x01 dez = %u ", a);
        TextZeile(buffer, 3); // Ausgabe auf Zeile 3
        sprintf(buffer, "\x01 char = %c ", a);
        TextZeile(buffer, 4); // Ausgabe auf Zeile 4
    }
}

```

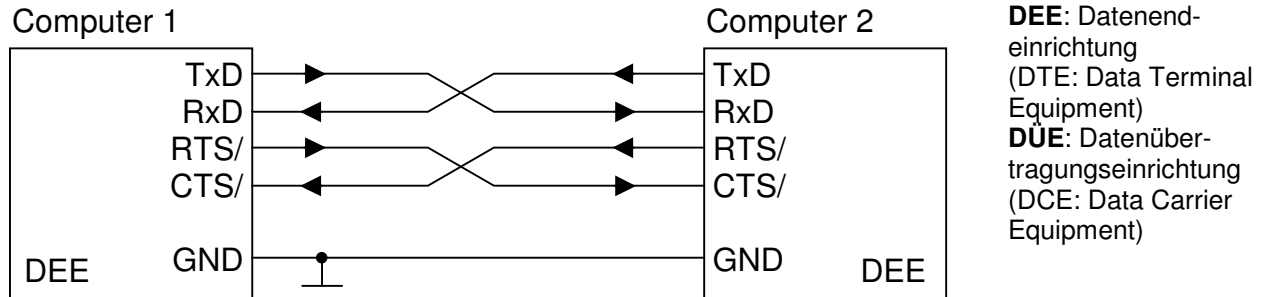


 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.3.1	Die serielle Schnittstelle (RS232)	Datum:

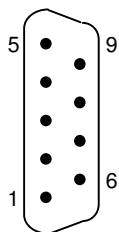
Normen: V.24 (DIN 66020 oder ITU-T; D, EU)
RS232C (EIA; USA)

Genormt sind: Steckerbelegung, Signalpegel („0“ = 3..15V; „1“ = -3..-15V)

Nullmodem-Verbindung zwischen 2 PC's



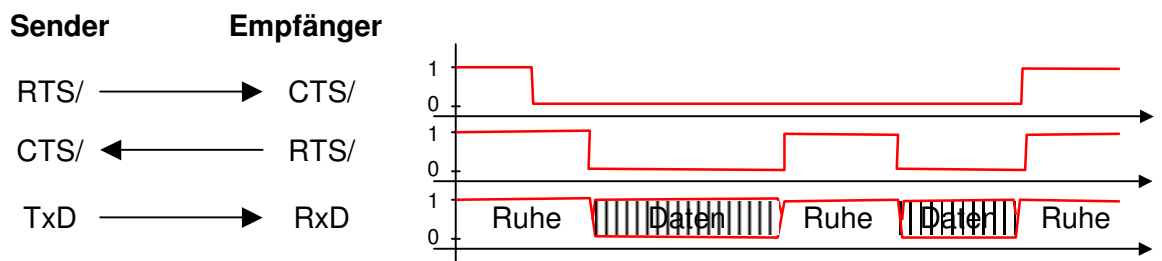
Signal- und Pinbelegung (9-polig SUB-D)



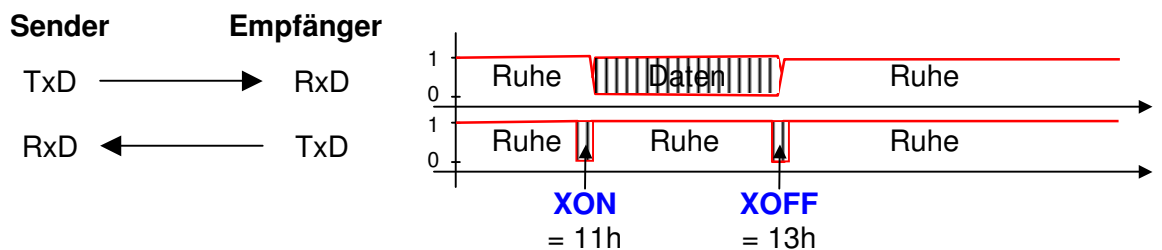
Pin	Signal	Beschreibung	I/O
3	TxD	Sendedatenleitung (Transmit Data)	Ausgang
2	RxD	Empfangsdatenleitung (Receive Data)	Eingang
7	RTS/	Sendeanforderung (Request to Send)	Ausgang
8	CTS/	Sendefreigabe (Clear to Send)	Eingang
5	GND	Signal-Ground	-

Flussteuerung (Handshake):

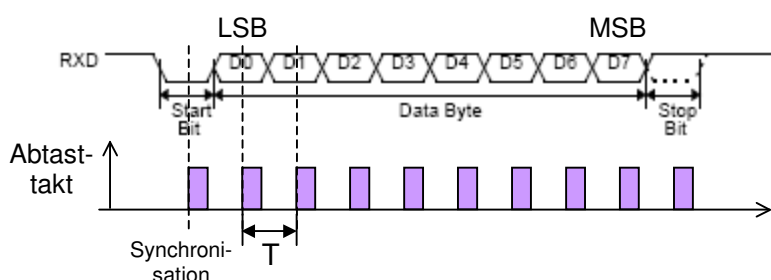
- Hardware-Handshake (mit RTS/ und CTS/)
Bsp.: Computer 1 möchte Daten zu Computer 2 senden!



- Software-Handshake (ohne RTS/ und CTS/)

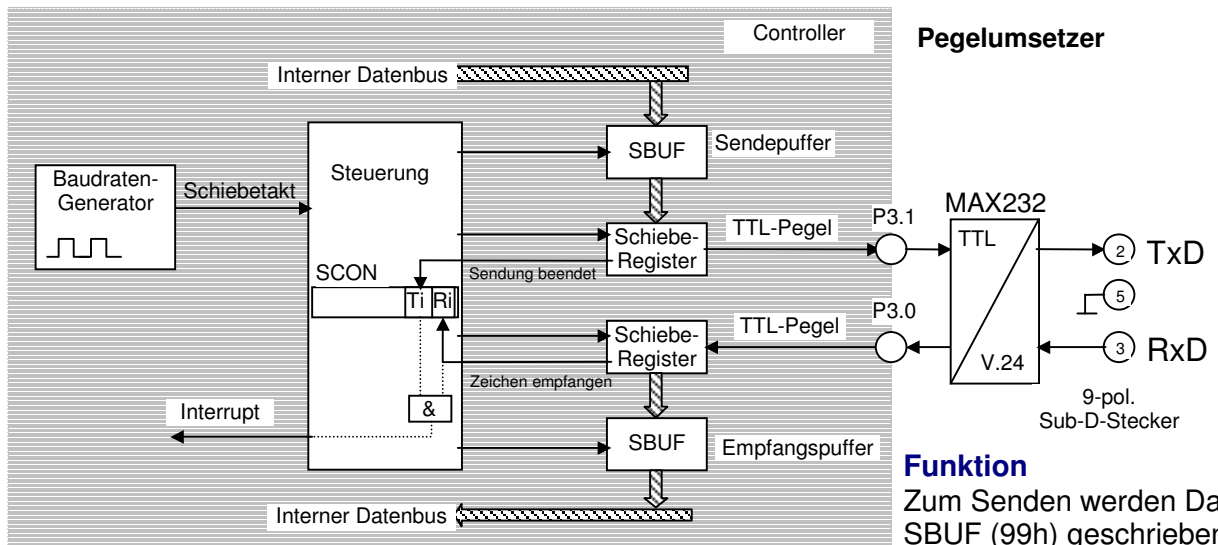


Übertragungsrahmen (Frame):



Die Signalabtastung erfolgt bei halber Bitdauer T . Da der Bittakt nicht mit übertragen wird, muss die Abtastrate (Baudrate) im Empfänger und Sender gleich gewählt sein.

Baudrate: 1 Baud = $1/T$
