Issue 11.2015

ELECTRONIC ASSEMBLY
new display design

embedded 2.0" ePAPER

172x72 with intelligence

Dimension: 62x35xx5.5mm

TECHNICAL DATA

* INTELLIGENT ELECTROPHORETIC-ACTIVE-MATIRX- DISPLAY (ePAPER) 2“
* WIDEVIEWING ANGLE
* HIGHEST CONTRAST DISPLY
* 4 GRAYSCALES - BLACK, DARK GRAY, LIGHT GRAY, WHITE
* 3 DIFFERENT INTERFACES ONBOARD: RS-232, I²C-BUS OR SPI-BUS
* 172x72 OR 72X172 DOTS ROTATED MOUNTABLE
* 8 INTEGRATED FONTS
* FONT ZOOM FROM 2mm UP TO 20mm, also rotated with 90°
* SUPPLY 3,0V/3,3V 0,2µA...16mA
* POWER-DOWN-MODE 0,2 µA, WAKE-UP BY SERIAL INTERFACE
* POSITIONING ACCURATE TO THE PIXEL WITH ALL FUNCTIONS
* LINE, DOT, AREA, AND/OR/EXOR, BARGRAPH, CLIPBOARD...
* UP TO 256 PICTURES INTERNALY STORED
* UP TO 256 MACROS PROGRAMMABLE (64kB EEPROM ONBOARD)
* OPERATING TEMPERATURE 0°...+50°C (STORAGE TEMPERATURE -25°..+75°C)

ORDERING CODES

INTELLIGENTES ePAPER 172x72 DOTS	**EA eLABEL20-A**
USB-TESTBOARD FOR PROGRAMMING	**EA 9780-3USB**

Documentation of revision				
Date	**Type**	**Old**	**New**	**Reason / Description**
June, 2013	0.1			preliminary Version
March, 2015	1.0			first release
November,2015	1.1			updated drawing

CONTENT

SPECIFICATION

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

GENERAL

The EA eLABEL20-A is an ePAPER with integrierted intelligence! The ease of use of this display dramatically reduces development times. In addition to a variety of integrated fonts that can be used with pixel accuracy it also offers a whole range of sophisticated graphic functions.

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

This display is programmed by means of commands, such as draw a rectangle from (0,0) to (64,15). No additional software or drivers are required. Strings and images can be placed with pixel accuracy. Text and graphics can be combined at any time. Different character sets can be used at same time. Each character set and the images can be zoomed from 2 to 4 times and rotated in 90° steps. With the largest character set, the words and numbers displayed will fill the screen. The display is designed to work at an operating voltage of +3.3V. Data transfer is either serial and asynchronous in RS-232 format or synchronous via the SPI or I²C specification. To improve data security, a simple protocol is used for all types of transfer.

RS-232 INTERFACE

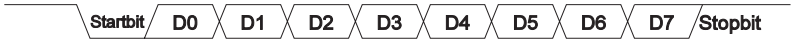
If the display is wired as shown below, the RS-232 interface is selected. The pin assignment is specified in the table on the right. The RxD and TxD lines lead CMOS level (VDD) to a microcontroller, for example, for direct connection.
If “genuine” RS-232 levels are required (e.g. for connection to a PC), an external level converter (e.g. MAX232) is required.

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

BAUD RATES

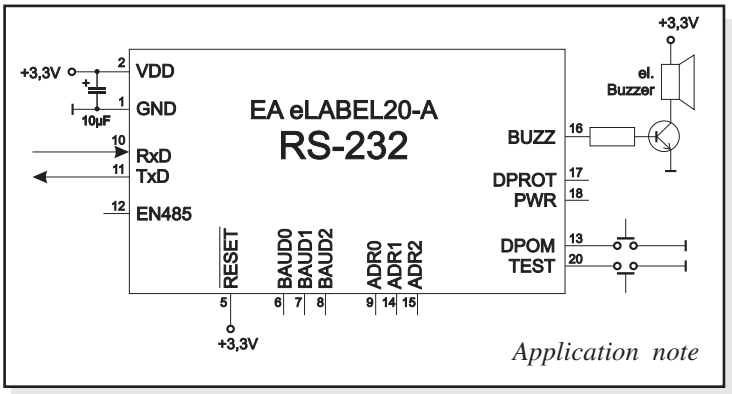
The baud rate is set by means of pins 6, 7 and 8 (baud 0 to 2). The data format is set permanently to 8 data bits, 1 stop bit, no parity.
RTS/CTS handshake lines are not required. The required control is taken over by the integrated software protocol (see pages 6 and 7).

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)



Adressing:

- *Up to eight hardware addresses (0 to 7) can be set by means of Pins ADR0..ADR2*
- *The eLABEL with the address 7 is selected and ready to receive after power-on.*
- *The eLABELs with the addresses 0 to 6 are deselcted after power-on*
- *Up to 246 further software addresses can be set by means of the ‘#KA adr’ command in the power-on macro (set eLABEL externally to address 0)*



Note:
The pins BAUD 0 to 2, ADR 0 to 2, DPOM, DPROT and TEST/SBUF have an internalpullup, which is why only the LO level (0=GND) is to be actively applied. These pins must be left open for a hi level.For RS232 operation (without addressing) the pins ADR 0 to ADR 2 must be left open. On pin 20 (SBUF) the display indicates with a low level that data is ready to be retrieved from the internal send buffer.The line can be connected to an interrupt input of the host system, for example.

SPI INTERFACE

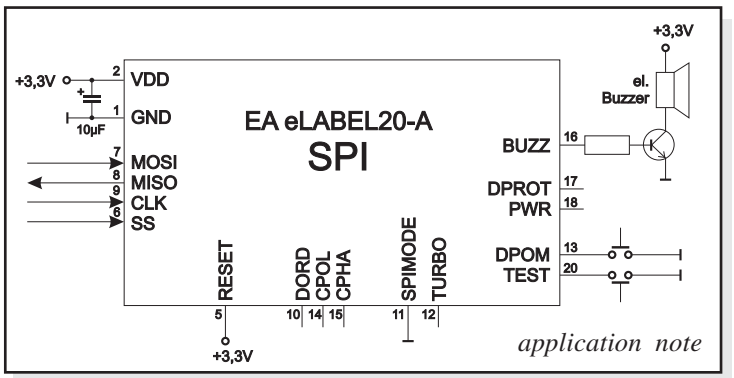
If the display is wired as shown below, SPI mode is activated. The data is then transferred via the serial, synchronous SPI interface.
The transfer parameter will be set via the pins DORD, CPOL and CPHA.

