EN: This Datasheet/Document is presented by the manufacturer.

Please visit our website for pricing and availability at www.hestore.hu.

Stand 2.2014

ELECTRONIC ASSEMBLY
new display design

embedded 2.0" ePAPER

172x72 mit Intelligenz

Abmessung: 62x35xx5,5mm

TECHNISCHE DATEN

* INTELLIGENTES ELEKTROPHORETISCH-AKTIV-MATIRX DISPLAY (ePAPER) 2“
* WEITER BLINKWINKELBEREICH
* KONTRASTREICHSTE ePAPER LCD ANZEIGE
* 4 GRAUSTUFEN - SCHWARZ, DUNKELGRAU, HELLGRAU, WEISS
* 3 VERSCHIEDENE INTERFACE ONBOARD: RS-232, I²C-BUS ODER SPI-BUS
* 172x72 ODER 72X172 PIXEL GEDREHT EINBAUBAR
* 8 EINGEBAUTE FONTS
* FONT ZOOM VON ca. 2mm BIS ZU ca. 20mm, auch um 90° GEDREHT
* VERSORGUNG 3,0V/3,3V 0,2µA...16mA
* POWER-DOWN-MODE 0,2 µA, MIT WAKEUP PER INTERFACE
* PIXELGENAUE POSITIONIERUNG BEI ALLEN FUNKTIONEN
* GERADE, PUNKT, BEREICH, RAHMEN, CLIPBOARD
* BIS ZU 256 BILDER INTERN SPEICHERBAR
* BIS ZU 256 MAKROS PROGRAMMIERBAR (64kB EEPROM ONBOARD)
* BETRIEBSTEMPERATURBEREICH 0°...+50°C (LAGERTEMPERATUR -25°..+75°C)

BESTELLBEZEICHNUNG

INTELLIGENTES ePAPER 172x72 DOTS	**EA eLABEL20-A**
TESTBOARD ZUM PROGRAMMIEREN MIT USB	**EA 9780-3USB**

Documentation of revision				
Date	**Type**	**Old**	**New**	**Reason / Description**
June, 2013	0.1			preliminary Version

INHALT

ELEKTRISCHE SPEZIFIKATIONEN 2
ALLGEMEINES 3
RS-232 3
SPI 4
I²C 5
SOFTWARE PROTOKOLL ..... 6 - 7
BEFEHLE / FUNKTIONEN IN TABELLENFORM ..... 8 - 9
TERMINAL-BETRIEB 10
RÜCKANTWORTEN 10
FÜLLMUSTER, RAHMEN 11
GEDREHTER EINBAU 11
POWER-DOWN-MODE 11
ZEICHENSÄTZE ..... 12 - 13
MAKROPROGRAMMIERUNG 14
APPLIKATIONEN ECHTE RS232, RS-485, USB 15
ABMESSUNGEN 16

SPEZIFIKATION

Characteristics					
Value	**Condition**	**min.**	**typ.**	**max.**	**Unit**
Operating Temperature		0		50	°C
Storage Temperature		-25		75	°C
Storage Humidity	< 40°C			90	%RH
Operating Voltage		2.8	3.0	3.3	V
Input Low Voltage		-0.5		0.2*VDD	V
Input High Voltage	Pin Reset only	0.9*VDD		VDD+0.5	V
Input High Voltage	except Reset	0.6*VDD		VDD+0.5	V
Input Leakage Current				1	uA
Input Pull-up Resistor		20		50	kOhms
Output Low Voltage	VDD = 3,0V			0.6	V
Output High Voltage	VDD = 3,0V	2.3			V
Output Current				20	mA
Current during display update (2s)	Turbo max.100kHz normal max. 38kHz		16,5 12,5		mA
Current normal operation	Turbo max.100kHz normal max. 38kHz		5,5 1,5		mA
Power Down	Mode 1 Mode 2, Timer On Mode 2, Timer Off		405 4,5 0,2		µA

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

ALLGEMEINES

EA eLABEL20-A ist ein ePAPER mit integrierter Intelligenz ! Die simple Verwendung dieses Displays verkürzt die Entwicklungszeit drastisch. Neben diversen eingebauten Schriften welche pixelgenau verwendet werden können, bietet es zudem eine ganze Reihe ausgefeilter Grafikfunktionen.

Pinout J2: RS-232			
Pin	**Symbol**	**In/Out**	**Function**
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	RxD	In	Receive Data
4	TxD	Out	Transmit Data

Die Programmierung dieses Displays erfolgt über Befehle wie z.B. *Zeichne ein Rechteck von (0,0) nach (64,15)*. Es ist keine zusätzliche Software oder Treiber erforderlich. Zeichenketten lassen sich **pixelgenau** platzieren. Das Mischen von Text und Grafik ist jederzeit möglich. Es können bis zu 16 verschiedene Zeichensätze verwendet werden. Jeder Zeichensatz kann wiederum 2- bis 4-fach gezoomt werden. Mit dem größten Zeichensatz lassen sich somit bildschirmfüllende Worte und Zahlen darstellen.

Das Display ist mit 3,0V...3,3V sofort betriebsbereit. Die Ansteuerung erfolgt über eine der 3 eingebauten Schnittstellen RS-232, SPI oder I²C.

Pinout eLABEL20-A: RS-232/RS-485 mode			
Pin	**Symbol**	**In/Out**	**Function**
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	NC		do not connect
4	NC		do not connect
5	RESET	In	L: Reset
6	BAUD0	In	Baud Rate 0
7	BAUD1	In	Baud Rate 1
8	BAUD2	In	Baud Rate 2
9	ADR0	In	Address 0 for RS-485
10	RxD	In	Receive Data
11	TxD	Out	Transmit Data
12	EN485	Out	Transmit Enable for RS-485 driver
13	DPOM	In	L: (Power-On) disable Power-On-Macro
14	ADR1	In	Address 1 for RS-485
15	ADR2	In	Address 2 for RS-485
16	BUZZ	Out	H: Buzzer output (L: Buzzer off)
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation
18	PWR	Out	L: Normal Operation H: Powerdownmode
19	NC		do not connect
20	TEST SBUF	IN Out	open-drain with internal pullup 20k..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer

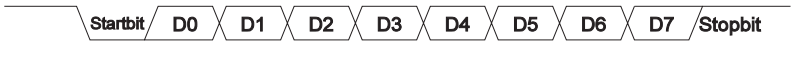
RS-232 INTERFACE

Wird das Display wie unten gezeigt beschaltet, so ist das RS-232 Interface ausgewählt. Die Pinbelegung ist in der Tabelle rechts bzw. oben angegeben. Die Leitungen RxD und TxD führen CMOS-Pegel (VDD) zur direkten Anbindung an z.B. einen Mikrokontroller.

BAUDRATEN

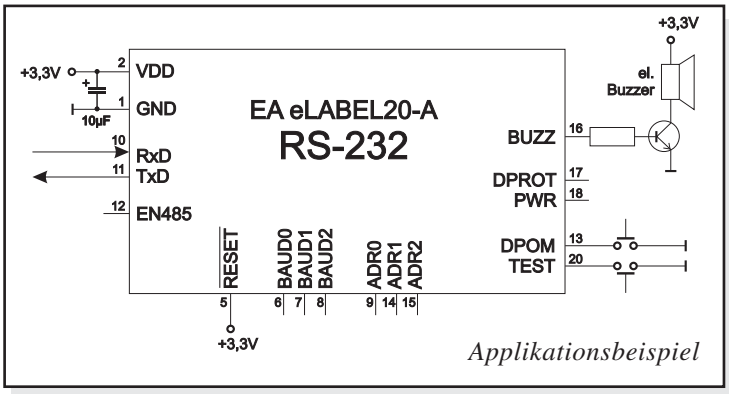
Die Baudrate wird über die Pins 6, 7 und 8 (Baud0..2) eingestellt. Das Datenformat ist fest eingestellt auf 8 Datenbits, 1 Stopbit, keine Parität.
Handshakeleitungen RTS/CTS sind nicht erforderlich. Die notwendige Steuerung wird von dem eingebauten Software-Protokoll übernommen.

