Bedienungsanleitung

LCD-Terminal

LCD-Terminal: Manual V1.03d

taskit GmbH

Groß-Berliner Damm 37
D 12487 Berlin

Telefon +49 (0) 30 611 295 – 0
Fax +49 (0) 30 611 295 – 10
www.taskit.de

Inhaltsverzeichnis

1. Verzeichnisübersicht

Abbildungsverzeichnis

Tabellenverzeichnis

Anwendungsbeispiele

2. Eigenschaften

Das LCD-Term ist ein direkt verwendbares Monochrom-Display mit VT100-Unter-stützung, drei integrierten Schriftarten, Unterstützung für Matrixtastaturen, einfachen E/A(I/O)-Funktionen und grafischen Routinen wie Zeichnen von Grundelementen, GIF-Bildern sowie Blitting von und zu mehreren Back-Buffer. Es ist in vier Displaygrößen von 1,2" bis 3,5" erhältlich.

Jede Hardware, die um einige Ausgabefähigkeiten oder volle Benutzerinteraktion erweitert werden soll, kann an das LCD-Term angeschlossen werden. Durch die Verwendung von Standard-Peripheriegeräten wie USB oder einer seriellen Schnittstelle, die auf vielen Plattformen leicht verfügbar sein sollte, werden die Anforderungen gering gehalten.

Das Design ist auf einfache Weise zu bedienen: Vier Leitungen reichen aus, um die volle Anzeigefunktionalität zu gewährleisten. Alle weiteren E/A(I/O)-Pins können für die Verwendung einer Matrixtastatur oder als E/A(I/O)-Pins konfiguriert werden. Die Konfiguration erfolgt über eine integrierte Setup-Shell, auf die über ein einfaches Terminalprogramm zugegriffen werden kann. Es ist keine Konfigurationssoftware erforderlich, die auf einem Host-PC ausgeführt werden müsste. Somit ist das Testen, Konfigurieren und Entwickeln unabhängig von einer bestimmten Betriebssystemplattform.

Das Integrieren der LCD-Term-Funktionalität in vorhandene Softwareprojekte ist ebenfalls einfach. Es gibt keine Bibliothek, die für die Interaktion mit der Firmware verwendet werden muss. Alle Befehle sind Textnachrichten, die direkt über die serielle Verbindung gesendet werden. Entwickler sind nicht auf eine bestimmte Programmiersprache beschränkt - Sie benötigen lediglich eine einfache Manipulation der Zeichenfolge und Zugriff auf die serielle Schnittstelle oder USB. Daher sind die meisten Beispiele entweder Nachrichten, die in ein serielles Terminal eingegeben wurden, oder ein kurzer LUA-Code[1].

[1]LUA ist eine kleine und simple Scriptsprache. Siehe: http://www.lua.org

3. Inbetriebnahme

Die Stromversorgung erfolgt entweder über die 5V-Pins eines USB-Hosts, der an den Micro-USB-Anschluss angeschlossen ist, oder über die seitlichen Anschlüsse mit dem 3,3-V- oder 5-V-Pin, falls verfügbar.

Abbildung 2.1. Pinbelegung LCDTerm12

Abbildung 2.2. Pinbelegung LCDTerm15

Abbildung 2.3. Pinbelegung LCDTerm25

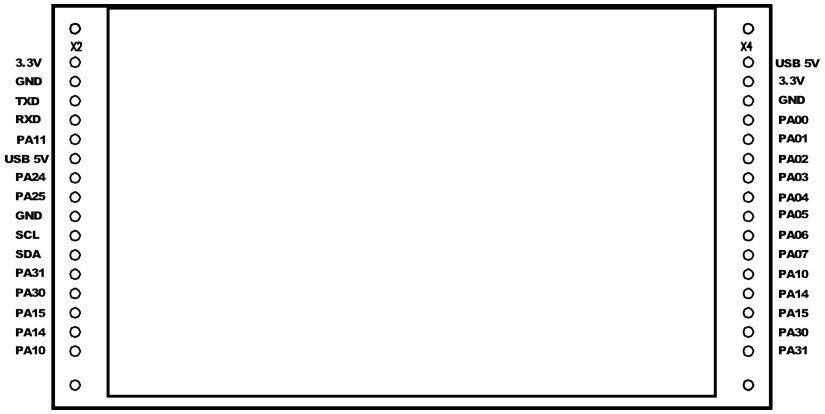

Abbildung 2.4. Pinbelegung LCDTerm35

3.1. USB

Wenn Sie ein USB-Kabel über den Micro-USB-Anschluss X3 anschließen, wird das Gerät als USB-CDC-Gerät aufgeführt. In modernen Betriebssystemen wie Windows 10, MAC OS X oder Linux sind entsprechende Treiber vorinstalliert.