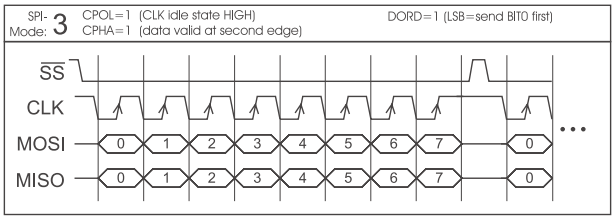
Pinout eLABEL20-A: SPI mode			
Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)
2	VDD		Power supply for logic (+3,0V..3,3V)
3	NC		do not connect
4	NC		do not connect
5	RESET	In	L: Reset
6	SS	In	Slave Select
7	MOSI	In	Serial In
8	MISO	Out	Serial Out
9	CLK	In	Shift Clock (38kHz / 100kHz 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

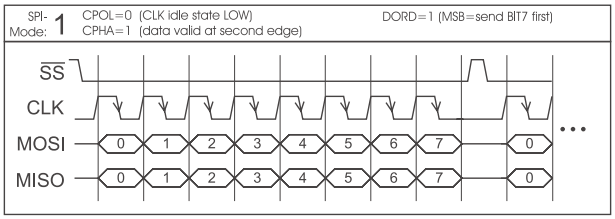
Note:
The pins DORD, CPOL, CPHA, DPOM, DPROT and TEST/SBUF have an internal pullup, which is why only the LO level (0=GND) is to be actively applied. These pins must be left open for a hi level.
On pin 20 (SBUF) the display indicates with a low level that data is ready to be retrieved from the internal send buffer. The line can be connected to an interrupt input of the host system, for example.

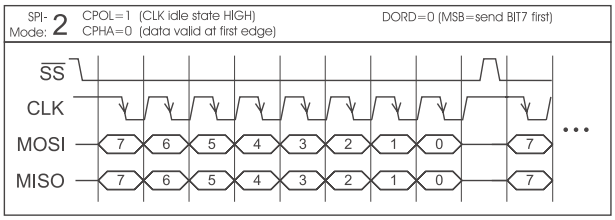
DATA TRANSFER SPI

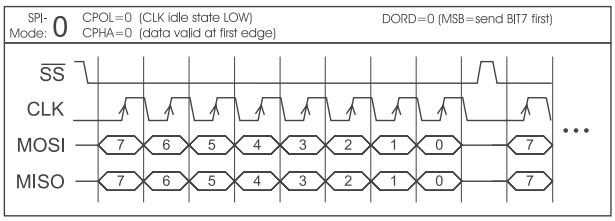
A clock rate up to 38 kHz is allowed in normal mode. In turbo mode (pin 12 low) a clock rate up to 100 KHz without any delay is allowed.
Read operation: to read data (e.g. the „ACK“ byte) a dummy byte (e.g . 0xFF) need to be sent.
Note: The EA eLABEL for internal operation does need a short time before providing the data; therefore a shortpause (no activity of CLK line) of min. 80µs (normal mode) or 10µs (turbo mode) is needed for each byte.









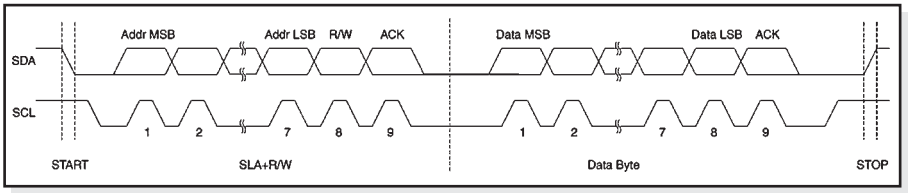


I²C-BUS INTERFACE

If the display is wired as shown below, it can be operated directly on an I²C bus.
8 different base addresses and 8 slave addresses can be selected on the display.
A clock rate up to 38 kHz is allowed in normal mode. In turbo mode (pin 12 low) a clock rate up to 100 KHz without any delay is allowed.

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

Note:
The pins DORD, CPOL, CPHA, DPOM, DPROT and TEST/SBUF have an internal pullup, which is why only the LO level (0=GND) is to be actively applied. These pins must be left open for a hi level.
On pin 20 (SBUF) the display indicates with a low level that data is ready to be retrieved from the internal send buffer. The line can be connected to an interrupt input of the host system, for example..



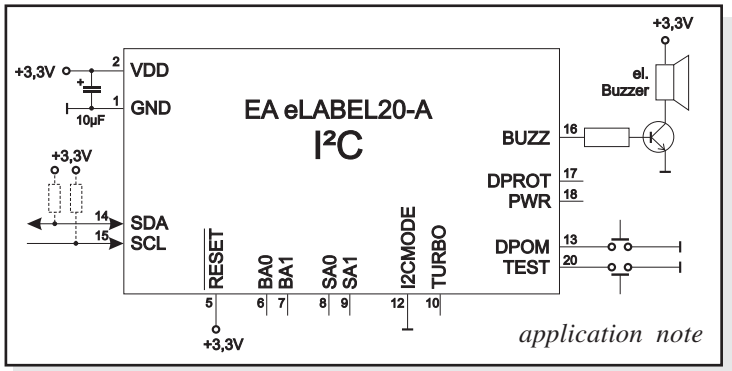
I²C - Address											
Pin 7,6		Base address		I²C address							
BA1	BA0			D7	D6	D5	D4	D3	D2	D1	D0
L	L	$78		0	1	1	1	1	SA1	SA0	RW
L	H	$98		1	0	0	1	1			
H	L	$B8		1	0	1	1	1			
H	H	$D8		1	1	0	1	1			

all pins openn: write $DE
read $DF

DATATRANSFER I²C INTERFACE

principle I2C-bus transfer:
- I²C-Start
- Master-Transmit: EA eLABEL-I²C-address (e.g. $DE), send smallprotocol package (data)
- I²C-Stop
- I²C-Start
- Master-Read: EA eLABEL-I²C-Address (e.g. $DF), read ACK-byte and opt. smallprotocoll package (data)
- I²C-Stop

