Daschke Ltd.
Krusenhof 42 / D-45731 Waltrop (Germany) / Tel.: +49(0)163 - 333 6917
Web: www.daschke-ltd.de / Email: **info@daschke-ltd.de**

DL100, DL101
Technisches Datenblatt - RS232 zu LCD-Konverter, Anzeigebaustein via RS232 (Nachfolger des LCDChipRS232)

Ausführungen:

DL100 im 28-poligen DIP-Gehäuse, RS232 zu LCD-Konverter
DL101 im 28-poligen DIP-Gehäuse, RS232 zu LCD-Konverter + 8 Bit I/O und 6 Bit I/O

Kurzspezifikationen:

* Extrem einfache Ansteuerung von alphanumerischen LCDs mittels eines Chips.
* Sie müssen sich nicht mehr sorgen um die Zeiten, die bei den Befehlen der alphanumerischen LCDs ein zu halten sind. Der DL100 erledigt dies von selbst. Sie geben nur noch Ihren Text über die 5V-RS232-Schnittstelle ein (und ab und an einen kurzen und sehr einfachen Befehl wie z.B. zum löschen der Anzeige). Einfacher geht es kaum noch!
* Es werden LCDs mit HD44780-Controller (und kompatible) im 4-Bit-Mode angesteuert.
* 2-Draht Schnittstelle: Kommunikation über TxD- & RxD-Leitung.
* LCD-Ansteuerung über die RS232 des PCs oder eines Microcontrollers. Asynchrone Befehlssteuerung – nur die TxD-Leitung der RS232 wird benötigt (RxD-Leitung kann optional genutzt werden, um die Antworten des DL100 auf dem Terminalprogramm zu überwachen).
* BEFEHLSSATZ:
    1. „#mein Text“ Raute-Zeichen gefolgt von beliebigem Text und schon wird der Text angezeigt!!!
    2. „$xxxxx“ Dollar-Zeichen gefolgt von ASCII-Zeichen ergibt einen Befehl für das LCD (z.B. Löschen des LCD, Cursorpositionierung etc.).

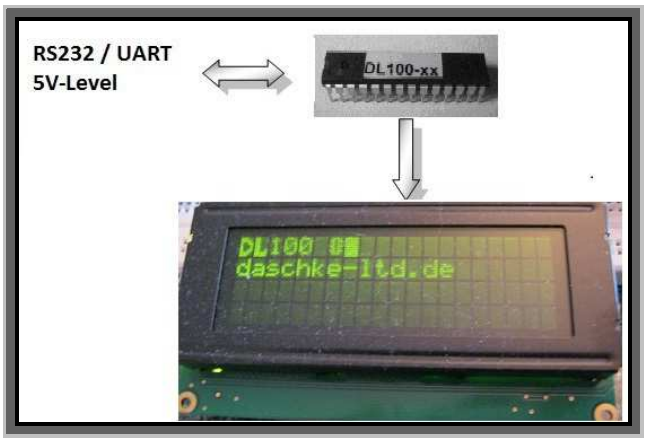
JEDER BEFEHL MUSS MIT <ENTER> bzw. <CR> ABGESCHLOSSEN WERDEN!!! Einfacher geht es kaum noch! „$“ gefolgt vom Befehl und „#“ gefolgt vom Text. Und schon reagiert das LCD!!!

Mit einer derartigen Befehlsstruktur kann ein alphanumerisches mehrzeiliges LCD ab sofort sehr einfach angesteuert werden.

Vorteile des DL100:

- Einfache Handhabung. Nur wenige Zusatzbauteile nötig (wenige Keramik-Kondensatoren & ein Widerstand)
- Akzeptiert Texte & Befehle im ASCII-Format
- Ansteuerung über die TxD- & RxD-Leitung einer RS232-Schnittstelle
- Keine Kenntnisse des Timings des HD44780-Controllers notwendig
- Keine Programmierkenntnisse notwendig. Befehle und Text als ASCII-Zeichen über die RS232 an den Chip senden. Fertig!
- Alle Einstellungen erfolgen via RS232
- Baudraten: 2400, 4800, 9600, 19200 (über RS232 umschaltbar)
- Erspart Portleitungen am Microcontroller
- Systemunabhängig: Egal, ob die Texte vom Microcontroller oder vom PC (z.B. über ein Terminalprogramm, über Ihre eigene Software) gesendet werden – Der DL100 zeigt Ihre Texte an.
- Lauffähig mit allen LCDs mit HD44780-Controller (sowie Kompatiblen) mit einer Chip Select Leitung.
- DIP28 Standardgehäuse: dadurch auch gut geeignet für Testschaltungen, Erprobungsphasen etc.