Baudraten			
Baud0	Baud1	Baud2	Datenformat 8,N,1
0	0	0	1200
1	0	0	2400
0	1	0	4800
1	1	0	9600
0	0	1	19200
1	1	1	38400
0	1	1	57600 (Turbo)
1	0	1	115200 (Turbo)



Adressierung:

- *Bis zu acht Hardware-Adressen (0..7) per Pins ADR0..ADR2 einstellbar*
- *Das eLABEL mit Adresse 7 ist nach PowerOn selektiert und Empfangsbereit*
- *Die eLABEL mit Adresse 0..6 sind nach PowerOn deselektiert*
- *Bis zu 246 weitere Software-Adressen per Befehl '#KA adr' im PowerOnMakro einstellbar (eLABEL extern auf Adresse 0 setzen)*



Hinweis:
Die Pins BAUD0..2, ADR0..2, WUP und TEST/SBUF haben einen internen Pull-UP, deshalb ist nur der LO-Pegel (0=GND) aktiv anzulegen. Für Hi-Pegel sind diese Pins offen zu lassen.
Für RS232 Betrieb (ohne Adressierung) sind die Pins ADR0..ADR2 offen zu lassen.
Am Pin 20 (SBUF) zeigt das Display mit einem low-Pegel, dass im internen Sendepuffer Daten zur Abholung bereit stehen. Diese Leitung kann z.B. mit einem Interrupteingang des Host Systems verbunden werden.

SPI INTERFACE

Wird das Display wie unten gezeigt beschaltet, ist der SPI-Mode aktiviert. Die Datenübertragung erfolgt dann über die serielle synchrone SPI-Schnittstelle.
Mit den Pins DORD, CPOL, CPHA werden die Hardwarebedingungen an den Master angepasst.

Pinout eLABEL20-A: SPI mode

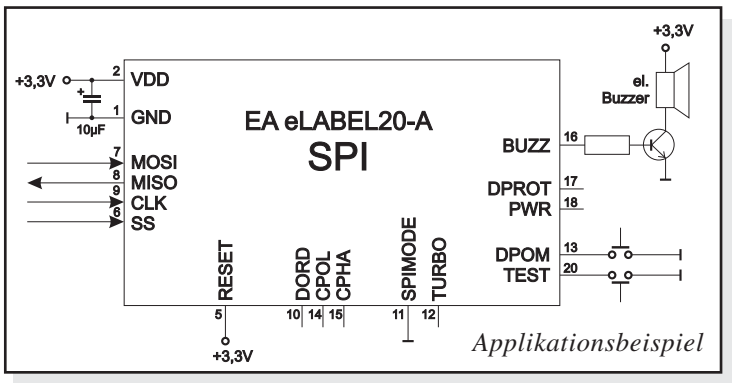
Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	NC		do not connect
4	NC		do not connect
5	RESET	In	L: Reset
6	SS	In	Slave Select
7	MOSI	In	Serial In
8	MISO	Out	Serial Out
9	CLK	In	Shift Clock (38kHz / 100kHz see Pin 12)
10	DORD	In	Data Order (0=MSB first; 1=LSB first)
11	SPIMOD	In	connect to GND for SPI interface
12	TURBO	In	L: CLK max.100kHz; H: CLK max.38kHz
13	DPOM	In	L: (Power-On) disable Power-On-Macro
14	CPOL	In	Clock Polarity (0=LO 1=HI when idle)
15	CPHA	In	Clock Phase sample 0=1st;1=2nd edge
16	BUZZ	Out	H: Buzzer output (L: Buzzer off)
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation
18	PWR	Out	L: Normal Operation H: Powerdownmode
19	NC		do not connect
20	TEST SBUF	IN Out	open-drain with internal pullup 20k..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer

Hinweis:
Die Pins DORD, CPOL, CPHA, WUP und TEST/SBUF haben einen internen Pull-UP, deshalb ist nur der LO-Pegel (0=GND) aktiv anzulegen. Für Hi-Pegel sind diese Pins offen zu lassen.
Am Pin 20 (SBUF) zeigt das Display mit einem low-Pegel, dass im internen Sendepuffer Daten zur Abholung bereit stehen. Diese Leitung kann z.B. mit einem Interrupteingang des Host Systems verbunden werden.

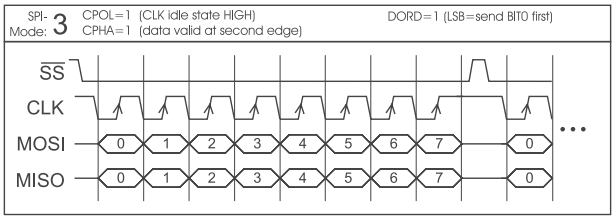
DATENÜBERTRAGUNG SPI

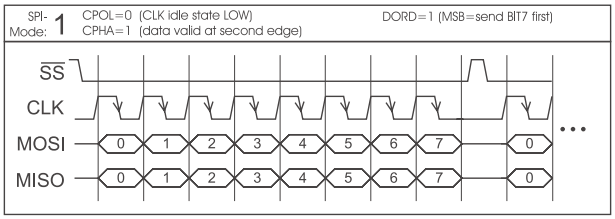
Eine Datenübertragung zum eLABEL ist im Normalbetrieb bis zu 38 kHz erlaubt. Ist der Turbomode aktiviert (PIN 12 auf LO) dann sind bis zu 100 kHz Nonstop möglich.

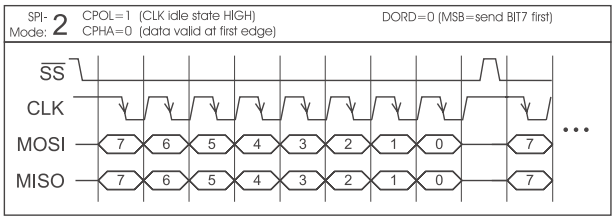
Um Daten vom eLABEL zu Lesen (z.B. das ACK-Byte) muss ein Dummy-Byte (z.B. 0xFF) gesendet werden. Das eLABEL benötigt eine bestimmte Zeit um die Daten bereit zu stellen; deshalb muss vor jedem zu lesenden Byte zusätzlich mindestens 80µs (Normalbetrieb) oder 10µs (Turbomode) gewartet werden (keine Aktivität auf der CLK Leitung).

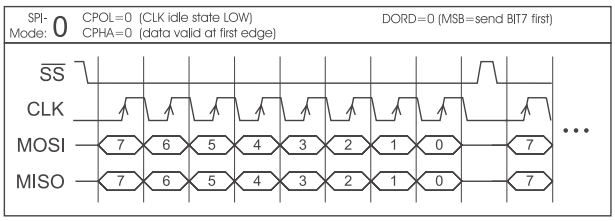


Applikationsbeispiel









Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

I²C-BUS INTERFACE

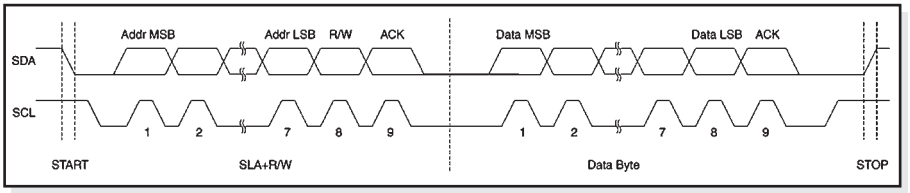
Eine Beschaltung des Displays wie unten ermöglicht den direkten Betrieb an einem I²C-Bus.
Am Display kann zwischen 8 unterschiedlichen Basisadressen und 8 verschiedenen Slave-Adressen ausgewählt werden.

Eine Datenübertragung ist im Normalbetrieb bis zu 38 kHz erlaubt. Ist der Turbomode aktiviert (PIN 10 auf LO) dann sind bis zu 100 kHz möglich.

Pinout eLABEL20-A: I2C mode

Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	NC		do not connect
4	NC		do not connect
5	RESET	In	L: Reset
6	BA0	In	Basic Address 0
7	BA1	In	Basic Address 1
8	SA0	In	Slave Address 0
9	SA1	In	Slave Address 1
10	TURBO	In	L: SCL max.100kHz; H: SCL max.38kHz
11	NC		do not connect
12	I2CMOD	In	connect to GND for I²C interface
13	DPOM	In	L: (Power-On) disable Power-On-Macro
14	SDA	Bidir.	Serial Data
15	SCL	In	Serial Clock (38kHz / 100kHz see Pin 10)
16	BUZZ	Out	H: Buzzer output (L: Buzzer off)
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation
18	PWR	Out	L: Normal Operation H: Powerdownmode
19	NC		do not connect
20	TEST SBUF	IN Out	open-drain with internal pullup 20k..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer

Hinweis:
Die Pins BA0..1, SA0..1, WUP, DPROT und TEST/SBUF haben einen internen Pull-Up, deshalb ist nur der LO-Pegel (0=GND) aktiv anzulegen. Für Hi-Pegel sind diese Pins offen zu lassen.
Am Pin 20 (SBUF) zeigt das Display mit einem LO-Pegel, dass im internen Sendepuffer Daten zur Abholung bereit stehen. Diese Leitung kann z.B. mit einem Interrupteingang des Host Systems verbunden werden.