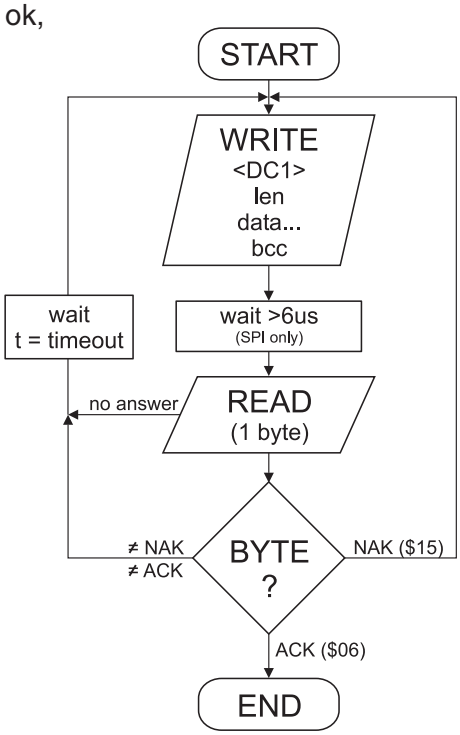
Note: For the read operation that the EA eLABEL needs for internal operation a short time before providing the data; therefore a shortpause (no activity of CLK line) of min. 80µs (normal mode) or 10µs (turbo mode) is needed for each byte.



DATATRANSFER PROTOCOL(SMALL PROTOCOL)

The protocol has an identical structure for all 3 interface types: RS-232, SPI and I²C. Each data transfer is embedded in a fixed frame with a checksum (protocol package). The EA eLABEL20-A acknowledges this package with the character <ACK> (=$06) on successful receipt or <NAK> (=$15) in the event of an incorrect checksum or receive buffer overflow. In the case of <NAK>, the entire package is rejected and must be sent again. Receiving the <ACK> byte means only that the protocol package is ok, there is no syntax check for the command.

Note: It is neccessary to read the <ACK> byte in any case. If the host computer does not receive an acknowledgment, at least one byte is lost. In this case, the set timeout has to be elapsed before the package is sent again. The raw data volume per package is limited to 255 bytes (len <=255). Commands longer than 255 bytes (e.g. Load image ESC UL...) must be split up between a number of packages. All data in the packages are compiled again after being correctly received by the EA eLABEL.



DEACTIVATINGTHE SMALL PROTOCOL

For tests the protocol can be switched off with an L-level at pin 17 = DPROT. In normal operation, however, you are urgently advised to activate the protocol. If you do not, any overflow of the receive buffer will not be detected.

BUILDINGTHE SMALL PROTOCOL PACKAGES

Command/data to the display

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

<DC1> = 17(dez.) = $11
<ACK> = 6(dez.) = $06
len = count of user data (without <DC1>, without checksum bcc)
bcc = 1 byte = sum of all bytes incl. <DC1> and len, modulo 256

The user data is transferred framed by <DC1>, the number of bytes (len) and the checksum (bcc). The display responds with <ACK>.