Nach der Initialisierung wird dem System ein neues Kommunikationsgerät hinzugefügt. Unter Windows heißt dies `//./COMx` wobei *x* eine Dezimalzahl ist. Linux- und UNIX-ähnliche Systeme verwenden im Allgemeinen entweder `/dev/ttyACMx` oder `/dev/ttyUSBx` als Namensschema.

Die einfache Textausgabe kann auf ähnliche Weise von fast jeder Shell aus erfolgen.

Beispiel 2.1. 'Hello world' aus der Windows Shell heraus

```
echo "Hello world !" > //./COM5
```

3.2. UART

Die UART-Schnittstelle ist für die Verwendung in 'embedded Systemen' konzipiert, um die Kosten so niedrig wie möglich zu halten. Demnach werden keine RS232-LVTTL-Pegelumsetzer verwendet. Die meisten in einer solchen Umgebung verwendeten Mikroprozessoren / Mikrocontroller können direkt an das LCD-Term angeschlossen werden.

Vorsicht

Für die Verwendung des UART als Verbindung zu einem gemeinsamen PC sind RS232-LVTTL-Pegelumsetzer erforderlich. Alternativ können LVTTL-zu-USB-Wandler wie ein FTDI-Kabel oder ein Chip verwendet werden. Diese Lösung wird bereits durch den USB-Anschluss von LCD-Term abgedeckt. Die Unterstützung von USB-Treibern kann jedoch bei älteren Betriebssystemen einfacher sein, wenn FTDI-Chips / -Kabel verwendet werden.

3.3. Konfiguration

LCD-Term bietet einen eingebauten Setup-Dialog zur Konfiguration. Sie kann entweder durch Kurzschließen von GND und PA00 während des Einschaltens oder durch eine in ein serielles Terminalprogramm eingegebene Escape-Sequenz eingegeben werden. Die Sequenz ist ESC [Q. Weitere Informationen finden Sie in Tabelle 3.4, „Zusätzliche Funktionen".

Der Setup-Dialog kann über ein Terminalprogramm aufgerufen werden. Sie können USB oder die UART-Verbindung verwenden. Befolgen Sie die Anweisungen im Menü, um die Hintergrundbeleuchtung, den Kontrast und die E/A(I/O)-Pins zu konfigurieren. Dann speichern und das Setup beenden. Bei Auslieferung wird LCD-Term mit angemessenen Werten konfiguriert.

LCD-Term kann GIF-Bilder aus seinem internen Flash-Speicher anzeigen. Zum Zeitpunkt des Schreibens dieser Dokumentation ist für das Hochladen von Bildern ein Drittanbieter-Tool des Herstellers des verwendeten Mikrocontrollers erforderlich. Da wir dieses Verfahren in Zukunft vereinfachen werden, wird es hier nicht behandelt.

Weitere Informationen und Tools finden Sie im GIF2LCD-Term Archiv *(anklicken)*

4. Befehlssatz

Alle Anweisungen folgen einem Grundschema, das mit ANSI-Escape-Sequenzen kompatibel ist. Eine Sequenz beginnt mit einem einleitenden Zeichen (Hexcode 0x1B), das im Folgenden als ESC geschrieben wird, und wird entweder durch einen lateinischen Groß- oder Kleinbuchstaben (A-Z, a-z) vervollständigt, der einen Funktionscode darstellt. Argumente für diese Funktionen werden als Dezimalzahlen geschrieben, die durch ein Semikolon zwischen ESC und dem Funktionscode getrennt sind.

Alle druckbaren ASCII-Codes, die nicht Teil einer aktiven Escape-Sequenz sind, werden im Display angezeigt. Der Textcursor bewegt sich dabei automatisch vorwärts.

4.1. Textfunktionen

Textfunktionen werden verwendet, um die Ausgabeposition der zu druckenden Zeichen zu steuern, die Anzeige oder Teile davon zu löschen, den Inhalt der Anzeige zu scrollen und verschiedene Betriebsmodi wie die inverse Darstellung umzuschalten. Hier ist eine Liste aller Textfunktionen.

