embedded 3.2" TFT-DISPLAY
320x240 BUILT-IN INTELLIGENCE

Issue 04.2012

WORLD NEW !

Dimension:
82,0x60,5x12mm

FEATURES

* TFT-GRAFIKDISPLAY WITH BUILD-IN GRAPHIC FUNCTIONS
* 320x240 DOTS, 16-BIT COLOR (65.536 COLORS) WITH LED-BACKLIGHT
* 4MB ON BOARD FLASH FOR FONTS, PICTURES, ANIMATIONS AND MACROS
* POWER SUPPLY WIDE RANGE +3,3V / 160mA ... +5V / 120mA
* 8 PRE-DEFINED FONTS, CAN BE EXPANDED
* FONT ZOOM FROM 2MM TO ABOUT 80MM, TURNABLE IN 90° STEPS
* 3 DIFFERENT INTERFACES ON BOARD: RS-232, I²C-BUS OR SPI-BUS
* POSITIONING ACCURATE TO THE PIXEL WITH ALL FUNCTIONS
* DRAW LINE, PLACE A DOT, AREA, BARGRAPH...
* ROTARY AND POINTER INSTRUMENTS
* PICTURES AND ANIMATIONS
* MIX TEXT AND GRAPHIC
* MULTI-LINGUAL WITH MACRO PAGES
* BACKLIGHT BRIGHTNESS BY SOFTWARE
* ANALOGUE TOUCH PANEL: VARIABLE GRID
* FREE DEFINABLE KEY AND SWITCH
* 8 DIGITAL IN- AND 8 DIGITAL OUTPUTS
* 2 ANALOGUE INPUTS, COMFORTABLE TO USE

ORDERING CODES

DISPLAYS	
TFT 320x240 DOTS, WHITE LED BACKLIGHT	**EA eDIPTFT32-A**
AS ABOVE, BUT WITH TOUCH PANEL	**EA eDIPTFT32-ATP**
STARTERKIT	
INCLUDES EA eDIPTFT32-ATP AND EVALUATION BOARD WITH USB FOR DIRECT CONNECTION TO PC AND INTERFACE BOARDS FOR CONNECTION WITH YOUR HOST SYSTEM	**EA EVALeDIPTFT32**
ADDTIOTNAL PARTS	
MOUNTING BEZEL (ALUMINIUM), BLACK ANODIZED	**EA 0FP322-32SW**
SOCKET 1x20, 4.5mm HIGH (1 piece)	**EA B254-20**

Zeppelinstr. 19 · D-82205 Gilching · Phone +49-(0)8105-778090 · Fax +49-(0)8105-778099 · www.lcd-module.de · info@lcd-module.de

Documentation of revision				
Date	**Type**	**Old**	**New**	**Reason / Description**
February, 2010	0.1			preliminary version
February, 2011	1.0		- Instrument - WinFonts	1st. Edition
January, 2012	1.1		- StringTable - Draw a X/Y-graph (#GX, #GY, #GS)	new functions

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.
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 between +3.3V to +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.

ANALOGUE 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

All displays are equipped with modern, energy-saving LED illumination. Brightness can be varied 0~100% by command.
In 24-hour operation, the illumination 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 (319,239)*.
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 8 times and rotated in 90° steps. With the largest character set, the words and numbers displayed will fill the screen.

ACCESSORIES

Evaluation-Board (Programmer) for internal data flash memory
The display is shipped fully programmed and with all fonts. The additional Evaluation-Board 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 Evaluation-Board EA 9777-2USB, which is available as an accessory, will burn the data/images you have created into the on-board data flash memory (4 MB) permanently.
The Evaluation-Board runs under Windows and is connected to the PC's USB interface. It is shipped with an interface cable and the installation software. The Evaluation-Board is equipped with serveral LEDs, pushbottons and potentiometer to test all peripherial modes of the eDIP.
Interface-Expansion for Evaluation-Board (included in the Starter-Kit):
Wtih the expansion EA 9777-2PE for the Evaluation-Board all interfaces of the display are made available with the help from small adapter boards: RS-232, RS-485, SPI, I²C, RS-232 (CMOS level).
Further information you will find in the datasheet of the Evalution-Board.

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 eDIPTFT32-A: RS-232/RS-485 mode

Pin	Symbol	In/Out	Function	Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)	21	GND		Ground (=Pin 1)
2	VDD		Power supply for logic (+3,3V ... +5V)	22	VDD		Power supply (=Pin 2)
3	NC		do not connect	23	AIN1	In	analogue input 0..VDD DC impedance 1MOhm
4	NC		do not connect	24	AIN2		
5	RESET	In	L: Reset	25	OUT1 / MO8	Out	8 digital outputs maximum current: IOL = IOH = 10mA alternativ up to 8 matrix keyboard output lines (reduces the digital output lines, see chapter external keyboard)
6	BAUD0	In	Baud Rate 0	26	OUT2 / MO7		
7	BAUD1	In	Baud Rate 1	27	OUT3 / MO6		
8	BAUD2	In	Baud Rate 2	28	OUT4 / MO5		
9	ADR0	In	Address 0 for RS-485	29	OUT5 / MO4		
10	RxD	In	Receive Data	30	OUT6 / MO3		
11	TxD	Out	Transmit Data	31	OUT7 / MO2		
12	EN485	Out	Transmit Enable for RS-485 driver	32	OUT8 / MO1		
13	DPOM	In	L: disable PowerOnMacro do not connect for normal operation	33	IN1 / MI8	In	8 digital inputs open-drain with internal pullup 20..50k alternativ up to 8 matrix keyboard input lines (reduces the digital input lines, see chapter external keyboard)
14	ADR1	In	Address 1 for RS-485	34	IN2 / MI7		
15	ADR2	In	Address 2 for RS-485	35	IN3 / MI6		
16	BUZZ	Out	Buzzer output	36	IN4 / MI5		
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation	37	IN5 / MI4		
18	DNC	Out	L: internal, do not connect	38	IN6 / MI3		
19	WP	In	L: Writeprotect for DataFlash	39	IN7 / MI2		
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	40	IN8 / MI1		

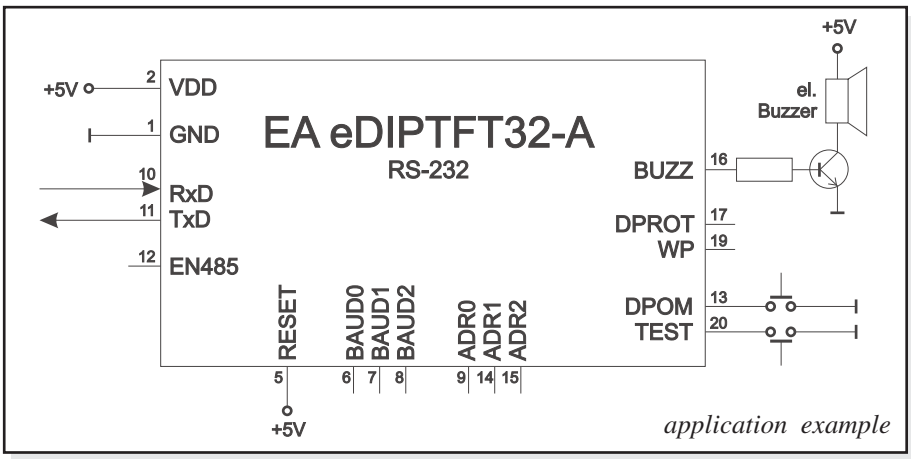
BAUD RATES

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 10 and 11).

Baud Rates

Baud0	Baud1	Baud2	data format 8,N,1
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
0	0	0	230400

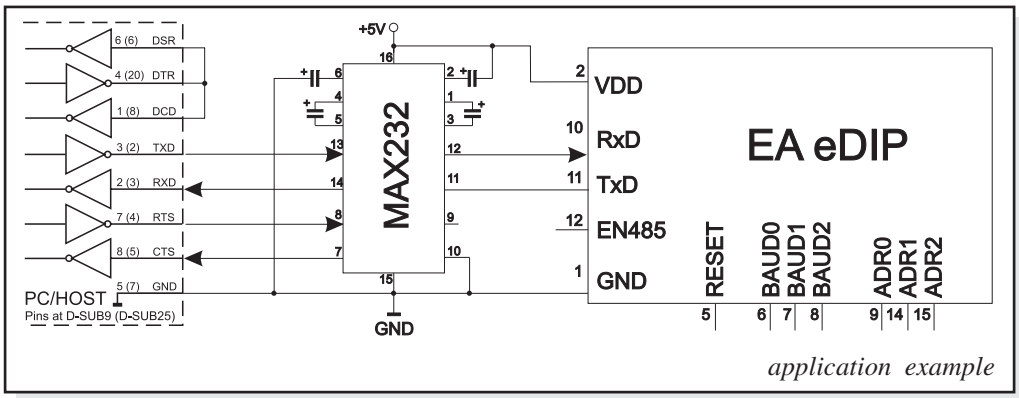


application example

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

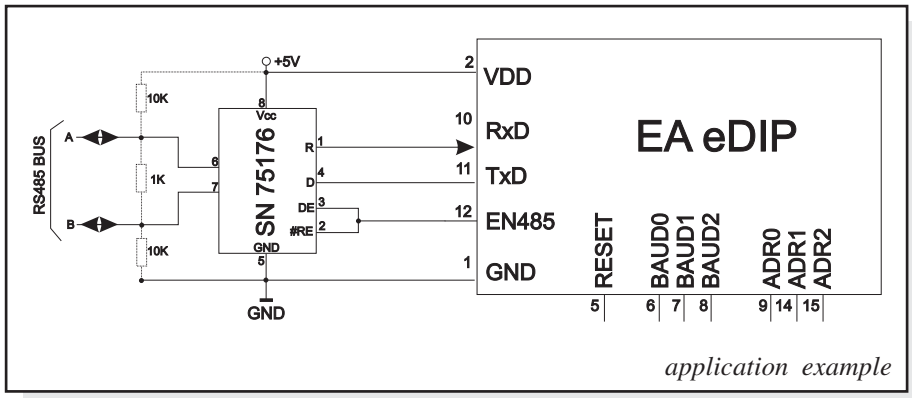
APPLICATION EXAMPLE „REAL“ RS-232 INTERFACE

The eDIP 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 eDIP 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 eDIP displays can be operated on a single RS-485 bus by setting addresses.