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	

Example fo a complete datapackage

```
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-example to send a datapcket

Request for content of send buffer

>	<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 = count of user data (without <DC1>, without checksum bcc)
bcc = 1 byte = sum of all bytes incl. <DC1> and len, modulo 256

The command sequence <DC2>, 1, S, bcc empties the display's send buffer. The display replies with the acknowledgement <ACK> and begins to send all the collected data.

Request for buffer information

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

This command queries whether user data is ready to be picked up and how full the display's receive buffer is.

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 I = 73(dez.) = $49
<ACK> = 6(dez.) = $06
send buffer bytes ready = count of bytes stored in send buffer
receive buffer bytes free = count of bytes for free receive buffer
bcc = 1 byte = sum of all bytes incl. <DC2>, modulo 256

Protocol settings

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

This is how the maximum package size that can be sent by the display can be limited. The default setting is a package size with up to 128 bytes of user data. The timeout can be set in increments of 1/100 seconds. The timeout is activated when individual bytes get lost. The entire package then has to be sent again.

<DC2> = 18(dec.) = $12 3 = 3(dez.) = $03 D = 68(dez.) = $44
packet size for send buffer = 1..128 (standard: 128)
timeout = 1..255 in 1/100 seconds (standard: 200 = 2 seconds)
bcc = 1 byte = sum of all bytes incl. <DC2>, modulo 256
<ACK> = 6(dec.) = $06

Request for protocol settings

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

This command is used to query protocol settings.

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 P = 80(dez.) = $50
<ACK> = 6(dez.) = $06
max. packet size = count of maximum user data for 1 package (eLEABEL20-A = 255)
akt. send packet size = current package size for send
akt. timeout = current timeout in 1/100 seconds
bcc = 1 byte = sum of all bytes incl. <DC2>, modulo 256

Repeat the last package

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

If the most recently requested package contains an incorrect checksum, the entire package can be requested again. The reply can then be the contents of the send buffer (<DC1>) or the buffer/protocol information (<DC2>).

<DC2> = 18(dez.) = $12 1 = 1(dez.) = $01 R = 82(dez.) = $52
<ACK> = 6(dez.) = $06
<DC1> = 17(dez.) = $11
len = count of user data in byte (without ckecksum, without <DC1> or <DC2>)
bcc = 1 byte = sum of all bytes incl. <DC2> and len, modulo 256

Adressing (only for RS232/RS485)

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

This command can be used to select or deselect the eLABEL with the address adr.

<DC2> = 18(dez.) = $12 3 = 3(dez.) = $03 A = 65(dez.) = $41
select or deselect: 'S' = $53 or 'D' = $44
adr = 0..255
bcc = 1 byte = sum of all bytes incl. <DC2> and adr, modulo 256
<ACK> = 6(dec.) = $06

USINGTHE SERIAL INTERFACE

The operating unit can be programmed by means of various integrated commands. Each command begins with ESCAPE followed by one or two command letters and then parameters. There are two ways to transmit commands:

1. ASCII mode

- The ESC character corresponds to the character '#' (hex: $23, dec: 35).
- The command letters follow directly after the '#' character.
- The parameters are transmitted as plain text (several ASCII characters) followed by a separating character (such as a comma ','), also after the last parameter e.g.: #GD0,0,159,103,
- Strings (text) are written directly without quotation marks and concluded with CR (hex: $0D) or LF (hex: $0A).

2. Binary mode

- The escape character corresponds to the character ESC (hex: $1B, dec: 27).
- The command letters are transmitted directly.
- The coordinates x and y are transmitted as 8-bit binary values
- All the other parameters are transmitted as 8-bit binary values (1 byte).
- Strings (text) are concluded with CR (hex: $0D) or LF (hex: $0A) or NUL (hex: $00).

No separating characters, such as spaces or commas, may be used in binary mode.
The commands require **no final byte**, such as a carriage return (apart from the string $00).).

ALL COMMANDS AT A GLANCE

The built-in intelligence allows an easy creation of your individual screen content. Below mentioned commands can be used either directly via the serial interface (see page 12) or together with the selfdefinable macro.

Display commands (effect the entire display)								after reset
Command	**Codes**						**Remarks**	
Set display orientation	ESC	D	O	n1			n1=0: 0° = 172x 72 dots; terminal: 21 columns, 9 lines n1=1: 90° = 72x172 dots; terminal: 9 columns, 21 lines	0°
Display update			U				The entire contents of the RAM is copied to the display	
