DISPLAY INDUSTRIAL SOLUTIONS VISIONS

Issue 04.2022

EA eDIP320-8

Intelligent HMI 320x240, RS-232, SPI, I2C

Dimensions: 138x105x10mm

TECHNICAL DATA

* LCD GRAPHIC DISPLAY WITH A RANGE OF GRAPHIC FUNCTIONS
* 8 BUILT-IN FONTS
* FONT ZOOM FROM approx. 2mm TO approx. 80mm, also rotated by 90°
* 3 DIFFERENT INTERFACE ONBOARD: RS-232, I²C-BUS OR SPI-BUS
* 320x240 DOTS WITH LED BACKLIGHT BLUE NEGATIVE OR
* BLACK&WHITE POSITIVE, FSTN TECHNOLOGY AND AMBER
* POWER SUPPLY +5V@ typ. 50mA / 240mA (WITHOUT / WITH LED BACKLIGHT)
* POSITIONING **ACCURATE TO THE PIXEL** WITH ALL FUNCTIONS
* DRAW LINE, DOT, AREA, AND/OR/EXOR, BARGRAPH...
* CLIPBOARD FUNCTION, PULL-DOWN MENU
* UP TO 16 PAGES á 256 PICTURES INTERNALY STORED
* UP TO 16 PAGES á 768 MACROS PROGRAMMABLE (80kB ON-BOARD FLASH)
* MIX TEXT AND GRAPHIC, FLASHING ATTRIBUTE: ON/OFF/ INVERT
* BACKLIGHT BRIGHTNESS PER SOFTWARE
* ANALOGUE TOUCH PANEL: VARIABLE GRID
* FREE DEFINABLE KEY AND SWITCH
* POWER-DOWN-MODE (TYP. 150µA) WITH WAKEUP BY TOUCH

ORDERING CODES

320x240 DOTS, WHITE LED BACKLIGHT, BLUE NEGATIVE	**EA eDIP320B-8LW**
AS ABOVE, BUT WITH TOUCH PANEL	**EA eDIP320B-8LWTP**
320x240 DOTS, WHITE LED BACKLIGHT, POSITIVE MODE, FSTN	**EA eDIP320J-8LW**
AS ABOVE, BUT WITH TOUCH PANEL	**EA eDIP320J-8LWTP**
MONTING FRAME (ALUMINIUM), BLACK ANODIZED	**EA 0FP321-8SW**
PROGRAMMER FOR USB INCL. CABLE, CD FOR WIN98/ME/2000/XP	**EA 9778-1USB**
STARTER KIT, (1x EA eDIP320B-8LWTP + USB-PROGRAMMER + CD)	**EA EVALeDIP320B**
STARTER KIT, (1x EA eDIP320J-8LWTP + USB-PROGRAMMER + CD)	**EA EVALeDIP320J**

Documentation of revision				
Date	**Type**	**Old**	**New**	**Reason / Description**
9.11.2006	V1.0			1st. edition
3.4.2007	V1.1	bug fix: - corrupted character chain - bargraph return code fixed - single picture for touch keys		
9.6.2011	V1.2		changed specification of pull-up resistor (RESET pin)	Changed specification, because of product change notification (SC112002) of ATMEL.
29.10.2012	V1.3		bug fix: - blink function within a string repaired ('@', '~') - automatic touch invert function repaired (when using '#AL')	
9.9.2012	V1.4	bug fix: - sometimes display starts with the "Test Mode"		

CONTENTS

GENERAL

The EA eDIP series of displays are the world's first displays with integrated intelligence. In addition to a variety of integrated fonts that can be used with pixel accuracy, they offer a whole range of sophisticated graphics functions.
The displays are ready for operation immediately with an operating voltage of 5V. They are controlled via one of the 3 integrated interfaces: RS-232, SPI or I²C. The displays are "programmed" by means of high-level language-type graphics commands. There is no longer any need for the time-consuming programming of character sets and graphics routines. The ease of use of this display with its touch panel dramatically reduces development times.

HARDWARE

The display is designed to work at an operating voltage of +5V. 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.

ANALOG TOUCH PANEL

All versions are also available with an integrated touch panel: You can make entries and menu or bar graph settings by touching the display. The labeling of the "keys" is flexible and can also be changed during runtime (different languages, icons). The drawing of the individual "keys" and the labeling is handled by the integrated software.

LED ILLUMINATION: WHITE

All displays are equipped with modern, energy-saving LED illumination. Brightness can be varied 0~100% by command. While the black&white display (J-LW) can also be read with the illumination switched off entirely, the blue-white display (B-LW) needs at least minimal illumination if it is to be read.
We recommend the black&white or amber version for use in direct sunlight. In all other cases we recommend the very high-contrast blue-white version.
In 24-hour operation, the illumination of the J-LW and B-LW types should be dimmed or switched off as often as possible to increase their lifetime.

SOFTWARE

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**. Flashing attributes can be assigned as often as you like. Text and graphics can be combined at any time. Up to 32 different character sets can be used. Each character set and the images can be zoomed from 2 to 8 times and rotated in 90° steps. With the largest character set, the words and numbers displayed will fill the screen.

ACCESSORIES

PROGRAMMER FOR INTERNAL DATA FLASH MEMORY

The display is shipped fully programmed and with all fonts. The additional programmer is thus generally not required.
However, if the internal character sets have to be changed or extended, or if images or macros have to be stored internally, the USB programmer EA 9778-1USB, which is available as an accessory, will burn the data/images you have created into the on-board data flash memory (80 kB) permanently.
The programmer runs under Windows and is connected to the PC's USB interface. It is shipped with an interface cable and the installation software.

SPEZIFICATION AND CHARACTERISTICS

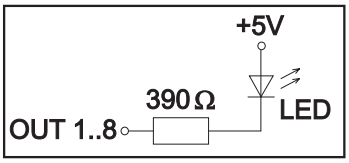
Characteristics					
Value	**Condition**	**min.**	**typ.**	**max.**	**Unit**
Operating Temperature		-20		+70	°C
Storage Temperature		-30		+80	°C
Storage Humidity	< 40°C			90	%RH
Operating Voltage		4.5	5.0	5.5	V
Input Low Voltage		-0.5		0.2*VDD	V
Input High Voltage	Pin Reset only	0.9*VDD		VDD+0.5	V
Input High Voltage	except Reset	0.6*VDD		VDD+0.5	V
Input Leakage Current	Pin MOSI only			1	uA
Input Pull-up Resistor		20		50	kOhms
Reset Pull-up Resistor		65		85	kOhms
Output Low Voltage				0.7	V
Output High Voltage		4.0			V
Output Current				20	mA
Power Supply	White Backlight 100%		230		mA
	Amber Backlight 100%		190		mA
	Backlight off		50		mA
	Powerdown (see page 18)	5	150		µA

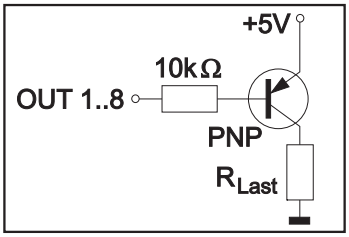
OUTPUTS

The EA eDIP320 offers up to 8 outputs, which can be used to control LEDs, for example. The configuration pins used depend on the interface selected (RS232, SPI or I²C). The configuration pins (open drain with internal pullup) are then evaluated as 1=HIGH level.

Each output can be controlled by means of the 'ESC YW n1 n2' command. Current can only flow when the level is at L (open drain with internal pullup). Each output can supply a maximum of 10 mA. It is thus possible to connect an LED to an output directly. Higher currents can be connected by using an external transistor.

Assignment output <-> pin no.						
	RS232/RS422		SPI		I2C	
output	**pin no.**	**symbol**	**pin no.**	**symbol**	**pin no.**	**symbol**
OUT1	6	BAUD0	10	DORD	6	BA0
OUT2	7	BAUD1	12	OUT2	7	BA1
OUT3	8	BAUD2	13	WUP	8	SA0
OUT4	9	ADR0	14	CPOL	9	SA1
OUT5	13	WUP	15	CPHA	10	SA2
OUT6	14	ADR1	17	DPROT	11	BA2
OUT7	15	ADR2			13	WUP
OUT8	17	DPROT			17	DPROT





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 5V (CMOS level) 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.