Prinzip des DL100:



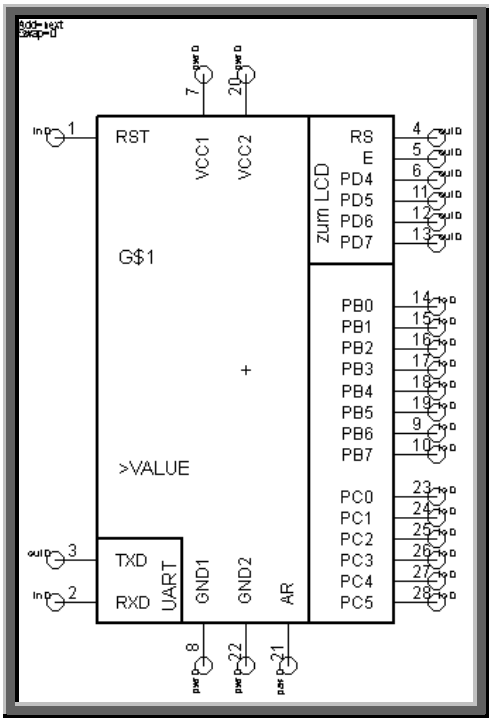
Anschlussbelegung:

Pin Nr.	Bezeichnung	Beschreibung
1	RST	Reset
2	RxD	Receive-Leitung (Datenempfang)
3	TxD	Transmit-Leitung (Daten senden)
4	RS	LCD H: DATA, L: Instruction code
5	E	LCD Chip enable signal
6	PD4	LCD Datenleitung D4
7	VCC1	+5VDC
8	GND1	Ground
9	PB6	Port B6 bei DL101
10	PB7	Port B7 bei DL101
11	PD5	LCD Datenleitung D5
12	PD6	LCD Datenleitung D6
13	PD7	LCD Datenleitung D7
14	PB0	Port B0 bei DL101
15	PB1	Port B1 bei DL101
16	PB2	Port B2 bei DL101
17	PB3	Port B3 bei DL101
18	PB4	Port B4 bei DL101
19	PB5	Port B5 bei DL101
20	VCC2	+5VDC
21	AR	AR
22	GND2	Ground
23	PC0	Port C0 bei DL101
24	PC1	Port C1 bei DL101
25	PC2	Port C2 bei DL101
26	PC3	Port C3 bei DL101
27	PC4	Port C4 bei DL101
28	PC5	Port C5 bei DL101

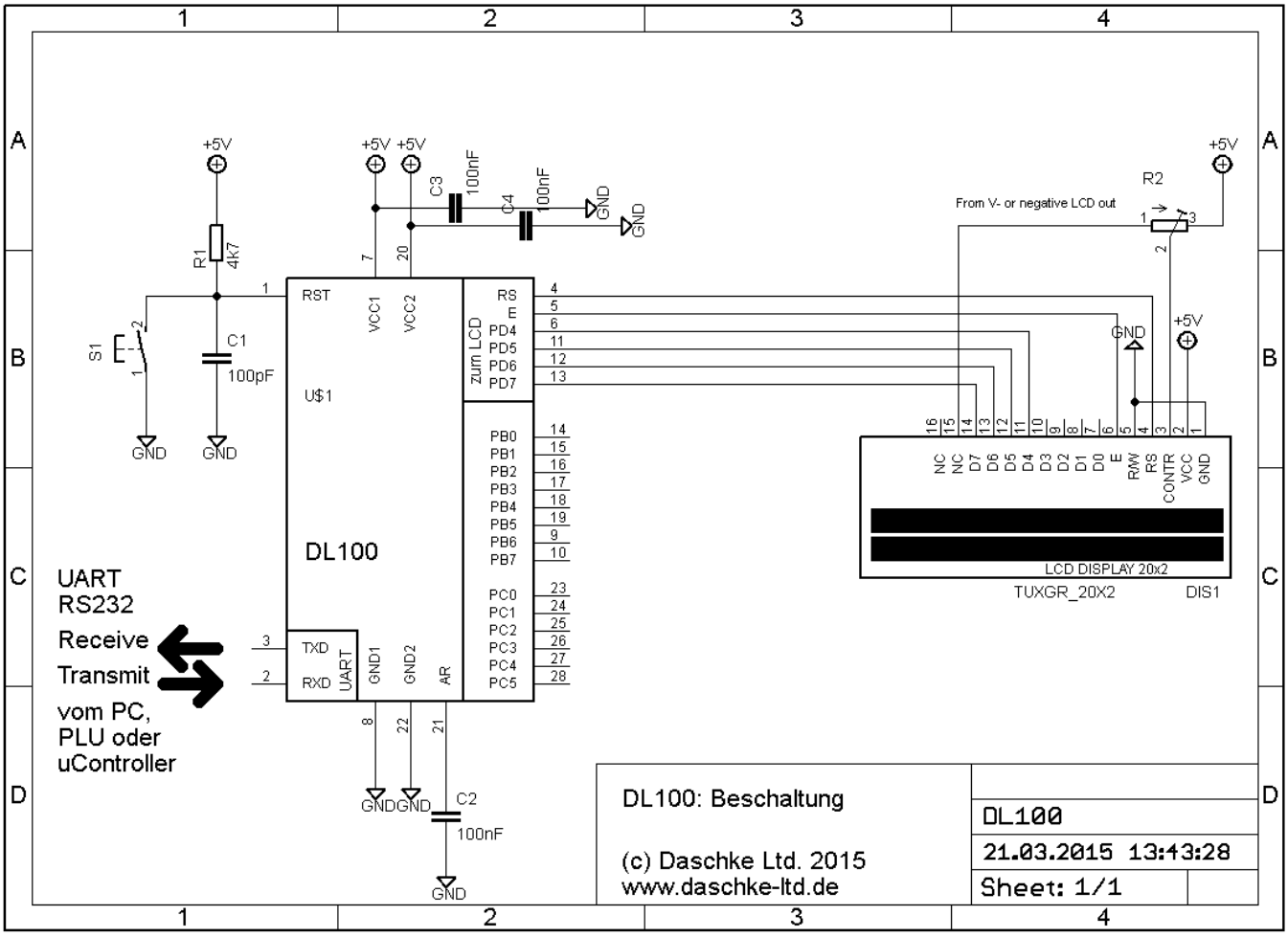
Chip-Spezifikationen:

- Eingang: RS232 (5V-Level)
- Ausgang : Alphanumerische LCDs (1x8, 1x16, 2x16… bis 4x20)
Einschränkung: LCDs mit zwei CS-Eingängen sowie 2x40 und 4x40 werden von diesem Chip nicht unterstützt.
- Übertragungsrate [Baud]: 2400, 4800, 9600, 19200
- Gehäuse: 28 Pin DIP
- Speisung: 3.3VDC - 5VDC
- ROHS (Pb-Free): JA
* Lieferumfang : 28 pol. DIP Chip (ohne LCD)

Blockschaltbild:



Beschaltung:



Anmerkungen zur Beschaltung:

Die RS232 / UART des DL100 arbeitet auf 3V bis 5V-Level. Um den DL100 an eine PC-RS232 an zu schließen, ist der Einsatz eines Treibers (z.B. MAX232) notwendig. Allerdings gibt es heut zu Tage viele USB/RS232-Converter, die an den Leitungen über Spannungen von etwa 4.7V bis ca. 5.5V verfügen. Diese Converter können evtl. ohne MAX232 direkt am DL100 betrieben werden.

In der obigen Schaltung ist eine Reset-Schaltung eingezeichnet (S1, R1, C1). Der Taster S1 ist in der finalen Beschaltung nicht notwendig, da der DL100 nach dem Einschalten einen Reset automatisch ausführt.

Die keramischen Kondensatoren C2, C3 und C4 sind zwingend notwendig, um die Speisespannung zu stabilisieren. Beachten Sie bitte, dass Spannungsschwankungen zu einem Reset des DL100 führen können. Daher sollte die Speisespannung schwankungsfrei sein. Insbesondere darf C2 nicht weggelassen werden!

Das LCD wird über 4 Portleitungen (D4 bis D7 am LCD-Modul) sowie die Steuerleitungen E und RS betrieben. Die Leitung R/W (Read/Write) wird am LCD auf Masse beschaltet. Damit lässt sich in die Register des LCD schreiben, aber nicht lesen. Das Lesen aus dem LCD ist für den Betrieb des DL100 nicht notwendig.

Die Kontrasteinstellung erfolgt über ein Trimmpoti R2.

Die Datenleitungen D0 bis D3 des LCD sind im obigen Bild nicht beschaltet. Je nach Hersteller müssen diese für den 4 Leitungsbetrieb eventuell verdrahtet werden. Hierzu ist das Datenblatt des Herstellers zu beachten.

Zum Betrieb des DL100 ist die TXD-Leitung des Chips nicht notwendig. Über diese Leitung sendet der Chip nach jedem korrekten Text/Befehl (die er empfangen hat) eine Information an die übergeordnete Einheit (PC, PLU, uController etc.). Es ist jedoch ratsam (zumindest in der Erprobungsphase) diese Leitung zu nutzen und die Antworten des Chips zu visualisieren bzw. zu überwachen.
WICHTIG: Falsch gesendete Befehle an den DL100 werden von diesem nicht beachtet und auch nicht quittiert!