Display autoupdate time			Z	n1	n2		n1: delaytime after last command before update n2: force update after first command (if commands are sent continuously) n1,n2=1..255 in 1/10sec; (n1,n2=0: autoupdate off; use ESC DU for update)	3, 50
Delete display	ESC	D	L				Delete display contents (all pixels off)	
Invert display			I				Invert display contents (invert all pixels)	
Fill display			F	n1			Fill display contents n1=1: white, n1=2: lightgray; n1=3: darkgrey; n2=4: black	
Switch display on	ESC	D	E				Display contents become invisible but are retained, commands are still possible	
Switch display off			A				Display contents become visible again	on

Text commands									after reset
Befehl	**Codes**							**Remarks**	
Settings									
Set text color	ESC	F	Z	fg	bg			fg=foreground- bg=background-color for strings and characters 1=black; 2=darkgrey; 3=grey; 4=white or 0=transparent; The sensless combination of transparent background and foreground is used to invert all dots (=complementary).	1,4
Set font	ESC	Z	F	n1				Set font with the number n1 = 0..15	0
Set font zoom factor			Z	n1	n2			n1 = x-zoom factor (1x..4x); n2 = y-zoom factor (1x..4x)	1,1
Additonal line spacing			Y	n1				Insert n1 = 0..15 dots between two lines as additional spacing	0
Spacewidth			J	n1				Spacewidth: n1=0 use from font; n1=1 same width as number; n1>=2 width in dot	0
Text angle			W	n1				Text angle: n1=0: 0°; n1=1: 90°;	0
Output strings									
Output string L: left justified C: centered R: right justified	ESC	Z	L C R	x1	y1	Text ...	NUL	A string is output to x1,y1; string termination is: 'NUL' ($00), 'LF' ($0A) or 'CR' ($0D); several lines are seperated by the character '|' ($7C); The character '\' ($5C, backslash) cancels the special funtion of characters '|\'; e.g. "name\\test.txt" => "name\test.txt"	
String for terminal	ESC	Z	T	Text ...				Command to output a string (text...) from a macro to the terminal	

Change / draw rectangular areas										after reset
Command	**Codes**								**Remarks**	
Delete area	ESC	R	L	x1	y1	x2	y2		Delete an area from x1,y1 to x2,y2 (all pixels off)	
Invert area	ESC	R	I	x1	y1	x2	y2		Invert an area from x1,y1 to x2,y2 (invert all pixels)	
Fill area	ESC	R	F	x1	y1	x2	y2	n1	Fill an area from x1,y1 to x2,y2 with color n1=0..4 1=black; 2=darkgrey; 3=grey; 4=white or 0=inverse	
Set pattern color	ESC	F	M	fg	bg				fg=foreground; bg=background color for patterns 1=black; 2=darkgrey; 3=grey; 4=white or 0=transparent; The sensless combination of transparent background and foreground is used to invert all dots (=complementary).	1,4
Area with fill pattern	ESC	R	M	x1	y1	x2	y2	n1	Draw an area from x1,y1 to x2,y2 with pattern number n1=0..15	
Draw box with fill pattern			O	x1	y1	x2	y2	n1	Draw a rectangle x1,y1 to x2,y2 and fill with pattern n1=0..15	
Set border color	ESC	F	R	fg	bg				fg=foreground (border); bg=background (filling) color for borders 1=black; 2=darkgrey; 3=grey; 4=white or 0=transparent; The sensless combination of transparent background and foreground is used to invert all dots (=complementary).	1,4
Draw border box		R	R	x1	y1	x2	y2	n1	Draw border number n1=1..9 from x1,y1 to x2,y2	

Straight lines and points									after reset
Command	**Codes**							**Remarks**	
Settings									
Set Line Color	ESC	F	G	n1				Set the color n1 for lines 1=black; 2=darkgrey; 3=grey; 4=white or 0=inverse	1
Point size / line thickness	ESC	G	Z	n1	n2			n1 = x-point size (1..15); n2 = y-point size (1..15);	1,1
Draw lines and points									
Draw point	ESC	G	P	x1	y1			Set a point at coordinates x1,y1	
Draw straight line	ESC	G	D	x1	y1	x2	y2	Draw a straight line from x1,y1 to x2,y2	
Continue straight line	ESC	G	W	x1	y1			Draw a straight line from the last end point to x1,y1	0, 0
Draw rectangle	ESC	G	R	x1	y1	x2	y2	Draw four straight lines as a rectangle from x1,y1 to x2,y2	

Bitmap commands									after reset
Command	**Codes**							**Remarks**	
Settings									
Set image color	ESC	F	U	fg	bg			fg=foreground- bg=background-color for monochrome images 1=black; 2=darkgrey; 3=grey; 4=white or 0=transparent; The sensless combination of transparent background and foreground is used to invert all dots (=complementary).	1,4
Image zoom factor	ESC	U	Z	n1	n2			n1 = x-zoom factor(1x..4x); n2 = y-zoom factor (1x..4x)	1,1
Image angle	ESC	U	W	n1				Image angle: n1=0: 0°; n1=1: 90°	0
Output									
Load internal image	ESC	U	I	x1	y1	nr		Load internal image with nr (0..255) from EEPROM to x1,y1	
Load image	ESC	U	L	x1	y1	BLG data ...		Load an image to x1,y1; data... = image in BLH-format	
Hardcopy									
Send hardcopy	ESC	U	H	x1	y1	x2	y2	An image area x1,y1 to x2,y2 is put into the sendbuffer. The image is send in BLG-format	

Clipboard commands (Buffer for display area)								after reset
Command	**Codes**							**Remarks**	
Save display contents	ESC	C	B					The entire contents of the display are copied to the clipboard as an image area	
Save area	ESC	C	S	x1	y1	x2	y2	The image area from x1,y1 to x2,y2 is copied to the clipboard	
Restore area	ESC	C	R					The image area on the clipboard is copied back to the display	