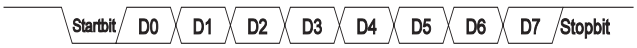
Pinout eDIP320-8: RS-232/RS-422 mode

Pin	Symbol	In/Out	Function		Pin	Symbol	Function
1	GND	-	Ground Potential for logic (0V)		25	N.C.	not connected
2	VDD	-	Power supply for logic (+5V)		26	N.C.	not connected
3	VADJ	In	Operating voltage for LC driving (input)		27	N.C.	not connected
4	VOUT	Out	Output voltage for LC driving		28	N.C.	not connected
5	RESET	-	L: Reset		29	N.C.	not connected
6	BAUD0	In	Baud Rate 0		30	N.C.	not connected
7	BAUD1	In	Baud Rate 1		31	N.C.	not connected
8	BAUD2	In	Baud Rate 2		32	N.C.	not connected
9	ADR0	In	Address 0 for RS-485		33	N.C.	not connected
10	RxD	In	Receive Data		34	N.C.	not connected
11	TxD	Out	Transmit Data		35	N.C.	not connected
12	EN485	Out	Transmit Enable for RS-485 driver		36	N.C.	not connected
13	WUP	In	L: (Power-On) disable Power-On-Macro L: Wakeup from Powerdownmode		37	N.C.	not connected
14	ADR1	In	Address 1 for RS-485		38	N.C.	not connected
15	ADR2	In	Address 2 for RS-485		39	N.C.	not connected
16	BUZZ	Out	Buzzer output		40	N.C.	not connected
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation		41	N.C.	not connected
18	DPWR	Out	L: Normal Operation H: Powerdownmode		42	N.C.	not connected
19	WP	In	L: Writeprotect for DataFlash		43	N.C.	not connected
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer		44	N.C.	not connected
21	PDI		internal use, do not connect		45	N.C.	not connected
22	PDO		internal use, do not connect		46	N.C.	not connected
23	N.C.		do not connect, reserved		47	N.C.	not connected
24	N.C.		do not connect, reserved		48	N.C.	not connected

Note:
The pins BAUD 0 to 2, ADR 0 to 2, WUP, 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.
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.

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 8 and 9).



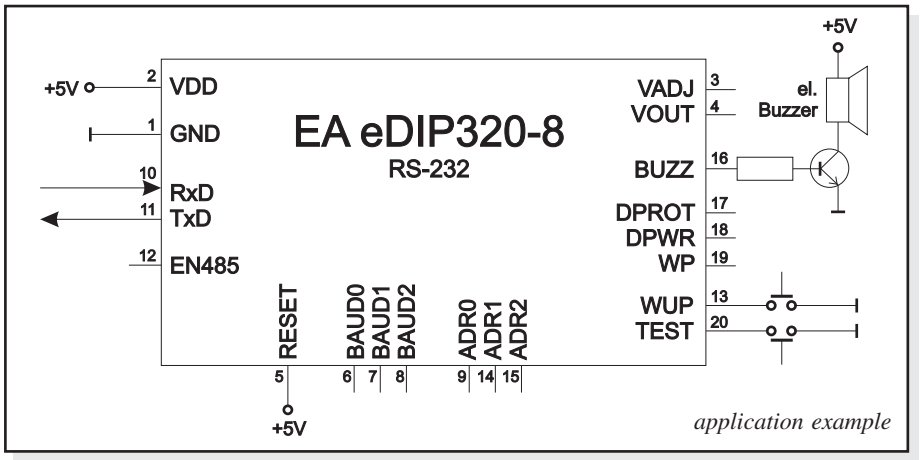
Baud Rates

Baud0	Baud1	Baud2	data format 8,N,1
0	0	0	1200
1	0	0	2400
0	1	0	4800
1	1	0	9600
0	0	1	19200
1	0	1	38400
0	1	1	57600
1	1	1	115200

RS-485 INTERFACE

With an external converter (e.g. SN75176), the EA eDIP320 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 eDIP320 displays can be operated on a single RS-485 bus by setting addresses.
We recommend the EA 9778-1RS485 board for development.
Addressing:

- Up to eight hardware addresses (0 to 7) can be set by means of pins ADR0 to ADR2.
- The eDIP with the address 7 is selected and ready to receive after power-on.
- The eDIPs with the addresses 0 to 6 are deselected 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 the eDIP externally to the address 0).



application example

SPI INTERFACE

If the display is wired as shown below, SP mode is activated. The data is then transferred via the serial, synchronous SPI interface.

Data transfer is possible at up to 100 kHz. However, if pauses of at least 100 µs are maintained between the individual bytes during transfer, a byte can be transferred at up to 3 MHz.

Note:
The pins DORD, CPOL, CPHA, WUP, 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.

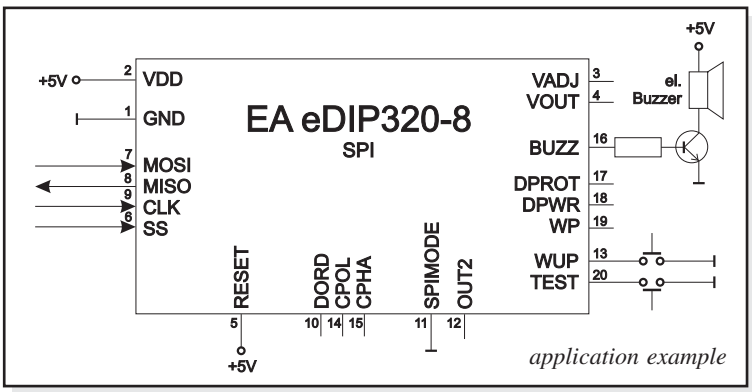
Pinout eDIP320-8: SPI mode

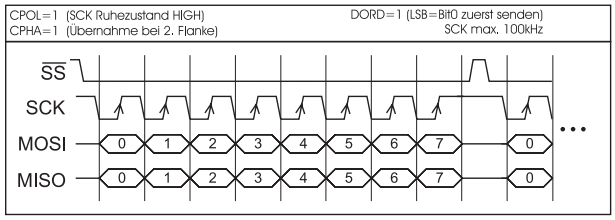
Pin	Symbol	In/Out	Function	Pin	Symbol	Function
1	GND	-	Ground Potential for logic (0V)	25	N.C.	not connected
2	VDD	-	Power supply for logic (+5V)	26	N.C.	not connected
3	VADJ	In	Operating voltage for LC driving (input)	27	N.C.	not connected
4	VOUT	Out	Output voltage for LC driving	28	N.C.	not connected
5	RESET	-	L: Reset	29	N.C.	not connected
6	SS	In	Slave Select	30	N.C.	not connected
7	MOSI	In	Serial In	31	N.C.	not connected
8	MISO	Out	Serial Out	32	N.C.	not connected
9	CLK	In	Shift Clock	33	N.C.	not connected
10	DORD	In	Data Order (0=MSB first; 1=LSB first)	34	N.C.	not connected
11	SPIMODE	In	connect to GND for SPI interface	35	N.C.	not connected
12	OUT2	Out	open-drain with internal pullup 20..50k	36	N.C.	not connected
13	WUP	In	L: (Power-On) disable Power-On-Macro L: Wakeup from Powerdownmode	37	N.C.	not connected
14	CPOL	In	Clock Polarity (0=LO 1=HI when idle)	38	N.C.	not connected
15	CPHA	In	Clock Phase (sampled on 0=1st 1=2nd edge)	39	N.C.	not connected
16	BUZZ	Out	Buzzer output	40	N.C.	not connected
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation	41	N.C.	not connected
18	DPWR	Out	L: Normal Operation H: Powerdownmode	42	N.C.	not connected
19	WP	In	L: Writeprotect for DataFlash	43	N.C.	not connected
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	44	N.C.	not connected
21	PDI		internal use, do not connect	45	N.C.	not connected
22	PDO		internal use, do not connect	46	N.C.	not connected
23	N.C.		do not connect, reserved	47	N.C.	not connected
24	N.C.		do not connect, reserved	48	N.C.	not connected

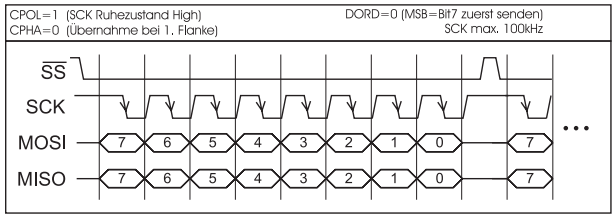
DATA TRANSFER SPI

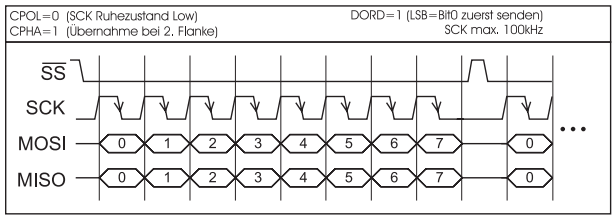
Via the pins DORD, CPOL and CPHA transfer parameter will be set.
Write operation: a clock rate up to 100 kHz is allowed without any stop. Together with a pause of 100 µs between every data byte a clock rate up to 3 MHz an be reached.