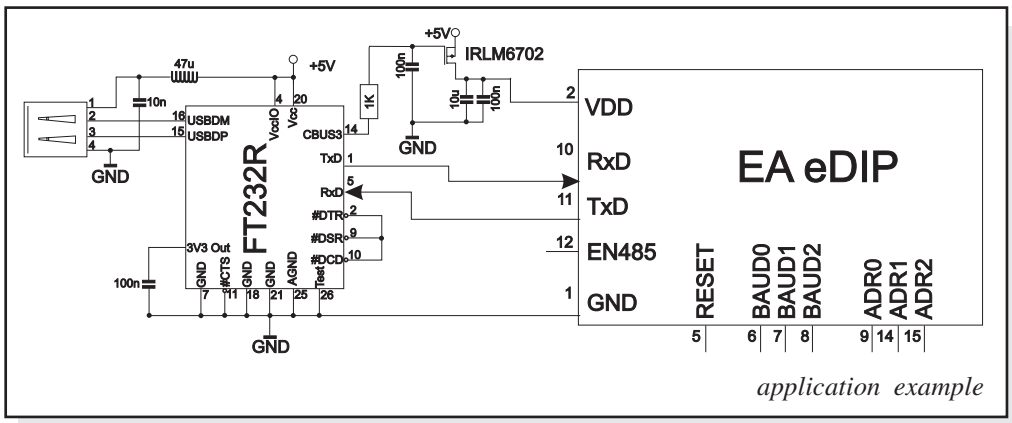


Addressing:

- *Up to eight hardware addresses (0 to 7) can be set by means of Pins ADR0..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 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 eDIP externally to address 0)*

APPLICATION EXAMPLE: USB INTERFACE

With an external converter (e.g. FT232R) from FTDI the eDIP 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.](http://www.ftdichip.com/drivers/vcp.htm)



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 eDIPTFT32-A: SPI mode

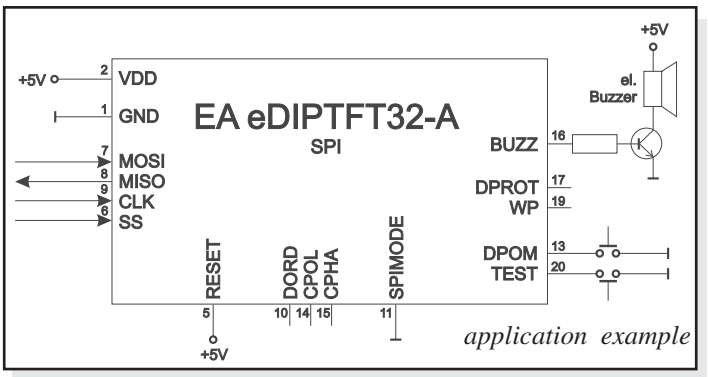
Pin	Symbol	In/Out	Function	Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)	21	GND		Ground (=Pin 1)
2	VDD		Power supply for logic (+3,3V ... +5V)	22	VDD		Power supply (=Pin 2)
3	NC		do not connect	23	AIN1	In	analogue input 0..VDD
4	NC		do not connect	24	AIN2		DC impedance 1MOhm
5	RESET	In	L: Reset	25	OUT1 / MO8	Out	8 digital outputs
6	SS	In	Slave Select	26	OUT2 / MO7		maximum current:
7	MOSI	In	Serial In	27	OUT3 / MO6		IOL = IOH = 10mA
8	MISO	Out	Serial Out	28	OUT4 / MO5		
9	CLK	In	Shift Clock	29	OUT5 / MO4		alternativ up to 8 matrix
10	DORD	In	Data Order (0=MSB first; 1=LSB first)	30	OUT6 / MO3		keyboard output lines
11	SPIMO	In	connect to GND for SPI interface	31	OUT7 / MO2		(reduces the digital
12	NC		do not connect	32	OUT8 / MO1		output lines, see chapter external keyboard)
13	DPOM	In	L: disable PowerOnMacro do not connect for normal operation	33	IN1 / MI8	In	
14	CPOL	In	Clock Polarity (0=LO 1=HI when idle)	34	IN2 / MI7		8 digital inputs
15	CPHA	In	Clock Phase sample 0=1st;1=2nd edge	35	IN3 / MI6		open-drain with internal
16	BUZZ	Out	Buzzer output	36	IN4 / MI5		pullup 20..50k
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation	37	IN5 / MI4		alternativ up to 8 matrix
18	DNC	Out	L: internal, do not connect	38	IN6 / MI3		keyboard input lines
19	WP	In	L: Writeprotect for DataFlash	39	IN7 / MI2		(reduces the digital input
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	40	IN8 / MI1		lines, see chapter external keyboard)

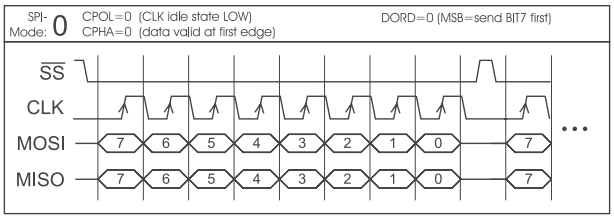
Note:
The pins DORD, CPOL, CPHA, DPOM, DPROT and TEST/SBUF have an internal pullup, which is why only the LO level (0=GND) is to be actively applied. These pins must be left open for a Hi level.
On pin 20 (SBUF) the display indicates with a low level that data is ready to be retrieved from the internal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

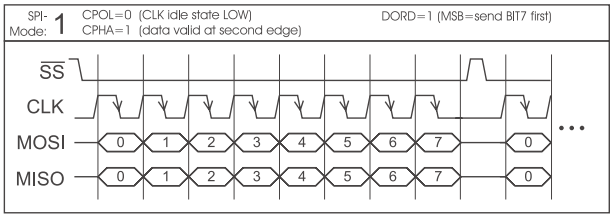
DATA TRANSFER SPI

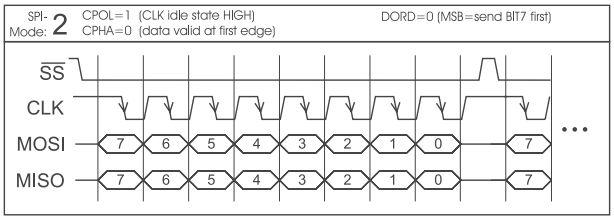
Write operation: a clock rate up to 200 kHz is allowed without any stop. Together with a pause of 100 µs between every data byte a clock rate up to 3 MHz can be reached.

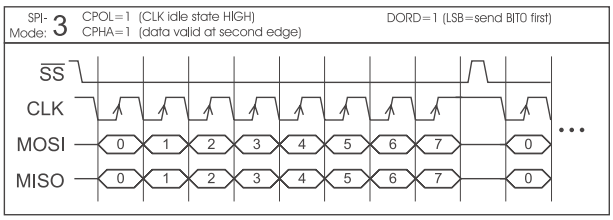
Read operation: to read data (e.g. the „ACK“ byte) a dummy byte (e.g . 0xFF) need to be sent.
Note that the EA eDIP for internal operation does need a short time before providing the data; therefore a short pause of min. 6µs (no activity of CLK line) is needed for each byte.











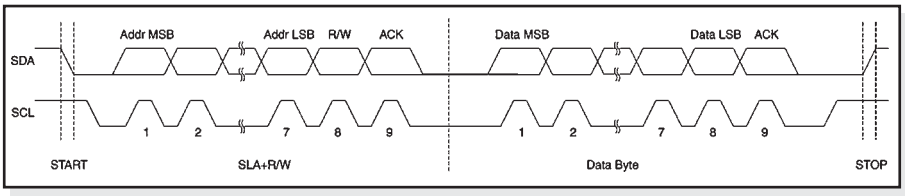
I²C-BUS INTERFACE

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.

Pinout eDIPTFT32-A: I2C mode

Pin	Symbol	In/Out	Function	Pin	Symbol	In/Out	Function
1	GND		Ground Potential for logic (0V)	21	GND		Ground (=Pin 1)
2	VDD		Power supply for logic (+3,3V ... +5V)	22	VDD		Power supply (=Pin 2)
3	NC		do not connect	23	AIN1	In	analogue input 0..VDD
4	NC		do not connect	24	AIN2		DC impedance 1MOhm
5	RESET	In	L: Reset	25	OUT1 / MO8	Out	8 digital outputs
6	BA0	In	Basic Address 0	26	OUT2 / MO7		maximum current:
7	BA1	In	Basic Address 1	27	OUT3 / MO6		IOL = IOH = 10mA
8	SA0	In	Slave Address 0	28	OUT4 / MO5		
9	SA1	In	Slave Address 1	29	OUT5 / MO4		alternativ up to 8 matrix
10	SA2	In	Slave Address 2	30	OUT6 / MO3		keyboard output lines
11	BA2	In	Basic Address 2	31	OUT7 / MO2		(reduces the digital
12	I2CMO	In	connect to GND for I²C interface	32	OUT8 / MO1		output lines, see chapter external keyboard)
13	DPOM	In	L: disable PowerOnMacro do not connect for normal operation	33	IN1 / MI8	In	
14	SDA	Bidir.	Serial Data Line	34	IN2 / MI7		8 digital inputs
15	SCL	In	Serial Clock Line	35	IN3 / MI6		open-drain with internal
16	BUZZ	Out	Buzzer output	36	IN4 / MI5		pullup 20..50k
17	DPROT	In	L: Disable Smallprotokoll do not connect for normal operation	37	IN5 / MI4		alternativ up to 8 matrix
18	DNC	Out	L: internal, do not connect	38	IN6 / MI3		keyboard input lines
19	WP	In	L: Writeprotect for DataFlash	39	IN7 / MI2		(reduces the digital input lines, see chapter
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	40	IN8 / MI1		external keyboard)

Note:
The pins BA0..2, SA0..2, 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

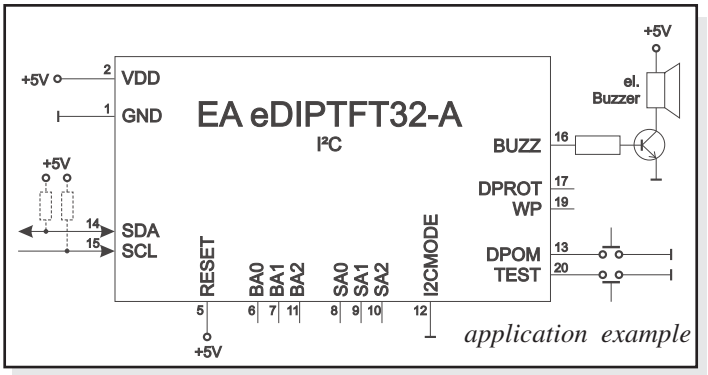
Pin 11,7,6 BA2	BA1	BA0	Base address	I²C address D7	D6	D5	D4	D3	D2	D1	D0
L	L	L	$10	0	0	0	1				
L	L	H	$20	0	0	1	0				
L	H	L	$30	0	0	1	1				
L	H	H	$40	0	1	0	0	SA2	SA1	SA0	RW
H	L	L	$70	0	1	1	1				
H	L	H	$90	1	0	0	1				
H	H	L	$B0	1	0	1	1				
H	H	H	$D0	1	1	0	1				

all pins open: Write $DE
Read $DF

DATA TRANSFER I²C INTERFACE

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).



ANALOGUE INPUT AIN1 AND AIN2 (PIN 23+24)

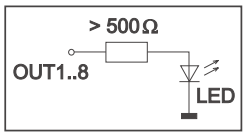
For analogue measurement 2 inputs with a range of 0..VDD are available. Each input is grounded (GND) and DC impedance is 1MΩ. Please make sure that only positive voltages will be supplied there. Internal resolution is 10 Bit, equal to a 3-digit DVM modul. Linearity (after adjustment) is around 0.5%.

Adjustment

Analogue inputs are not calibrated when shipped out. A procedure for adjustment may be like that:
1.) Put a well known voltage within a range of 2-VDD to analogue input (example: 3,0V, AIN1)
2.) Run command for calibration (see page 15). Example: „ESC V @ 1 3000“.

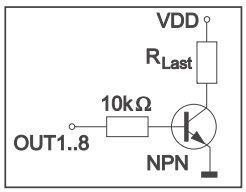
Each input query can be done via serial interface or directly shown on display (as digits or bargraph in various colors and sizes).
Best way for direct visualisation are Process-macros or one of Analogue-macros (e.g. starting at every voltage change, or above/below a limit).
Both input lines are scaleable from 0 to ±9999.9. Scaling will be done via definition at 2 voltages „value1=string1;value2=string2“ (see table on page 16).

DIGITAL INPUT AND OUTPUT



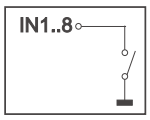
The EA eDIP is featured with 8 digital input and 8 digital output lines (CMOS level, grounded).

8 outputs (Pin 25-32)



Each line can be controlled individually using the „ESC Y W“ command. A maximum current of 10mA can be switched per line. This give the opportunity to drive a low power LED in direct way. To source higher current please use an external transistor.