Copy area	ESC	C	K	x1	y1			The image area on the clipboard is copied to x1,y1 in the display	

General commands								after reset
Command	**Codes**							**Remarks**	
Send commands									
Send bytes	ESC	S	B	len	data ...			len (=1 to 255) bytes are sent to the sendbuffer data... = data to send. In the source text of the macro programming, the number len must not be specified. This is counted by the eLABEL-compiler and entered.	
Send version	ESC	S	V					The version is sent as a string to sendbuffer, e.g. "EA eLABEL20-A V1.0 Rev.A"	
Send projectname	ESC	S	J					The macro project name is sent as a string to sendbuffer, e.g. "init / delivery"	
Send internal infos	ESC	S	I					Internal information about the eLABEL is sent to the sendbuffer	
Power commands									
Enter Power Down	ESC	P	D	mode				Enter eLABEL to PowerDown in mode=1..2 mode=1 (~405µA): switch off microcontollor mode=2 (~ 0,2µA): switch off microcontollor + ePaper (requires an additional blankcycle)	
Enter Powerdown and run macros	ESC	P	M	mode	n1	n2		Enter PowerDown in mode mode=1..2 and run macros from n1 to n2 cyclically mode=1 (~410µA): switch off microcontollor mode=2 (~ 4,5µA): switch off microcontollor + ePaper (requires an additional blankcycle)	
Set time between macros	ESC	P	Z	h	m	s		define the time between macros in powerdown h=hours; m=minutes; s=seconds; (±10%)	0,0,8
Other commands									
Run macro	ESC	M	N	n1				Call the macro with the number n1(0..255) (max. 7 levels)	
Set RS485 address	ESC	K	A	adr				For RS232/RS485 operation only and only possible when Hardware address is 0. The eLABEL is assigned a new address adr (in the Power-On-Macro).	
Buzzer output	ESC	Y	S	n1				The buzzer output (pin 16) becomes n1=0:OFF; n1=1:ON; n1=2 to 255:ON for n1/10s	OFF
Wait (pause)	ESC	X	n1					Wait n1/10s before next command is executed	

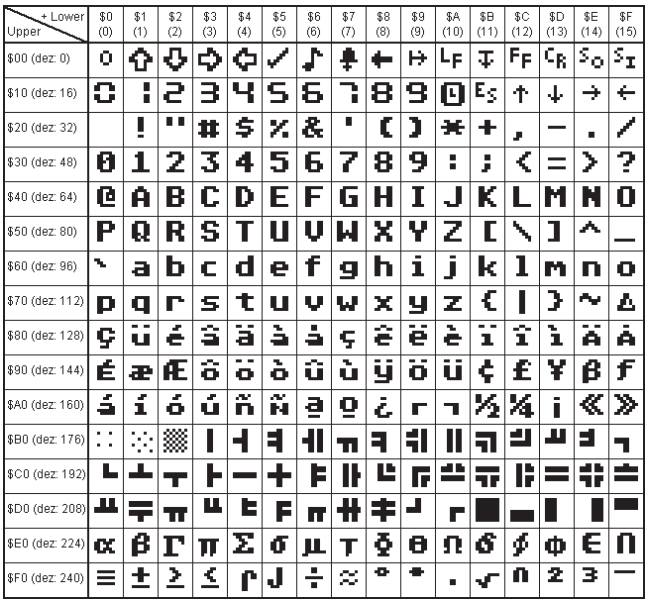
TERMINAL MODE

All the incoming characters are displayed in ASCII format on the terminal (exception: CR,LF,FF,ESC,'#'). The prerequisite for this is a working protocol frame (pages 10 and 11) or a deactivated protocol.

Line breaks are automatic or can be executed by means of the 'LF' character. If the last line is full, the contents of the terminal scroll upward. The 'FF' character (page feed) deletes the terminal. The character '#' is used as an escape character and thus cannot be displayed directly on the terminal. If the character '#' is to be output on the terminal, it must be transmitted twice: '##'.

The terminal has its own level for displaying and is thus entirely independent of the graphic outputs.

If the graphics screen is deleted with 'ESC DL', for example, that does not affect the contents of the terminal window. The terminal font is fixed in the ROM and can also be used for graphic outputs 'ESC Z...' (set FONT nr=0).



Terminal-Font (Font 0): 8x8 monospaced

Terminal commands						After reset
Command	**Codes**				**Remarks**	
Form feed FF (dec:12)	**^L**				The contents of the screen are deleted and the cursor is placed at pos. (1,1)	
Carriage return CR(13) Line feed LF (dec:10)	**^M** **^J**				Cursor to the beginning of the line on the extreme left Cursor 1 line lower, if cursor in last line then scroll	
Position cursor Save cursor position	**ESC**	**T**	**P** **S**	C | L	C=column; L=line; origin upper-left corner (1,1) The current cursor position is saved	1,1
Restore cursor position Terminal off	**ESC**	**T**	**R** **A**		The last saved cursor position is restored Terminal display is switched off; outputs are rejected	
Terminal on	**ESC**	**T**	**E**		Terminal display is switched on;	On
Output version	**ESC**	**T**	**V**		The version no. is output in the terminal (e.g. "EA eLABEL20-A V1.0 Rev.A")	
Output project name Output information	**ESC**	**T**	**J** **I**		The macro project name is output to the terminal (e.g. "init / delivery state") The terminal is initialized and deleted; software version, hardware revision, the macro project name and the CRC-checksum is ouput to the terminal	

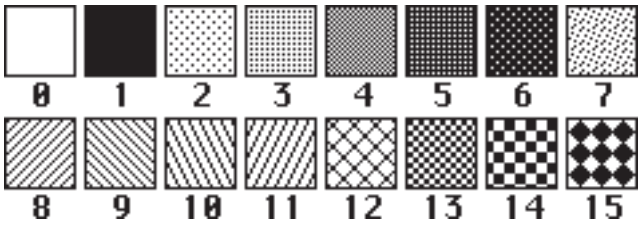
RESPONSES OFTHE EA ELABEL20-AVIA SERIAL INTERFACE

The table below contains all response codes. Some response data will come automatically some others on request. In addition to that with command 'ESC SB ...' user is able to transmit individual data packages. All reponses are placed into the sendbuffer. With the smallprotocol command 'Request for content of send buffer' (see page10) the host can read out the sendbuffer. This can be done per polling, alternatively pin 20 'SBUF' shows with Low-level that data is ready to transmit.

Responses of the eLABEL						
Id		**num**	**data**			**Remarks**
Response with length specification num (placed into sendbuffer)						
ESC	**V**	num	version string...			After the 'ESC S V' command, the version of the eLABEL firmware is transmitted as a string e.g. "EA eLABEL20-A V1.0 Rev.A"
ESC	**J**	num	projectname string...			After the 'ESC S J' command, the macro-projectname is transmitted. e.g. "init / delivery state"