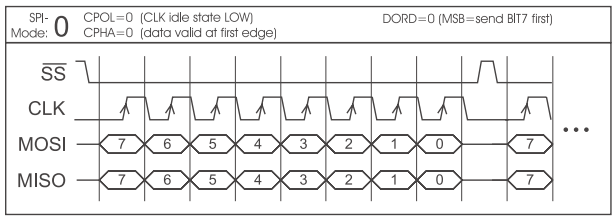
Read operation: to read data (e.g. the "ACK" byte) a dummy byte (e.g . 0xFF) need to be sent. Note that the EA eDIP320-8 for internal operation does need a short time before providing the data; therefore a short pause of min. 6µs (no activity of CLK line) is needed for each byte. Same is with 100kHz operation.











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.

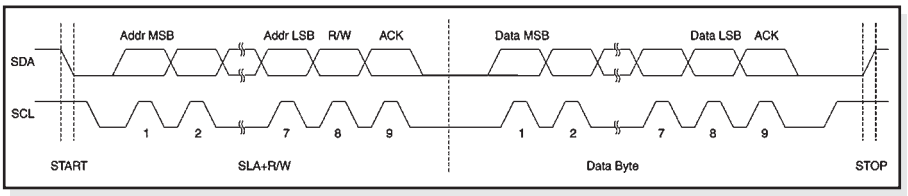
Data transfer is possible at up to 100 kHz. However, if pauses of at least 100 µs are maintained between the individual bytes during transfer, a byte can be transferred at up to 400 kHz.

Note:
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.

Pinout eDIP320-8: I2C mode

Pin	Symbol	In/Out	Function	Pin	Symbol	Function
1	GND	-	Ground Potential for logic (0V)	25	N.C.	not connected
2	VDD	-	Power supply for logic (+5V)	26	N.C.	not connected
3	VADJ	In	Operating voltage for LC driving (input)	27	N.C.	not connected
4	VOUT	Out	Output voltage for LC driving	28	N.C.	not connected
5	RESET	-	L: Reset	29	N.C.	not connected
6	BA0	In	Basic Address 0	30	N.C.	not connected
7	BA1	In	Basic Address 1	31	N.C.	not connected
8	SA0	In	Slave Address 0	32	N.C.	not connected
9	SA1	In	Slave Address 1	33	N.C.	not connected
10	SA2	In	Slave Address 2	34	N.C.	not connected
11	BA2	In	Basic Address 2	35	N.C.	not connected
12	I2CMODE	In	connect to GND for I²C interface	36	N.C.	not connected
13	WUP	In	L: (Power-On) disable Power-On-Macro L: Wakeup from Powerdownmode	37	N.C.	not connected
14	SDA	Bidir.	Serial Data Line	38	N.C.	not connected
15	SCL	In	Serial Clock Line	39	N.C.	not connected
16	BUZZ	Out	Buzzer output	40	N.C.	not connected
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation	41	N.C.	not connected
18	DPWR	Out	L: Normal Operation H: Powerdownmode	42	N.C.	not connected
19	WP	In	L: Writeprotect for DataFlash	43	N.C.	not connected
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	44	N.C.	not connected
21	PDI		internal use, do not connect	45	N.C.	not connected
22	PDO		internal use, do not connect	46	N.C.	not connected
23	N.C.		do not connect, reserved	47	N.C.	not connected
24	N.C.		do not connect, reserved	48	N.C.	not connected

Note:
The pins BA0 to 2, SA0 to 2, DPOM, DPROT and TEST/SBUF have an internal pullup, which is why only the LO level (L=0=GND) is to be actively applied. These pins must be left open for a Hi level (H=1).
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 11,7,6 BA2	BA1	BA0	Base address	D7	D6	D5	D4	D3	D2	D1	D0
L	L	L	$10	0	0	0	1	SA2	SA1	SA0	RW
L	L	H	$20	0	0	1	0				
L	H	L	$30	0	0	1	1				
L	H	H	$40	0	1	0	0				
H	L	L	$70	0	1	1	1				
H	L	H	$90	1	0	0	1				
H	H	L	$B0	1	0	1	1				
H	H	H	$D0	1	1	0	1				

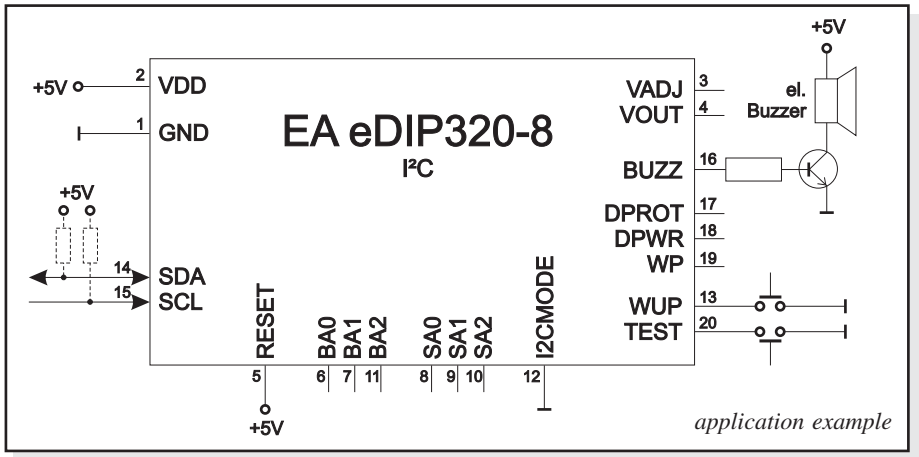
all pins open: Write $DE
Read $DF

DATA TRANSFER I²C-BUS

principle I2C-bus transfer:

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

Read operation: for internal operation the EA eDIP does need a short time before providing the data; therefore a short pause of min. 6µs is needed for each byte (no activity of SCL line).



application example

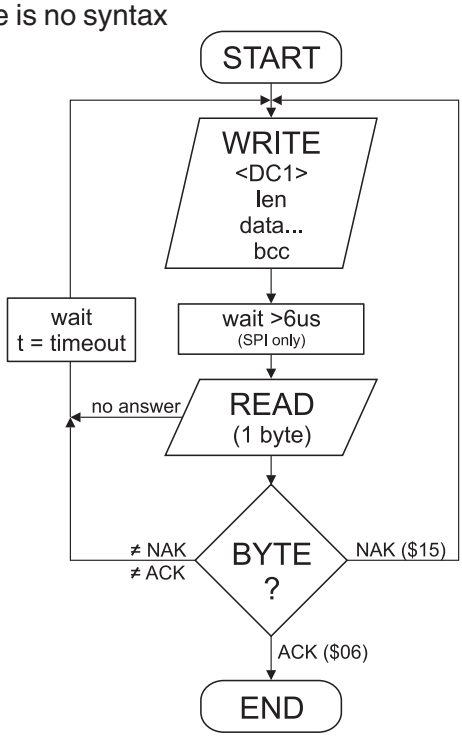
DATA TRANSFER PROTOCOL (SMALL PROTOCOL)

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 eDIP320-8 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 elapse before the package is sent again.

The raw data volume per package is limited to 128 bytes (len <= 128). Commands longer than 128 bytes (e.g. Load image ESC UL...) must be divided up between a number of packages. All data in the packages are compiled again after being correctly received by the EA eDIP320-8.



DEACTIVATING THE 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.

BUILDING THE SMALLPROTOCOL PACKAGES

Command/Data to the display

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

<DC1> = 17(dec.) = $11 <ACK> = 6(dec.) = $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 319,239

<DC1>	len	ESC	D	L	ESC	G	D	0	0	319	239	bcc	>		
$11	$0E	$1B	$44	$4C	$1B	$47	$44	$00	$00	$00	$00	$3F	$01	$EF	$00	$9F	
													<	<ACK>	
														$06	