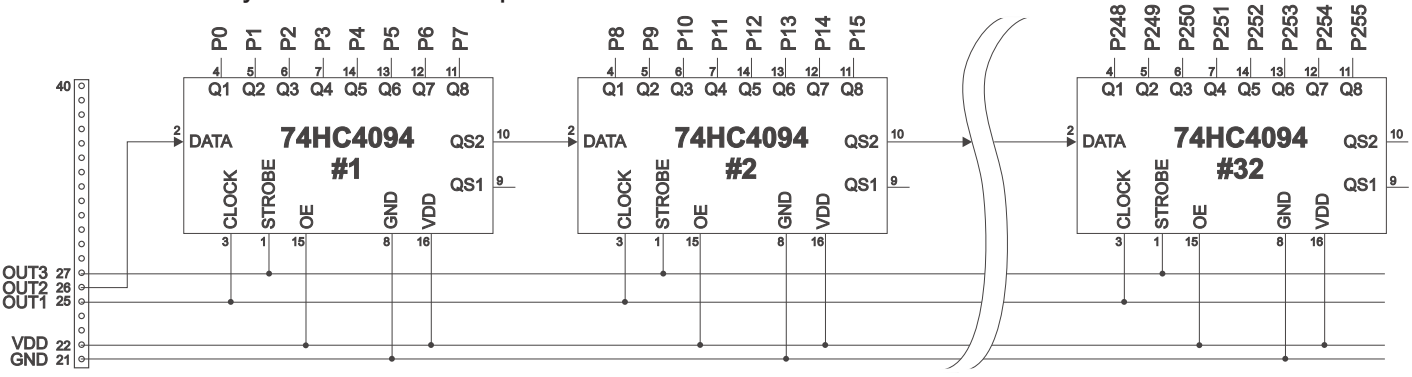
8 inputs (Pin 33-40)



Each input provides an internal 20..50 kΩ pull-up resistor, so it is possible to connect a key or switch directly between input and GND. The inputs can be queried and evaluated directly via the serial interface („ESC Y R“).
In addition to that every port change may start an individual port - or bit- macro (see p. 28).
The command "ESC Y A 1" activates automatic port query. Every alteration of inputs firstly calls bit macros and afterwards port macros. If there is no definied macro, the new status is transfered into the send buffer (refer to p. 19).
Note: The logic circuitry is designed for slow operations; in other words, more than 3 changes per second cannot be easily executed.

EXTENDED OUTPUTS

It is possible to connect 1 to 32 chips like 74HC4094 to the eDIP (OUT1...OUT3), this is why it is attainable to have 8 to 256 additional outputs. The command "ESC Y E n1 n2 n3" (see p. 17) provides a comfortable way to control the outputs.



EXTERNAL KEYBOARD

A keyboard (anything from individual keys to a 8x8 matrix keyboard) can be connected to the I/O- ports. The command 'ESC Y M n1 n2 n3' defines the count of input lines (n1=1..8) and output lines (n2=1..8). n3 set debounce function with 10ms steps (n3=0..15). Please note that count of digital input and output lines will be reduced while connecting an external keyboard at the same port.
Each key is connected with 1 output and 1 input. All inputs are terminated with a 20..50kΩ pull-up resistor. For double-keytroke function decoupling of outputs is necessary. For that please use schottky diodes (e.g. BAT 46).

Transmitting the keystrokes

At each keystroke, the associated key number (1..64) is transmitted or if a corresponding Matrix-Macro is defined, Matrix-Macro will be started. The release of the key is not transmitted. If the release of the key is to be transmitted as well, this can be done by defining Matrix-Macro no. 0.
(see page 17: Responses of EA eDIP)

Calculating of key numbers:

Key_number = (output-1) * count_of_inputs + input (output = MOx, input = MIx).
Examples

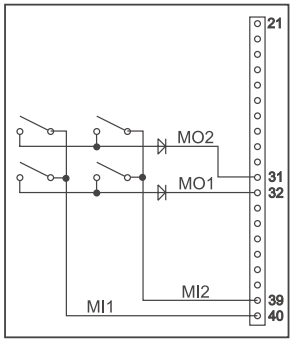
- 2x2 matrix: Command 'ESC Y M 2 2 ..' defines the 2x2 matrix. Keypad will need input lines MI1, MI2 and output lines MO1, MO2. Output lines are decoupled by diodes; this is for double keystrokes necessary. 6 input and 6 output lines remain free for other requirements.

- 1x4 matrix: Command 'ESC Y M 1 4 ..' defines the 1x4 matrix. Keypad will need output lines MO1..MO4 and a single input line MI1. With that connection 7 input and 4 output lines remain free for other requirements.

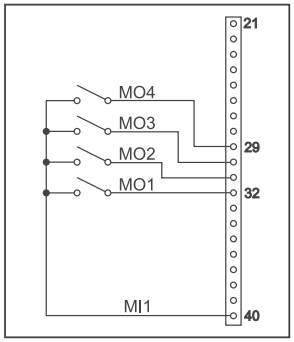
- 4x0 matrix: Using one single output only (physically 4x1 Matrix), all keys can switch to GND. So no output line is necessary and comman 'ESC Y M 4 0 ..' defines 4 input lines onyl. With that connection 4 input and 8 output lines remain free for other requirements.

- 4x4 matrix: Command 'ESC Y M 4 4 ..' defines the 4x4 matrix. Keypad will need input lines MI1..MI4 and output lines MO1..MO4. Output lines are decoupled by diodes; this is for double keystrokes necessary. 4 input and 4 output lines remain free for other requirements.

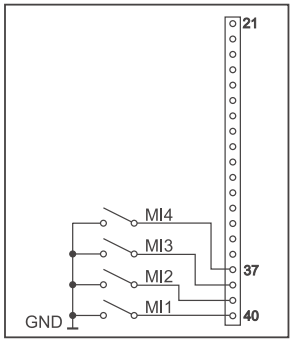
.



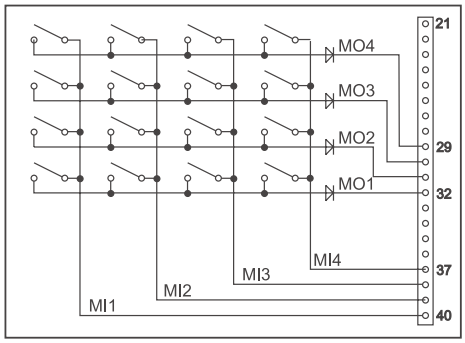
2x2 Matrix



1x4 Matrix



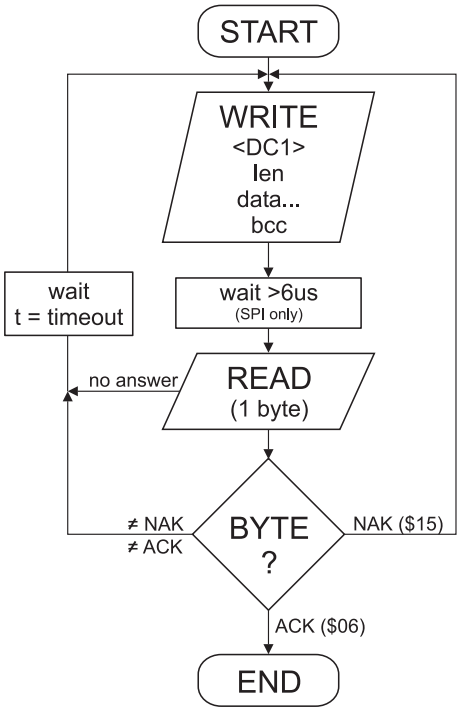
4x0 Matrix



4x4 Matrix

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 eDIPTFT32-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 elapse 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 divided up between a number of packages. All data in the packages are compiled again after being correctly received by the EA eDIP.



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 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 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 for 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-Code to transmit data package

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 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(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

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(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 (eDIPTFT32-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

This command is used to query protocol settings.

Repeat the last package

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

<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

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(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

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

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 or a deactivated protocol (see pages 10 and 11).
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 size of the terminal-window can be set by command 'ESC TW'.
Attention: Graphic commands are able to draw inside terminal window. For example 'ESC DL' will delete terminal window, too.

+ 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)
$00 (dez: 0)	NUL	⇧	⇩	⇨	⇦	√	♪	♦	←	↦	LF	↧	FF	CR	SO	SI
$10 (dez: 16)	0	1	2	3	4	5	6	7	8	9	ESC	↑	↓	→	←	
$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)	Ç	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 2: 8x16

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. Binär 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).

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 (see page 28).

EA eDIPTFT32-A: Terminal commands										after reset
Command	**Codes**								**Remarks**	
Set terminal color	ESC	F	T	fg	bg				Preset color for terminal mode: fg= foreground color; bg= background color	8,1
Define window	ESC	T	W	n1	C	L	W	H	The terminal output is executed with font n1: 1=8x8; 2=8x16 only within the window from column C and line L (=upper-left corner) with a width of W and a height of H (specifications in characters). Display organisation 480x272: C=1..60; L=1..34/17; 272x480: C=1..34; L=1..60/30	8x16 1,1 60,17
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	ESC	T	V						The version no. is output in the terminal e.g. "EA eDIPTFT43-A V1.0 Rev.A"	
Output projectname			J						The macrofile-projectname is output in the terminal e.g. "init / delivery state"	
Output interface			Q						The used interface is output in the terminal e.g "RS232,115200 baud,ADR: $07"	
Output informationen	ESC	T	I						The terminal is initialisized and cleared; the software version, hardware revision, macrofile-projectname and CRC-checksum are output in the terminal	

EA eDIPTFT32-A: Graphic commands											after reset
Command	**Codes**									**Remarks**	
Display commands (effect on the entire display)											
Set display color	ESC	F	D	fg	bg					Defines color 1..32 for display and areas: fg=foreground color; bg=background color	8,1
Delete display	ESC	D	L							Delete display contents (all pixels to background color)	
Fill display	ESC		S							Fill display contents (all pixels to foreground color)	
Fill display with color	ESC		F	n1						Fill complete display content with color n1=1..32	
Invert display	ESC		I							Invert display content	
Commands for outputting strings											
Set text color	ESC	F	Z	fg	bg					Color 1..32 (0=transparent) for string and character: fg=text color; bg=background color	8,0
Set font	ESC	Z	F	n1						Set font with the number n1	3
Font zoom factor			Z	n1	n2					n1 = X-zoom factor (1x to 8x); n2 = Y-zoom factor (1x to 8x)	1,1
Additional width/height			Y	n1	n2					n1=0..15: additional width left/right; n2=0..15: additional height top/bottom	0, 0
Spacewidth			J	n1						n1=0: use spacewidth from font; n1=1: same witdh as a number; n1>=2 width in dot	0
Text angle			W	n1						Text output angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
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, pipe) the character '\' ($5C, backslash) cancles the special function of '|' and '\'	
Output string in an area (since V1.2)	ESC	Z	B	xx1	yy1	xx2	yy2	n1	text ...	NUL	Output a string (...) inside area from xx1,yy1 to xx2,yy2 at position n1=1..9; the area will be filled with background color; n1=1: Top Left; n1=2: Top Center; n1=3: Top Right n1=4: Middle Left; n1=5: Middle Center; n1=6: Middle Right n1=7: Bottom Left; n1=8: Bottom Center; n1=9: Bottom Right	
String for terminal	ESC	Z	T	text ...						Command for outputting a string from a macro to the terminal	
Draw straight lines and points											
Set color for lines	ESC	F	G	fg	bg					Colors 1..32 (0=transparent): fg = color for line; bg = pattern background	8,1
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			D	xx1	yy1	xx2	yy2			Draw straight line from xx1,yy1 to xx2,yy2	
Continue straight line			W	xx1	yy1					Draw a straight line from last end point to xx1, yy1	
Draw point			P	xx1	yy1					Set a point at coordinates xx1, yy1	
Point size/line thickness			Z	n1	n2					n1 = X-point size (1 to 15); n2 = Y-point size (1 to 15);	1,1
Pattern			M	n1						Set straight line/point pattern no. n1=1..255; 0=do not use pattern	0
Set start point	ESC	G	S	xx1	yy1					Set the last end point at coordinates xx1, yy1 for commands 'GW', 'GX' and 'GY' (since V1.2)	
Draw X-Graph			X	xs	ya	yy1				Draw graph with fix x-steps (xs=1..127 or 129..255 for neg. steps) and variable amount (ya=1..255) of y-values (since V1.2)	
Draw Y-Graph			Y	ys	xa	xx1				Draw graph with fix y-steps (ys=1..127 or 129..255 for neg. steps) and variable amount (xa=1..255) of x-values (since V1.2)	
Change/draw rectangular areas											
Delete area	ESC	R	L	xx1	yy1	xx2	yy2			Delete an area from xx1,yy1 to xx2,yy2 (fill with background color)	
Fill area			S	xx1	yy1	xx2	yy2			Fill an area from xx1,yy1 to xx2,yy2 (fill with foreground color)	
Fill area with color			F	xx1	yy1	xx2	yy2	n1		Fill an area from xx1,yy1 to xx2,yy2 with color n1=1..32	
Invert area			I	xx1	yy1	xx2	yy2			Invert an area from xx1,yy1 to xx2,yy2	
Copy area			C	xx1	yy1	xx2	yy2	xx3	yy3	Copy an area from xx1,yy1 to xx2,yy2 to new position xx3,yy3	
Patterncolor	ESC	F	M	fg	bg					Color 1..32 (0=transp.) for monochrome pattern: fg=foreground; bg=background color	8,1
Area with fill pattern		R	M	xx1	yy1	xx2	yy2	n1		Draw an area from xx1,yy1 to xx2,yy2 with pattern n1	
Draw box			O	xx1	yy1	xx2	yy2	n1		Draw a rectangle xx1,yy1 to xx2,yy2 and fill with pattern n1	
Set color for border	ESC	F	R	c1	c2	c3				Set color for border segments: c1=frame outside; c2=frame inside; c3=filling	8,1,1
Set border type		R	E	n1	n2					Set border type n1=1..255; border angle: n2=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	1, 0
Draw border box			R	xx1	yy1	xx2	yy2			Draw a border box from xx1,yy1 to xx2,yy2	