I²C - Address

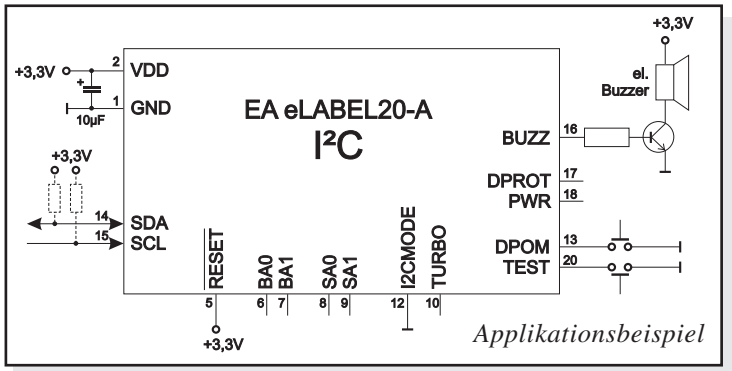
Pin 7,6		Base address		I²C address							
BA1	BA0			D7	D6	D5	D4	D3	D2	D1	D0
L	L	$78		0	1	1	1	1	SA1	SA0	RW
L	H	$98		1	0	0	1	1			
H	L	$B8		1	0	1	1	1			
H	H	$D8		1	1	0	1	1			

alle Pins offen: Schreiben $DE
Lesen $DF

DATENÜBERTRAGUNG I²C-BUS

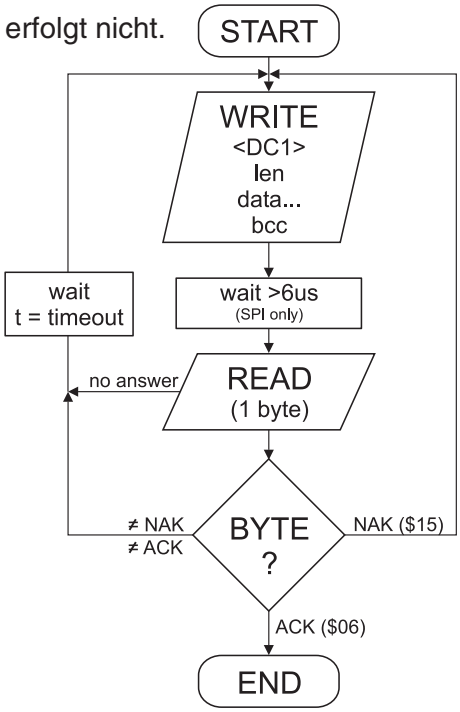
Prinzip der Übertragung:
- I²C-Start
- Master-Transmit: Display-I²C-Adr. (z.B. $DE), Smallprotokollpaket (Daten) senden
- I²C-Stop
- I²C-Start
- Master-Read: Display-I²C-Adr. (z.B. $DF), ACK-Byte und evtl. Smallprotokollpaket (Daten) lesen
- I²C-Stop

Das eLABEL benötigt eine bestimmte Zeit um die Daten bereit zu stellen; deshalb muss vor jedem zu lesenden Byte zusätzlich mindestens 80µs (Normalbetrieb) oder 10µs (Turbomode) gewartet werden (keine Aktivität auf der SCL Leitung).



DATENÜBERTRAGUNGSPROTOKOLL (SMALL PROTOKOLL)

Das Protokoll ist für alle 3 Schnittstellenarten RS-232, SPI und I²C identisch aufgebaut. Die Datenübertragung ist jeweils eingebettet in einen festen Rahmen mit Prüfsumme „bcc“. Das eLABELquittiert dieses Paket mit dem Zeichen <ACK> (=$06) bei erfolgreichem Empfang oder <NAK> (=$15) bei fehlerhafter Prüfsumme oder Empfangspufferüberlauf. In jedem Fall wird bei <NAK> das komplette Paket verworfen und muss erneut gesendet werden.
Ein <ACK> bestätigt lediglich die korrekte Übertragung. Ein Syntax-Check erfolgt nicht.
Hinweis: <ACK> muss eingelesen werden.
Empfängt der Hostrechner keine Quittierung, so ist mindestens ein Byte verloren gegangen. In diesem Fall muss die eingestellte Timeoutzeit abgewartet werden, bevor das Paket komplett wiederholt wird.
Die Anzahl (len) der Rohdaten pro Paket kann max. 255 Byte betragen. Befehle die grösser als 255 Byte sind (z.B. Bild laden ESC UL ...) müssen auf mehrere Pakete aufgeteilt werden. Alle Daten in den Paketen werden nach korrektem Empfang im Display wieder zusammengefügt.



SMALL PROTOLKOLL DEAKTIVIEREN

Das Protokoll ist für alle drei Schnittstellen RS-232, I²C und SPI identisch. Für Tests kann das Protokoll durch L-Pegel an Pin17(DPROT) abgeschaltet werden. Im normalen Betrieb ist allerdings die Aktivierung des Protokolls unbedingt zu empfehlen. Andernfalls wäre ein möglicher Überlauf des Empfangspuffers oder eine fehlerhafte Datenübertragung nicht zu erkennen.

DIE PAKETVARIANTEN IN EINZELNEN

Befehle/Daten zum Display senden

>	<DC1>	len	data...	bcc
<	<ACK>			

<DC1> = 17(dez.) = $11
<ACK> = 6(dez.) = $06
len = Anzahl der Nutzdaten in Byte (ohne Prüfsumme, ohne <DC1>)
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC1> und len, Modulo 256

Clear display and draw a line from 0,0 to 171,71

<DC1>	len	ESC D L		ESC G D 0 0 171 71							bcc	>
$11	$0A	$1B	$44	$4C	$1B	$47	$44	$00	$00	$AB	$47	$5E

< <ACK> $06

Beispiel für ein komplettes Datenpaket

Eingerahmt von <DC1>, der Anzahl der Daten "len" und der Prüfsumme "bcc" werden die jeweiligen Nutzdaten übertragen. Als Antwort sendet das Display <ACK> zurück.

```
void SendData(unsigned char *buf, unsigned char len)
{
 unsigned char i, bcc;

 SendByte(0x11);            // Send DC1
 bcc = 0x11;

 SendByte(len);             // Send data length
 bcc = bcc + len;

 for(i=0; i < len; i++)     // Send buf
 { SendByte(buf[i]);
   bcc = bcc + buf[i];
 }

 SendByte(bcc);             // Send checksum
}
```

C-Beispiel zum Senden eines Datenpaketes

Inhalt des Sendepuffers anfordern

>	<DC2>	1	S	bcc
<	<ACK>			
<	<DC1>	len	data...	bcc

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 S = 83(dez.) = $53
<ACK> = 6(dez.) = $06
len = Anzahl der Nutzdaten in Byte (ohne Prüfsumme, ohne <DC1>)
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC1> und len, Modulo 256

Die Befehlsfolge <DC2>, 1, S, bcc entleert den Sendepuffer des Displays. Das Display antwortet zuerst mit der Quittierung <ACK> und beginnt dann alle gesammelten Daten zu senden.

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

Pufferinformationen anfordern

>	<DC2>	1	I	bcc
<	<ACK>			
<	<DC2>	2	send buffer bytes ready	receive buffer bytes free	bcc

Mit diesem Befehl wird abgefragt, ob Nutzdaten zur Abholung bereit stehen und wie voll der Empfangspuffer des Displays bereits ist.

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 I = 73(dez.) = $49
<ACK> = 6(dez.) = $06
send buffer bytes ready = Anzahl abholbereiter Bytes
receive buffer bytes free = verfügbarer Platz im Empfangspuffer
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC2> Modulo 256

Protokolleinstellungen

>	<DC2>	3	D	packet size for send buffer	timeout	bcc
<	<ACK>					

Hierüber läßt sich die maximale Paketgröße welche das Display senden darf begrenzen. Voreingestellt ist eine Paketgröße mit bis zu 128 Byte Nutzdaten.
Weiterhin läßt sich der Timeout in 1/100s einstellen. Der Timeout spricht an, wenn einzelne Bytes verloren gegangen sind. Danach muß das gesamte Paket nochmals übertragen werden.