Example to a complete data package

```
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“ source code to transmit a data package

Request for content of send buffer

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

<DC2> = 18(dec.) = $12 1 = 1(dez.) = $01 S = 83(dez.) = $53
<ACK> = 6(dec.) = $06
len = count of user data (without <DC2>, without checksum bcc)
bcc = 1 byte = sum of all bytes incl. <DC2>, modulo 256

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

Request for buffer information

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

<DC2> = 18(dec.) = $12 1 = 1(dez.) = $01 I = 73(dez.) = $49
<ACK> = 6(dec.) = $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

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

Protocol settings

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

<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

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.

Request for protocol settings

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

<DC2> = 18(dec.) = $12 1 = 1(dez.) = $01 P = 80(dez.) = $50
<ACK> = 6(dec.) = $06
max. packet size = count of maximum user data for 1 package (eDIP320-8 = 128)
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

This command is used to query protocol settings.

Repeat the last package

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

<DC2> = 18(dec.) = $12 1 = 1(dez.) = $01 R = 82(dez.) = $52
<ACK> = 6(dec.) = $06
<DC1> = 17(dec.) = $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

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>).

Adressing (only for RS232/RS485)

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

<DC2> = 18(dec.) = $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

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

TOUCH PANEL (EA EDIP320X-8LWTP VERSIONS)

The -xxxTP versions are shipped with an analog, resistive touch panel. Up to 80 touch areas (keys, switches, menus, bar graph inputs) can be defined simultaneously. The fields can be defined with pixel accuracy. The display supports user-friendly commands (see page 16). When the touch "keys" are touched, they can be automatically inverted and an external tone can sound (pin 16), indicating they have been touched. The predefined return code of the "key" is transmitted via the interface, or an internal touch macro with the number of the return code is started instead (see page 19, *Macro programming*).

TOUCH PANEL ADJUSTMENT

The touch panel is perfectly adjusted and immediately ready for operation on delivery. As a result of aging and wear, it may become necessary to readjust the touch panel.

Adjustment procedure:

1. Touch the touch panel at power-on and keep it depressed. After the message *"touch adjustment ?"* appears, release the touch panel again (or issue the 'ESC @' command).
2. Touch the touch panel again within a second for at least a second.
3. Follow the instructions for adjustment (press the 2 points *upper left* and *lower right*).

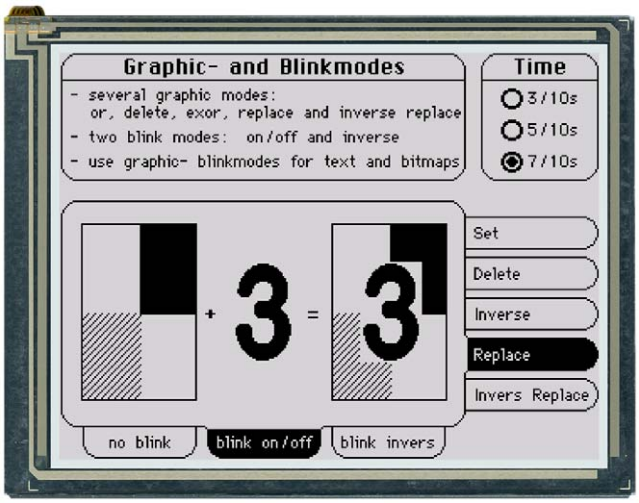
FRAMES AND KEY FORMS

A frame type can be set by using the *Draw frame* or *Draw frame box* command or by drawing touch keys. 18 frame types are available (0 = do not draw a frame). The frame size must be at least 16x16 pixels.

BITMAPS AS KEYS

Apart from the frame types, which are infinitely scalable, it is also possible to use bitmaps (2 each for *not printed* and *printed*) as touch keys or touch switches.

You can use ELECTRONIC ASSEMBLY LCD-Tools*) to integrate your own buttons as images ("PICTURE" compiler statement). A button always consists of two monochrome Windows BMPs of equal size (one bitmap to display the touch key in its normal state and one for when it is pressed). The active area of the touch key automatically results from the size of the button bitmaps.



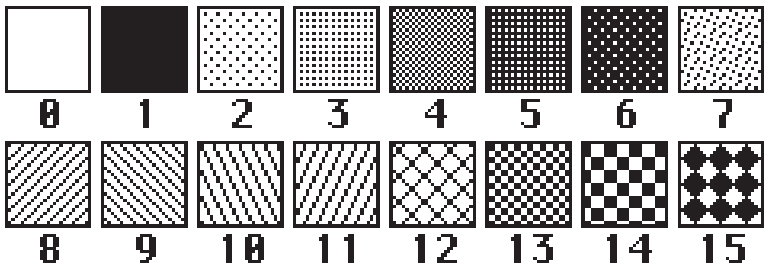
SWITCHES IN GROUPS (RADIO GROUPS)

Touch switches (radio buttons) change their status from *ON* to *OFF* or vice versa each time they are touched. Several touch switches can be included in a group ('ESC A R nr' command). If a touch switch in the group 'nr' is switched on, all the other touch switches in this group are automatically switched off. Only one switch is ever on.

*) *full version is free available on web at* http://www.lcd-module.com/products/touch.html

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.



TERMINAL MODE

When you switch the unit on, the cursor flashes in the first line, indicating that the display is ready for operation. 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 8 and 9) 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).

+ 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	¯

Terminal-Font (Font 0): 8x8 monospaced

EA eDIP320-8: 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)	**^M**							Cursor to the beginning of the line on the extreme left	
line feed lf (dec:10)	**^J**							Cursor 1 line lower, if cursor in last line then scroll	
Position cursor	**ESC**	**T**	**P**	C	L			C=column; L=line; origin upper-left corner (1,1)	1,1
Cursor on/off			**C**	n1				n1=0: Cursor is invisible; n1=1: Cursor flashes;	1
save cursor position			**S**					the current cursor position is saved	
restore cursor position			**R**					the last saved cursor position is restored	
Terminal off			**A**					Terminal display is switched off; outputs are rejected	
Terminal on			**E**					Terminal display is switched on;	On
output version			**V**					the version no. is output in the terminal (e.g. "EA eDIP320-8 V1.0 Rev.A")	
Define window	**ESC**	**T**	**W**	C	L	B	H	w	The terminal output is executed only within the window from column C and line Z (=upper-left corner) with a width of b and a height of h (specifications in characters); w=angle (0=0°; 1=90°; 2=180°; 3=270°) of the terminal display	1,1 40,30 0

INTEGRATED AND EXTERNAL FONTS

As standard, there are 3 monospaced character sets, 3 proportional character sets and 1 large digit font integrated in addition to the 8x8 terminal font (font no. 0). The proportional character sets (which have a narrow “I” and a wide “W”, for example) look better and take up less space on the screen. Each character can be placed with **pixel accuracy**, and its height and width can be increased by a factor of 1 to 8.

A text can be output left justified, right justified or centered. Rotation in 90° steps is possible (for vertical installation of the display, for example).

Macro programming permits further fonts to be integrated (up to 31). All kinds of fonts can be created using a text editor and loaded using the eDIP320 compiler*) (the USB programmer EA 9778-1USB is required).

+ 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 2: 6x8 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	:	;	<	=	>	?
$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	{	|	}	~	⌂
$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

+ 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: large Digits BigZif57

FONT APPEARANCE

This hard copy shows all the fonts with which the product is shipped.
Macro programming permits them to be modified or further fonts to be integrated. All kinds of fonts (including Cyrillic and Chinese) can be created using a text editor and programmed using the kit compiler/ LCD toolkit*) (the EA 9778-1USB programmer is required).

*) *full version is free available on web at http://www.lcd-module.com/products/touch.html*

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 17) or together with the self-definable macro (see pages 19/20).

EA eDIP320-8: Command table 1																	After reset
Command	**Codes**															**Remarks**	
Commands for outputting strings																	
Output string L: left justified C: centered R: right justified	ESC	Z	L C R	xx1	yy1	text ...	NUL									A string (...) is output to xx1,yy1; end of string: 'NUL' ($00), 'LF' ($0A) or 'CR' ($0D); several lines are separated by the character '|' ($7C); text between two '~' ($7E) characters flashes on/off; text between two '@' ($40) characters flashes inversely;	
Set font	ESC	Z	F	n1												Set font with the number n1 (0 to 31)	0
Font zoom factor	ESC	Z	Z	n1	n2											n1 = X zoom factor (1x to 8x); n2 = Y zoom factor (1x to 8x)	1,1