EA eDIPTFT32-A: Bitmap / Animation commands												after reset
Command	**Codes**										**Remarks**	
Bitmap image commands												
Set bitmap colors	ESC	F	U	fg	bg						painting color for monchrome bitmaps fg=foreground color; bg=background color	1,8
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; n1=1: the image is mirrored horizontally	0
Transparency for color bitmaps	ESC	U	T	n1							n1=0: no transparency; show picture with all colors rectangular n1=1: color of the first dot at top left side will be defined as transparent (like a mask) n1=2: if defined - use transparent color from bitmap-file (.GIF .TGA .G16) n1=3: replace transparent color from bitmap-file with actually background color	2
Load internal image	ESC	U	I	xx1	yy1	nr					Load internal image with the no (0 to 255) from the data flash memory to xx1,yy1	
Load image	ESC	U	L	xx1	yy1	G16 data ...					Load an image to xx1,yy1; see image structure (G16 format) for image data	
RLE compression	ESC	U	R								the next hardcopy ('ESC U H xx1,yy1,xx2,yy2) is send with RLE compression (since V1.2)	
Send hardcopy	ESC	U	H	xx1	yy1	xx2	yy2				After this command, the image extract is sent (to sendbuffer) in G16 format	
Animation image commands												
Set animation colors	ESC	F	W	fg	bg						color for monchrome animation images fg=foreground color; bg=background color	1,8
Animation zoom factor	ESC	W	Z	n1	n2						n1 = X-zoom factor (1x to 8x); n2 = Y-zoom factor (1x to 8x)	1,1
Animation angle	ESC	W	W	n1							output angle of the animation image n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
Mirror animation	ESC	W	X	n1							n1=0: normal display; n1=1: the animation image is mirrored horizontally	0
Transparency for color animation	ESC	W	T	n1							n1=0: no transparency; show animation with all colors rectangular n1=1: color of the first dot at top left side will be defined as transparent (like a mask) n1=2: if defined - use transparent color from animation-file (.GIF .G16) n1=3: replace transparent color from animation-file with actually background color	2
Load single image	ESC	W	I	xx1	yy1	n1	n2				Load from animation n1=0..255 the single image n2 to xx1,yy1	
Define animationprocess	ESC	W	D	no	xx1	yy1	n2	type	time		Define an animationprocess no=1..4 at position xx1,yy1 (=left top edge) with animation image n2=0..255. type: 1=run once; 2=cyclically; 3=pingpong; 4=once backwards; 5=cyclic backwards 6=pingpong backwards; 7=manually (use command ESC W N|P|F|M) time: 0=stop; 1..254=time in in 1/10 sec; 255=use time from animation-file	
Change animation type	ESC	W	Y	no	type						Assign a new type=1..7 to animationprocess no=1..4	
Change animation time	ESC	W	C	no	time						Assign a new time=0..255 to animationprocess no=1..4	
Next animation image	ESC	W	N	no							Show the next image from animationprocess no=1..4	
Previous animation image	ESC	W	P	no							Show the previous image from animationprocess no=1..4	
Show animation image	ESC	W	F	no	n2						Show image n2 from animationprocess no=1..4	
Run to animation image	ESC	W	M	no	n2						Run animationprocess no=1..4 from actually image to image n2	
Stop animationprocess	ESC	W	L	no							Stop animationprocess no=1..4 and clear last image with actually background color	

EA eDIPTFT32-A: Bargraph commands													after reset
Command	**Codes**								**Remarks**				
Bargraph commands													
Set color for bargraph	ESC	F	B	fg	bg	fc			Colors: fg = foreground; bg = background; fc = color for frame				8,1,8
Bargraph pattern	ESC	B	M	n1					Pattern for bargraph n1=1..255; n1=0 no pattern/solid (valid for type=0..3)				0
Bargraph border			E	n1					Border for bargraph n1=1..255 (valid for type=4..7)				1
Bargraph linewidth			B	n1					Linewidth for bargraph n1=1..255; n1=0 automatic (valid for type=2,3,6,7)				0
Define bargraph	ESC	B	R L O U	no	xx1	yy1	xx2	yy2	sv	ev	type	Define bargraph no=1..20 to L(eft), R(ight), O(up), U(down) xx1,yy1,xx2,yy2 rectangle enclosing the bar graph. sv, ev are the values for 0% and 100%. type: 0=pattern bar; 1=pattern bar in rectangle; type: 2=pattern line; 3=pattern line in rectangle; type: 4=border bar; 5=border bar in rectangle; type: 6=border line; 7=border line in rectangle;	no bar defined
Update bargraph	ESC	B	A	no	val				Set and draw the bargraph with the number no=1..20 to the new value val				
Draw bargraph			N	no					Entirely redraw the bargraph with the number no=1..20				
Send bargraph value			S	no					Send the current value of bargraph number no=1..20 to sendbuffer				
Delete bargraph			D	no	n2				The definition of the bar graph with the number no=1..20 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				
User values - Format text output													
User value color	ESC	F	X	fg	bg				Set color for bargraph user value; fg=foreground, bg=background color				8,1
User value font	ESC	B	F	n1					Set font n1 for bargraph user value				5
User value zoom			Z	n1	n2				Set zoom factor for bargraph user value; n1=X-Zoom 1x..8x; n2=Y-Zoom 1x..8x				1,1
User value additional width/height			Y	n1	n2				n1=0..15: additional width left/right; n2=0..15: additional height top/bottom for bargraph user value;				0, 0
User value angle			W	n1					Set writing angle for bargraph user value; n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°;				0°
User values / scaling	ESC	B	X	no	xx1	yy1	Format String	NUL	Define user value for bargraph no=1..20. Output is always right justified to xx1,yy1; Format String: "bv1=uservalue1;bv2=uservalue2". 'NUL' ($00) = termination; Assign two bar values (bv1,bv2 =0..254) to user defined values max. range: 4 1/2 digits 19999 + decimal point ('.' oder ',') + sign e.g. display "-123.4" for bar value bv1=0 and "567.8" for bar value bv2=100 Format String: "0=-123.4;100=567.8"				

EA eDIPTFT32-A: Instrument commands												after reset
Command	**Codes**								**Remarks**			
Define, use instruments												
Definine instrument	ESC	I	P	n1	xx1	yy1	n2	n3	aw	ew	Define instrument n1=1..4 an xx1,yy1 (=left top edge); Use intrument image n2=0..255 Output angle n3=0: 0°; n3=1: 90°; n3=2: 180°; n3=3: 270°; aw, ew (0..254) are start and endvalue (0% and 100%).	nothing definied
Update instrument	ESC	I	A	n1	val				Update instrument with new value and redraw			
Redraw instrument			N	n1					Redraw entirely instument n1=1..4			
Send instrument value			S	n1					Send actual instrument value n1=1..4 to send buffer			
Delete instrument			D	n1	n2				The definition of the instrumets gets invalid. If the instrument was adjustable by touch, the touch area will be deleted, too. n2=0: Instrument stays visible; n2=1: Instrument is deleted completely			
User values - formatted string output												
User value color	ESC	F	I	vf	hf				Set color 1..32 for instrument user value fg=foreground; bg=background color			8,1
User value font	ESC	I	F	n1					Set font nr for instrument user value			5
User value zoom			Z	n1	n2				Set zoom factor for instrument user value: n1=X-Zoom 1x..8x; n2=Y-Zoom 1x..8x			1,1
User value additional hight/width			Y	n1	n2				n1=0..15: additional width left/right; n2=0..15: additional height top/bottom for instrument user value;			0, 0
User value angle			W	n1					Set writing angle for instrument user value: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°			0°
User values / scaling	ESC	I	X	n1	xx1	yy1	Format String	NUL	Define user value for instrument no=1..4. Output is right justified to x1,y1 Format String: "iv1=uservalue1;iv2=uservalue2" Assign two instrument values (iv1,iv2 =0..254) to user defined values max. range: 4 1/2 digits 19999 + decimal point ('.' or ',') + sign e.g. display "-123.4" for iv1=0 and "567.8" for iv2=100 Format String: "0=-123.4;100=567.8"			

EA eDIPTFT32-A: Macro commands											after reset
Command	**Codes**									**Remarks**	
Macro commands											
Run macro	ESC	M	N	no						Call the (normal) macro with the number no (max. 7 levels)	
Run touch macros			T	no						Call the touch macro with the number no (max. 7 levels)	
Run port macro			P	no						Call the port macro with the number no (max. 7 levels)	
Run bit macro			B	no						Call the bit macro with the number no (max. 7 levels)	
Run matrix macro			X	no						Call the matrix macro with the number (max. 7 levels)	
Run process macro			C	no						Call the process macro with the number (max. 7 levels)	
Run analogue macro			V	no						Call the analogue macro with the number no (max. 7 levels)	
Disable macros	ESC	M	L	type	n1	n2				Macros of the type'N','T','P','B','X','C' or 'V' (type 'A' = all macro types) are disabled from the number n1 to n2; i.e. no longer run when called.	
Enable macros			U	type	n1	n2				Macros of the type 'N','T','P','B','X','C' or 'V' (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			W							the current macro/image page is saved (when used in process macros)	
Restore macro/imagepage			R							the last saved macro/image page is restored	
Automatic (normal-) macro											
Macro with delay	ESC	M	G	n1	n2					Call the (normal) macro with the number n1 in n2/10s. Execution is stopped by commands (e.g. receipt or touch macros).	
Autom. macros once only			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			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			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 4) is defined (1=highest priority). The process 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			Z	no	zs					a new time zs in 1/10s is assigned to the macro process with the number no (1 to 4). if the time zs=0, execution is stopped.	
Stop macro processes			S	n1						All macro processes and animations are stopped with n1=0 and restarted with n1=1 in order, for example, to execute settings and outputs via the interface undisturbed	1

