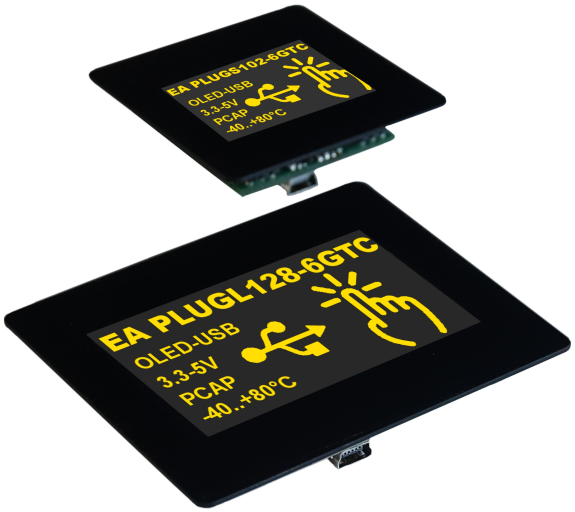
THE DATASHEET OF

EA PLUGL128-6GTC

DISPLAY INDUSTRIAL SOLUTIONS VISIONS

Issue 12.2023

EA PLUG-Series



TECHNICAL DATA

- WITH CAPACITIVE TOUCH PANEL
- 4 SERIAL INTERFACES USB, RS232, SPI, I²C
- 8 DIGITAL, FREELY DEFINABLE I/Os BUILT-IN
- 2 ANALOG INPUTS/ 1 ANALOG OUTPUT
- 8 BUILT-IN FONTS
- POSITIONING ACCURATE TO THE PIXEL WITH ALL FUNCTIONS
- SCREENSAVER MODES
- UP TO 256 PICTURES INTERNAL STORED
- UP TO 256 MACROS PROGRAMMABLE
- MIX TEXT AND GRAPHIC, FLASHING ATTRIBUTE: ON/OFF/INVERT
- CHANGE DISPLAY BRIGHTNESS BY SOFTWARE

ORDERING CODES

2,9" OLED WITH USB AND TOUCHPANEL	**EA PLUGL128-6GTC**
...WITH SCREW TERMINAL AND CONNECTOR	**EA PLUGL128-6GTCZ**
1,7" OLED WITH USB AND TOUCHPANEL	**EA PLUGS102-6GTC**
...WITH SCREW TERMINAL AND CONNECTOR	**EA PLUGS102-6GTCZ**

ACCESSORIES

USB CABLE MINI (~1m)	**EA KUSB-MINI**
IDC CABLE 25cm 1 CONNECTOR (PLUGL128-6)	**EA KB-126**
... WITH 2 CONNECTOR	**EA KB-226**
IDC CABLE 25cm 1 CONNECTOR (PLUGS102-6)	**EA KB-120**
... WITH 2 CONNECTOR	**EA KB-220**

INDIVIDUAL LENGTH ON REQUEST

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

Table of contents

REVISION HISTORY

EA PLUG-Series Firmware

Date	Version	Info
06.02.2019	Version 1.1	Bugs: Formated strings are displayed too large by one zoom level Partly missing reset of protocol send buffer length Internal: Change flash routine for testmacro
07.12.2018	Version 1.0	First release