Add. line spacing	ESC	Z	Y	n1												Insert n1 pixels (0 to 15) between two lines as additional line spacing	
Text angle	ESC	Z	W	n1												Text output angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
Text link mode	ESC	Z	V	n1												Mode n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace;	4
Text pattern	ESC	Z	M	n1												link Text with pattern number n1 (0 to 15);	1
Text flashing attribute	ESC	Z	B	n1												n1: 0=no flashing; 1=Text flashes on/off; 2=Text flashes inversely	0
String for terminal	ESC	Z	T				text ...									Command for outputting a string from a macro to the terminal	
Draw straight lines and points																	
Draw rectangle	ESC	G	R	xx1	yy1	xx2	yy2									Draw four straight lines as a rectangle from xx1,yy1 to xx2,yy2	
Draw straight line	ESC	G	D	xx1	yy1	xx2	yy2									Draw straight line from xx1,yy1 to xx2,yy2	
Continue straight line	ESC	G	W	xx1	yy1											Draw a straight line from last end point to xx1, yy1	0
Draw point	ESC	G	P	xx1	yy1											Set a point at coordinates xx1, yy1	
Point size/line thickness	ESC	G	Z	n1	n2											n1 = X point size (1 to 15); n2 = Y point size (1 to 15);	1,1
Link mode	ESC	G	V	n1												Set drawing mode n1: 1=set; 2=delete; 3=inverse;	1
Pattern	ESC	G	M	n1												set straight line/point pattern number n1 (0 to 15)	1
Change/draw rectangular areas																	
Delete area	ESC	R	L	xx1	yy1	xx2	yy2									Delete area from xx1,yy1 to xx2,yy2 (all pixels off)	
Invert area	ESC	R	I	xx1	yy1	xx2	yy2									Invert area from xx1,yy1 to xx2,yy2 (invert all pixels)	
Fill area	ESC	R	S	xx1	yy1	xx2	yy2									Fill area from xx1,yy1 to xx2,yy2 (all pixels on)	
Area with fill pattern	ESC	R	M	xx1	yy1	xx2	yy2	n1								Draw area from xx1,yy1 to xx2,yy2 with pattern n1 (always set)	
Draw box	ESC	R	O	xx1	yy1	xx2	yy2	n1								Draw rectangle from xx1,yy1 to xx2,yy2 with pattern n1 (always replace)	
Draw frame	ESC	R	R	xx1	yy1	xx2	yy2	n1								Draw frame of type n1 from xx1,yy1 to xx2,yy2 (always set)	
Draw frame box	ESC	R	T	xx1	yy1	xx2	yy2	n1								Draw frame box of type n1 from xx1,yy1 to xx2,yy2 (always replace)	
Bitmap image commands																	
Image from clipboard	ESC	U	C	xx1	yy1											The current contents of the clipboard are loaded to xx1,yy1 with all the image attributes	
Load internal image	ESC	U	I	xx1	yy1	no										Load internal image with the no (0 to 255) from the data flash memory to xx1,yy1	
Load image	ESC	U	L	xx1	yy1	Bh7 data ...										Load an image to xx1,yy1; see image structure (BH7 format) for image data	
Image zoom factor	ESC	U	Z	n1	n2											n1 = X zoom factor (1x to 8x); n2 = Y zoom factor (1x to 8x)	1,1
Image angle	ESC	U	W	n1												output angle of the image: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
Mirror Image	ESC	U	X	n1												n1: 0=normal display; 1=the image is mirrored horizontally	0
Image link mode	ESC	U	V	n1												Mode n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace;	4
Image pattern	ESC	U	M	n1												link Text with pattern number n1 (0 to 15)	1
Image flashing attribute	ESC	U	B	n1												n1: 0=no flashing; 1=image flashes on/off; 2=image flashes inversely: 3=flashes with flash image	0
Send hard copy	ESC	U	H	xx1	yy1	xx2	yy2									After this command, the image extract is sent in BH7 Format.	
Display commands (effect on the entire display)																	
Delete display	ESC	D	L													Delete display contents (all pixels off)	
Invert display	ESC	D	I													Invert display contents (invert all pixels)	
Fill display	ESC	D	S													Fill display contents (all pixels on)	
Switch display off	ESC	D	A													Display contents become invisible but are retained, commands are still possible	
Switch display on	ESC	D	E													Display contents become visible again	On
Flashing area commands																	
Delete flashing attribute	ESC	Q	L	xx1	yy1	xx2	yy2									Delete the flashing attribute from xx1,yy1 to xx2,yy2	
Flash inversely	ESC	Q	I	xx1	yy1	xx2	yy2									Defines an inverted flashing area from xx1,yy1 to xx2,yy2	
Flashing area pattern	ESC	Q	M	xx1	yy1	xx2	yy2	n1								Defines a flashing area with pattern n1 (on/off) from xx1,yy1 to xx2,yy2	
Set flashing time	ESC	Q	Z	n1												Set the flashing time n1= 1 to 15 in 1/10s; 0=deactivate flashing function	6
Bar graph commands																	
Define bar graph	ESC	B	R L O U	no	xx1	yy1	xx2	yy2	SV	EV	Typ	pat				Define bar graph to L(eft), R(ight), O(ben) (up), U(nten) (down) with number no. xx1,yy1,xx2,yy2 form the rectangle enclosing the bar graph. sv, ev are the values for 0% and 100%. Type: 0=bar; 1=bar in rectangle; pat=bar pattern type: 2=line; 3=line in rectangle; pat=line width	No bar define
Update bar graph	ESC	B	A	n1	valu											Set and draw the bar graph with the number n1 to the new user 'value'.	
Draw new bar graph	ESC	B	Z	n1												entirely reDraw the bar graph with the number n1	
Send bar graph value	ESC	B	S	n1												Send the current value of bar graph number n1	
Delete bar graph	ESC	B	D	n1	n2											the definition of the bar graph with the number n1 becomes invalid. If the bar graph was defined as input with touch, this touch field will also be deleted. n2=0: Bar graph remains visible; n2=1: Bar graph is deleted	

EA eDIP320-8: Command table 2												After reset
Command	**Codes**										**Remarks**	
Clipboard commands (buffer for image areas)												
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	xx1	yy1	xx2	yy2				The image area from xx1,yy1 to xx2,yy2 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	xx1	yy1						The image area on the clipboard is copied to xx1,yy1 in the display	
settings for menu box/touch menu												
Set menu font	ESC	N	F	n1							Set font with the number n1 (0 to 31) for menu display	0
Menu font zoom factor	ESC	N	Z	n1	n2						n1 = X zoom factor (1x to 8x); n2 = Y zoom factor (1x to 8x)	1,1
Add. line spacing	ESC	N	Y	n1							Insert n1 pixels (0 to 15) between two menu items as additional line spacing	
Menu angle	ESC	N	W	n1							Menu display angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
Touch menu automation	ESC	N	T	n1							n1=1: Touch menu opens automatically; n1=0:Touch menu does not open automatically; instead, the request 'ESC T 0' to open is sent to the host computer, which can then open the touch menu with 'ESC N T 2'.	1
Menu box commands (control with keys rather than by touch)												
Define and display menu	ESC	N	D	xx1	yy1	no	text ...	NUL			A menu is drawn as of the corner xx1,yy1 with the current menu font. no= currently inverted entry (e.g.: 1 = 1. entry) Text:= string with menu items. The different items are separated by the character '|' ($7C,dec:124) (e.g. "item1|item2|item3"). The background of the menu is saved automatically. If a menu is already defined, it is automatically canceled+deleted.	
Next item	ESC	N	N								The next item is inverted or remains at the end	
Previous item	ESC	N	P								The previous item is inverted or remains at the beginning	
End of menu/send	ESC	N	S								The menu is removed and replaced with the original background. The current item is sent as a number (1 to n) (0=no menu displayed)	
End of menu/macro	ESC	N	M	n1							The menu is removed and replaced with the original background. Menu macro n1 is called for item 1, menu macro nr+1 for item 2, and so on	
End of menu/cancel	ESC	N	A								The menu is removed and replaced with the original background	
Macro commands												
Run normal macro	ESC	M	N	n1							Call the (normal) macro with the number n1 (0 to 255) (max. 7 levels)	
Run touch macro	ESC	M	T	n1							Call the touch macro with the number n1 (0 to 255) (max. 7 levels)	
Run menu macro	ESC	M	M	n1							Call the menu macro with the number n1 (0 to 255) (max. 7 levels)	
Disable macros	ESC	M	L	type	n1	n2					Macros of the type 'N', 'T' or 'M' (type 'A' = all macro types) are disabled from the number n1 to n2; i.e. no longer run when called.	
Enable macros	ESC	M	U	type	n1	n2					Macros of the type 'N', 'T' or 'M' (type 'A' = all macro types) are enabled from number n1 to n2; i.e. run again when called.	