Tabelle 3.1: Textfunktionen

Befehl	Beschreibung	Beispiel(e)	LUA-String
Curser-Funktionen (Text)			
A	Bewege Curser *n* Zeilen aufwärts. Ohne Angabe von *n* entspricht das *n*=1.	`ESC[A ESC[2A`	`"\27[2A"`
B	Bewege Curser *n* Zeilen abwärts. Ohne Angabe von *n* entspricht *n* = 1.	`ESC[B ESC[2B`	`"\27[2B"`
C	Bewege Curser *n* Zeilen nach rechts. Ohne Angabe von *n* entspricht *n* = 1.	`ESC[C ESC[12C`	`"\27[12C"`
D	Bewege Curser *n* Zeilen nach links. Ohne Angabe von *n* entspricht *n* = 1.	`ESC[D ESC[16D`	`"\27[16D"`
a	Setze Curser auf Zeile *n*. Ohne Angabe von *n*, wird auf die erste Zeile gesetzt.	`ESC[a ESC[2a`	`"\27[2a"`
c	Setze Curser auf Spalte *n*. Ohne Angabe von *n*, wird auf die erste Spalte gesetzt.	`ESC[c ESC[5c`	`"\27[5c"`
H, f	Setze Curser auf Poition [y; x]. Ohne Parameter wird der Curser auf [1; 1] gesetzt.	`ESC[H ESC[2;12f`	`"\27[2;12H"`
s	Aktuelle Cuserposition speichern.	`ESC[s`	`"\27[s"`
u	Gespeicherte Curserposition wieder herstellen.	`ESC[u`	`"\27[u"`
Modus setzen			
z	Fontgröße auf *n* setzen. Verfügbare Größen sind: 0(8x6; default), 1(8x8), 2(16x8). Ohne Angabe von *n*, wird 8x6 gesetzt. Jede Änderung setzt den Curser auf Position [1;1].	`ESC[z ESC[2z`	`"\27[2z"`
l	Deaktiviere Modus *n*. Unterstützte Modi sind: 1, 2, 3, 8, 25, 75. Siehe Tabelle 3.2, “Optionen” für Details.	`ESC[?2l ESC[?25l`	`"\27[?2l"`
h	Aktiviere Modus *n*. Unterstützte Modi sind: 1, 2, 3, 8, 25, 75. Siehe Tabelle 3.2, “Optionen” für Details.	`ESC[?2h ESC[?25h`	`"\27[?2h"`

Befehl	Beschreibung	Beispiel(e)	LUA-String
Display Löschfunktionen			
J	Löschen den Anzeigeinhalt gemäß Option *n*. Optionen sind: 0 = vom Cursor bis zum Ende, 1 = vom Anfang bis zum Cursor, 2 oder weglassen von *n* löscht die gesamte Anzeige.	`ESC[2J ESC[J`	`"\27[2J"`
K	Zeileninhalt gemäß Option löschen *n*. Folgende Optionen stehen zur Verfügung: 0 = vom Cursor zum Zeilenende, 1 = vom Zeilenanfang zum Cursor, 2 = ganze Zeile. Durch Weglassen von *n* wird die aktuelle Zeile vom Cursor bis zum Ende gelöscht.	`ESC[2K ESC[K`	`"\27[2K"`
P	Löschen *n* Zeichen rechts vom Cursor. Das Zeichen an der Cursorposition wird ebenfalls gelöscht. Wenn *n* weggelassen wird, wird ein Zeichen gelöscht.	`ESC[P ESC[4P`	`"\27[4P"`
M	Löschen *n* Zeichen links vom Cursor. Das Zeichen an der Cursorposition wird ebenfalls gelöscht. Wenn *n* weggelassen wird, wird ein Zeichen gelöscht.	`ESC[M ESC[4M`	`"\27[4M"`
Scroll-Funktionen			
S	Scrollt den Anzeigeinhalt *n* Textzeilen nach oben. Wenn n weggelassen wird, wird um eine Zeile gescrollt.	`ESC[S ESC[1S`	`"\27[2S"`
T	Scrollt den Anzeigeinhalt *n* Textzeilen nach unten. Wenn n weggelassen wird, wird um eine Zeile gescrollt.	`ESC[T ESC[1T`	`"\27[2T"`

Mit den Befehlen l und h werden mehrere Optionen deaktiviert bzw. aktiviert, die die Textdarstellung beeinflussen. Die Beschreibung dieser Optionen finden Sie in Tabelle 3.2, „Optionen“.

Tabelle 3.1: Optionen

Option	***h*: aktivieren**	***l*: deaktivieren**
1	Wenn die Cursoranzeige aktiviert ist, wird der Cursor als großer Block gezeichnet.	Wenn die Cursoranzeige aktiviert ist, wird der Cursor als dünne Linie gezeichnet.
2	Cursoranzeige wird eingeschaltet.	Cursoranzeige wird ausgeschaltet.
25	Cursorblinken wird eingeschaltet.	Cursorblinken wird ausgeschaltet.
3	Der Text wird invers dargestellt. Der Hintergrund ist dunkel	Der Text wird normal dargestellt. Der Hintergrund ist (normal) hell
8	Der automatische Zeilenumbruch wird eingeschaltet. Beim Umbruch an der letzten Zeile wird um eine Zeile gescrollt.	Der automatische Zeilenumbruch wird ausgeschaltet.