EA eDIPTFT32-A: Analogue input AIN1, AIN2 commands											after reset
Command	**Codes**									**Remarks**	
Commands for analogue inputs											
Calibration	ESC	V	@	ch	xx1					Calibration procedure is as follows: 1.) Apply defined voltage (3..5V) to AIN1 (channel1) or AIN2 (channel2) 2.) Run this command with channel information ch=1..2 and xx1=voltage value [mV] (16-Bit) e.g. 4.0V on AIN1; Command: '#V@1,4000;'	not calib rated
Enable/disable AIN scan	ESC	V	A	n1						n1=0 disables input scan for AIN1 and AIN2; n1=1 enable input scan	0
Send analog value			D	ch						Voltage in [mV] will be sent (to sendbuffer) for channel ch=1..2	
Limit for analog macro			K	ch	n1	n2	n3			Sets two limits for channel ch=1..2. n1=lower limit [mV/20]; n2=upper limit [mV/20]; n3=hysteresis [mV] Related to this limits serveral analogmacros can be started automatically.	0
Redefine analoguemacro (since V1.1)	ESC	V	M	n1	n2					Assign analoguemacrofunction n1=0..19 with analoguemacro number n2=0..255.	
Bargraph for AIN1/AIN2	ESC	V	B	ch	no					Assigns bargraph no=1..20 to analogue input ch=1..2 (it is possible to assign more than one bargraph to an anlogue input). Define start- endvalues (sv, ev) for bargraph in [mV/20] (see comand 'ESC B RLOU')	
Instrument for AIN1/AIN2			+	ch	no					Assigns instrument no=1..4 to analogue input ch=1..2 Define start- endvalues (sv, ev) for bargraph in [mV/20]	
Redraw bargraph			R	ch						Redraw all bar graphs defined for channel ch=1..2	
User values - Format text output											
User value color	ESC	F	V	ch	fg	bg				Set color for string output of channel ch=1..2; fg= foreground, bg= background color	8,1
User value Font		V	F	ch	n1					Set font n1 for channel ch=1..2	5
User value zoom			Z	ch	n1	n2				Set zoom factor for channel ch=1..2; n1=X-Zoom 1x..8x; n2=Y-Zoom 1x..8x	1,1
User value additional width/height			Y	ch	n1	n2				n1=0..15: additional width left/right; n2=0..15: additional height top/bottom for channel ch=1..2;	0, 0
User value angle			W	ch	n1					Set writing angle for channel ch=1..2; n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°;	0
User values / scaling	ESC	V	E	ch	format string ...				NUL	Set user value for channel ch=1..2. Format String: "mV1=uservalue1;mV2=uservalue2". 'NUL' ($00) = termination Assign two voltages (0..5000mV) to user defined values max. range: 4 1/2 digits 19999 + decimal point ('.' oder ',') + sign e.g. display for 2000 mV input should be "-123.45" and "0.00" for 1000mV Format String: "2000=-123.45;1000=0"	0 =0.00 5000 =5.00
Send user value	ESC	V	S	ch						This will send current voltage as formated string for channel ch=1..2 to sendbuffer	
Display on terminal			T	ch						Show formated string of channel ch=1..2 on termial window	
Display user value			G	ch	xx1	yy1				Show formated string of channel ch=1..2 at coordinate xx1,yy1	

EA eDIPTFT32-A: Commands for backlight, I/O-port and misc											after reset
Command	**Codes**									**Remarks**	
Backlight commands											
Illumination brightness	ESC	Y	H	n1						Set brightness of the LED illumination n1=0 to 100%.	100
Increase brightness			N							Increase brightness of the LED illumination (one step=1%)	
Decrease brightness			P							Decrease brightness of the LED illumination (one step=1%)	
Brightness changetime			Z	n1						Time n1=0..31 in 1/10sec for changing brightness from 0 to 100%	5
Illumination on/off			L	n1						LED n1=0: OFF; n1=1: ON; n1=2 to 255: LED switched ON for n1/10sec	1
Assign bar with backlight			B	no						Assign bar no=1..20 for changing brightness of the backlight	1
Asign instrument with backlight			+	n1						Brightness is connected to instrument 1..4.	1
Save parameter			@							Save the actual brightness and changetime for poweron to EEPROM	
Port commands											
Write output port	ESC	Y	W	n1	n2					n1=0: Set all 8 output ports in accordance with n2 (=8-bit binary value) n1=1..8: Reset output port n1 (n2=0); set (n2=1); invert (n2=2)	Ports 1-8=0
Read input port			R	n1						n1=0: Read all 8 input ports as 8-bit binary value (to sendbuffer) n1=1..8: Read input port <n1> (1=H-level=VDD, 0=L-level=GND)	
Port scan on/off			A	n1						The automatic scan of the input port is n1=0: deactivated; n1=1: activated	1
Invert input port			I	n1						The input port is n1=0: normal; n1=1: evaluated inverted	0
Matrix keyboard			M	n1	n2	n3				Specifies an external matrix keyboard at the inputs and outputs. n1=number of inputs (1..8); n2=number of outputs (0..8); n3=debouncing (0..7)	0
Redefine input bitmacro	ESC	Y	D	n1	n2	n3				input port n1=1..8 is assigned by falling edge n2=0 to new BitMacro number n3=0..255 input port n1=1..8 is assigned by rising edge n2=1 to new BitMacro number n3=0..255	
Redefine matrixmacro for keys			X	n1	n2					Assign keynumber n1=1..65 with matrixmacro number n2=0..255 After release the key n1=0 run matrixmacro number n2=0..255	
Other commands											
Define color	ESC	F	P	no	R5	G6	B5			Set a new RGB value for color no. n1=1..32 (R5:Bit7..3; G6:Bit7..2; B5:Bit7..3)	
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:ON for n1/10s	OFF
String table code	ESC	S	T	n1						n1=0: no use of internal strings n1>0: after code n1 appears following codes are internal string numbers (since V1.2)	0
Send bytes	ESC	S	B	num	data...					num (=1 to 255) bytes are sent to the sendbuffer data... = num Bytes. In the source text of the macro programming, the number num must not be specified. This is counted by the ediptftcompiler and entered.	
Send version			V							The version is sent as a string to sendbuffer e.g. "EA eDIPTFT32-A V1.1 Rev.A TP+"	
Send projectname			J							The macro-projectname is sent as a string to the sendbuffer e.g. "init / delivery state"	
Send internal infos			I							Internal information about the edip is sent to the sendbuffer.	

TOUCH PANEL

The Version EA eDIPTFT32-ATP is shipped with an analog, resistive touch panel. Up to 60 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. 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.

EA eDIPTFT32-A: Commands for the touch panel														after reset
Command	**Codes**												**Remarks**	
Touch presets														
Touch bordercolors	ESC	F	E	n1	n2	n3	s1	s2	s3				Set the colors (0..32) for touch borders (ESC AT AK). n=normal; s=selected; 1=frame outside; 2=frame inside; 3=filling	8,1,2 8,1,7
Touch borderform		A	E	n1	n2								n1=1..255 border number; n1=0 no border; n2=angle 0=0°; 1=90°; 2=180°; 3=270°	1,0
Touch button colors	ESC	F	C	nf	nb	sf	sb						Set the colors (0..32) for monochrome touch buttons (ESC AU AJ). n=normal; s=selected; f=foreground; b=background	8,1 8,1
Touch button number		A	C	n1	n2	n3	n4						n1=0..255 button number; n2=button angle; n3=X-Zoom 1..8; n4=Y-Zoom 1..8	1,0,1,1
Radio group for switches	ESC	A	R	n1									n1=0: newly defined switches do not belong to a group. n1=1 to 255: newly defined switches belong to the group with the number n1. Only 1 switch in a group is active at any one time; all the others are deactivated. In the case of a switch in a group, only the down code is applicable. the up code is ignored.	0
Label font presets														
Font color	ESC	F	A	nf	sf								Color for touch labeling. nf=normal fontcolor; sf= fontcolor for selection	8,1
Label font	ESC	A	F	n1									Set font with the number n1 for touch key label	5
Label zoom factor			Z	n1	n2								n1 = X-zoom factor (1x to 8x); n2 = Y-zoom factor (1x to 8x)	1,1
Additional width/height			Y	n1	n2								n1=0..15: additional width left/right; n2=0..15: additional height top/bottom	0,0
Label angle			W	n1									Label output angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°	0
Offset for selected label			O	n1	n2								n1=X-offset; n2=Y-offset; n1,n2=0..7 (add +8 for negative direction)	0, 0
Define touch areas														
Define touch key	ESC	A	T	xx1	yy1	xx2	yy2	dow Cod	up Cod	text ...	NUL		'T': The area from xx1,yy1 to xx2,yy2 is defined as a key 'K': The area from xx1,yy1 to xx2,yy2 is defined as a switch 'U': The actual button is loaded to xx1,yy2 and defined as a key 'J': The actual button is loaded to xx1,yy2 and defined as a switch 'down code':(1-255) return/touchmacro when key pressed. 'up code': (1-255) return/touchmacro when key released. (down/up code = 0 press/release not reported). 'text': this is a string that is placed in the key with the current touch font. The first character determines the alignment of the text (C=centered, L=left justified, R=right justified). Multiline texts are separated with the character '|' ($7C, dec: 124); optional: after the character '~' ($7E, dec: 126) you can write a 2nd text for a selected touch key/switch e.g. "LED|on~LED|off" 'nul': ($00) = end of string	
			U	xx1	yy1	dow Cod	up Cod	text ...	NUL					
Define touch switch (status of the switch toggles after each contact)	ESC	A	K	xx1	yy1	xx2	yy2	dow Cod	up Cod	text ...	NUL			
			J	xx1	yy1	dow Cod	up Cod	text ...	NUL					
Define drawing area	ESC	A	D	xx1	yy1	xx2	yy2	n1	fg				A drawing area is defined. You can then draw with a line width of n1 and color fg 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 bargraph with number n1 is defined for input by touch panel.	
Set instrument by touch	ESC	A	+	n1									The instrument with number n1 is defined for input by touch panel.	
Global settings														
Touch query on/off	ESC	A	A	n1									Touch query is deactivated (n1=0) or activated (n1=1)	1
Touch key response	ESC	A	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
Send bar value on/off	ESC	A	Q	n1									Automatic transmission of a new bar graph / instrument value by touch input is n1=0: deactivated; n1=1: is placed in the sendbuffer once at the end of input n1=2: changes are placed continious in the sendbuffer during input	1
Other functions														
Invert touch key	ESC	A	N	code									The touch key with the assigned return code is inverted manually	
Set touch switch			P	code	n1								The status of the switch is changed by means of a command (n1=0=off; n1=1=on)	
Query touch switch			X	code									The status of the switch with the return code (off=0; on=1) is placed in the sendbuffer	
Query radio group			G	n1									down code of the activated switch from the radio group n1 is placed in the sendbuffer	
Delete touch area	ESC	A	L	code	n1								The touch area with the return code (code=0: all touch areas) is removed from the touch query. n1=0 the area remains visible on the display; n1=1, the area is deleted.	
			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	

TOUICH 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:

1a. Send Command 'ESC A@' or

1b. Touch the touch panel at power-on and keep it depressed. After the message “touch adjustment ?” appears, release the touch panel. Touch the touch panel again within a second for at least a second.

2. Follow the instructions for adjustment (press the 2 points upper left and lower right).

RESPONSE OF THE EA EDIPTFT32-A VIA 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, alternativly pin 20 ’SBUF’ shows with LO-signal that data is ready to transmit.