Select macro/image page	ESC	M	K	n1							A page is selected for macros and images n1=0 to 15. if a macro/image is not defined in the current page 1 to 15, this macro/image is taken from page 0 (e.g. to switch languages or for horizontal/vertical installation).	
Save macro/image page	ESC	M	W								the current macro/image page is saved (when used in process macros)	
Restore macro/image page	ESC	M	R								the last saved macro/image page is restored	
automatic (Normal) Macros												
Macro with delay	ESC	M	G	n1	n2						Call the (normal) macro with the number n1 (0 to 255) in n2/10S. Execution is stopped by commands (e.g. receipt or touch macros).	
Autom. macros once only	ESC	M	E	n1	n2	n3					Automatically run macros n1 to n2 once only; n3=pause in 1/10s. Execution is stopped by commands (e.g. receipt or touch macros).	
Autom. macros cyclical	ESC	M	A	n1	n2	n3					Automatically run macros n1 to n2 cyclically; n3=pause in 1/10s. Execution is stopped by commands (e.g. receipt or touch macros).	
Autom. macros ping pong	ESC	M	J	n1	n2	n3					Automatically run macros n1 to n2 to n1 (ping pong); n3=pause in 1/10s. Execution is stopped, for example, by receipt or touch macros.	
macro processes												
Define macro process	ESC	M	D	no	type	n3	n4	zs			A macro process with the number no (1 to 8) is defined (1=highest priority). The (Normal) Macros n3 to n4 are run successively every zs/10s. type: 1=once only; 2=cyclical; 3=ping pong n3 to n4 to n3	
Macro process interval	ESC	M	Z	nr	zs						a new time zs is assigned to the macro process with the number nr (1 to 8) in 1/10s. if the time zs=0, execution is stopped.	
Stop macro processes	ESC	M	S	n1							all macro processes are stopped with n1=0 and restarted with n1=1 in order, for example, to execute settings and outputs via the interface undisturbed	1
Other commands												
Wait (pause)	ESC	X	n1								Wait n1 tenths of a second before the next command is executed.	
Set RS485 address	ESC	K	A	adr							for RS232/RS485 operation only and only possible when Hardware address is 0. The eDIP is assigned a new address adr (in the Power-On macro).	
Tone on/off	ESC	Y	S	n1							The tone output (pin 16) becomes n1=0: OFF; n1=1: ON; n1=2 to 255: switched on for n1 tenths of a second	OFF
Illumination on/off	ESC	Y	L	n1							LED illumination n1=0: OFF; n1=1: ON; n1=2 to 255: illumination switched on for n1 tenths of a second.	1
Illumination brightness	ESC	Y	H	n1							set brightness of the LED illumination n1=0 to 100%. n1=250 save current brightness as starting brightness; n1=254 switch LED off immediately; n1=255 switch to 100% immediately.	100
Write output port	ESC	Y	W	n1	n2						n1=0: Set all output ports in accordance with n2 (=6/8-bit binary value). n1=1 to 6/8: Reset port n1 (n2=0); set (n2=1); invert (n2=2);	to 1
Send bytes	ESC	S	B	num	data ...						num (=1 to 255) bytes are sent to the send buffer... = num Bytes. in the source text of the macro programming, the number nUM must not be specified. This is counted by the edip compiler and entered.	
Send version	ESC	S	V								the version is sent as a string (e.g. "EA eDIP320-8 V1.0 Rev.A tp+")	
Send internal information	ESC	S	I								internal information is sent by the edip.	
Power down	ESC	P	D	n1							After this command, the display goes into power-down mode. n1=0: wake up only after reset; n1=1: wake up on L level at WUP Pin n1=2: wake up on touch; n1=3: wake up on WUP Pin or Touch	

EA eDIP320-8: Commands for the touch panel													After reset
Command	**Codes**											**Remarks**	
Touch: Define areas													
Define touch key (key remains depressed as long as there is contact)	ESC	A	T xx1	yy1	xx2	yy2	dow code	up code	text ...	NUL		'T': The area from xx1,yy1 to xx2,yy2 is defined as a key. 'U': Image no. n1 is loaded to xx1,yy2 and defined as a key. 'down code':(1-255) Return/touch macro when key pressed. 'up code': (1-255) Return/touch macro when key released. (down/up code = 0 press/release not reported). ´text´: the first character determines the alignment of the text (C=centered, L=left justified, R=right justified). this is followed by a string that is placed in the key with the current touch font. multiline texts are separated with the character '|' ($7C, dec: 124); 'nul': ($00) = end of string	
			U xx1	yy1	n1	dow code	up code	text ...	NUL				
Define touch switch (status of the switch toggles after each contact)	ESC	A	K xx1	yy1	xx2	yy2	dow code	up code	text ...	NUL		'K': The area from xx1,yy1 to xx2,yy2 is defined as a switch. 'J': Image no. n1 is loaded to xx1,yy2 and defined as a switch. 'down code': (1-255) Return/touch macro when switched on. 'up code': (1-255) Return/touch macro when switched off. (down/up code = 0 on/off not reported). ´text´: the first character determines the alignment of the text (C=centered, L=left justified, R=right justified). this is followed by a string that is placed in the key with the current touch font. multiline texts are separated with the character '|' ($7C, dec: 124); 'nul': ($00) = end of string	
			J xx1	yy1	n1	dow code	up code	text ...	NUL				
Define touch key with menu function	ESC	A	M xx1	yy1	xx2	yy2	dow code	up code	mnu code	text ...	NUL	The area from xx1,yy1 to xx2,yy2 is defined as a menu key. 'down code':(1-255) Return/touch macro when pressed. 'up Code':(1-255) Return/touch macro when menu canceled 'mnu Code':(1-255) Return/menu macro+(item no. 1) after selection of a menu item. (down/up code = 0: activation/cancellation is not reported.) 'text':= string with the key text and the menu items. the first character determines the direction in which the menu opens (R=right, L=left, O=up, U=down). The second character determines the alignment of the touch key text (C=centered, L=left justified, R=right justified). The menu items are separated by the character '|' ($7C,dec:124) (e.g. "uckey|item1|item2|item3". The key text is written with the current touch font and the menu items are written with the current menu font. The background of the menu is saved automatically.	
Define drawing area	ESC	A	D xx1	yy1	xx2	yy2	n1	A drawing area is defined. You can then draw with a line width of n1 within the corner coordinates xx1,yy1 and xx2,yy2.					
Define free touch area	ESC	A	H xx1	yy1	xx2	yy2		A freely usable touch area is defined. Touch actions (down, up and drag) within the corner coordinates xx1,yy1 and xx2,yy2 are sent.					
Set bar by touch	ESC	A	B n1					The bar graph with the no. n1 is defined for input by touch panel.					
Touch: settings													
Touch frame form	ESC	A	E n1					The frame type for the display of touch keys/switches is set with n1					1
Touch key response			I n1					Automatic inversion when touch key touched: n1=0=OFF; n1=1=ON;					1
			S n1					Tone sounds briefly when a touch key is touched: n1=0=OFF; n1=1=ON					1
Invert touch key			N Cod					The touch key with the assigned return code is inverted manually					
Query touch switch			X Cod					The status of the switch (off=0; on=1) is placed in the send buffer.					
Set touch switch			P Cod	n1				The status of the switch is changed by means of a command (n1=0=off; n1=1=on).					
Radio group for switches			R n1					Only 1 switch in a group is active at any one time; all the others are deactivated. nr=0: newly defined switches do not belong to a group. nr=1 to 255: newly defined switches belong to the group with the number nr. In the case of a switch in a group, only the down code is applicable. the up code is ignored.					0
Query radio group			G n1					the down code of the activated switch from the radio group with the number n1 is placed in the send buffer.					
			L Cod	n1				The touch area with the return code (code=0: all touch areas) is removed from the touch query When n1=0, the area remains visible on the display; when n1=1, the area is deleted.					
Delete touch area			V xx1	yy1	n1			remove the Touch area that includes the coordinates xx1,yy1 from the touch query. n1=0: area remains visible; n1=1: Delete area					
Send bar value automatically			Q n1					The Automatic transmission of a new bar graph value by touch input is deactivated (n1=0); a new value is sent after setting (n1=1); each change is sent during setting (N1=2).					1
Touch query on/off			A n1					Touch query is deactivated (n1=0) or activated (n1=1);					