Die bidirektionalen Ports B & C haben beim DL100 keine Funktion und sind nur beim DL101 vorhanden.

Befehlsübersicht:

Um Befehle über den PC an den DL100 zu senden, eignen sich Terminal-Programme, die eine ASCII-Übertragung über die TxD- und RxD-Leitungen ermöglichen.

Der DL100 sendet nach jedem empfangenen korrekten Befehl eine Nachricht an die übergeordnete Einheit (PC, PLU, uController etc.) über seine TxD-Leitung zurück. Damit ist es möglich, die Aktivität des DL100 aus der Ferne zu prüfen und zu überwachen.
Empfängt der DL100 einen falschen Befehl, bleibt eine Antwort aus.
HINWIES: Die Antworten des DL100 erfolgen automatisch, wenn die Kommunikation zwischen DL100 und PC funktioniert und der Befehl der Sytax entspricht. Diese Antworten geben keine Information darüber, ob der Chip an der LCD-Anzeige korrekt angeschlossen ist. Die Antworten sollen dem Anwender zeigen, dass der DL100 korrekt funktioniert und dass die Kommunikation über die RS232 intakt ist. Ist der DL100 falsch an der LCD-Anzeige angeschlossen, wird die Anzeige nicht korrekt reagieren und möglicherweise nichts anzeigen, aber die zurückgesandten Antworten des DL100 sind korrekt, sobald die RS232-Kommunikation funktioniert.

LCD-Befehle:

Auslieferungszustand des DL100:

1. Cursor ON, blinkend, eigener Text in Zeilen 1 & 2
2. RS232: 19200Baud, No Parity, 8 Bits, 1 Stopbit

Bezüglich der Kommunikation lässt sich lediglich die Baudrate ändern. Die anderen Parameter der RS232 sind fest und unveränderlich.

Jeder Befehl beginnt mit einem $-Zeichen, jeder an zu zeigende Text beginnt mit einem #-Zeichen. Damit kann der DL100 unterscheiden, ob es sich um einen Befehl, oder einen Text handelt.
Nach dem $-Zeigen erfolgt ein Befehl in Grossbuchstaben.

Syntax für LCD, RS232	Befehl DL100	Antwort des DL100 über die RS232	Beschreibung
$CLS	Lösche LCD-Bildschirm	CLS	Löscht alle LCD-Zeilen
$CHTL	Chiptyp an LCD	-	Zeigt den Chiptyp im LCD an
$CHTR	Chiptyp an RS232	DL100	Sendet den Chiptyp über die RS232 an den PC
$SHR	Shift Right LCD	SHR	Schiebt den Inhalt der LCD-Anzeige um eine Stelle nach rechts
$SHL	Shift Left LCD	SHL	Schiebt den Inhalt der LCD-Anzeige um eine Stelle nach links
$2400	Setze Baudrate	2400	Baudrate wird auf 2400 Baud gesetzt / geändert
$4800	Setze Baudrate	4800	Baudrate wird auf 4800 Baud gesetzt / geändert
$9600	Setze Baudrate	9600	Baudrate wird auf 9600 Baud gesetzt / geändert
$19200	Setze Baudrate	19200	Baudrate wird auf 19200 Baud gesetzt / geändert (Werkseinstellung)
$LCxyy	Locate x,yy	x,yy	Locate, Setzen des Kursors an die Zeile x, Spalte yy (gilt NICHT für 4x16 stellige Anzeigen)
$LDxyy	Locate x,yy	X,yy	Locate, Setzen des Kursors an die Zeile x, Spalte yy (gilt NUR für 4x16 stellige Anzeigen)
$CON	Cursor On	CON	Schaltet den Cursor ein
$COFF	Cursor Off	COFF	Schaltet den Cursor aus
$CBL	Cursor Blink	CBL	Blinkender Cursor eingeschaltet
$CNBL	Cursor No Blink	CNBL	Blinkender Cursor ausgeschaltet
$DON	Display on	DON	Display einschalten (Schrift sichtbar)
$DOFF	Display off	DOFF	Display ausschalten (Schrift unsichtbar)
#Text	Text anzeigen	LCD Text OK	Zeigt Text im LCD an und meldet das OK über die RS232
$SCxxx	Special Character	SCxxx	Darstellung von Sonderzeichen (x = 0…255)

Hinweise & Tips:

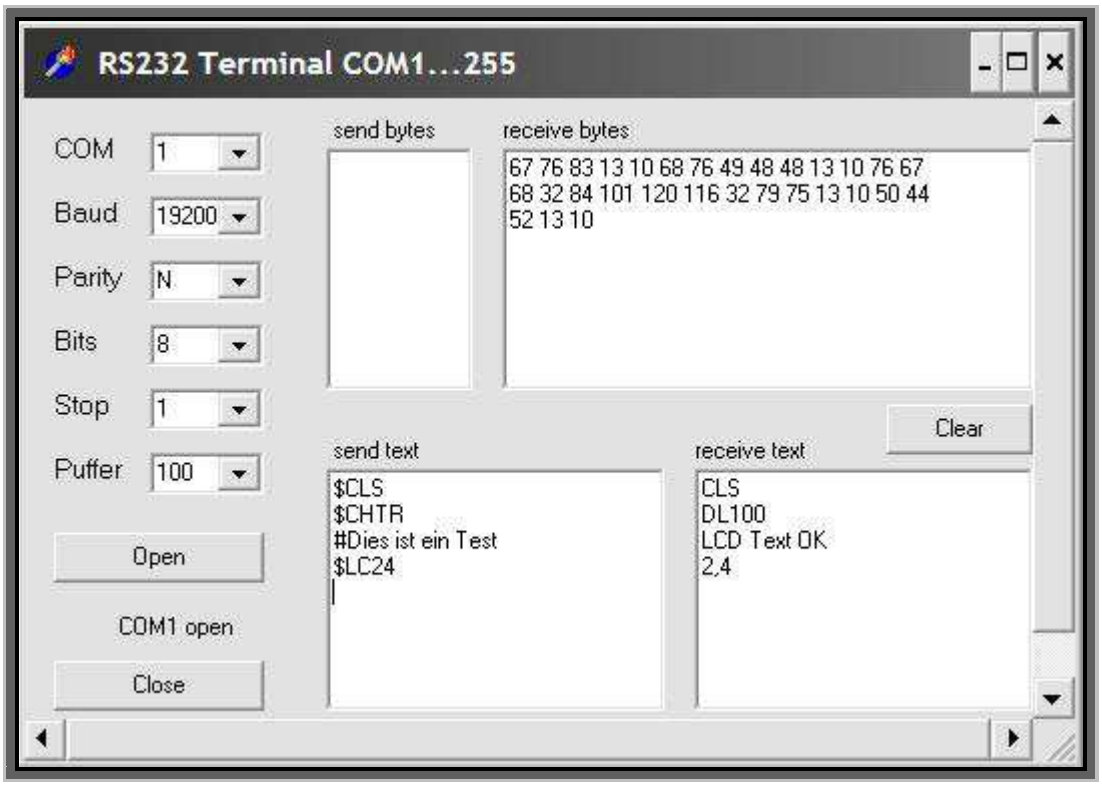
1. Achten Sie stets darauf, dass die anzuzeigenden Texte auch der Textbreite Ihres LCD entsprechen. Ansonsten können vorhandene Texte auf dem LCD durch die neuen (und zu langen) Texte überschrieben werden. Dies hängt von der Art der LCD-Anzeige ab. Hinweise dazu finden Sie im Datenblatt des LCD-Herstellers.
2. Für den Testbetrieb empfiehlt sich immer den Cursor blinkend ein zu schalten.
3. Bei kompatiblen HD44780-LCD-Controllern (z.B. ST7066U, der vom Hersteller Raystar eingesetzt wird) können die Cursorfunktionen unterschiedlich sein. Bei nicht blinkendem Cursor gibt es möglicherweise keinen Unterschied zwischen dem eingeschalteten und ausgeschalteten Cursor. Wenn der Cursor sichtbar sein soll, empfiehlt es sich, die Befehle $CON und $CBL (Cursor On und Cursor Blink) nacheinander zu verwenden. Dann ist sichergestellt, dass ein sichtbarer Cursor - unabhängig vom Controllertyp - vorhanden ist.
4. Ein Reset-Taster ist nicht zwingend notwendig (s. obiges Bild zur Beschaltung des DL100)
5. Das Auslesen der TxD-Leitung des DL100 ist zum Betrieb nicht notwendig. Es genügt, Befehle und Texte an den Chip zu senden, ohne seine Antworten zu kennen. Für Überwachungszwecke oder in der Erprobungsphase ist jedoch die Kenntnis über die Antworten des DL100 durchaus zu empfehlen.
6. Beim Umschalten der Baudrate ist wie folgt vor zu gehen:
   a. Mittels Terminalsoftware die neue Baudrate an DL100 senden ($-Zeichen gefolgt von der Baudrate). Die Kommunikation mit dem DL100 ist nun nicht mehr möglich, da der DL100 auf eine neue Baudrate umgeschaltet hat.
   b. COM-Port des PCs schließen.
   c. Gleiche Baudrate (wie die zuvor an DL100 gesendete) am Terminalprogramm einstellen
   d. COM-Port des PCs wieder öffnen
   e. Kommunikation mit DL100 fortsetzen

LCD-Zustand nach einem Reset oder nach dem Einschalten:
Erste Zeile: DL (in Fettschrift) + 100 + Leerzeichen + Symbol eines 6-poligen ICs + dahinter blinkender Cursor
Zweite Zeile: daschke-ltd.de

Bei korrekter Beschaltung erscheint das obige Bild auf dem LCD. Jegliche Abweichung davon zeugt von einer falschen Beschaltung des DL100 bzw. des alphanumerischen LCD.