<DC2> = 18(dez.) = $12 3 = 3(dez.) = $03 D = 68(dez.) = $44
packet size for send buffer = 1..128 (Standard: 128)
timeout = 1..255 in 1/100 Sekunden (Standard: 200 = 2 Sekunden)
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC2>, Modulo 256
<ACK> = 6(dez.) = $06

Protokollinformationen anfordern

>	<DC2>	1	P	bcc		
<	<ACK>					
<	<DC2>	3	max. packet size	akt. send packet size	akt. timeout	bcc

Mit diesem Befehl werden Protokoll-einstellungen abgefragt.

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 P = 80(dez.) = $50
<ACK> = 6(dez.) = $06
max. packet size = maximale Anzahl der Nutzdaten eines Protokollpaketes (eLABEL20-A = 255)
akt. send packet size = eingestellte Paketgrösse zum Senden
akt. timeout = eingestellter timeout in 1/100 Sekunden
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC2>, Modulo 256

Letztes Datenpaket wiederholen

>	<DC2>	1	R	bcc
<	<ACK>			
<	<DC1> <DC2>	len	data...	bcc

Falls das zuletzt angeforderte Paket eine falsche Prüfsumme enthielt, kann das komlette Paket nochmals angefordert werden. Die Antwort kann dann der Inhalt des Sendepuffers (<DC1>) oder die Puffer-/Protokoll-Information (<DC2>) sein.

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 R = 82(dez.) = $52
<ACK> = 6(dez.) = $06
<DC1> = 17(dez.) = $11
len = Anzahl der Nutzdaten in Byte (ohne Prüfsumme, ohne <DC1> bzw. <DC2>)
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC2> und len, Modulo 256

Adressierung nur bei RS232/RS485 Betrieb

>	<DC2>	3	A	select or deselect	adr	bcc
<	<ACK>					

Mit diesem Befehl läst sich das eLABEL mit der Adresse adr Selektieren oder Deselektieren.

<DC2> = 18(dez.) = $12 3 = 3(dez.) = $03 A = 65(dez.) = $41
select or deselect: 'S' = 83(dez.) = $53 oder 'D' = 68(dez.) = $44
adr = 0..255
bcc = 1 Byte = Summe aus allen Bytes inkl. <DC2> und len, Modulo 256
<ACK> = 6(dez.) = $06

BEFEHLE ÜBER DIE SERIELLE SCHNITTSTELLE SENDEN

Das eLABEL läßt sich über diverse eingebaute Befehle programmieren. Jeder Befehl beginnt mit ESCAPE gefolgt von einem oder zwei Befehlsbuchstaben und einigen Parametern.
Es gibt zwei Möglichkeiten Befehle zu senden:

1. ASCII-Modus

- Das Escape-Zeichen entspricht dem Zeichen '**#**' (hex: $23, dez: 35).
- Die Befehlsbuchstaben folgen direkt im Anschluss an das '#' Zeichen.
- Die Parameter werden im Klartext (mehrere ASCII Ziffern) mit einem nachfolgenden Trennzeichen (z.B. das Komma '**,**') gesendet, auch hinter dem letzten Parameter z.B.: **#GD0,0,171,71,**
- Zeichenketten (Texte) werden direkt ohne Anführungsstrichen geschrieben und mit CR (hex: $0D), oder LF (hex: $0A) abgeschlossen.

2. Binär-Modus

- Das Escape-Zeichen entspricht dem Zeichen ESC (hex: $1B, dez: 27).
- Die Befehlsbuchstaben werden direkt gesendet.
- Die Koodinaten x und y und alle anderen Parameter werden als 8-Bit Binärwert (1 Byte) gesendet.
- Zeichenketten (Texte) werden mit CR (hex: $0D), LF (hex: $0A) oder NUL (hex: $00) abgeschlossen.

Im Binär-Modus dürfen keine Trennzeichen z.B. Leerzeichen oder Kommas verwendet werden.
Die Befehle benötigen **kein Abschlussbyte** wie z.B Carriage Return (außer Zeichenkette: $00).

ALLE BEFEHLE AUF EINEN BLICK

Die eingebaute Intelligenz erlaubt den Aufbau eines Bildschirmes über unten stehende Befehle. Alle Befehle können sowohl über die serielle Schnittstelle als auch in selbst-definierten Makros verwendet werden.

Displayfunktionen (Wirkung auf das gesamte Display)									**nach Reset**
Befehl	**Codes**							**Anmerkung**	
Display Orientierung	ESC	D	O	n1				n1=0: 0° = 172x 72 pixel; Terminal: 21 Spalten, 9 Zeilen n1=1: 90° = 72x172 pixel; Terminal: 9 Spalten, 21 Zeilen	0°
Display Update	ESC	D	U					Der aktuelle Inhalt des Display-RAM wird ins ePAPER kopiert	
Display Autoupdate Zeiten	ESC	D	Z	n1	n2			n1: Verzögerungszeit nach dem letzten Befehl bevor geupdatet wird n2: Zwangsupdatezeit nach dem ersten Befehl, wenn ohne Unterbrechung gesendet wird n1,n2=1..255 in 1/10sec; (n1,n2=0: Autoupdate Aus; ESC DU für Update benutzen)	3, 50
Display löschen	ESC	D	L					Displayinhalt löschen (alle Pixel aus)	
Display invertieren	ESC	D	I					Displayinhalt invertieren (alle Pixel umkehren)	
Display füllen	ESC	D	F	n1				Displayinhalt mit Farbe n1 füllen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß	
Display ausschalten	ESC	D	E					Displayinhalt wird unsichtbar bleibt aber erhalten, Befehle weiterhin möglich	
Display einschalten	ESC	D	A					Displayinhalt wird wieder sichtbar	Ein

Textfunktionen									**nach Reset**
Befehl	**Codes**							**Anmerkung**	
Einstellungen									
Textfarbenfarben einstellen	ESC	F	Z	vf	hf			vf=Vordergrund-; hf=Hintergrundfarbe für Zeichenketten 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Transparent; Die sinnlose Kombination von transparenten Vorder- und Hintergrund invertiert alle Pixel	1,4
Font einstellen	ESC	Z	F	n1				Font mit der Nummer n1 (0..15) einstellen	0
Font-Zoomfaktor	ESC	Z	Z	n1	n2			n1 = X-Zoomfaktor (1x..4x); n2 = Y-Zoomfaktor (1x..4x)	1,1
zus. Zeilenabstand	ESC	Z	Y	n1				zwischen zwei Textzeilen n1 Pixel (0..15) als zusätzlichen Zeilenabstand einfügen	
Leerzeichenbreite	ESC	Z	J	n1				Leerzeichenbreite: n1=0 aus Zeichensatz; n1=1 wie Ziffer; n1>=2 Breite in Pixel	0
Text-Winkel	ESC	Z	W	n1				Text-Ausgabewinkel: n1=0: 0°; n1=1: 90°;	0
Ausgabe von Zeichenketten									
Zeichenkette ausgeben L: Linksbündig C: Zentriert R: Rechtsbündig	ESC	Z	L C R	x1	y1	Text ...	NUL	Eine Zeichenkette an x1,y1 ausgegeben; Zeichenkettenende: 'NUL' ($00), 'LF' ($0A) oder 'CR' ($0D); Mehrere Zeilen werden durch das Zeichen '|' ($7C) getrennt; Das Backslash-Zeichen '\' ($5C) hebt die Sonderfunkion der Zeichen '|' auf; z.B. "name\\test.txt" => "name\test.txt"	
Zeichenkette für Terminal	ESC	Z	T	Text ...				Befehl um eine Zeichenkette in einem Makro an das Terminal ausgeben zu können	

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

Bereichsfunktionen										nach Reset
Befehl	**Codes**								**Anmerkung**	
Bereich löschen	ESC	R	L	x1	y1	x2	y2		Einen Bereich von x1,y1 nach x2,y2 löschen (alle Pixel aus)	
Bereich invertieren	ESC	R	I	x1	y1	x2	y2		Einen Bereich von x1,y1 nach x2,y2 invertieren (alle Pixel umkehren)	
Bereich füllen	ESC	R	F	x1	y1	x2	y2	n1	Einen Bereich von x1,y1 nach x2,y2 mit Farbe n1 füllen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Invers	
Musterfarben einstellen	ESC	F	M	vf	hf				vf=Vordergrund-; hf=Hintergrundfarbe für Muster einstellen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Transparent; Die sinnlose Kombination von transparenten Vorder- und Hintergrund invertiert alle Pixel	1,4
Bereich mit Füllmuster	ESC	R	M	x1	y1	x2	y2	n1	Einen Bereich von x1,y1 nach x2,y2 mit Muster n1 zeichnen	
Box mit Füllmuster zeichnen	ESC	R	O	x1	y1	x2	y2	n1	Ein Rechteck von x1,y1 nach x2,y2 mit Muster n1 zeichnen	
Rahmenfarben einstellen	ESC	F	R	vf	hf				vf=Vordergrund (Rahmen); hf=Hintergrundfarbe (Füllung) für Rahmen einstellen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Transparent; Die sinnlose Kombination von transparenten Vorder- und Hintergrund invertiert alle Pixel	1,4
Rahmen zeichnen	ESC	R	R	x1	y1	x2	y2	n1	Einen Rahmen Typ n1 von x1,y1 nach x2,y2 zeichnen	