Responses of the EA eDIPTFT32-A								
Id		num	data					Remarks
automatic responses (placed into sendbuffer)								
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 onl transmitted if no touch macro with the number code is defined !
ESC	B	2	no	value				When a bargraph is set by touch, the current value of the bar no is transmitted. Transmission of the bar balue must be activated (see the 'ESC A Q n1' command).
ESC	F	2	no	value				When a instrument is set by touch, the current value of the instrument no is transmitted. Transmission of the intstrument must be activated (see the 'ESC A Q n1' command).
ESC	P	1	value					After the input port is changed, the new 8-bit value is transmitted. The automatic port scan must be activated. See the 'E A n1' command. It is only transmitted when there is no corresponding port/bit macro defined !
ESC	M	1	no					When a keystroke of the external matrix keyboard is detected, the newly pressed key number no is transmitted. Only transmitted if no corresponding matrix macro is defined !
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; type=1 is touch; type=2 is drag with free touch area at the coordinates xx1, yy1
Response only when requested by command (placed into sendbuffer)								
ESC	B	2	no	value				After the 'ESC B S n1' command, the current value of the bar with the number no is transmitted.
ESC	F	2	no	value				After the 'ESC I S n1' command, the current value of the instrument with the number no is transmitted.
ESC	X	2	code	value				After the 'ESC A X' command, the current status (value=0 or 1) of the touch switch code is transmitted.
ESC	G	2	no	code				After the 'ESC A G nR' command, the code of the active touch switch in the radio group no is sent.
ESC	Y	2	no	value				After the 'ESC Y R' command, the requested input port is transmitted. no=0: value is an 8-bit binary value of all 8 inputs. no=1..8: value is 0 or 1 depending on the status of the input no
ESC	D	3	ch	LOval	HIval			After the 'ESC V D ch' command, the requested voltage of channel ch=1..2 will be sent (value = 0..5000mV)
ESC	W	num	ch	scaled ASCII string...				After the 'ESC V S ch' command, the requested voltage of channel ch=1..2 will be set as scaled ASCII characters (length string = num-1).
ESC	V	num	version string...					After the 'ESC S V' command, the version of the edip firmware is transmitted as a string e.g. "EA eDIPTFT43-A V1.0 Rev.A TP+"
ESC	J	num	projectname string...					After the 'ESC S J' command, the macro-projectname is transmitted. e.g. "init / delivery state"
ESC	I	21	X-dots, Y-dots, Version, Touchinfo, CRC-ROM, CRC-ROMsoll DF in KB, CRC-DF, CRC-DFsoll, DFlen					after the 'ESC S I' command, internal information is sent by eDIP (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 DFlen: number of user bytes in data flash memory (3 Bytes: LO-, MID- HI-Byte)
Responses without length specification (num)								
ESC	U	L	xx1	yy1	image data... (G16-FORMAT)			after the 'ESC UH....' command, a hard copy is sent in G16-format. xx1,yy1 = Start coordinates of the hard copy (upper left corner)

PRELOADED FONTS

As standard, there are 3 monospaced, 3 proportional character sets and 2 large digit fonts integrated. 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. Macro programming permits further fonts to be integrated. All kinds of fonts can be converted from True-Type Fonts by using using the LCD toolkit/ eDIPTFTcompiler (the USB Evaluation Board EA 9777-2USB is required).

*) refer to web: http://www.lcd-module.com/products/edip.html

+ 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										¥	
$90 (dez: 144)					ä					¥	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	∞	ø	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: big numbers BigZif50

+ 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	:					

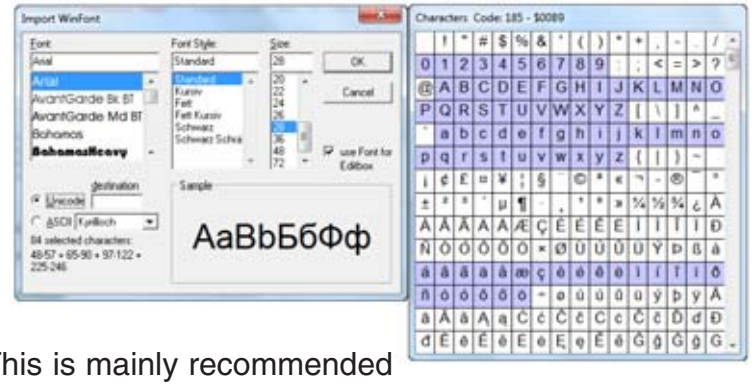
Font 8: big numbers BigZif100

This hard copy shows all the fonts with which the product is shipped

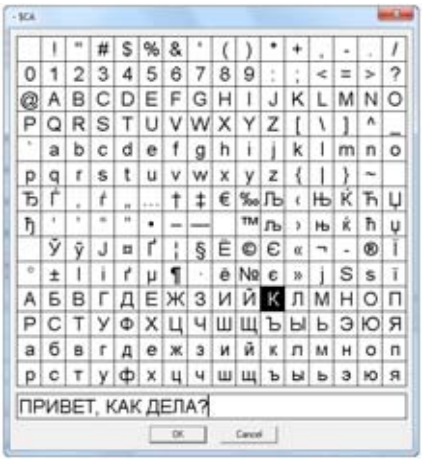
ADDITIONAL FONTS

Up to 256 fonts á 16 pages can be loaded into the internal DataFlash.

COMPILER OPTION "WinFont:"



It is possible to raster TrueType-Fonts in different sizes witch can be used. A double-click to the fontname within the KitEditor opens the font selection box.
To simplify the use of fonts, there is the possibilty of a 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}).



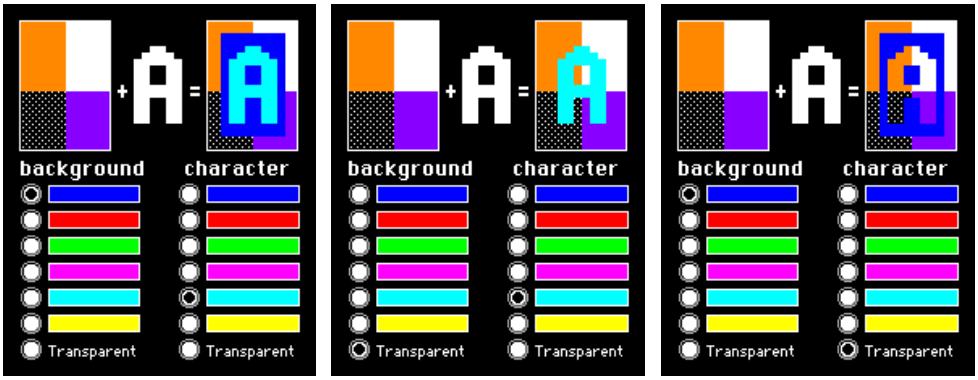
COMPILER OPTION "Font:"

Following font formats can be used:
- FXT: Textfont as used by eDIP240/320 and KIT series
- G16: internal eDIPTFT format (with this format it is possible to user color fonts)

65,536 COLORS

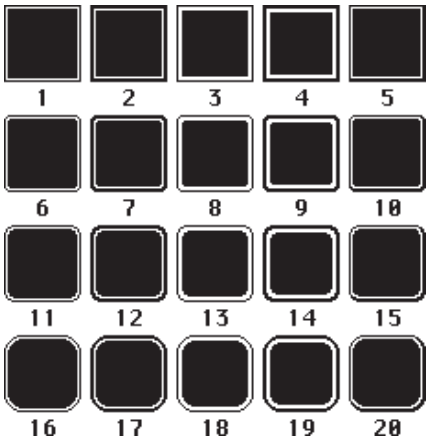
EA eDIPTFT32-A is able to work with 65,536 colors for true-color pictures/icons and animations. For an easy use there exists a color palette with 32 entrys (16 colors are predefined after PowerOn). This color palette can be redefined at any time without changing the content of the display (command: ESC FP no R G B). To use a color for text and graphic functions you set only a number between 1..32. The dummy color number 255 means that the actually color is not changed e.g you want only to change the foreground- and not the background color. The color number 0=transparent is special and can be used for background of character e.g. that means that for placing a character no rectangular field will be deleted around the character itself. The sensless combination of transparent background and transparent foreground is used to invert all dots (=complementary). Two times inverted will end same as action was started (original drawing is restored).

Color	R	G	B
1	0	0	0
2	0	0	255
3	255	0	0
4	0	255	0
5	255	0	255
6	0	255	255
7	255	255	0
8	255	255	255
9	111	111	111
10	255	143	0
11	143	0	255
12	255	0	143
13	0	255	143
14	143	255	0
15	0	143	255
16	175	175	175

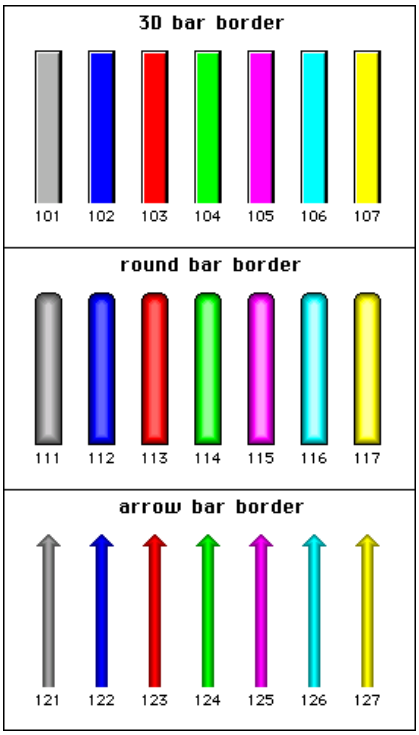


some examples to show the letter 'A' onto a fixed background

BORDERS, KEY STYLES AND BARGRAPH

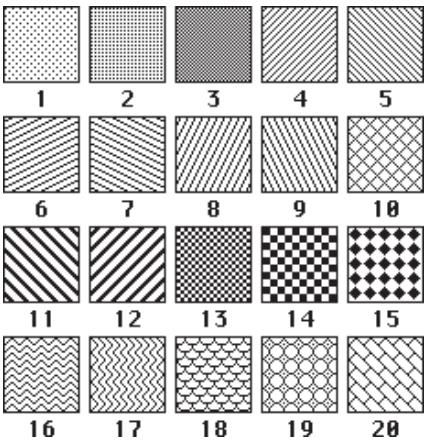


The eDIP is shipped with 20 predefined border (no:1..20) for the commands draw box frame and draw touchkeys.There are also three special borders in various colors for using with bargraph commands (no:101..107, 111..117 and 121..127). All of them can be used in various sizes via coordinates. The frames 1..20 are split into 3 segments: the outer frame, inner frame and filling. Each segment will get an individual colour for normal and for selected state. This will give the user the opportunity, when touching a field, the individual part of the key will be inverted only.



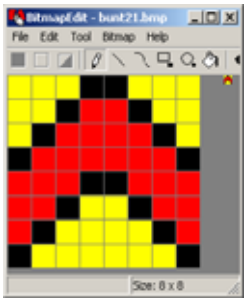
FILL PATTERNS

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. The eDIP is shipped with 20 predefined fill patterns.



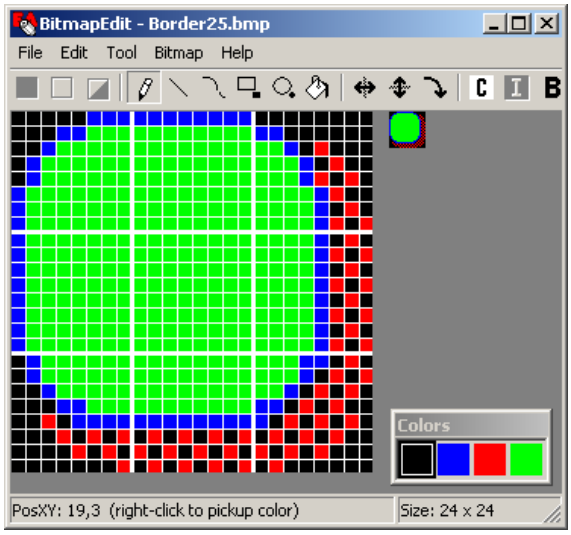
Define own pattern:

With the LCD-tools it is possible to define new pattern (=bitmaps with exactly 8x8 dots).(Compileroption "Pattern:"). The foreground- and backgound color can be set for monochrome pattern (as the 20 preloaded pattern) incl. transparency. It is also possible to define full colored pattern.With the LCD-tools some sample pattern has been installed (see folder 'Pattern').