Dieser LUA-Beispielcode zeigt Text sowie die aktuelle Uhrzeit und das aktuelle Datum in zwei verschiedenen Modi oben auf dem Bildschirm an.

Beispiel 3.1. example1.lua

```
-- All commands will be sent using standard file IO
-- on the USB CDC port my LCD-Term is connected to.
-- On Linux it will be either "/dev/ttyACMx" or "/dev/ttyUSBx";
-- on Windows the device is "//./COMx".
f = io.open("/dev/ttyACM0", "wb")

function wait(seconds)
 local start = os.clock()
 repeat until os.clock() > start + seconds
end

function clear_display(f)
 f:write("\27[2J")
 f:flush()
 wait(0.01)
end

function clear_line(f)
 f:write("\27[2K")
 f:flush()
 wait(0.01)
end

function gotoYX(f, y, x)
 -- insert y and x at their appropriate positions
 -- format: ESC[y;xH
 f:write(string.format("\27[%d;%dH", y, x))
 f:flush()
 wait(0.01)
end

-- disable inverse mode
f:write("\27[?3l")
-- enable automatic line wrap
f:write("\27[?8h")

clear_display(f)

gotoYX(f, 4, 1)
f:write("\nSome text here...\nand here...\n\nand there.")

-- enable inverse mode
f:write("\27[?3h")
-- disable automatic line wrap
f:write("\27[?8l")

-- clear first three lines
for y = 1, 3, 1 do
 gotoYX(f, y, 1)
 clear_line(f)
end

date_format = "%H:%M:%S %Y/%m/%d"

while true do
 -- output current time
 gotoYX(f, 2, 2)
 f:write(os.date(date_format))
 f:flush()
end

-- never reach this because of loop for infinity
f:close()
```

4.2. Grafikfunktionen

Neben der Textfunktionalität bietet LCD-Term eine Reihe von Sequenzen, die zum Zeichnen verwendet werden können. Einfache Formen wie Linien, Rechtecke und Kreise können mit unterschiedlichen Mustern gefüllt werden, Text kann pro Pixel anstatt pro Zeichen positioniert werden, ein Bildschirm kann im Hintergrund-Buffer vorbereitet und in den sichtbaren Buffer eingeblendet werden. Symbole und spezielle Symbole werden über die GIF-Funktion unterstützt.

Die meisten Funktionen werden nicht direkt auf dem Bildschirm angezeigt, sondern in einen der beiden Hintergrund-Buffer gerendert. Dies ermöglicht das Einrichten eines Bildschirms, ohne den Benutzer mit mehreren Aktualisierungen derselben Region zu belästigen (z. B. wenn etwas über ein Bild gezeichnet wird). Es gibt eine Funktion, die die Bildschirmaktualisierung steuert und erzwingt, die aufgerufen werden kann, nachdem das Rendern des Hintergrund-Buffers abgeschlossen ist.

Tabelle 3.3: Grafikfunktionen

Befehl	Beschreibung	Beispiel(e)	LUA-String
Grafische Grundelemente			
x	Setzt oder löscht ein Pixel an Position [y; x]. Der dritte Parameter c entscheidet, ob das Pixel gesetzt (c> 0) oder gelöscht (c = 0) ist.	ESC[*y*;*x*;*c*x ESC[10;40;1x	"\27[10;20;1x"
y	Zeichnet eine Linie von [y0; x0] bis [y1; x1]. Der letzte Parameter p gibt ein sich wiederholendes Muster an, bei dem jedes Bit einem Pixel entspricht. Ein gesetztes Bit führt zu einem gesetzten Pixel.	ESC[*y0*;*x0*;*y1*;*x1*;*p*y ESC[0;0;63;127;255y	"\27[63;0;0;127;85y"
v	Zeichnet ein Rechteck von [y0; x0] bis [y1; x1]. Der letzte Parameter *p* gibt ein sich wiederholendes Muster an, das für die Umrisse des Rechtecks verwendet werden soll.	ESC[*y0*;*x0*;*y1*;*x1*;*p*v ESC[0;0;63;127;15v	"\27[63;0;0; 127;111v"
w	Füllt ein Rechteck von [y0; x0] bis [y1; x1]. Der letzte Parameter *p* (0-15) ist die Nummer eines von sechzehn Füllmustern.	ESC[*y0*;*x0*;*y1*;*x1*;*p*w ESC[0;0;63;127;7w	"\27[62;1;1;126;3w"
k	Zeichnet / füllt einen Kreis mit dem Mittelpunkt [y; x] und Radius *r*. Der Parameter *f i l l* wählt aus, ob *p* eine Füllung (*f i l l* = 1) oder ein Umrissmuster ist.	ESC[*y*;*x*;*r*;*fill*;*p*k ESC[32;64;20;1;7k	"\27[32;64;30;0; 111k"
Grafik-Cursor			
G	Stellt den Grafikcursor auf die Pixelkoordinaten [y; x]. Textfunktionen sowie Druckzeichenfolgen funktionieren relativ zu dieser Position.	ESC[*y*;*x*G ESC[10;30G	"\27[57;100G"