Geradenfunktionen									nach Reset
Befehl	**Codes**							**Anmerkung**	
Einstellungen									
Geradenfarbe einstellen	ESC	F	G	n1				Linienfarbe n1 einstellen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Invertieren	1
Punktgröße / Liniendicke	ESC	G	Z	n1	n2			n1 = X-Punktgröße (1..15); n2 = Y-Punktgröße (1..15);	1,1
Geraden und Punkte zeichnen									
Punkt zeichnen	ESC	G	P	x1	y1			Ein Punkt an die Koordinaten x1,y1 setzen	
Gerade zeichnen	ESC	G	D	x1	y1	x2	y2	Eine Gerade von x1,y1 nach x2,y2 zeichnen	
Gerade weiter zeichnen	ESC	G	W	x1	y1			Eine Gerade vom letzten Endpunkt bis x1, y1 zeichnen	0, 0
Rechteck zeichnen	ESC	G	R	x1	y1	x2	y2	Vier Geraden als Rechteck von x1,y1 nach x2,y2 zeichnen	

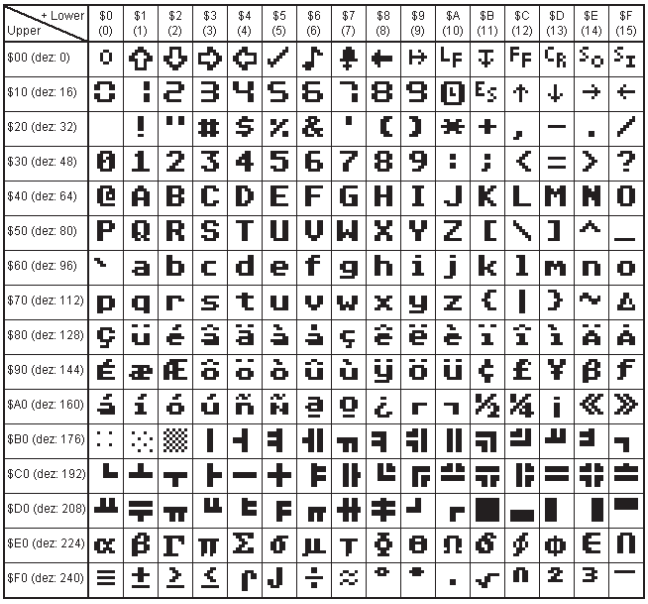
Bildfunktionen									nach Reset
Befehl	**Codes**							**Anmerkung**	
Einstellungen									
Monochrombild Farben	ESC	F	U	vf	hf			vf=Vordergrund-; hf=Hintergrundfarbe für monochrome Bilder einstellen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Transparent; Die sinnlose Kombination von transparenten Vorder- und Hintergrund invertiert alle Pixel	1,4
Bild-Zoomfaktor	ESC	U	Z	n1	n2			n1 = X-Zoomfaktor (1x..4x); n2 = Y-Zoomfaktor (1x..4x)	1,1
Bild-Winkel	ESC	U	W	n1				Ausgabewinkel des Bildes: n1=0: 0°; n1=1: 90°	0
Ausgabe									
internes Bild laden	ESC	U	I	x1	y1	nr		internes Bild mit der nr (0..255) aus dem EEPROM nach x1,y1 laden	
Bild laden	ESC	U	L	x1	y1	BLG daten ...		Ein Bild nach x1,y1 laden; daten des Bildes siehe Bildaufbau BLG-Format	
Hardcopy									
Hardcopy senden	ESC	U	H	x1	y1	x2	y2	Der Bildausschitt wird im BLG-Format gesendet (landet im Sendepuffer)	

Clipboard Befehle (Zwischenspeicher für Bildbereiche)									nach Reset
Befehl	**Codes**							**Anmerkung**	
Displayinhalt sichern	ESC	C	B					Der gesamte Displayinhalt wird als Bildbereich ins Clipboard kopiert	
Bereich sichern	ESC	C	S	x1	y1	x2	y2	Der Bildbereich von x1,y1 bis nach x2,y2 wird ins Clipboard kopiert	
Bereich restaurieren	ESC	C	R					Der Bildbereich im Clipboard wird wieder ins Display kopiert	
Bereich kopieren	ESC	C	K	x1	y1			Der Bildbereich im Clipboard wird ins Display nach x1,y1 kopiert	

Allgemeine Befehle									nach Reset
Befehl	**Codes**							**Anmerkung**	
Sende-Befehle									
Bytes senden	ESC	S	B	anz	daten ...			Es werden anz (=1..255) Bytes zum Sendepuffer gesendet; im Quelltext der Makroprogrammierung darf die Anzahl anz nicht angegeben werden, diese wird vom eLABEL-Compiler automatisch eingetragen.	
Version senden	ESC	S	V					Version wird als String gesendet z.B."EA eLABEL20-A V1.0 Rev.A" (Sendepuffer)	
Projektname senden	ESC	S	J					Es wird der Makro-Projektname als String gesendet z.B. "init / delivery" (Sendepuffer)	
Interne Infos senden	ESC	S	I					Es werden interne Informationen vom eLABEL gesendet (landen im Sendepuffer)	
Power Down-Befehle									
Power Down	ESC	P	D	mode				Nach diesem Befehl geht das eLABEL in den Power-down mode=1..2 mode=1 (~405µA): Mikrokontroller aus mode=2 (~ 0,2µA): Mikrokontroller aus + ePaper aus (benötigt zusätzlichen blankcycle)	
Power Down und Makros ausführen	ESC	P	M	mode	n1	n2		Das eLABEL geht in Power-down mode=1..2 und führt die Makros n1..n2 zyklisch aus mode=1 (~410µA): Mikrokontroller aus mode=2 (~ 4,5µA): Mikrokontroller aus + ePaper aus (benötigt zusätzlichen blankcycle)	
Zeit zwischen Makros	ESC	P	Z	h	m	s		Zeit zwischen Makros während powerdown h=Stunden; m=Minuten; s=Sekunden (±10%)	0,0,8
Sonstige-Befehle									
Makro ausführen	ESC	M	N	n1				Das Makro mit der Nummer n1 (0..255) aufrufen (max. 7 Ebenen)	
RS485 Adresse einstellen	ESC	K	A	adr				nur für RS232/RS485 Betrieb und nur bei Hardwareadresse 0 möglich Dem eLABEL wird eine neue Adresse adr zugewiesen (im PowerOn-Makro).	
Summer Ein / Aus	ESC	Y	S	n1				Summerausgang (PIN16) wird n1=0:AUS;n1=1:EIN;n1=2..255;für n1/10s eingeschaltet	AUS
Warten (Pause)	ESC	X	n1					n1/10s abwarten bevor der nächste Befehl ausgeführt wird.	

TERMINAL-BETRIEB

Das Display enthält eine integrierte Terminalfunktion. Nach dem Einschalten steht der unschtbare Cursor in der ersten Zeile und das Display ist empfangsbereit. Alle ankommenden Zeichen werden als ASCII's dargestellt (Ausnahme: CR,LF,FF,ESC,'#'). Voraussetzung dafür ist ein funktionierender Portokollrahmen oder ein abgeschaltetes Protokoll (siehe Seite 6+7).
Der Zeilenvorschub erfolgt automatisch oder durch das Zeichen 'LF'. Ist die letzte Zeile voll, scrollt der Terminalinhalt nach oben. Beim Zeichen 'FF' wird das Terminal gelöscht.
Das Zeichen '#' wird als Escape-Zeichen benutzt und ist somit nicht direkt im Terminal darstellbar. Soll das Zeichen '#' im Terminal ausgegeben werden, so muß es doppelt gesendet werden '##'.