z.B.: 1200, 2400, 4800, 9600, 14400,



Pegelumsetzer

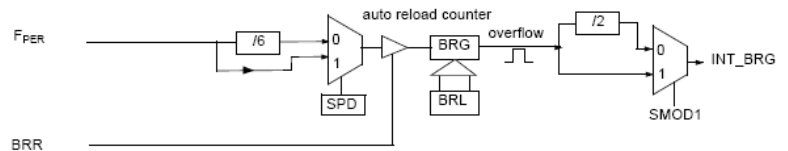
Funktion

Zum Senden werden Daten in SBUF (99h) geschrieben. Beim Empfang werden die Daten aus SBUF gelesen.

Interner Baudratengenerator

Der interne Baudratengenerator besitzt einen Timer **BRG**, der beim Überlauf automatisch mit dem Wert im Register **BRL** (9Ah) nachgeladen wird. Mit Hilfe der Steuerbits SPD und SMOD1 kann

Internal Baud Rate



der erzeugte Schiebetakt heruntergeteilt werden (1/6 bzw. 1/2). Die Eingangsfrequenz F_{PER} ist die halbe Oszillatorfrequenz, also $F_{PER} = 6\text{MHz}$. Es gelten folgende Formeln:

$$\text{Baud_Rate} = \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot (256 - \text{BRL})}$$

Beispiel: Oszillator 12 MHz $\rightarrow F_{PER} = 6\text{ MHz}$

Gewünschte Baudrate: 9600 Bit/s; SMOD1 = 1; SPD = 1

$$\text{BRL} = 256 - \frac{2^{\text{SMOD1}} \cdot F_{\text{PER}}}{6^{(1-\text{SPD})} \cdot 32 \cdot \text{Baud_Rate}}$$

$$\text{BRL} = 256 - \frac{2^1 \cdot 6\text{MHz}}{6^{(1-1)} \cdot 32 \cdot 9600 \frac{1}{s}} = 256 - 39 = 217$$


Initialisierung

PCON : Power Control Register (87h; nicht bitadressierbar)								Befehl
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
SMOD1 serial port mode bit 1	SMOD0	reserve	POF	GF1	GF2	PD	IDL	
1	x	x	x	x	x	x	x	orl PCON,#1000000b

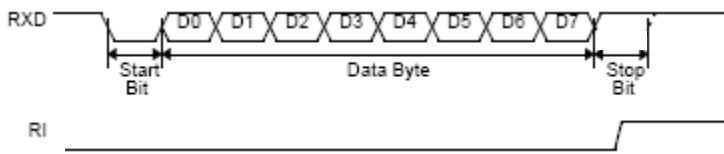
doppelte Baudrate