Befehl	Beschreibung	Beispiel(e)	LUA-String
GIF-Untersützung			
Y	GIF-Nummer *num* an Position [x; y]. Der Parameter *sub-image* wählt das Unterbild aus, das in Multibild-GIFs angezeigt werden soll. Nur *num* ist obligatorisch; ausgelassene Parameter sind standardmäßig 0.	ESC[*num*;*x*;*y*;*sub-image*Y ESC[1;32;16Y	"\27[3;16;16;4Y"
Pufferspeicherverwaltung			
V	Wechselt zur Puffernummer *num*. Alle weiteren Zeichnungen werden im ausgewählten Puffer ausgeführt.	ESC[*num*V ESC[0V	"\27[1V"
U	Aktualisierungsbildschirm: Je nach Modus *mode* ist die Funktion Automatische Aktualisierung deaktiviert (*mode* = 0) oder aktiviert (*mode* = 1). Wenn *mode* = 2 ist oder weggelassen wird, wird der aktuell ausgewählte Puffer auf den Bildschirm kopiert.	ESC[*mode*U ESC[0U ESC[1U ESC[2U	"\27[U"
W	Quellrechteck (Block) an [x0; y0] mit der Größe [dx; dy] bis [x1; y1]. Der Quellpuffer ist *src*, der Zielpuffer ist *dst*.	ESC[*src*;*x0*;*y0*;*dx*;*dy*;*dst*;*x1*;*y1*W ESC[1;0;0;32;32;0;10;15W	"\27[1;10;20;16;16;0;0;0W"

Hier ist ein komplexeres Beispiel mit grafischen Funktionen zum Zeichnen eines animierten Zufallsgraphen.

Beispiel 3.2. example2.lua

```
-- All commands will be sent using standard file IO
-- on the USB CDC port my LCD-Term is connected to.
-- On Linux it will be either "/dev/ttyACMx" or "/dev/ttyUSBx";
-- on Windows the device is "//./COMx".
f = io.open("/dev/ttyACM0", "wb")

function wait(seconds)
 local start = os.clock()
 repeat until os.clock() > start + seconds
end

math.randomseed(os.time())

graph_x = 10
graph_y = 10
graph_dx = 128 - 2 * graph_x
graph_dy = 64 - 2 * graph_y
variance = 5
steps = 4
pattern = 0xFF
values = {}
values[0] = math.random() * graph_dy

function calc_value(old_y)
 y = old_y + (math.random() - 0.5) * variance, graph_dy
 -- shrink value to graph size
 y = math.max(math.min(y, graph_dy - 1), 0)

 return y
end

function calc_graph(start_x, end_x)
 for x = start_x + 1, end_x, 1 do
  -- round value to the nearest integer
  y = math.floor(calc_value(values[x - 1]) + 0.5)
  values[x] = y
  old_y = y
 end
end

function draw_graph(f, start_x, end_x)
 for x = start_x, end_x - 1, 1 do
  -- draw a line between the two points
  f:write(string.format("\27[%d;%d;%d;%d;%dy",
                        graph_y + values[x], graph_x + x,
                        graph_y + values[x + 1], graph_x + x + 1,
                        pattern))
  f:flush()
  -- here we don't use XON/XOFF, so make a little pause after
  -- having drawn some lines
  if (x % 8) == 0 then
   wait(0.01)
  end
 end
end
```

```
-- draw frame
f:write(string.format("\27[%d;%d;%d;%d;%dw", 0, 0, 63, 127, 4))
f:flush()
if (graph_y > 4) and (graph_x > 4) then
 f:write(string.format("\27[%d;%d;%d;%d;%dw",
                       graph_y - 4, graph_x - 4,
                       graph_y + graph_dy + 3,
                       graph_x + graph_dx + 3, 0))
 f:flush() f:write(string.format("\27[%d;%d;
 %d;%d;%dv",
                       graph_y - 2, graph_x - 2,
                       graph_y + graph_dy + 1,
                       graph_x + graph_dx + 1, 0x33))
 f:flush()
end
wait(0.01)

calc_graph(0, graph_dx - 1)

while true do
 -- clear graph using filled rectange
 f:write(string.format("\27[%d;%d;%d;%d;%dw",
                       graph_y, graph_x,
                       graph_y + graph_dy - 1,
                       graph_x + graph_dx - 1, 0))
 f:flush()
 wait(0.05)

 -- shift graph left, add new values
 for x = 0, graph_dx - 1 - steps, 1 do
  values[x] = values[x + steps]
 end
 calc_graph(graph_dx - 1 - steps, graph_dx - 1)

 -- draw graph
 draw_graph(f, 0, graph_dx - 1)
 f:flush()
 wait(0.05)

 -- update screen
 f:write("\27[U");
 f:flush()

 wait(0.05)
end

-- never reach this because of loop for infinity
f:close()
```