Terminal-Font (Font 0): 8x8 monospaced

Das Terminal besitzt eine eigene Ebene zur Darstellung und ist somit völlig unabhänging von den Grafikausgaben. Wird z.B. der Grafikbildschirm mit 'ESC DL' gelöscht, so beeinflusst das nicht den Inhalt des Terminalfensters.
Der Terminalfont ist fest im ROM vorhanden und kann auch für Grafikausgaben 'ESC Z...' verwendet werden (FONT nr=0 einstellen).

Terminal Befehle								nach Reset
Befehl	**Codes**						**Anmerkung**	
Formfeed FF (dez:12)	**^L**						Bildschirm wird gelöscht und der Cursor nach Pos. (1,1) gesetzt	
Carriage Return CR(13)	**^M**						Cursor ganz nach links zum Zeilenanfang	
Linefeed LF (dez:10)	**^J**						Cursor 1 Zeile tiefer, falls Cursor in letzter Zeile dann wird gescrollt	
Cursor positionieren	**ESC**	**T**	**P**	n1	n2		n1=Spalte; n2=Zeile; Ursprung links oben ist (1,1)	1,1
Cursorposition sichern			**S**				die aktuelle Cursorposition wird gesichert	
Cursorposition restoren			**R**				die letzte gesicherte Cursorposition wird wieder hergestellt	
Terminal AUS			**A**				Terminal Anzeige ist ausgeschalten; Ausgaben werden verworfen	
Terminal EIN			**E**				Terminal Anzeige ist eingeschalten;	Ein
Version ausgeben	**ESC**	**T**	**V**				Die Versions-Nr. wird im Terminal ausgegeben z.B "EA eLABEL20-A V1.0 Rev.A"	
Projektname anzeigen			**J**				Der Makro-Projektname wird im Terminal ausgegeben z.B. "init / delivery state"	
Informationen anzeigen			**I**				Das Terminal wird initialisiert und gelöscht, die Software Version, Hardware Revision, der Makro-Projektname und die CRC-Checksummen werden im Terminal ausgegeben.	

ANTWORTEN/RÜCKMELDUNGEN

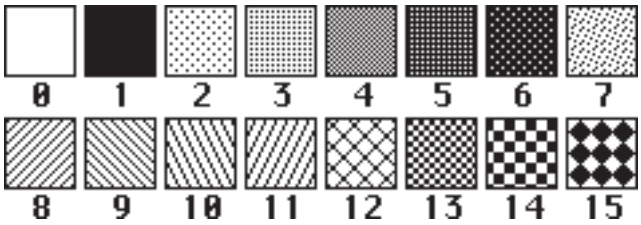
Alle Antworten des EA eLABEL20-A werden in einen Sendepuffer gestellt. Über das Small-Protokoll werden diese dann vom Host angefordert (siehe Seite 10). Dies kann per „Polling“ geschehen, oder altenativ dazu zeigt der Pin 20 „SBUF“ mit einem LO-Pegel an, dass Daten zur Abholung bereit stehen.

Antworten über die serielle Schnittstelle						
Kennung		**anz**	**daten**			**Anmerkung**
Antworten mit Längenangabe anz (landen im Sendepuffer)						
ESC	**V**	anz	Zeichenkette...			Nach dem Befehl 'ESC S V' wird die Version der eLABEL Firmware als Zeichenkette gesendet. z.B. "EA eLABEL20-A V1.0 Rev.A"
ESC	**J**	anz	Zeichenkette Projektname...			Nach dem Befehl 'ESC S J' wird der Makro-Projektname als Zeichenkette gesendet. z.B. "init / delivery state"
ESC	**I**	anz	X-Pixel, Y-Pixel, Version, Touchinfo CRC-ROM, CRC-ROMsoll EEP in KB, CRC-EEP, CRC-EEPsoll, EEPanz			anz = 21: Nach dem Befehl 'ESC S I' werden interne Informationen vom eLABEL gesendet (16-Bit integer Werte LO- HI-Byte) Version: LO-Byte = Versionsnr. Software; HI-Byte = Hardwarerevisionsbuchstabe Touchinfo: LO-Byte = '-|+' X-Richtung erkannt; HI-Byte = '-|+' Y-Richtung erkannt EEPanz: Anzahl benutzter Bytes im EEPROM (3 Byte: LO-, MID- HI-Byte)
Antworten ohne Längenangabe (landet im Sendepuffer)						
ESC	**U**	L	x1	y1	Bilddaten... (BLG-Format)	Nach dem Befehl 'ESC UH....' wird ein Hardcopy im BLG-Format gesendet. x1,y1 = Startkoordinaten des Hardcopys (Linke obere Ecke)

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

FÜLLMUSTER

Bei diversen Befehlen kann als Parameter ein Mustertyp eingestellt werden. So können z.B. rechteckige Bereiche mit unterschiedlichen Mustern gefüllt werden. Dabei stehen 16 interne Füllmuster zur Verfügung.



RAHMEN

Mit den Befehl *Rahmen zeichnen* kann ein Rahmentyp eingestellt werden. Es stehen dabei 9 Rahmentypen zur Verfügung. Die Rahmengröße muß mindestens 16x16 Pixel betragen.

GEDREHTER EINBAU

Es ist möglich das eLABEL um **90**° gedreht einzubauen, um ein hochkant-Display mit 72x172 Pixeln zu erhalten. Zur Richtigstellung des Bildinhaltes muss der Befehl 'ESC DO 1' (siehe Seite 13) ausgeführt werden (z.B. im PowerOnMakro).

0°: 'ESC DO 0'

90°: 'ESC DO 1'

POWER-DOWN

Um Strom zu sparen (Betrieb mit Batterie) kann man mit dem Befehlen 'ESC PD mode' und 'ESC PM mode n1 n2' verschiedene Power-down-modes aktivieren.

'ESC PM mode n1 n2':
Nach diesem Befehl werden die Makros n1 bis n2 zyklisch ausgeführt, zwischen den Makros (Zeit ist mit 'ESC PZ h m s' einstellbar) geht das eLABEL in Powerdown.
mode=1 (410µA): Der interne Kontroller wird in den Powedown versetzt der Timer bleibt aktiv.
mode=2 (4,5µA): Zusätzlich zu mode 1 wird das ePAPER komplett abgeschaltet.

'ESC PD mode':
Nach diesm Befehl geht das eLABEL in den Powerdown.
mode=1 (405µA): Der interne Kontroller wird in den Powedown.
mode=2 (0,2µA): Zusätzlich zu mode 1 wird das ePAPER komplett abgeschaltet.

Hinweis: Nach dem Abschalten des ePAPERs in mode 2 wird beim nächsten Displayupdate ein zusätzlicher Blankcycle benötigt.

Das eLABEL wird aus dem Powerdown, je nach gewählten Interface, wie folgt aufgeweckt.
RS232: durch einen L-Pegel an Pin10 RXD (eine Binäre 0 senden)
SPI: durch einen L-Pegel an Pin6 SS
I2C: durch Senden der eingestellten I²C Adresse

nach dem Aufwecken muss mindestens 10ms gewartet werden bevor das eLABEL wieder Empfangsbereit ist.

VORGELADENE FONTS

Es sind standardmäßig, außer dem 8x8 Terminalfont (Font-Nr. 0), noch 3 monospaced, 3 proportionale Zeichensätze und 1 grosser Ziffernfont integriert. Die proportionalen Zeichensätze ergeben ein schöneres Schriftbild, gleichzeitig benötigen sie weniger Platz auf dem Bildschirm (z.B. schmales "i" und breites "W"). Jedes Zeichen kann **pixelgenau** platziert werden und in der Höhe und Breite von 1- bis 4-fach vergrössert werden.

Texte lassen sich linksbündig, rechtsbündig und zentriert ausgeben. Auch eine 90° Drehung ist möglich.