Zu beachten ist, dass bei 1x8-zeiligen bzw. 2x8-zeiligen LCDs logischerweise die Schrift nicht komplett ins LCD passt. Entsprechend abgeschnitten bzw. verschoben erscheinen die Texte, die Funktion des DL100 bleibt erhalten.

Ansteuerung mittels Terminalprogramm über PC (Beispiel):



Im obigen Bild sind unten 2 Fenster zu sehen: „send Text“ zeigt die an den DL100 gesendeten Texte & Befehle an. Fenster „receive Text“ zeigt die vom DL100 gesendeten und vom PC empfangenen Daten an.

Zeile 1: Befehl $CLS – Löschen der LCD-Anzeige
Antwort: CLS
Zeile 2: Befehl $CHTR – ChipTyp an RS232 senden
Antwort: DL100
Zeile 3: #Dies ist ein Text – Text wurde gesendet und als „Dies ist ein Text“ im LCD angezeigt
Antwort LCD Text OK
Zeile 4: $LC24 – Locate (Cursorpositionierung) in der 2ten Zeile an der 4ten Stelle (4te Spalte)
Antwort 2,4

Anmerkungen zu den Locate-Befehlen:

Der DL100 steuert alle alphanumerischen Anzeigen an, die über einen Enable-Eingang verfügen. Anzeigen mit 2 Enable-Eingängen (meist bei 4x40 Zeichen LCDs zu finden) werden nicht unterstützt.

Abhängig vom Anzeigentyp werden 2 unterschiedliche Locate-Befehle verwendet. Für 4x16 stellige Anzeigen lautet der Befehl zur Cursorpositionierung $LDxyy, wobei x die Zeile und y die Spalte/Stelle repräsentieren. Für alle anderen Anzeigentypen lautet der Befehl zur Cursorpositionierung: $LCxyy.

Hintergrund:
Bei 4x20 stelligen Anzeigen beginnt die DDRAM-Adresse der 3ten Zeile bei Hex 14, während die 3te Zeile einer 4x16 stelligen Anzeige die DDRAM-Adresse H10 für die erste Stelle hat. Intern wird beim $LD-Befehl die Zahl 4 abgezogen, so dass die Positionierung wieder angepasst wird. Ähnlich verhält es sich bei der 4ten Zeile (Adressen sind H50 und H54).
Weitere Daten zum DDRAM einer alphanumerischen Anzeige finden Sie im Datenblatt des jeweiligen Herstellers.

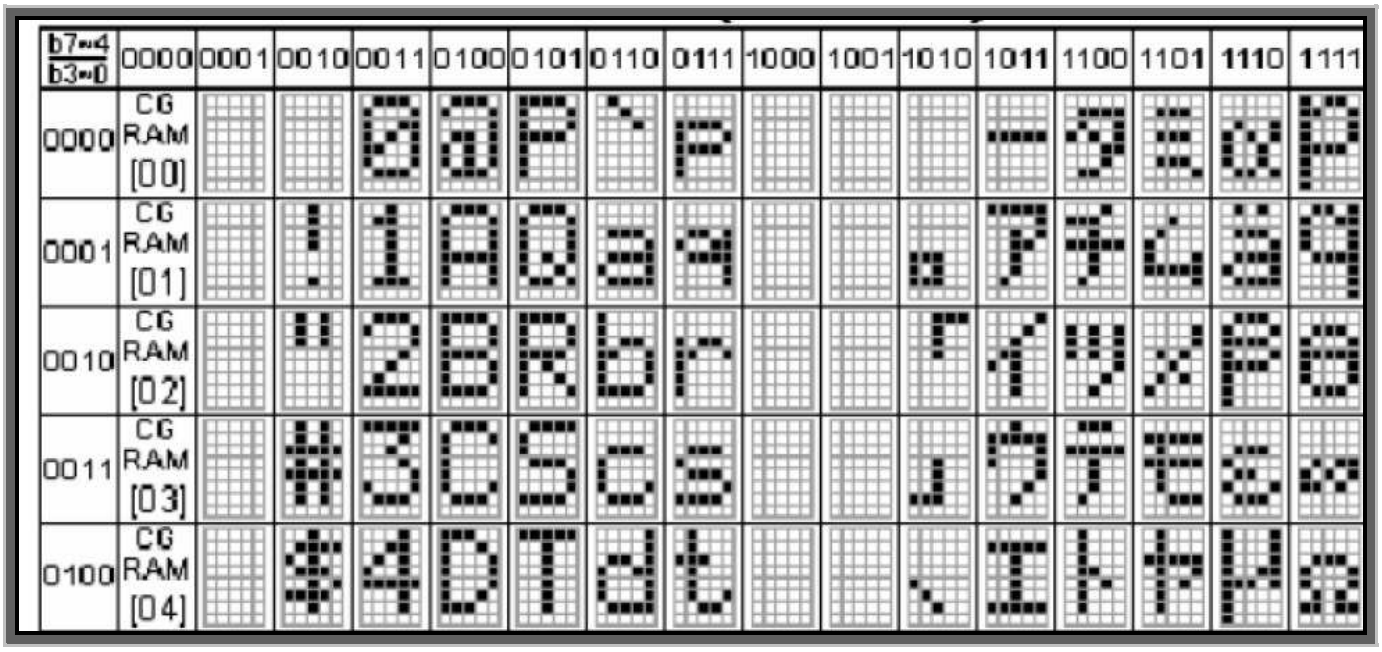
Prinzipiell ist es für den Anwender nur wichtig, bei 4x16 stelligen Anzeigen den Befehl $LD und für alle anderen Anzeigentypen den Befehl $LC zur Cursorpositionierung zu verwenden.