BDRCON : Baudrate Generator Control Register (9Bh; nicht bitadressierbar)								Befehl
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
reserve	reserve	reserve	BRR Baud Rate Run Bit	TBCK Transmitter Control	RBCK Receiver Control	SPD Speed Control	SRC Receive Interrupt flag	
0	0	0	1	1	1	1	1	mov BRL,#217 mov BDRCON,#00011111b
--	--	--	internen Baudraten Generator starten	Transmitter und Receiver mit internem Baudrate Generator		Fast Baud Rate Generator Speed		egal in Modus 1

SCON : Serial Port Control Register (98h; bitadressierbar)								Befehl
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
SM0	SM1	SM2 Multiproc Comm. bit	REN Reception Enable bit	TB8 Transmitter bit 8	RB8 Receiver bit 8	TI Transmit Interrupt flag	RI Receive Interrupt flag	
Serial Port Mode								mov SCON,#0101000b
0	1	0	1	0	0	0	0	Wird nach Empfang vom μC gesetzt! Muss manuell zurückgesetzt werden!
Modus 1: 8-Bit UART		--	Empfang erlaubt	egal in Modus 1				Wird nach dem Senden vom μC gesetzt! Muss manuell zurückgesetzt werden

 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.3.2.2	Programmierung der seriellen Schnittstelle	Datum:

Datenframe beim Empfangen (Modus 1):



Im Modus 1 wird zunächst ein Startbit gesendet. Anschließend 8 Datenbit und ein Stoppbit. Der UART erzeugt kein Paritätsbit. Ist dies gewünscht, muss Bit D7 vor der Übertragung entsprechend gesetzt werden.

Initialisierung:

```

orl PCON,#1000000b ;SMOD=1 bei 9600 Baud
mov BDRCON,#00011111b ;int. Baudraten-Generator für Senden und Empfangen, SPD=1
mov BRL,#217 ;Reloadwert für int. Baudratengenerator
mov SCON,#01010000b ;8-Bit-UART (Mode 1), Empfang zulassen
clr ES ;Seriellen Interrupt sperren
  
```

Programmteil Zeichen_Senden:

```

mov sbuf,#'A' ;Ascii „A“ (= 41h) ausgeben
warte: jnb ti,warte ;warten bis gesendet
clr ti ;Sende-Flag löschen
  
```

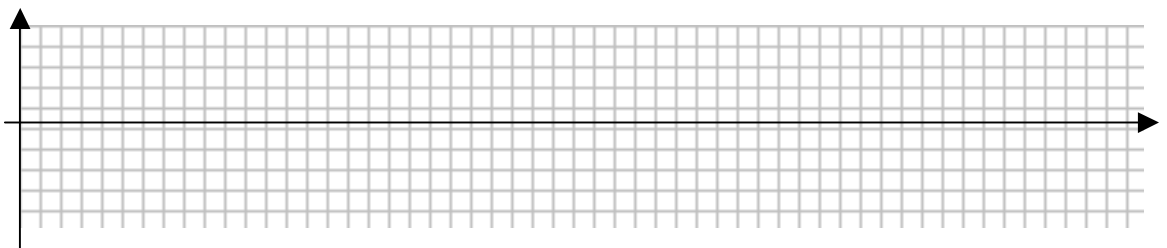
Programmteil Zeichen_empfangen:

```

weiter: jnb ri,weiter ;Sprung wenn kein Zeichen empfangen
mov a,sbuf ;Zeichen abholen
clr ri ;Empfangsflag löschen (Ab jetzt kann nächstes Zeichen
; empfangen werden)
  
```

Arbeitsauftrag:

- Schreibe ein Programm, welches bei Betätigen der Sendetaste (P3.3 „Blau“) das Ascii-Zeichen 'I' (=49h) an der seriellen Schnittstelle ausgibt. Der Empfang erfolgt am PC mit dem Programm **Hyperterminal**. Starte Hyperterminal mit der Steuerdatei **µC8252.ht**.
- Stelle das V.24-Signal der Sendedatenleitung TxD am Oszilloskopschirm dar und Skizziere es in das Liniendiagramm. Zeichne Startbit, Datenbits (LSB, MSB), Stoppbit und die Ruhephasen ein.:



- Ermittle mit dem Skope die genaue Baudrate

Baudrate = _____ =


Zusatzaufgabe:

- Erweitere das Programm, so dass bei Tastendruck der Text „Hallo Welt“ am PC ausgegeben wird! (**Hinweis:** Lege den Text als Tabelle im Programmspeicher ab. Eine 0 als letzten Tabellenwert signalisiert das Textende. 0Ah ist das Line-Feed Zeichen (Zeilenwechsel)!

```

text: db "Hallo Welt",0Ah,0
  
```

Kommentiere Deine Programme und drucke es aus!

 Friedrich-Ebert-Schule Esslingen	MIKROCONTROLLER	Name:
4.3.3	Text über RS232 übertragen	Datum:

Assemblerlisting

```

include c51rd2.inc

senden bit    p3.3
BRL    data   9Ah
BDRCON data   9Bh

code at 0

init_V24:
    orl    PCON, #10000000b
    mov    BDRCON, #00011111b
    mov    BRL, #217
    mov    SCON, #01010000b
    clr    ES

    mov    dptr, #text

loop:   jnb    senden, loop
flanke: jnb    senden, flanke

next_c: mov    r0, #0
        mov    a, r0
        movc   a, @a+dptr
        jz     loop