Rotate touch query			O n1					n1=0: normal query; n1=1: Touch query for top view (solder straps changed over)					1
Touch: Label font													
Label font	ESC	A	F n1					Set font with the number n1 (0 to 31) for touch key label					0
Label zoom factor			Z n1	n2				n1 = X zoom factor (1x to 8x); n2 = Y zoom factor (1x to 8x)					1,1
Add. line spacing			Y n1					Insert n1 pixels (0 to 15) between two lines of text as additional line spacing					
Label angle			W n1					Text output angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°					0

Responses of the EA eDIP320-8								
Id		num	data					Remarks
automatic responses								
ESC	A	1	code					Response from the analog touch panel when a key/switch is pressed. code = down or up code of the key/switch. it is Only transmitted if no touch macro is defined with the no. co
ESC	N	1	code					After a menu item is selected by touch, the selected menu item code is transmitted. it is Only transmitted if no touch macro is defined with the number code.
ESC	B	2	no	value				When a bar graph is set by touch, the current value of the bar is transmitted with the nur Transmission of the bar balue must be activated (see the 'ESC A Q n1' command).
ESC	T	0						if automatic opening of a touch menu is disabled (see the 'ESC N T n1' command), this request is sent to the host computer. The host can then open the touch menu with the 'E T 2' command.
ESC	H	5	type	xLO	xHI	yLO	yHI	The following is transmitted in the case of a free touch area event: type=0 is release; typ is touch; type=2 is drag within the free touch area at the coordinates XX1, YY1
Response only when requested by command								
ESC	N	1	no					After the 'ESC N S' command, the currently selected menu item is transmitted. no=0: no menu item is selected.
ESC	B	2	no	value				After the 'ESC B S n1' command, the current value of the bar is transmitted with the num nr.
ESC	X	2	code	value				After the 'ESC A X' command, the current status of the touch switch is transmitted with c (the return code). value = 0 or 1
ESC	G	2	no	code				After the 'ESC A G nR' command, the code of the active touch switch in the radio group sent.
ESC	V	num	String...					After the 'ESC S V' command, the version of the edip firmware is transmitted as a string (e.g. "ea edip320-8 v1.0 rev.a tp+")
ESC	I	num	X Pixel, Y Pixel, Version, Touch info, CRC-ROM, CRC-ROM target DF in KB, CRC-DF, CRC-DF target, DF num					num = 21 after the 'ESC S I' command, internal information is sent by eDIP (16-Bit integ values LO-HI Byte) Version: LO-Byte = version number Software; HI-Byte = Hardware revison letter touch info: LO-Byte = '-|+' X direction detected; HI-Byte = '-|+' Y direction detected DF num: number of user bytes in data flash memory (3 Bytes: LO-, MID- HI-By
Responses without length specification (num)								
ESC	U	L	xx1	yy1	image data... (BH7 FORMAT)			after the 'ESC UH....' command, a hard copy is sent in BH7 Format. xx1,yy1 = Start coordinates of the hard copy (upper left corner)

USING THE 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 come 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.: **#GD0,0,319,239,**
- Strings (text) are written directly without quotation marks and concluded with CR (hex: $0D) or LF (hex: $0A).

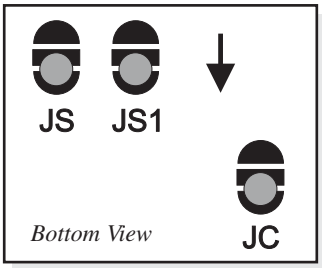
2. Binary mode

- The escape character corresponds to the character ESC (hex: $1B, dec: 27).
- The command letters are transmitted directly.
- The coordinates xx and yy are transmitted as 16-bit binary values (first the LOW byte and then the HIGH byte).
- 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).

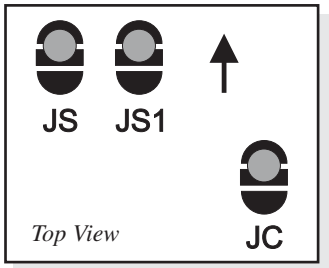
TOP VIEW AFTER 180° ROTATION

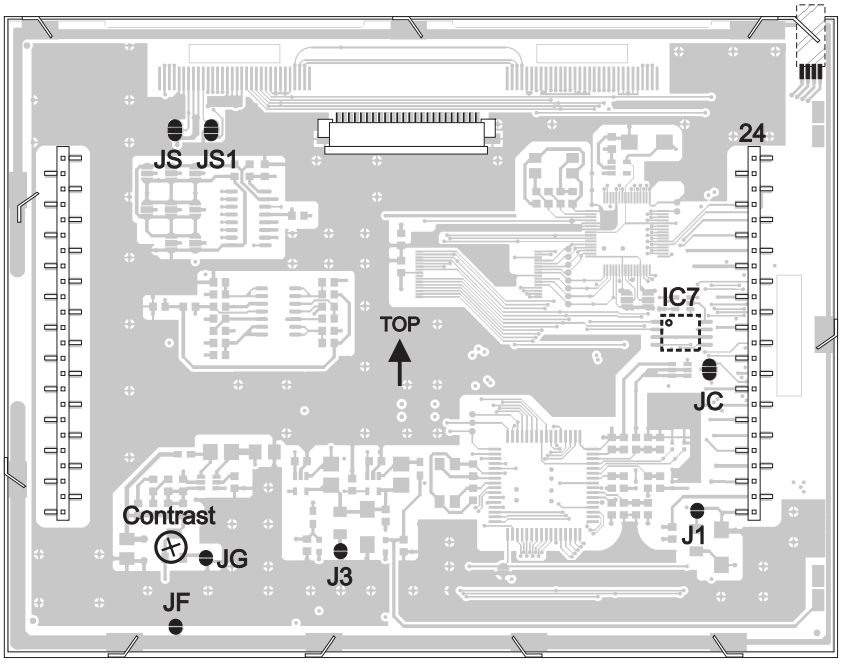
The best way to view the EA eDIP320 is diagonally from below (bottom view, 6 o'clock).
The eDIP320 can be **installed rotated by 180°** to get the top view (12 o'clock).
To correct the contents of the screen, three solder straps (JS, JS1 and JC) have to be resoldered.



Important: Always place all solder straps in the same position and desolder them cleanly. Short circuits destroy the eDIP320-8.

If an eDIP320-8 with a touch panel is used, the touch interpretation must also be changed with the command 'ESC AO 1'.





POWER-DOWN MODE

To save energy (battery operation), you can activate power-down mode by means of the command 'ESC PD n1' (see page 15 below). The LED illumination is switched off, and the contents of the display become invisible although they are still there.
In power-down mode including suppressor diodes, the eDIP20 typically requires 150 µA.
Thanks to the integrated suppressor diodes, however, the shunt current can also be 1000 µA and more.
The suppressor diodes can be deactivated by opening the solder straps J1 and J3. Then power-down current of typically 20 µA is reached.

Important: When the solder straps J1 and J3 are open, it is essential that the polarity of the display is correct at all the time: VDD, GND (pin 1 + 2). Even very brief polarity reversal or overvoltage can damage the display immediately and irreparably.

The eDIP320 can be woken from power-down mode by a level of L at pin 13 (WUP), when the screen is touched or through the I2C address.

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. There are different types of macro (compiler directive marked in green letters):

Normal macro (0 to 255) `Makro:`
These are started by means of an 'ESC MN xx' command via the serial interface or from another macro. A series of macros occurring one after the other can be called cyclically (movie, hourglass, multi-page help text). These automatic macros continue to be processed until either a command is received via the interface or a touch macro with a corresponding return code is activated. These macros are also called by macro processes at defined intervals. Macro processes are not interrupted when commands are received from the interface or when touch macros are triggered.
Touch macro (1 to 255) `TouchMakro:`
Started when you touch/release a touch field (only in versions with a touch panel - TP) or issue an 'ESC MT xx' command.
Menu macro (1 to 255) `MenuMakro:`
Started when you choose a menu item or issue an 'ESC MM xx' command.
Power-on macro `PowerOnMakro:`
Started after power-on. You can switch off the cursor and define an opening screen, for example.
Reset macro `ResetMakro:`
Started after an external reset (low level at pin 5).
Watchdog macro `WatchdogMakro:`
Started after a fault/error (e.g. failure).
Brown-out macro `BrownoutMakro:`
Started after a voltage drop under 3V.
WakeUpPin macro `WakeUpPinMakro:`
Started after waking from power-down mode at pin 13 (WUP).
WakeUpTouch macro `WakeUpTouchMakro:`
Start after waking from power-down mode by touch contact (WUP).
WakeUpI²C macro `WakeUpI2CMakro:`
Started from power-down mode via the I²C bus.

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

CREATING INDIVIDUAL MACROS AND IMAGES

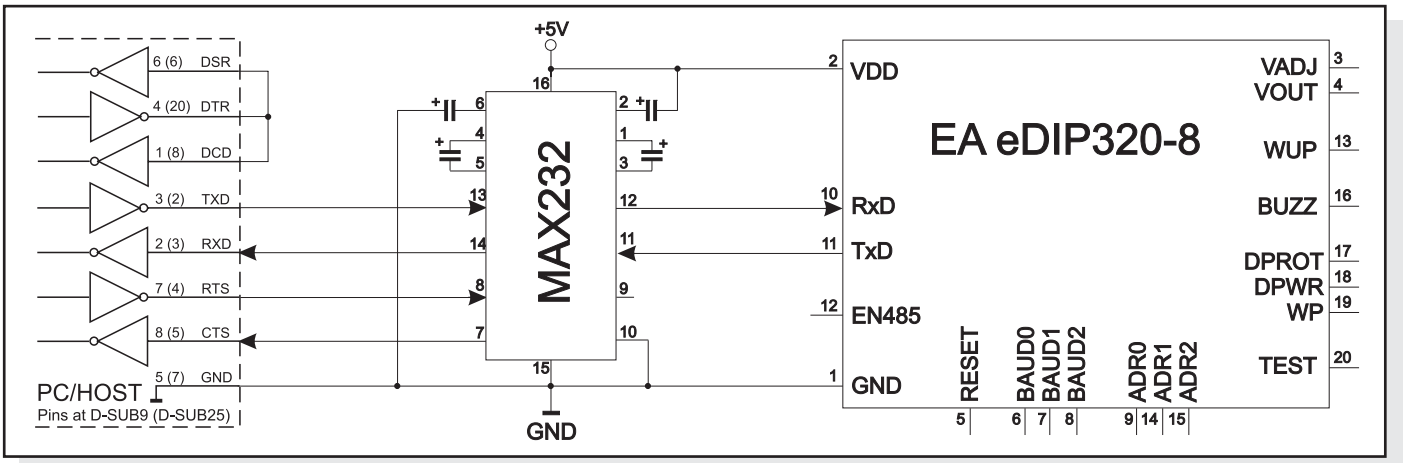
To create your own macros, you need the following:

- To connect the display to the PC, you need the EA 9778-1USB USB programmer, which is available as an accessory, or a self-built adapter with a MAX232 level converter (see the application example below).
- ELECTRONIC ASSEMBLY LCD-Tools*), which contains a kit editor, kit compiler and examples and fonts (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 eDIP320 compiler using F5. This creates a file called DEMO.DF. If an EA 9778-1USB programmer is also connected or the display is connected to the PC via a MAX232, this file is automatically burned in the display's data flash memory. The eDIP320 compiler recognizes the display regardless of whether the small protocol is switched on.

You will find a detailed description of the programming of the macros together with examples in the ELECTRONIC ASSEMBLY LCD-Tools*) help system.



Adaptor for interfacing to a PC

*) *full version is free available on web at* http://www.lcd-module.com/products/touch.html

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 data flash memory (80 kB) using 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.
You can use the “PICTURE” compiler directive to integrate two monochrome BMPs of equal size for touch keys, screen masks or flashing images.