ESC	**I**	num	X-dots, Y-dots, Version, Touchinfo, CRC-ROM, CRC-ROMsoll EEP in KB, CRC-EEP, CRC-EEPsoll, EEPlen			num=21; after the 'ESC S I' command, internal information is sent by eLABEL (16-Bit integer values LO-HI Byte) Version: LO-Byte = version number Software; HI-Byte = Hardware revison letter touch Touchinfo: LO-Byte = '-|+' X direction detected; HI-Byte = '-|+' Y direction detected EEPlen: number of user bytes in eeprom memory (3 Bytes: LO-, MID- HI-Byte)
Responses without length specification (num)						
ESC	**U**	L	x1	y1	image data... (BLG-Format)	after the 'ESC UH....' command, a hard copy is sent in BLG Format x1,y1 = Start coordinates of the hard copy (upper left corner)

FILL PATTERN

A pattern type can be set as a parameter with various commands. In this way, for example, rectangular areas and bar graphs can be filled with different patterns. There are 16 internal fill patterns available.



FRAMES

A frame type can be set by using the Draw frame or Draw frame box command. 9 frame types are available. The frame size must be at least 16x16 pixels.

TOPVIEW ANDTWISTED MOUNTING

The prefered view of the eLABEL20-A is bottom view, (6 o'clock). It is possible to mount the display turned with 90° to gain a portrait mode display with 72x172 pixels.
To set the viewing direction you have to run (e.g. in PowerOnMacro) the command 'ESC DO 1' (refer to p. 8).

0°: 'ESC DO 0'

90°: 'ESC DO 1'

POWER DOWN MODE

To save energy (battery operation), you can activate one of three power-down modes by means of the command ‘ESC PD n1’ (see page 9 below).

'ESC PM mode n1 n2':
With this command the macros n1 to n2 are cyclically called. Between the macros (time is set by command 'ESC PZ h m s') the displa's powerdown mode is entered.
mode=1 (410µA): internal controller is in power down state, timer and display stays active.
mode=2 (4,5µA): additionally to mode 1 the ePAPER is shut down

'ESC PD mode':
This command sets the eLABEL to powerdown mode.
mode=1 (405µA): internal controller is in power down state, timer and display stays active.
mode=2 (0,2µA): additionally to mode 1 the ePAPER is shut down

NOTE: After shut down of the eLABEL (mode 2) one blank cycle is needed to show new content.

Waking up the eLABEL from powerdown mode:
RS232: low-level on pin 10 RXD (send a binary 0)
SPI: low-level on pin 6 SS
I2C: sending the correct I^2C adress

After wakre up, the eLABEL needs another 10 ms to be fully operable again.

PRELOADED FONTS

Apart from the 8x8 terminal font (font no. 8), 3 additional monospaced fonts, 3 proportional fonts and 1 large numeric font are integrated as standard. The proportional fonts result in a more attractive appearance, and at the same time require less space on screen (e.g. the “i” is narrow and the “W” is wide). Each character can be positioned with pixel accuracy and the width and height can be scaled. Each text can be output left justified, right justified or centered. 90° rotation is also possible. Macro programming permits additional fonts to be integrated (up to 15). This is done using the LCD-Tools.

+ 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										Ä	
$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: big digits 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 4: GENEVA10 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)		ß														
$F0 (dez: 240)									°							

Font 5: CHICAGO14 proportional

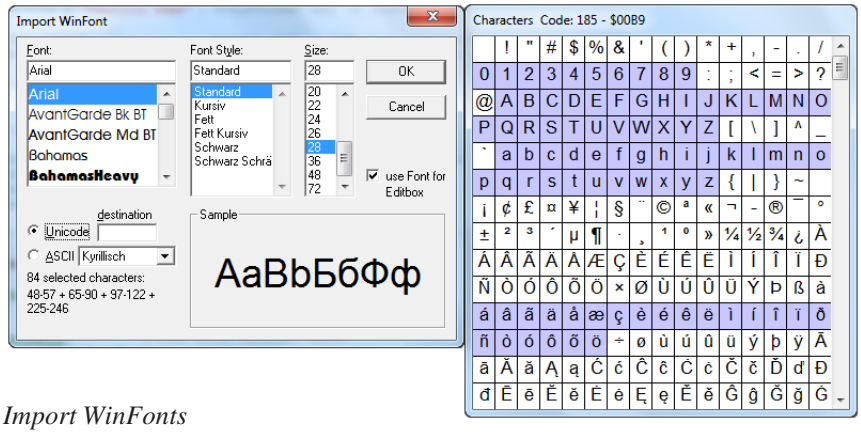
+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	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)		ß														
$F0 (dez: 240)									°							

Font 6: Swiss30 Bold proportional

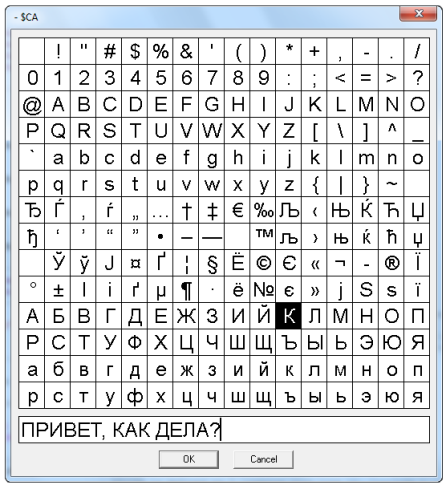
ADDITIONAL FONTS

integrated charset at delivery state

Compile statement "`WinFont:`"
It is possible to raster TrueType-Fonts in different sizes whitch can be used. A doubleclick to the fontname within the KitEditor opens the font selection box. To simplify the use of fonts, there is the possibilty of an edit box. If you output a string with KitEditor (e.g. #ZL 5,5, "Hello"), you can perform a double click on the string to open it. Now you can select the desired characters. This is mainly recommended using cyrillic, asian or symbol fonts.
In that way, the KitEditor automatically places the right ASCII-Code. Alternativly you can use instead of the quotation mark curly brackets (e.g. #ZL 5,5, {48656C6C6F}).
Compiler option "`Font:`"
Following font formats can be used:
- FXT: Textfont as used by eDIP and KIT series



Import WinFonts



Edit Box

MACRO PROGRAMMING

Single or multiple command sequences can be grouped together in macros and stored in the data flash memory. You can then start them by using the Run macro commands.

Normal macro `Macro:`

These are started by means of an 'ESC MN xx' command via the serial interface or from another macro. These macros can be called from the powerdown mode with user defined intervals.

Power-On-Macro `PowerOnMacro:`