4.3. Zusätzliche Funktionen

Alle anderen Funktionen wie die Steuerung von E/A(I/O), Hintergrundbeleuchtung oder die Kontrasteinstellungen des Displays sind in Tabelle 3.4, „Zusätzliche Funktionen“ zusammengefasst.

Tabelle 3.4: Zusätzliche Funktionen

Befehl	Beschreibung	Beispiel(e)	LUA-String
Controlling display settings			
i, I	Set display background light to *num*.	ESC[0i ESC[255i	"\27[128i"
j	Set display's contrast to *num*.	ESC[0j ESC[63j	"\27[35j"

Befehl	Beschreibung	Beispiel(e)	LUA-String
IO-Funktionen			
N	Get pin level of requested IO pin *n*[a].	ESC[2N	"\27[4N"
O	Set pin level of selected IO pin *n*[a] to `level`.	ESC[2;0O ESC[2;1O	"\27[1;1O"
Konfiguration			
Q	Enter LCD-Term's setup dialog.	ESC[Q	"\27[Q"
e	Jump into the device's bootloader.	ESC[e	"\27[e"

[a]Siehe Abschnitt 4.2, “Anschluss-Schema” für Details bezüglich *n*.

Beim Hochladen von GIF-Bildern muss die Firmware beendet und zuerst der Bootloader aufgerufen werden. Dieses Beispiel zeigt, wie dies erreicht wird.

Beispiel 3.3. Den Bootloader in der Linux Shell aktivieren

```
echo -e "\033[e" > /dev/ttyACM0
```

5. Technische Daten

Tabelle 4.1: Elektrische und mechanische Hauptdaten

Spezifikation	Wert				Einheit
	LCD-Term12	LCD-Term15	LCD-Term25	LCD-Term35	
Versorgungsspannung	3.3 .. 5				V
Leistungsaufnahme (min.)	38				mW
Leistungsaufnahme (max.)	160				mW
Zulässige Betriebstemperatur	-30 .. +85				°C
Auflösung	128 x 64				Pixel
Breite	25	29.4	43.2	53	mm
Länge	49.1	59.3	74.4	94.8	mm
Höhe	5.5	6.0	6.8	7.5	mm
Pixelfläche	26.8 x 12.0	35.8 x 19.8	47.3 x 26.2	66.5 x 33.2	mm^2
IO-Pins	5	8	12	14	

Eine detailliertere Auflistung des Energiebedarfs in Abhängigkeit von der Hintergrundbeleuchtung ist in Tabelle 6, „Stromverbrauch“, dargestellt.

Tabelle 4.2: Stromaufnahme

Hintergrundbeleuchtung / Betriebsmodi	aus	25%	50%	100%	Einheit
Leerlauf [a]	7.6	13.4	18.0	26.6	mA
Vollast [a]	13.1	18.5	23.1	31.5	mA

[a]LCD-Term35: V = 4.9V; Baud = 115200; Kontrasteinstellung = 35

5.1. Spannungsversorgung

Bei der Stromversorgung sind einige Dinge zu beachten. Beim Anschließen des USB-Geräteanschlusses verwendet LCD-Term die + 5V-Leitung vom angeschlossenen Host. In diesem Fall werden aus dieser Quelle auch +3,3 V erzeugt.

Vorsicht

Bei Verwendung von USB darf keiner der als 5 V bezeichneten Pins an eine zweite Stromquelle angeschlossen werden!

Wenn USB nicht verwendet wird - weder zur Stromversorgung noch als Kommunikationsanschluss - kann ein beliebiger 5-V-Pin als Versorgungseingang verwendet werden.

Alternativ kann die Stromversorgung auch über einen der 3,3-V-Pins erfolgen. Die Verwendung eines 3,3-V-Pins als Stromquelle kollidiert nicht mit einer der vorherigen Optionen. Die Kombination der 5-V- und 3,3-V-Versorgung wird jedoch nicht empfohlen.

5.2. Anschluss-Schema

Alle mit PA*xx* / IO*yy* gekennzeichneten Pins können als IO-Pins verwendet werden. Sie werden im Setup von LCD-Term referenziert und IO fungiert als IO*yy*. Die Pinbelegungstabellen enthalten die Zuordnung zwischen der Pinnummer eines Steckers und den IO*yy*-Pins für die verschiedenen LCD-Term-Karten.