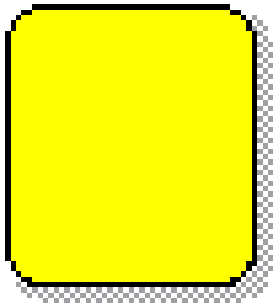


DEFINE OWN BORDER

With the LCD-tools it is possible to define new borders (Compileroption "Border:"). Each of these new border is a bitmap with exactly 24x24 dots (9 segments with 8x8 dots: 4x edge, 4x middle part, 1x filling).



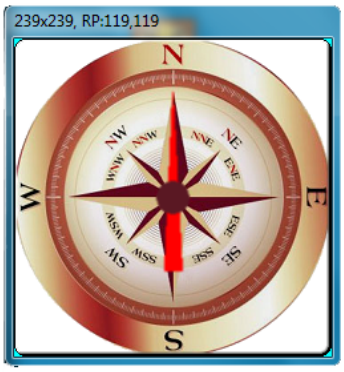
Scaling for bigger touchkeys/ frames will be done by repetition of these 8x8 dot segments. This makes it necessary to keep the 8x8 size in every case. If 4-color bitmaps are used (as the preloaded border 1..20) the color can be set individually (the first color is always transparent and is not used by the eDIP). It is also possible to define full colored border (as the preloaded border for bargraphs 101...127).With the LCD-tools some sample border has been installed (see folder 'Bitmaps\Color\Border').



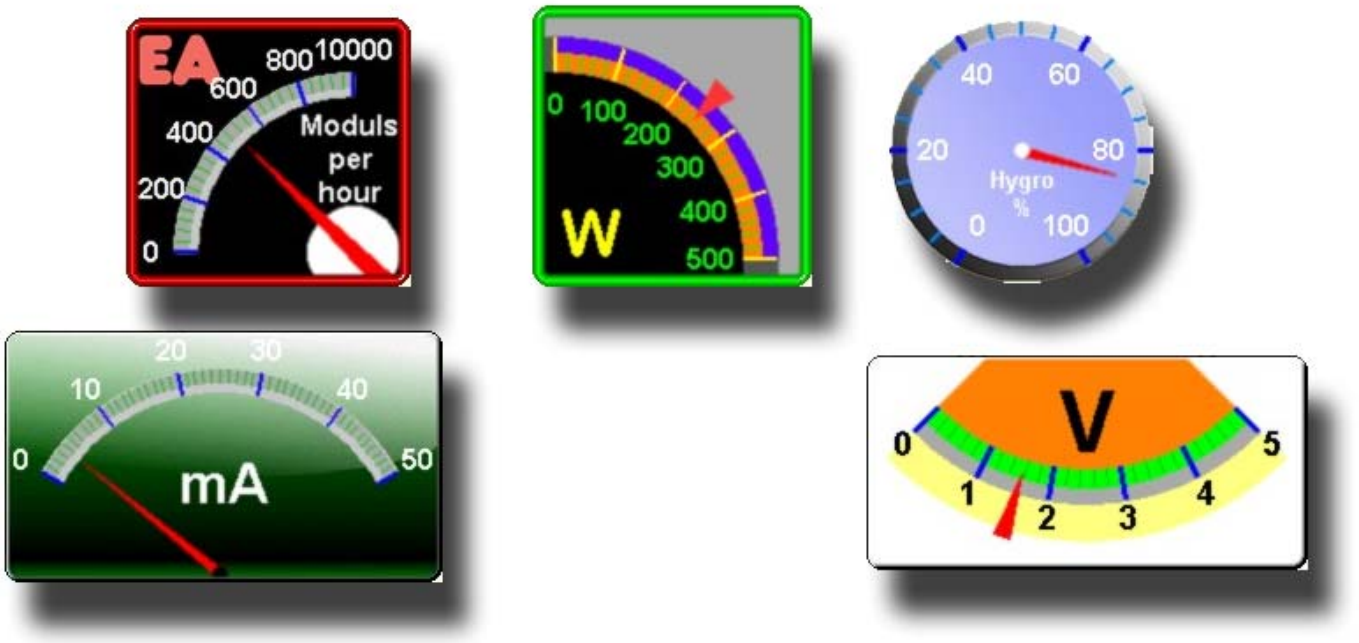
border25: 50x56 Dot size

ROTARY AND POINTER INSTRUMENTS

With the help of the LCD-Tools it is possible to include instruments (Compiler command: `Instrument: 4, <instrument.i16>`). Performing a double click in the KitEditor on the instrument file opens the instrument editor.
The instruments are supported by convinient commands ('ESC I..'). For example the instruments are connectable to an analog input. In addition they are configurable by touch.



Some instrument examples:
With the LCD-tools some sample instruments have been installed (see folder 'instruments\').

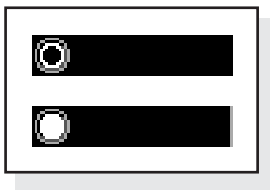


BUTTONS AS KEYS

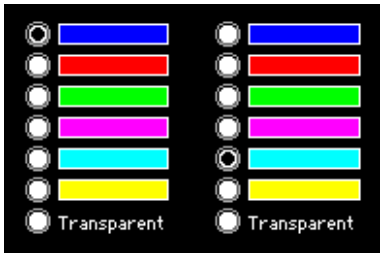
Apart from the border types, which are infinitely scalable, it is also possible to use bitmaps as touch keys or touch switches (Compilerotion "Button:"). A button always consists of two Bitmaps of equal size (one bitmap to display the touch key in its normal state and another for when it is pressed). The active area of the touch key automatically results from the size of the button bitmaps.

SWITCHES IN GROUPS (RADIO GROUPS)

Touch switches 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 n1' command). If a touch switch in the group 'n1' is switched on, all the other touch switches in this group are automatically switched off. Only one switch is ever on (see table on page 18).



RadioBlack75x15_0.bmp
RadioBlack75x15_1.bmp



two radio grooups with touch switches

CREATING INDIVIDUAL MACROS AND IMAGES

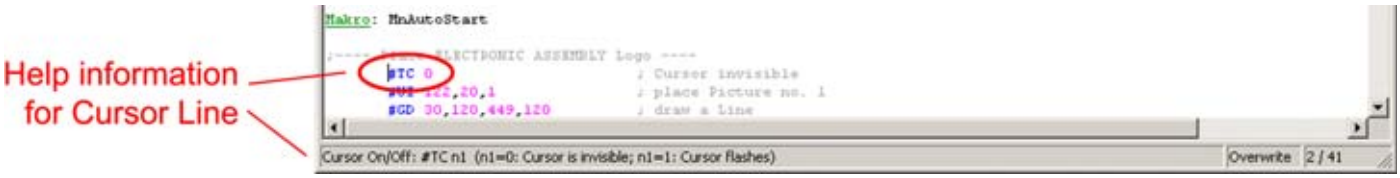
To create your own fonts, images, animations and macros you need the following:

- To connect the display to the PC, you need the EA 9777-2USB 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 5).
- ELECTRONIC ASSEMBLY LCD-Tools*), which contains a kiteditor, bitmapeditor, ediptftcompiler, 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 eDIPTFT compiler using F5. This creates a file called DEMO.DF. If an EA 9777-2USB 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 flash memory. You can send the created macrofile *.DF with any other system to the EA eDIPTFT32-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 eDIPTFT32-A.

KIT-EDITOR HELP (ELECTRONIC ASSEMBLY LCD TOOLS)

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.



*) Web: http://www.lcd-module.de/deu/dip/edip.htm

IMAGES

To save transfer time via serial interface, it is possible to store up to 256 bitmaps á 16 pages into internal dataflash (Compileroption "Picture:"). Following image file-formats can be used:
- BMP: Windows Bitmap with 1-, 4-, 8-, 16-, 24-, 32-BIT colordepth incl. RLE.
- GIF: Graphics Interchange Format incl. optionally transparency
- JPG: JPEG Compressed Images
- TGA: TARGA Images with 8-, 16-, 24-, 32-BIT colordepth incl. RLE and transparency.
- PNG: Portable Network Graphics incl. colour map, gray scale and transparency
- G16: internal eDIPTFT format, incl. RLE and transparency

All pictures are converted into internal G16 format with RLE encoding (saves memory). Too big pictures are resized proportional (Compileroption "MaxSize:"). It is also possible to reduce the colordepth (Compileroption "MaxColorDepth:"). One color can be defined as transparent (Compileroption "MakeTransparent:“) The internal pictures can be used with the command "ESC U I" via serial interface or from a macro. The foreground- and backgound color can be set for monochrome pictures incl. transparency.

ANIMATIONS

It is possible to store up to 256 animations á 16 pages into internal dataflash. (Compileroption "Animation:"). Following image file-formats can be used:
- GIF: animated GIF (only identically transparent areas, transparency can be switched off).
- G16: internal animated eDIPTFT format
- two or more single bitmaps (BMP, GIF, JPG, TGA, PNG, G16) e.g. two bitmaps for blinking

Note that max. 4 animations (animationprocesses) can run at the same time . The animations are selfrunning pictures, but you can use the animations manually too. The foreground- and backgound color can be set for monochrome animations.

PATTERN

Patterns are used to fill a box, a bargraph or to draw a line. It is possible to store up to 256 pattern á 16 pages into internal dataflash (Compileroption "Pattern:").
Each bitmap (BMP, GIF, JPG, TGA, PNG, G16) with a size of 8x8 dots can be imported as a pattern. The foreground- and backgound color can be set for monochrome pattern incl. transparency.

BORDER / BARGRAPH

A border can be scaled and is used for rectangles, bargraphs and touch keys/switches. It is possible to store up to 256 border á 16 pages into internal dataflash (Compileroption "Border:")
Each bitmap (BMP, GIF, JPG, TGA, PNG, G16) with a size of 24x24 dots can be imported as a border. The transparency of GIF,TGA and G16 bitmaps is used by the EA eDIP. It is possible to change the colors for 4-color borders, the first color ist always transparent and is not used by the EA eDIP. When used for a touch key/switch a second border can be loaded witch will be used if the touch key/ switch is pressed.

IMAGES AS TOUCHKEYS (BUTTONS)

It is possible to store up to 256 touchkeys/buttons á 16 pages into internal dataflash. (Compileroption "Button:").
A button consists of one or two images with the same size (BMP, GIF, JPG, TGA, PNG, G16). The transparency of GIF,TGA, PNG and G16 bitmaps is used by the EA eDIP and should be identical. The first bitmap is used when the touch key/switch is released and the second bitmap is used if the touch key/ switch is pressed.

MACROS

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 `Macro:`
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.

Touch macro `TouchMacro:`
Started when you touch/release a touch field (only in versions with a touch panel - TP) or issue an 'ESC MT xx' command.

Bit macro `BitMacro:`

will be started by a single line IN 1..8 (bit) will change or by command 'ESC MB xx'. Bit- Macro 1..8 are good for falling edge and Bit Macro 9..16 are good for rising edge at input 1..8. It is possible to change the assignment between Bitmacro and intput with command 'ESC YD n1 n2 n3' (see page 17).

Port macro `PortMacro:`
These are started when voltage (binary) is applied to IN 1..8 or by command 'ESC MP xx'.