Started after power-on. You can define an opening screen, for example.

Reset-Macro `ResetMacro:`

Started after an external reset (low level at pin 5).

Brown-Out-Macro `BrownOutMacro:`

Started after a voltage drop under 2.0V (typ.).

Important: *If a continuous loop is programmed in a power-on, reset, watchdog or brown-out macro, the display can no longer be adressed. In this case, the execution of the power-on macro must be suppressed. You do this by wiring DPOM:*
- PowerOff - connect pin 13 (DPOM) to GND
- PowerOn - open pin 13 (DPOM) again.

STORING IMAGES IN THE DATA FLASH MEMORY

To reduce the transmission times of the interface or to save storage space in the processor system, up to 256 images can be stored in the internal EEPROM with the "PICTURE" compiler directive. They can be called using the "ESC U I" command or from within a macro.
All images in the Windows BMP format (monochrome images only) can be used. They can be created and edited using widely available software such as Windows Paint or Photoshop or the bitmap editor shipped with the product.

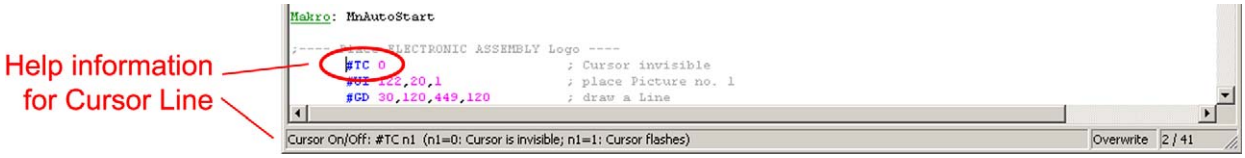
CREATING INDIVIDUAL MACROS AND IMAGES

To create your own fonts, images, animations and macros you need the following:
- To connect the display to the PC, you need the EA 9780-3USB USB evaluation board, which is available as an accessory, or a self-built adapter with a MAX232 level converter (see the application example on page 15).
- ELECTRONIC ASSEMBLY LCD-Tools*), which contains a kiteditor, bitmapeditor, compiler, fonts, images, border, pattern and examples (for Windows PCs)
- A PC with an USB or serial COM interface

To define a sequence of commands as a macro, all the commands are written to a file on the PC (e.g. DEMO.KMC). You specify which character sets are to be integrated and which command sequences are to be in which macros. If the macros are defined using the kit editor, you start the compiler using F5. This creates a file called DEMO.EEP. If an EA 9780-3USB evaluation board is also connected or the display is connected to the PC via a MAX232, this file is automatically burned in the display's data memory.
You can send the created macrofile *.EEP with any other system to the EA eLABEL20-A. All programming commands are inside this file, so you only need to send the content of the *.df file (via RS232, SPI or I2C with smallprotocol in packets) to the EA eLABEL-A.

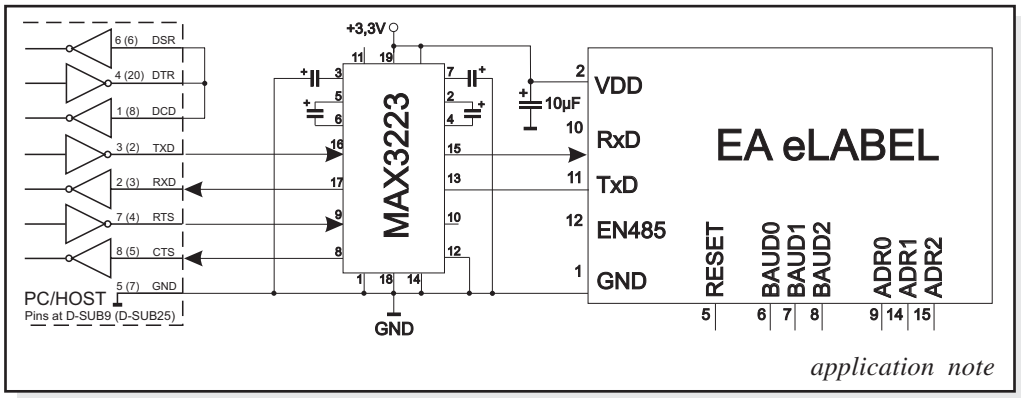
KIT-EDITOR HELP (ELECTRONIC ASSEMBLY LCDTOOLS)

At bottom from the KitEditor window in the statusline you can see a short description for the current command and the parameters. For more information press F1.



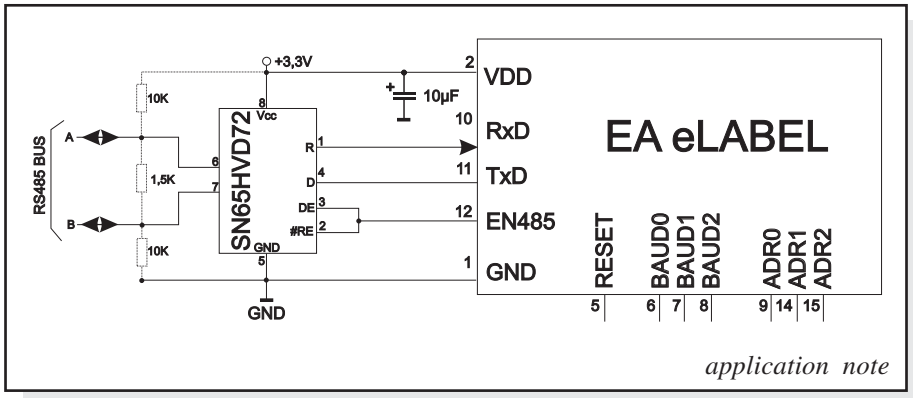
APPLICATION EXAMPLE „REAL“ RS-232 INTERFACE

The eLABEL fits for direct connection to a RS-232 interface with CMOS level (VDD).
If you have an interface with ±12V level, an external levelshifter is needed.



APPLICATION EXAMPLE: RS-485 INTERFACE

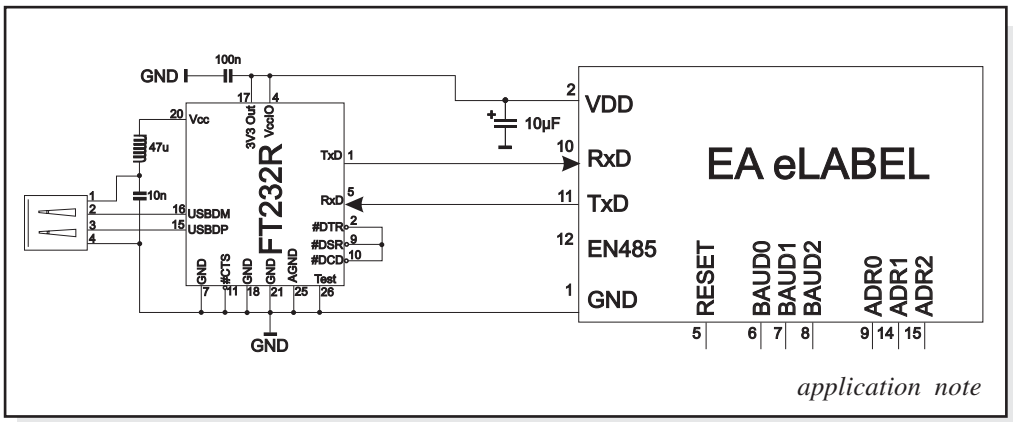
With an external converter (e.g. SN75176), the EA eLABEL can be connected to a 2-wire RS-485 bus. Large distances of up to 1200 m can thus be implemented (remote display).
Several EA eLABEL displays can be operated on a single RS-485 bus by setting addresses.



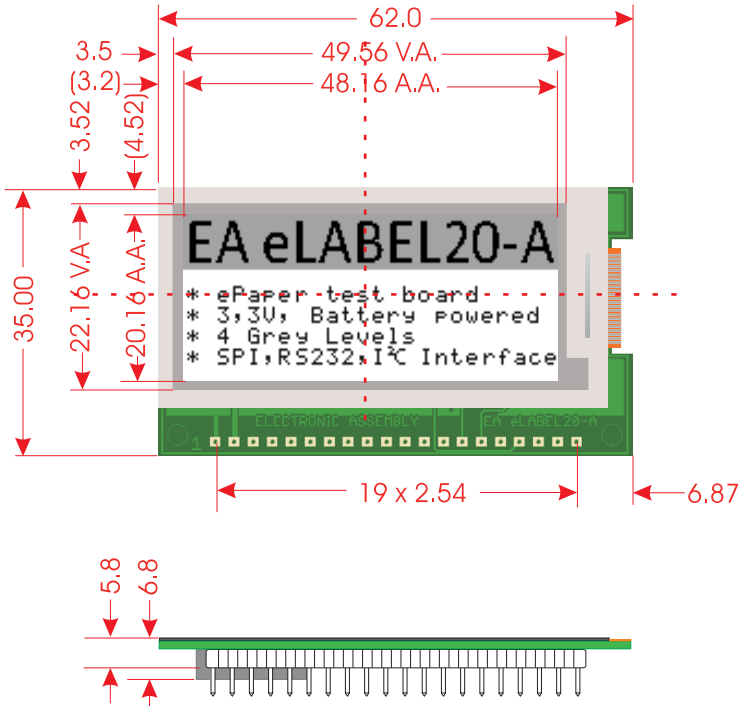
Adressing:
- *Up to eight hardware addresses (0 to 7) can be set by means of Pins ADR0..ADR2*
- *The eLABEL with the address 7 is selected and ready to receive after power-on.*
- *The eLABELs with the addresses 0 to 6 are deselcted after power-on*
- *Up to 246 further software addresses can be set by means of the '#KA adr' command in the power-on macro (set eLABEL externally to address 0)*

APPLICATION EXAMPLE: USB INTERFACE

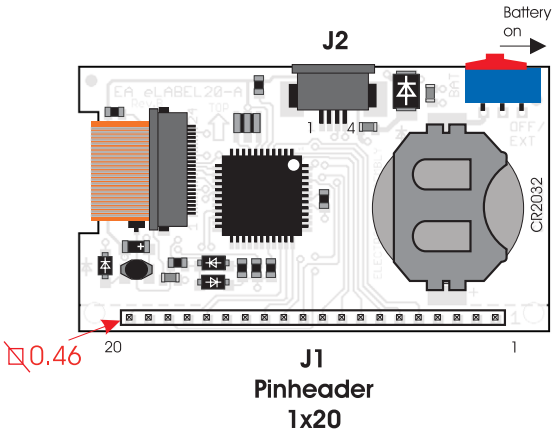
With an external converter (e.g. FT232R from FTDI) the eLABEL can be connected to an USB-Bus. Virtual-COM-Port drivers are available for different Systems on the FTDI Homepage: http://www.ftdichip.com/drivers/vcp.htm.



DIMENSION



all dimensions are 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

NOTES ON HANDLING AND OPERATION

- The module can be destroyed by polarity reversal or overvoltage of the power supply; overvoltage, reverse polarity or static discharge at the inputs; or short-circuiting of the outputs.
- It is essential that the power supply is switched off before the module is disconnected. All inputs must also be deenergized.
- The display are made of plastic and must not come into contact with hard objects. The surfaces can be cleaned using a soft cloth without solvents.
- The module is designed exclusively for use in buildings. Additional measures have to be taken if it is to be used outdoors. The maximum temperature range of 0 to +50°C must not be exceeded. If used in a damp environment, the module may malfunction or fail. The display must be protected from direct sunshine.

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.com

ELECTRONIC ASSEMBLY
new display design