Anzeigen von Sonderzeichen:

Befehl: $SCxxx

Eine alphanumerische Anzeige verfügt über ein so genanntes „Character Generator ROM“ abgekürzt CG ROM. Dies ist ein nichtflüchtiger Speicher, in dem alle darstellbaren Zeichen abgespeichert sind. Dabei handelt es sich um die Zahlen 0 bis 9, das gesamte Alphabet, einige Zusatzzeichen ($%&/! etc.) sowie weitere (herstellerspezifische) Sonderzeichen.

Im Datenblatt des Herstellers befindet sich eine Tabelle, die Auskunft über die Zeichen und die dazugehörigen Speicherstellen Auskunft gibt. Diese Tabelle sieht in etwa so aus:



Je nach Hersteller befinden sich im rechten Block unterschiedliche Zeichen, abhängig davon, für welchen Kunden bzw. für welches Land das LCD-Modul hergestellt wurde.
Der Zugriff auf alle 256 Zeichen erfolgt im DL100 über den Befehl $SCxxx, wobei xxx der Speicherstelle entspricht, die dem gewünschten Zeichen angehört.

Beispiele:
$SC65 LCD zeigt den Buchstaben A an.
$SC127 LCD zeigt „←“ an

Somit lassen sich alle Standardzeichen (Alphabet + Zahlen) aber auch Sonderzeichen im LCD anzeigen.

Kundenspezifische Sonderzeichen:

Der DL100 verfügt über 8 Sonderzeichen, die von uns kundenspezifisch und gegen Aufpreis für Sie angefertigt werden können. Nach der Implementation in den Chip stehen Ihnen diese Sonderzeichen in den Speicherstellen 0 bis 7 zur Verfügung.

Im Auslieferungszustand sind folgende Sonderzeichen implementiert:
Speicher 0: Fettes D / Speicher 1: Fettes L / Speicher 2: Zeichen eines 6-poligen ICs
Mit diesen 3 Sonderzeichen wird das Startbild des Displays erzeugt.
Des Weiteren sind folgende Sonderzeichen implementiert
Speicher 3: senkrechter Balken / Speicher 4: zwei senkrechte Balken / Speicher 5: drei senkrechte Balken / Speicher 6: vier senkrechte Balken / Speicher 7: fünf senkrechte Balken
Damit lässt sich eine Balkenanzeige (Bargraph) realisieren.

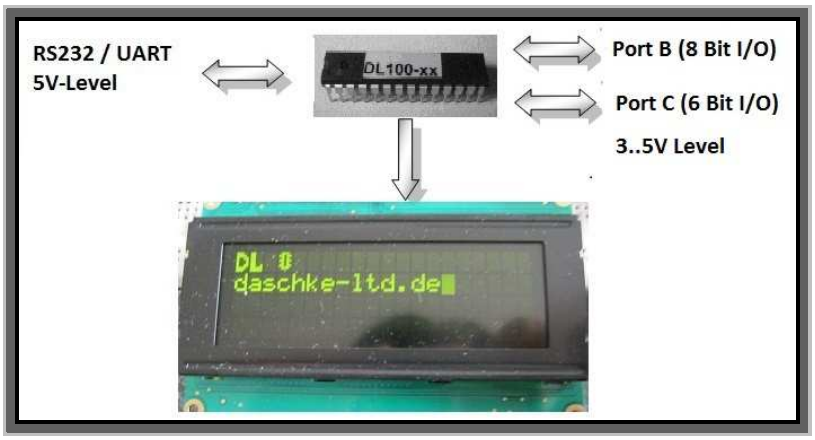
Sollten Sie 8 Sonderzeichen wünschen, die nach Ihrer Spezifikation implementiert sein sollen, kontaktieren Sie uns bitte. Wir programmieren Ihnen diese Zeichen in den Chip ein und Sie können diese über den Befehl $SC0 bis $SC7 ansprechen und zur Anzeige ins LCD bringen.