Matrix macro `MatrixMacro:`

Matrix Macro 1..64: start when keypressed or by command 'ESC MX xx'. Matrix Macro 0: start after release of key or by command. It is possible to change the assignment between keynumber and Matrixmacro with command 'ESC YX n1 n2 n3' (see page 17).

Analogue macro `AnalogMacro:`

will start whenever voltage changes or limit exceeds or by command 'ESC MV xx'. See table at the right: It is possible to change the assignment between analoguemacrofunction and Analoguemacronumber with command 'ESC VM n1 n2' (see page 15).

Analogue Macro		
Macro No.		Macro starts at
AIN1	AIN2	
0	10	every change of input voltage
1	11	falling input voltage
2	12	rising input voltage
3	13	below lower limit
4	14	above lower limit
5	15	below upper limit
6	16	above upper limit
7	17	outside of both limits
8	18	inside of both limits
9	19	lower than other channel

Process Makro `ProcessMacro:`

automatic start at fixed periode (0.1s up to 25s) or by command 'ESC MC xx'. Up to 4 individual process may be defined by command 'ESC MD ..'. These Process Makro will never be stopped by other commands or activities.

Power-on-macro `PowerOnMacro:`
Started after power-on. You can switch off the cursor and define an opening screen, for example.

Reset-macro `ResetMacro:`
Started after an external reset (low level at pin 5).

Watchdog-macro `WatchdogMacro:`
Started after a fault/error (e.g. failure).

Brown-out-macro `BrownOutMacro:`
Started after a voltage drop under 3.0V (typ.).

Important: If a continuous loop is programmed in a power-on, reset, watchdog or brown-out macro, the display can no longer be addressed. 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.

MACRO PAGES (MULTILINGUAL CAPABILITY)

There are 16 complete macro sets available as well as the internal images and fonts. 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] <BIER.BMP>
PICTURE: 100[1] <BEER.BMP>
PICTURE: 100[2] <BIRRA.BMP>

MACRO: 2[0]                    ; SAME AS "MACRO: 2"
         #FZ 3,1
         #ZL 25,0 "DEUTSCH   "
         #UI 0,20, 100

MACRO: 2[1]                    ; ENGLISH
         #FZ 3,1
         #ZL 25,0 "ENGLISH   "
         #UI 0,20, 100

MACRO: 2[2]                    ; ITALIAN
         #FZ 3,1
         #ZL 25,0 "ITALIAN   "
         #UI 0,20, 100
```

If there is only the need of handling strings differently, stringtables might be used. Depending on the active macropage (ESC M K n1) the desired string is called. Please check the following example:

```
STRINGCODE=$01

STRING: 100[0] "HALLO WELT    "
STRING: 100[1] "HELLO WORLD   "
STRING: 100[2] "CIAO A TUTTI  "

MACRO: 1
  #ST StringCode
  #ZL 10,5, StringCode, 100
```

WRITE PROTECTION FOR MACRO PROGRAMMING AND FONTS

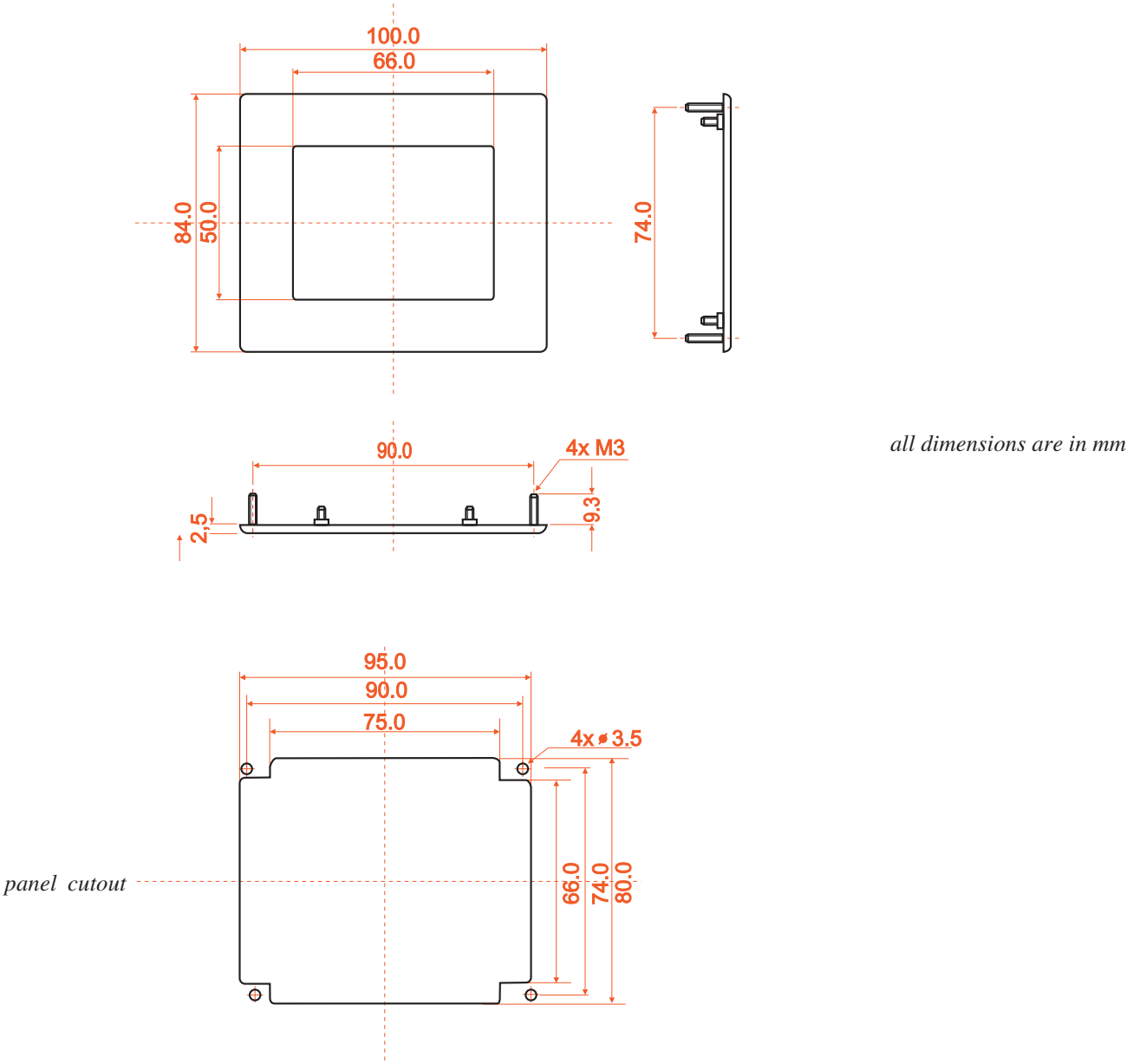
A LO level at pin 19 (WP) prevents the macros, images and fonts in the data flash memory from being overwritten inadvertently (so it is highly recommended !).

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		3.2	3.3 / 5.0	5.1	V
Input Low Voltage		-0.5		0.3*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
Output Low Voltage				0.7	V
Output High Voltage	VDD=5V VDD=3.3V	4.2 2.4			V
Brightness (white)	w./o. Touch		700		cd/m²
	with Touch		550		cd/m²
Output Current	OUT1..8			10	mA
Power Supply Backlight on (100%)	VDD=5V VDD=3.3V		120 160		mA
Power Supply Backlight off (0%)	VDD=5V VDD=3.3V		37 25		mA

MOUNTING BEZEL EA 0FP322-32SW

As accessory we deliver an optional black anodized mounting bezel. The mounting clips are included in the supplied EA eDIPTFT32-A(TP).



all dimensions are in mm

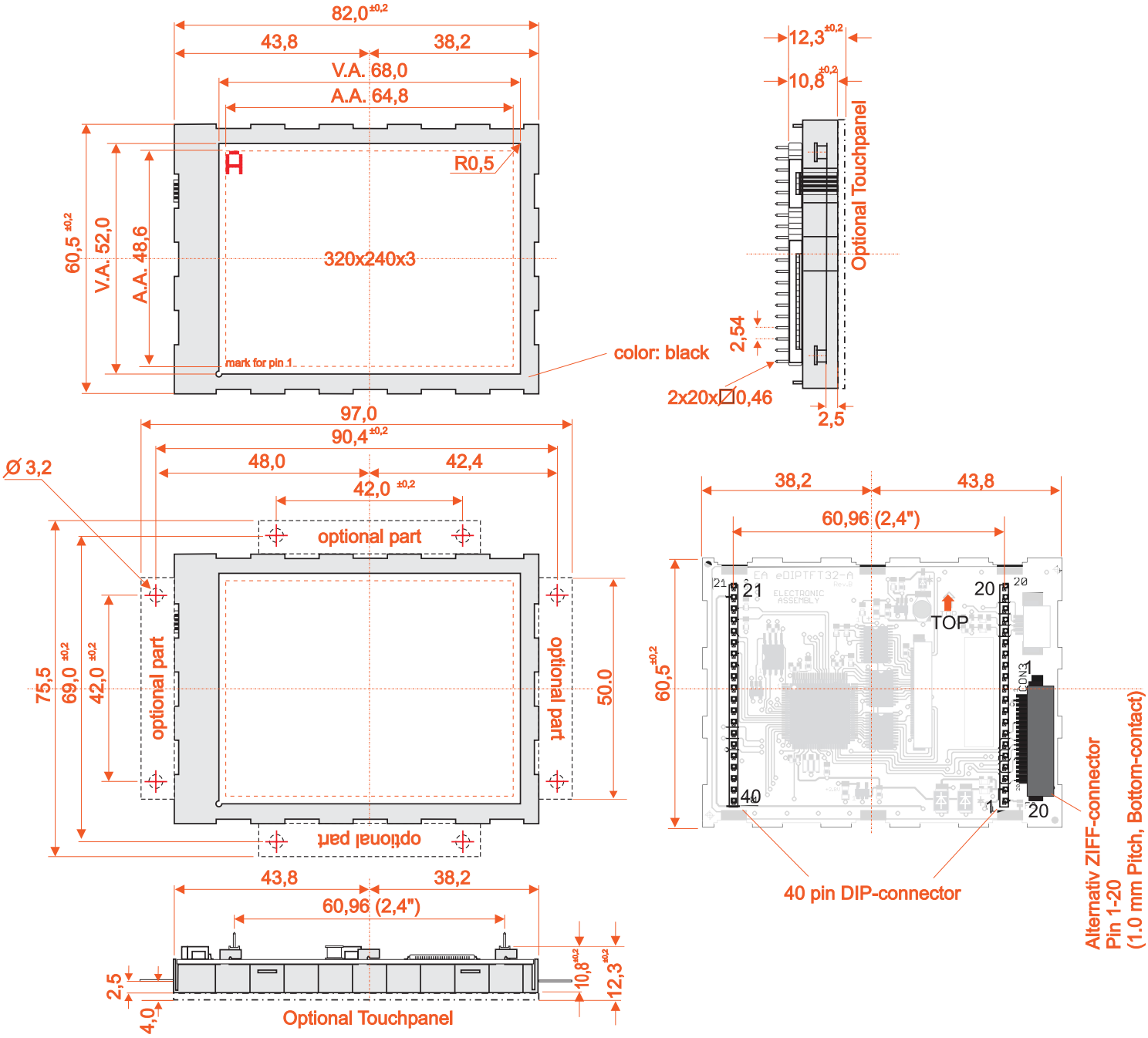
NOTES ON HANDLING AND OPERATION

ATTENTION

handling precautions!

- 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.

DIMENSIONS



all dimensions are in mm

Note:
LC displays are generally not suited to wave or reflow soldering.
Temperatures of over 80°C can cause lasting damage.
Two mounting clips are included.