Beispiel 4.1. Pin IO03[1] auf ‚Low' setzen.

```
echo -e -n "\033[3;0O" > /dev/ttyACM0
```

1) Dieser Pin besetzt unterschiedliche Pinnummern auf dem rechten Anschlussfeld (siehe Tabelle 4.4)

5.2.1. Linkes Anschlussfeld

Dieser Anschluss enthält im Wesentlichen die serielle Schnittstelle und das Netzteil.

Tabelle 4.3: Linkes Anschlussfeld

Pin	Beschreibung			
	LCD-Term12 (X4)	LCD-Term15 (X1)	LCD-Term25 (X1)	LCD-Term35 (X2)
1	3.3V[a]	3.3V[a]	3.3V[a]	3.3V[a]
2	GND	GND	GND	GND
3	TXD	TXD	TXD	TXD
4	RXD	RXD	RXD	RXD
5	PA11 / IO04	PA11 / IO07	PA11 / IO10	PA11 / IO13
6		5V[b]	5V[b]	5V[b]
7		PA24 / USB_D-	PA24 / USB_D-	PA24 / USB_D-
8		PA25 / USB_D+	PA25 / USB_D+	PA25 / USB_D+
9		GND	GND	GND
10			SCL[c]	SCL[c]
11			SDA[c]	SDA[c]
12			PA31 / IO11	PA31 / IO12
13				PA30 / IO11
14				PA15 / IO10
15				PA14 / IO09
16				PA10 / IO08

[a] Kann Zur Spannungsversorgung verwendet werden oder als Ausgang für externe Hardware dienen, wenn die Versorgung mit +5V über USB oder Pin 6 erfolgt

[b] Verbunden mit USB +5V; kann zur Versorgung verwendet werden wenn USB nicht verbunden ist

[c] Aktuell keine Funktion und darf nicht angeschlossen werden

5.2.2. Rechtes Anschlussfeld

Der rechte Anschluss besteht hauptsächlich aus E/A(I/O)-Pins.

Tabelle 4.4: Rechtes Anschlussfeld

Pin	Beschreibung			
	LCD-Term12 (X2)	LCD-Term15 (X2)	LCD-Term25 (X2)	LCD-Term35 (X4)
1	GND	3.3V[a]	3.3V[a]	5V[b]
2	PA00 / IO00	GND	GND	3.3V[a]
3	PA01 / IO01	PA00 / IO00	PA00 / IO00	GND
4	PA02 / IO02	PA01 / IO01	PA01 / IO01	PA00 / IO00
5	PA03 / IO03	PA02 / IO02	PA02 / IO02	PA01 / IO01
6		PA03 / IO03	PA03 / IO03	PA02 / IO02
7		PA04 / IO04	PA04 / IO04	PA03 / IO03
8		PA05 / IO05	PA05 / IO05	PA04 / IO04
9		PA06 / IO06	PA06 / IO06	PA05 / IO05
10			PA07 / IO07	PA06 / IO06
11			PA10 / IO08	PA07 / IO07
12			PA14 / IO09	PA10 / IO08
13				PA14 / IO09
14				PA15 / IO10
15				PA30 / IO11
16				PA31 / IO12

Auf einigen Pins liegen verschiedene Funktionen.

[a] Kann Zur Spannungsversorgung verwendet werden oder als Ausgang für externe Hardware dienen, wenn die Versorgung mit +5V über USB oder Pin 6 erfolgt

[b] Verbunden mit USB +5V; kann zur Versorgung verwendet werden wenn USB nicht verbunden ist

5.3. Maße

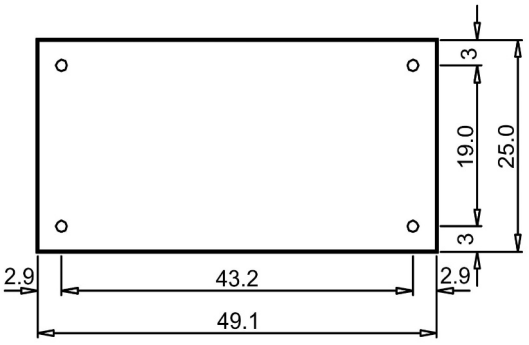

Abbildung 4.1. LCDTerm12-Maße[2]

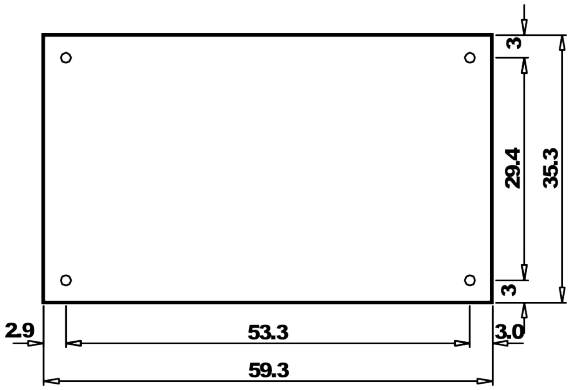

Abbildung 4.2. LCDTerm15-Maße[2]