        mov    SBUF, a
send:   jnb    ti, send
        clr    ti

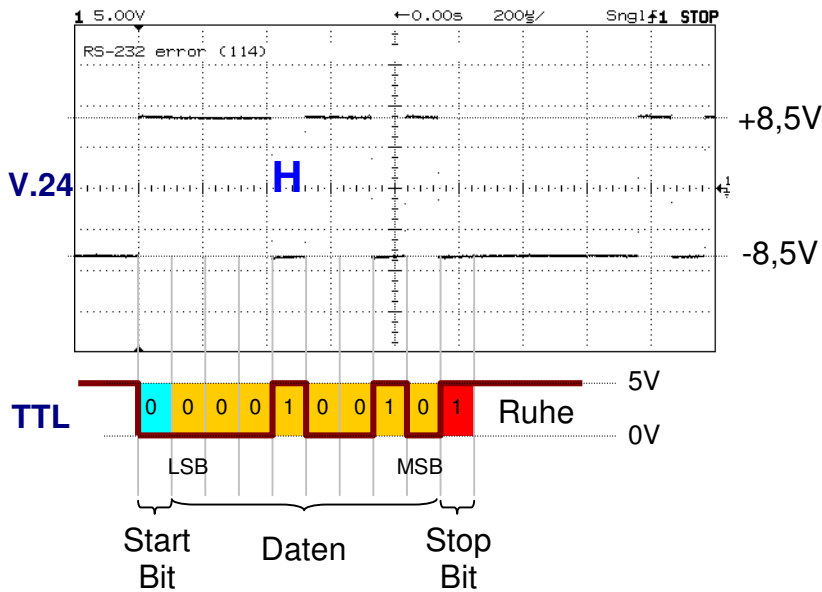
        inc    r0
        sjmp  next_c

text:   db    "Hallo Welt", 0Ah, 0

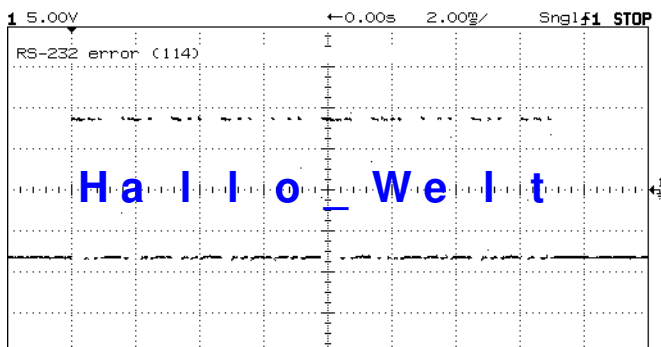
```


0Ah = LF
„Line Feed“

Oszillogramm des Sendevorgangs



Übertragenes Zeichen:
01001000b ⇒ 48h ⇒ „H“



 Friedrich-Ebert-Schule Esslingen	Mikrocontroller	Name:
4.3.4	Zeichencodes	Datum:

ASCII = American Standard Code for Information Interchange

= 7-Bit Zeichencode → 128 Zeichen:

94 Schriftzeichen:

0...9, A...Z, a...z, ...
(keine Umlaute!)

33 Steuerzeichen:

zur Datenübertragung:

SOH: Start of Header
STX: Start of Text
ETX: End of Text
EOT: End of Transmission
ACK: Acknowledge

zur Gerätesteuerung:

DCx: Device Control 1..4
DEL: Delete
CAN: Cancel
NUL: Null
SUB: Substitute

Formatierung:

LF: Line Feed
CR: Carriage Return
VT: Vertical Tabulator
BS: Backspace

Code-Erweiterung:

ESC: Escape

ASCII		Zeichen	ASCII		Zch.	ASCII		Zch.	ASCII		Zch.
hex	dez		hex	dez		hex	dez		hex	dez	
00	0	NUL	20	32		40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	\$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(48	72	H	68	104	h
09	9	TAB	29	41)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	Esc	3B	59	;	5B	91	[7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	
1D	29	GS	3D	61	=	5D	93]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

UTF-8 (Unicode Transformation Format)

= 1 ... 4 Byte langer Zeichencode für zur Zeit 1.114.112 internationale Zeichen
(Quelle: Wikipedia)

Aufbau	Zeichenanzahl
0xxxxxxx	$2^7 = 128$ (7 Bit- ASCII)
110xxxxx 10xxxxxx	$2^{11} = 2.048$
1110xxxx 10xxxxxx 10xxxxxx	$2^{16} = 65.536$
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	$2^{21} = 2.097.152$