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)	€	ü			ä										ß	
$90 (dez: 144)					ä					ß	ü				ß	

Font 1: 4x6 monospaced

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	⌂
$80 (dez: 128)	€	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	ß	ƒ
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	n	2	3	¯

Font 3: 7x12 monospaced

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)												+		−	.	
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:					

Font 7: grosse Ziffern BigZif57

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)	€	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü					
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º								
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)		ß														
$F0 (dez: 240)									°							

Font 2: 6x8 monospaced

Font 4: GENEVA10 proportional

Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)	€	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü					
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º								
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)		ß														
$F0 (dez: 240)									°							

Font 5: CHICAGO14 proportional

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	Δ
$80 (dez: 128)	€	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü					
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º								
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)		β														
$F0 (dez: 240)									°							

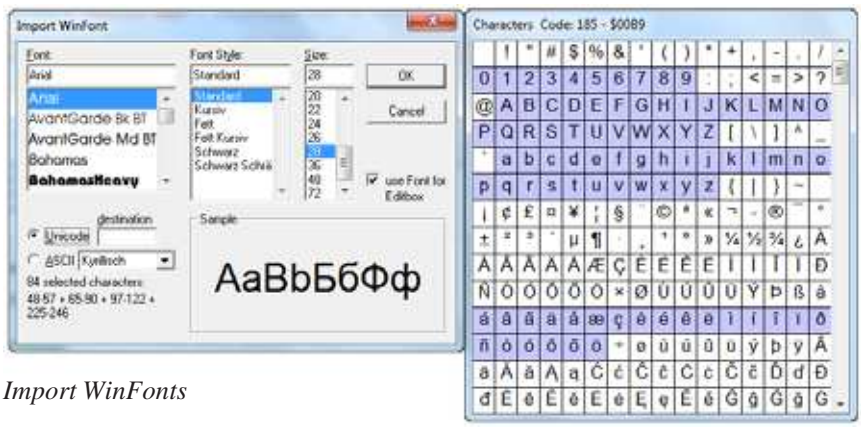
Font 6: Swiss30 Bold proportional

LADBARE ZEICHENSÄTZE

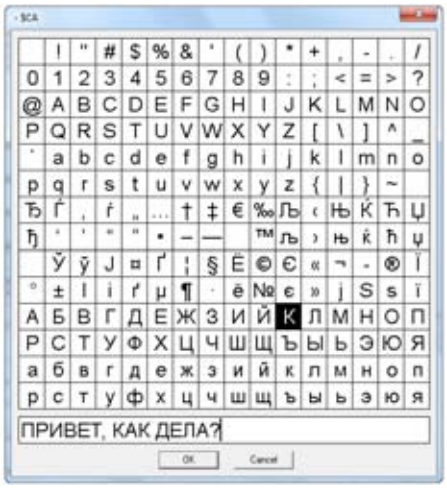
integrierte Schriften im Auslieferungszustand

Compileranweisung "`WinFont:`"
Damit ist es möglich, TrueType-Fonts in verschiedenen Größen zu rastern und einzubinden. Sie können entweder den kompletten Zeichensatz (ASCII) einbinden oder Sie wählen aus dem gesamten Unicode-Zeichensatz bestimmte Zeichen aus. Ein Doppelclick im KitEditor auf den Fontnamen öffnet dazu die Font-Auswahlbox. Um die Verwendung dieser Zeichensätze zu vereinfachen gibt es die kompfortable Möglichkeit einer Zeichen-Auswahlbox. Wird im KitEditor ein String ausgegeben (z.B. #ZL 5,5, "Hallo") kann mit einem Doppelclick auf den String diese geöffnet werden. Es können nun die gewünschten Zeichen ausgewählt werden. Dies ist vor allem bei kyrillischen, asiatischen oder Symbolschriftarten zu empfehelen. Der KitEditor setzt darauf hin automatischen den richtigen ASCII-Code ein. Alternativ zu den Anführungsstrichen können geschweifte Klemmern genutzt werden (z.B. +ZL5,5, {48616C6C6F}).

Compileranweisung "`Font:`"
Verwendet werden kann folgendes Font-Format:
- FXT: Textfont von eDIP- und KIT-Serie



Import WinFonts



Edit Box

MAKRO PROGRAMMIERUNG

Einzelne oder mehrere Befehlsfolgen können als sog. Makros zusammengefasst und im EEPROM fest abgespeichert werden. Diese können dann mit den Befehlen *Makro ausführen* gestartet werden.

Makro (0..255) `Makro:`

Start per Befehl 'ESC MN xx' über serielle Schnittstelle oder von einem anderen Makro aus. Die Makros können auch in definierten Intervallen aus dem Powerdown heraus aufgerufen werden.

Power-On-Makro `PowerOnMakro:`

Start nach dem Einschalten. Hier kann man zB. den Cursor abschalten und einen Startbildschirm definieren.

Reset-Makro `ResetMakro:`

Start nach einem externen Reset.

Brown-Out-Makro `BrownOutMakro:`

Start nach einem Spannungseinbruch <2V.

Achtung: *Wird im PowerOn-, Reset-, oder BrownOut-Makro eine Endlosschleife programmiert, ist das Display nicht mehr ansprechbar. In diesen Fall muss die Ausführung des Power-On Makros unterdrückt werden. Das erreicht man durch die Beschaltung von DPROT: -PowerOff - Pin13 (DPROT) auf GND legen -PowerOn - Pin13 (DPROT) wieder öffnen.*

BILDER IM EEPROM ABGELEGT

Um die Übertragungszeiten der Schnittstelle zu verkürzen, oder auch um Speicherplatz im Prozessorsystem zu sparen, können bis zu 256 Bilder im internen EEPROM abgelegt werden. Der Aufruf erfolgt über den Befehl "ESC U I" oder aus einem Makro heraus. Verwendet werden können alle Bilder im Windows BMP-Format und das interne BLG-Format. Die Erstellung und Bearbeitung erfolgt über Standardsoftware wie z.B. Windows Paint oder Photoshop (nur schwarz/weiss = 1 Bit).

ERSTELLEN INDIVIDUELLER MAKROS UND BILDER

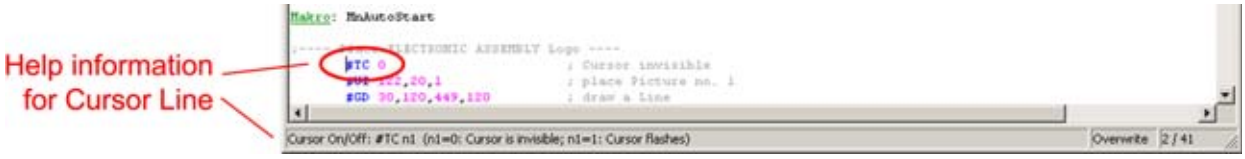
Um nun Ihre speziellen Makros erstellen zu können, benötigen Sie folgende Hilfsmittel:

- um das Display an den PC anschliessen zu können benötigen Sie das als Zubehör erhältlichen Testboard EA 9780-3USB oder einen selbstgebauten Adapter mit Pegelwandler MAX3223 (Applikationsbeispiel siehe Seite 15)
- die Software ELECTRONIC ASSMBLY LCD-Tools; sie enthält einen KitEditor, Compiler sowie Beispiele und Fonts (für PC-Win)
- einen PC mit USB oder serieller Schnittstelle COM

Um eine Befehlsfolge als Makro zu definieren, werden alle Befehle auf dem PC in eine Datei z.B. DEMO.KMC geschrieben. Hier bestimmen Sie, welche Zeichensätze eingebunden werden und in welchen Makros welche Befehlsfolgen stehen sollen.
Sind die Makros über den KitEditor definiert, startet man über F5 den Compiler. Dieser erzeugt eine Datei DEMO.EEP, ist auch das Testboard EA 9780-3USB angeschlossen, oder das Display über einen MAX232 an den PC angeschlossen, dann wird diese Datei automatisch in das EEPROM des Displays gebrannt. Der Programmiervorgang selbst dauert nur wenige Sekunden und sofort danach können die selbstdefinierten Makros und Bilder auch im Display genutzt werden. Eine ausführliche Beschreibung zur Programmierung der Makros finden Sie zusammen mit Beispielen in der Hilfefunktion der ELECTRONIC ASSEMBLY LCD-Tools Software.

HILFE IM KIT-EDITOR (ELECTRONIC ASSEMBLY LCDTOOLS)

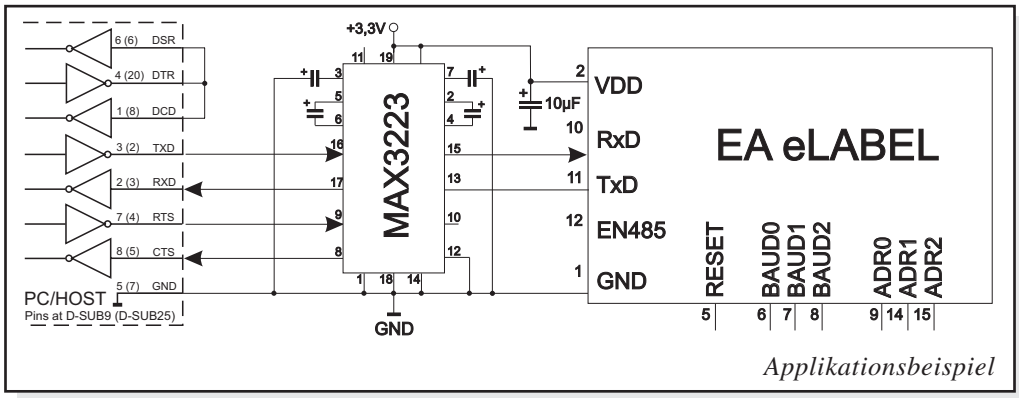
In der Statuszeile am unteren Rand des Editorfensters werden für den aktuellen Befehl mögliche Parameter kurz erläutert. Der Cursor muss dazu in der entsprechenden Zeile stehen.
Für mehr Informationen drücken Sie F1.