DL101:

Der DL101 verfügt über die gleichen Spezifikationen (und die gleichen Befehle), wie der DL100. Zusätzlich stehen beim DL101 zwei bidirektionale I/O- Ports zur Verfügung: ein 8 Bit I/O-Port und ein 6 Bit I/O-Port. Entsprechend existieren zusätzliche Befehle, die die Steuerung der Ports managen.

Für den DL101 gelten alle zuvor beim DL100 genannten Befehle, Anmerkungen, Spezifikationen und Skizzen. Der DL101 ist also die Erweiterung des DL100 um 2 bidirektionale Ports.

Prinzip des DL101:



Erweiterter Befehlssatz für DL101:

Syntax für Ports, RS232	Befehl DL101	Antwort des DL101 über die RS232	Beschreibung
$PBI	Port B = Input	PBI	Setzt Port B als Eingangsport
$RPB	Read Port B	xxx	Liest die 8 Bit breite Zahl von Port B (0 bis 255) und sendet diese Zahl an die RS232
$PBO	Port B = Output	PBO	Setzt Port B als Ausgangsport. Danach hat Port B den Wert 255 (Alle 8 Leitungen auf 1 gesetzt)
$BBxxx	Set Port B to xxx	BBxxx	Setzt Port B auf die Zahl xxx (0 bis 255)
$PCI	Port C = Input	PCI	Setzt Port C als Eingangsport
$RPC	Read Port C	xxx	Liest die 6 Bit breite Zahl von Port C (0 bis 63) und sendet diese Zahl an die RS232
$PCO	Port C = Output	PCO	Setzt Port C als Ausgangsport. Danach hat Port B den Wert 63 (Alle 6 Leitungen auf 1 gesetzt)
$CCxxx	Set Port C to xxx	CCxxx	Setzt Port C auf die Zahl xxx (0 bis 63)

Anmerkungen zu den Ports B & C:

Port B ist 8 Bit breit. Entsprechend lassen sich an ihn die Zahlen 0 bis 255 darstellen (bzw. daraus ablesen).
0 = 0000 0000
255 = 2^8 -1 = 1111 1111
Analog dazu sehen die Berechnungen für Port C aus, wobei Port C über nur 6 Bits verfügt.
0 = 00 0000
63 = 2^6 -1 = 11 1111

Per Definition werden alle Leitungen der beiden Ports auf 1 gesetzt, wenn die Ports als Ausgänge definiert werden. Dies bedeutet, dass nach dem Befehl $PBO der Port B die Zahl 255 und nach dem Befehl $PCO der Port C die Zahl 63 aufweisen (alle Leitungen liegen auf Niveau der Speisespannung).

Auf Kundenwunsch (ohne Aufpreis) können beide Ports die Zahl 0 (nach dem Setzen als Ausgangsport) aufweisen (alle Leitungen führen dann 0V Spannung). Kontaktieren Sie uns bitte, wenn Sie diesbezüglich eine Änderung für Ihren Chip wünschen.

Offene Porteingänge können Schwanken.
Beispiel: Senden Sie bitte den Befehl $PBI (Port B ist als Eingang). Wenn nun keine Spannung an den Portleitungen anliegt (also alle Eingänge des Port B offen sind), so erhält man über die RS232 (nach Absetzen des Befehls $RPB = Read Port B) unterschiedliche Werte: 0, 16, 34, 24 etc.

Nicht belegte Eingangs-Portleitungen können auf Masse verschaltet werden und erhalten dann den Wert 0. Zu empfehlen ist es, nicht belegte Eingangs-Portleitungen mittels eines Widerstandes zu Masse zu schalten (z.B 10KΩ). Dies verhindert Kurzschlüsse, für den Fall, dass danach versehentlich der Befehl $PBO (Port B = Ausgang) abgesetzt wird.
Man kann sich aber die Verdrahtung der offenen Portleitungen ersparen. In diesem Falle sind nicht belegte Portleitungen in der übergeordneten Software zu ignorieren. Dies ist prinzipiell die bessere Alternative.