```
PICTURE: 1 <BITMAP1.BMP>
PICTURE: 2 <BITMAP2.BMP>,<MASK2.BMP>
PICTURE: 3 <BITMAP3.BMP>,<BLINK3.BMP>
PICTURE: 4 <TOUCH.BMP>,<TOUCHPRESSED.BMP>
```

MACRO PAGES (MULTILINGUAL CAPABILITY)

There are 16 complete macro sets available in each case for the normal, touch and menu macros as well as the internal images. By simply switching the active macro page (ESC M K n1), for example, up to 16 different languages can thus be supported.
If a macro/picture is defined in the kit editor, a page number can be specified in square brackets after the macro/picture number.
If a macro/image is not defined in the currently set page [1] to [15], this macro/picture is automatically taken from page [0]. Thus, not all macros and images have to be stored separately for each language when they are identical in each language.

```
PICTURE: 100[0] <SAUSAGE.BMP>
PICTURE: 100[1] <BEER.BMP>
PICTURE: 100[2] <WINE.BMP>

MACRO: 2[0]                    ; SAME AS "MACRO: 0"
         #ZV REPLACE
         #ZL 25,0 „DEUTSCH  „
         #UI 0,20, 100

MACRO: 2[1]                    ; ENGLISH
         #ZV REPLACE
         #ZL 25,0 „ENGLISH  „
         #UI 0,20, 100

MACRO: 2[2]                    ; FRENCH
         #ZV REPLACE
         #ZL 25,0 „FRANCAISE“
         #UI 0,20, 100
```

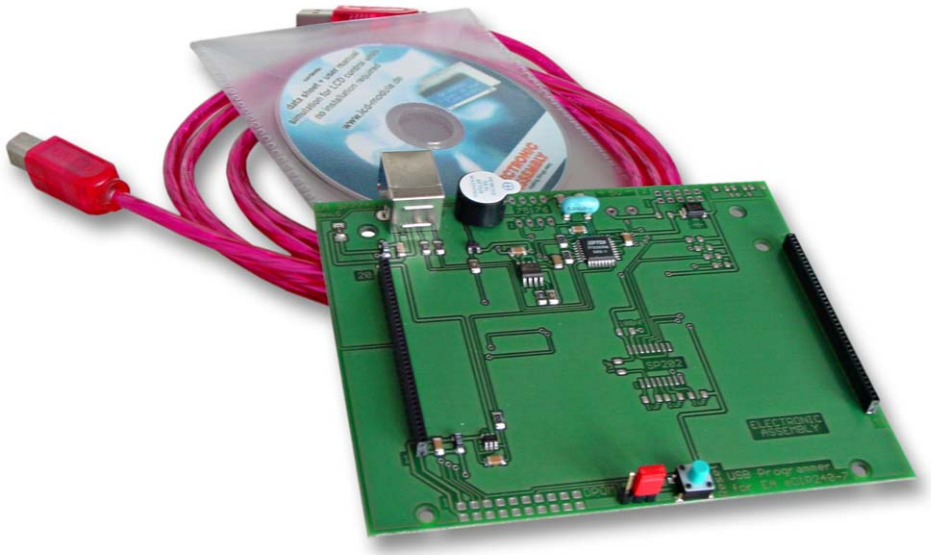
WRITE PROTECTION FOR MACRO PROGRAMMING AND FONTS

A LO level at pin 19 (WP) prevents the macros, images and fonts in the data flash memory from being overwritten inadvertently (so it is highly recommended !).

ADDING MEMORY

The internal data flash memory is 80 kB. That means there is generally enough space for a large number of icons and macros. However, if a very large number of images (full images, in particular) or several large character sets are to be stored, it may be necessary to add more memory (max. 8192 kB). This can be done by directly soldering a data flash memory from the AT45DBxxxD-SU series onto the eDIP320 (see page 18 IC7).
For example: AT45DB041D-SU = 512 kB, AT45DB081D-SU = 1024 kB or AT45DB161D-SU = 2048 kB.

ADAPTOR BOARD FOR EA eDIP320-8



TECHNICAL DATA

* **EA 9778-1USB**
* PROGRAMMING BOARD FOR USB
* INCLUDING USB CABLE
* VERY EASY TO USE, NO POWER SUPPLY REQUIRED
* REQUIRES USB DRIVER, WHICH IS INCLUDED

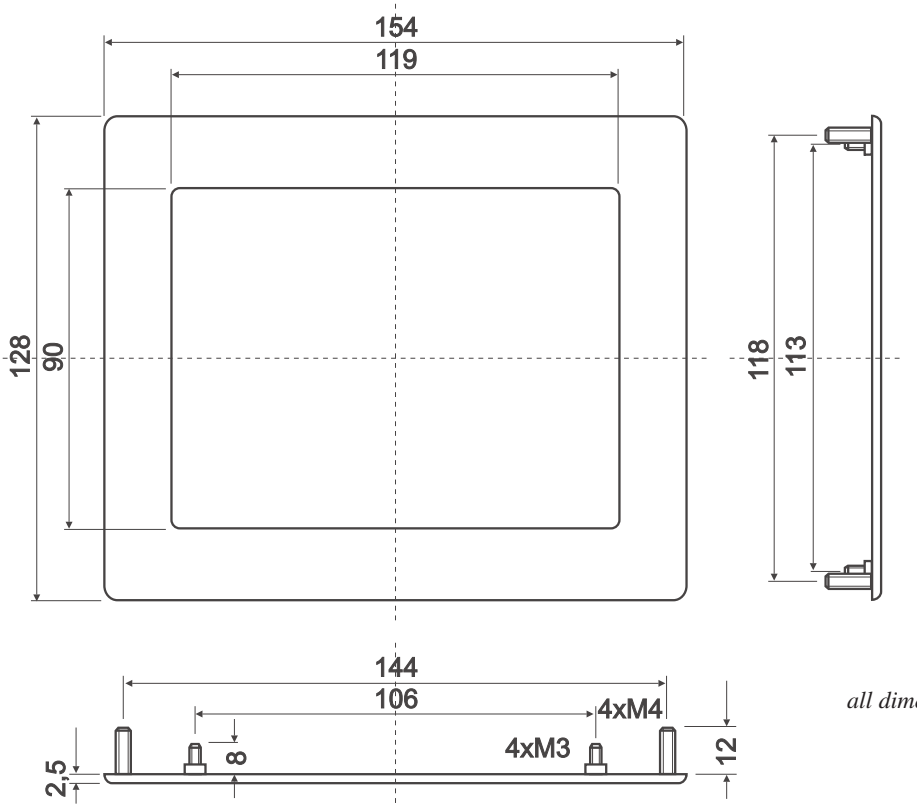
* **EA 9778-1RS232**
* RS-232 INTERFACE BOARD WITH ±12V LEVELS AT RXD AND TXD
* INCLUDING EA KV24-9B CABLE WITH 9-PIN D-SUB CONNECTOR
* REQUIRES EXTERNAL SUPPLY +5V/TYPICALLY 270 mA
* OPTIONAL SUPPLY 9 TO 35VDC INSTEAD OF 5V (EA OPT-9/35V)

* **EA 9778-1RS485**
* INTERFACE BOARD FOR RS-485 2-WIRE CONNECTION
* REQUIRES EXTERNAL SUPPLY +5V/TYPICALLY 300 mA
* OPTIONAL SUPPLY 9 TO 35VDC INSTEAD OF 5V (EA OPT-9/35V)

ORDER DESIGNATION

STARTER KIT, BLUE (1x EA eDIP320B-8LWTP + EA 9778-1USB)	**EA EVALeDIP320B**
STARTER KIT, B/W (1x EA eDIP320J-8LWTP + EA 9778-1USB)	**EA EVALeDIP320J**
PROGRAMMING BOARD INCLUDING USB CABLE AND CD FOR PC	**EA 9778-1USB**
RS-232 BOARD WITH ±12V LEVELS AT RXD AND TXD	**EA 9778-1RS232**
INTERFACE BOARD FOR RS-485 2-WIRE CONNECTION	**EA 9778-1RS485**
SUPPLY 9 TO 35VDC INSTEAD OF 5V (9778-1RS232,-1RS485 ONLY)	**EA OPT-9/35V**

MOUNTING BEZEL EA 0FP321-8SW



all dimensions in mm

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 and touch screen 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 -20 to +70°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.