Technische Änderung vorbehalten.
Wir übernehmen keine Haftung für Druckfehler und Applikationsbeispiele.

APPLIKATIONSBEISPIEL „ECHTES“ RS-232 INTERFACE

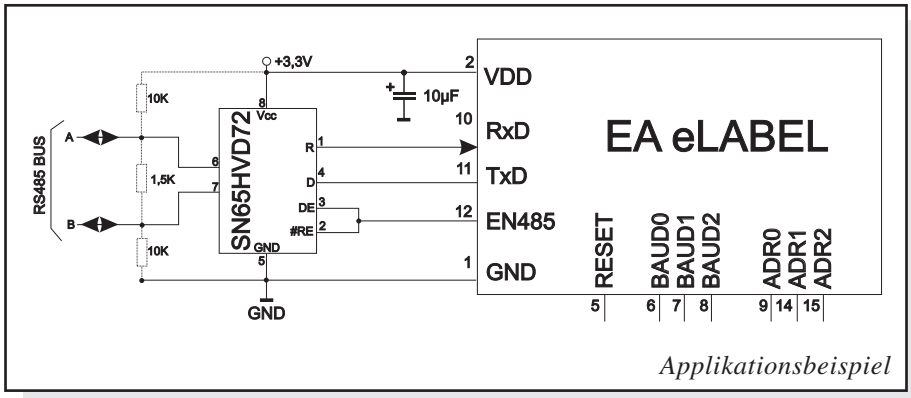
Das eLABEL ist für den direkten Anschluss an eine RS-232 Schnittstelle mit CMOS-Pegeln (VDD) geeignet. Steht jedoch nur eine Schnittstelle mit ±12V Pegeln, so ist ein externer Pegelwandler erforderlich.



Applikationsbeispiel

APPLIKATIONSBEISPIEL: RS-485 INTERFACE

Mit einem externen Umsetzer (z.B. SN65HVD72) kann das eLABEL an einen 2-Draht RS-485 Bus angeschlossen werden. Somit können grosse Entfernungen bis zu 1200m (Ferndisplay) realisiert werden. Betrieb von mehreren eLABELs an einem RS-485 Bus durch Einstellen von Adressen.



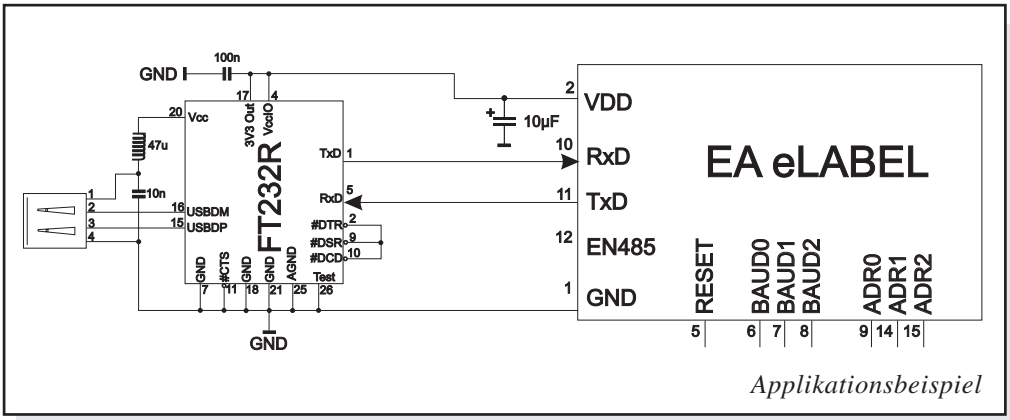
Applikationsbeispiel

Adressierung:

- *Bis zu acht Hardware-Adressen (0..7) per Pins ADR0..ADR2 einstellbar*
- *Das eLABEL mit Adresse 7 ist nach PowerOn selektiert und Empfangsbereit*
- *Die eLABEL mit Adresse 0..6 sind nach PowerOn deselektiert*
- *Bis zu 246 weitere Software-Adressen per Befehl '#KA adr' im PowerOnMakro einstellbar (eLABEL extern auf Adresse 0 setzen)*

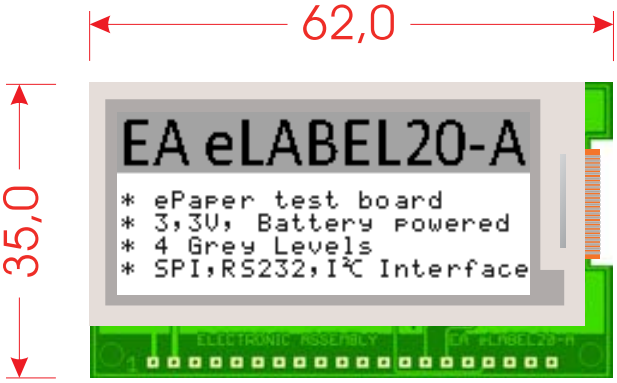
APPLIKATIONSBEISPIEL: USB ANSCHLUSS

Mit einem externen Umsetzer (z.B. FTZ232R) von FTDI kann das eLABEL an einen USB-Bus angeschlossen werden. Virtuelle-COM-Port Treiber gibt es für viele Betriebssyteme auf der FTDI Homepagehttp://www.ftdichip.com/drivers/vcp.htm.

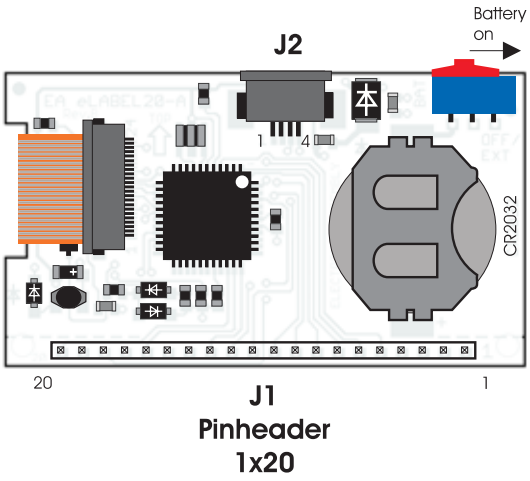


Applikationsbeispiel

ABMESSUNGEN



alle Maße in mm



Pinout J2: RS-232			
Pin	**Symbol**	**In/Out**	**Function**
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	RxD	In	Receive Data
4	TxD	Out	Transmit Data

Hinweise zur Handhabung und zum Betrieb

- *Zur elektrischen Zerstörungs des Moduls kann führen: Verpolung oder Überspannung der Stromversorgung, Überspannung oder Verpolung bzw. statische Entladung an den Eingängen, Kurzschließen der Ausgänge.*
- *Vor dem Abstecken des Moduls muß unbedingt die Stromversorgung abgeschaltet sein. Ebenso müssen alle Eingänge stromlos sein.*
- *Das Display besteht aus Kunststoff und darf nicht mit harten Gegenständen in Berührung kommen. Die Oberfläche kann mit einem weichen Tuch ohne Verwendung von Lösungsmitteln gereinigt werden.*
- *Das Modul ist ausschließlich für den Betrieb innerhalb von Gebäuden konzipiert. Für den Betrieb im Freien müssen zusätzliche Vorkehrungen getroffen werden. Der maximale Temperaturbereich darf nicht überschritten werden. Bei Einsatz in feuchter Umgebung kann es zu Funktionsstörungen und zum Ausfall des Moduls kommen. Das Display ist vor direkter Sonneneinstrahlung zu schützen.*

ELECTRONIC ASSEMBLY GmbH
Zeppelinstraße 19
D-82205 Gilching
Germany

Fon: +49 (0)8105-778090
Fax: +49 (0)8105-778099
e-Mail: info@lcd-module.de
Web: www.lcd-module.de

ELECTRONIC ASSEMBLY
new display design