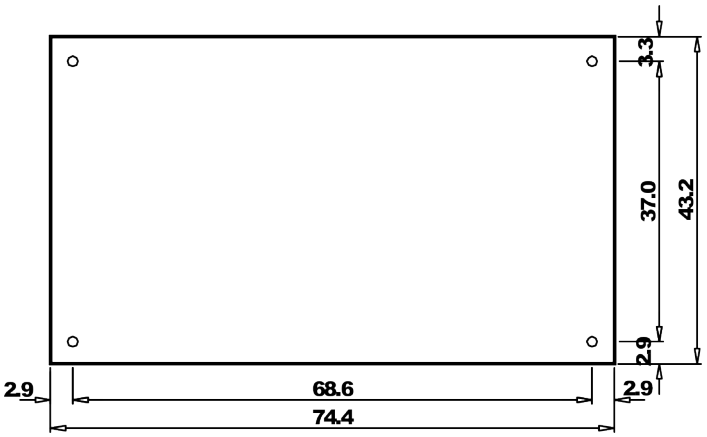

Abbildung 4.3. LCDTerm25-Maße[2]

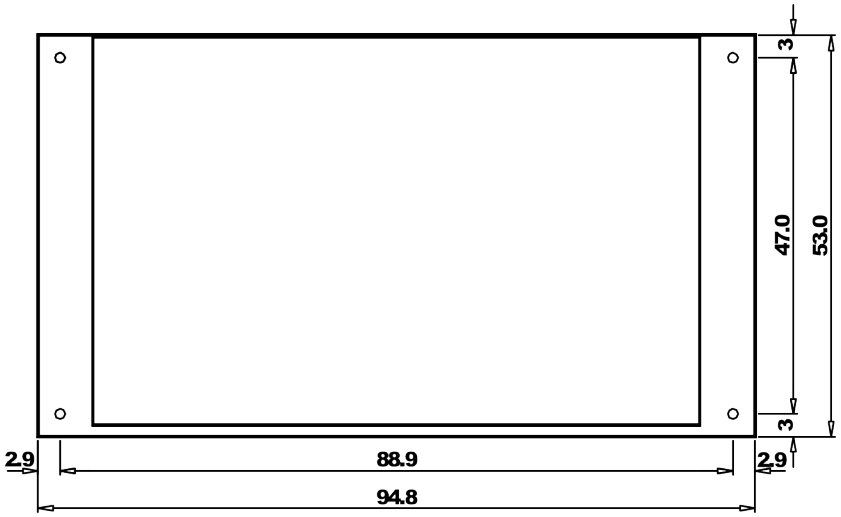

Abbildung 4.4. LCDTerm35-Maße[2]

[2]Alle Maße in mm.

6. Konformitätserklärung

6.1. Konformitätserklärung für Europa

Für folgende Produkte:

Allgemeiner Produktname: LCD-Term

Art.-Nr. 545959: LCD-Term12 - 1,2" LCD, 128x64, FSTN
Art.-Nr. 545960: LCD-Term15 - 1,5" LCD, 128x64, FSTN
Art.-Nr. 545961: LCD-Term25 - 2,5" LCD, 128x64, FSTN
Art.-Nr. 545962: LCD-Term35 - 3,5" LCD, 128x64, FSTN

Wir erklären, dass sie den folgenden Richtlinien der Europäischen Gemeinschaft entsprechen:

Konformität nach EMV-Richtlinie 2014/30/EU
European Conformity According to EMC-Directive 2014/30/EU

6.2. Konformität nach EMV-Richtlinie 2014/30/EU

Die Produkte, auf die sich diese Erklärung bezieht, entsprechen den folgenden Normen oder normativen Dokumenten der elektromagnetischen Verträglichkeit.

Sicherheit / Gesundheit:	EN 62368-1:2016-05 EN 62479:2011-09
EMV:	EN 301 489-1 V2.1.1:2017-02 EN 301 489-17 V3.1.1:2017-02
RoHS:	EU-Richtlinie 2011/65/EU

Aufgrund des in der Richtlinie 2014/30/EU beschriebenen Konformitätsbewertungsverfahrens sollte die Endkundenausrüstung wie folgt gekennzeichnet werden:

CE

Die Endkundenausrüstung muss den tatsächlichen Sicherheits- / Gesundheitsanforderungen nach den gen. Normen entsprechen.