Datasheet

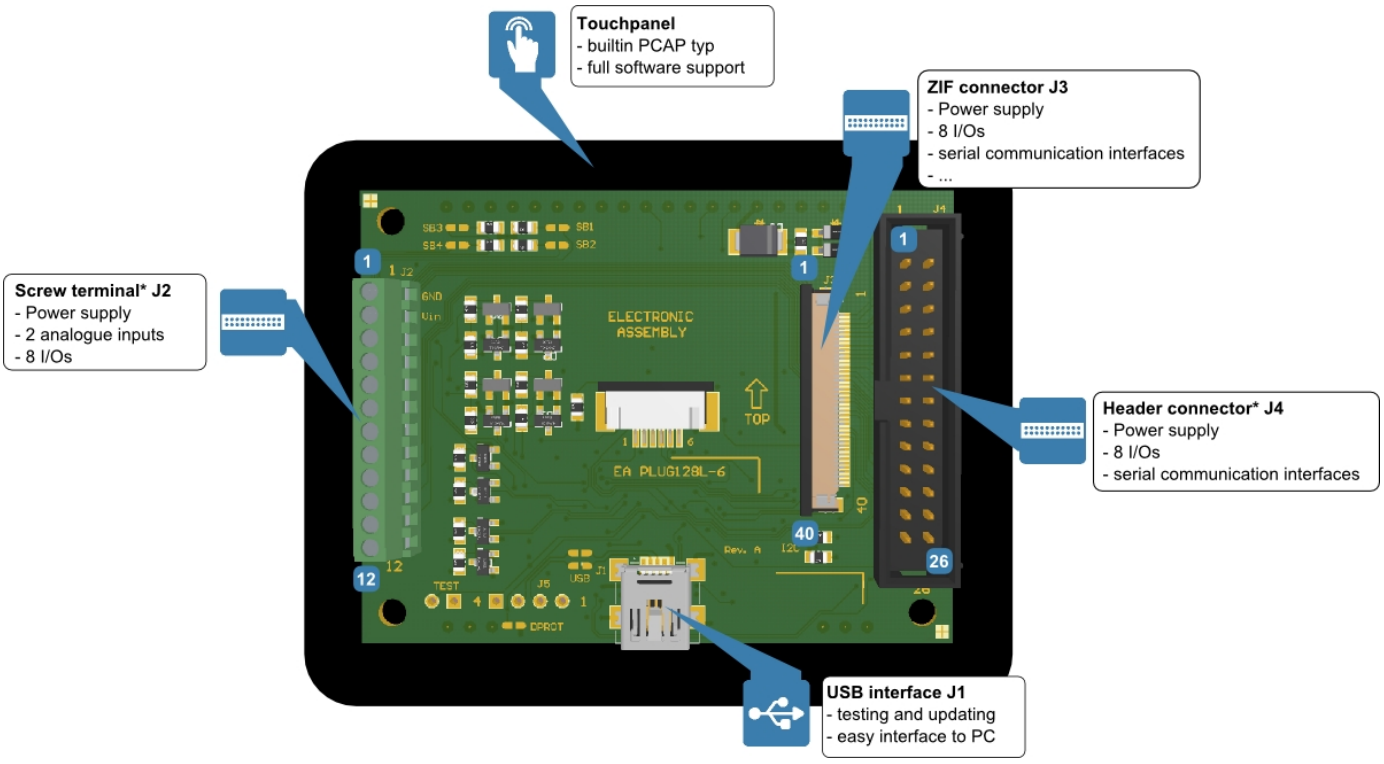
Date	Version	Info
05.2019		First release Adding cable and connector accessories

GENERAL

The EA PLUG-Series provides a simple graphical display with the built-in instruction set.
In addition to various built-in fonts, which can be used pixel-perfect, it also offers a whole range of sophisticated graphics functions. The display is immediately ready for operation with 3.3V ... 5V. It is controlled via one of the four built-in serial interfaces RS232, SPI, I²C or USB.
Programming is via high-level language-like graphics commands; the time-consuming programming of character sets and graphics routines is completely eliminated here. The simple use of this display with touch panel drastically shortens the development time.

HARDWARE

The EA PLUG-Series consists of a OLED display, which is dimmable using software commands. The most easy way to bring it into operation is via USB. The module is designed to work with 3.3~5V VDD. Furthermore serial data transfer is possible through RS232, SPI and I²C. For simple control tasks, the module has 8 freely usable I/Os, 2 analog inputs, one PWM and one analog output.



** Connectors are only included in hardware version Z (e.g. EA PLUGL128G-6TCZ)*

PIN ASSIGNMENT ZIF CONNECTOR (J3)

Pin	Symbol	I/O	Description	
1	GND		Ground 0V	
2	V_{in}		Power Supply 3.3~5V	
3	$\overline{RES}$	I	$\overline{Reset}$	internal pull-up (10kΩ), low active
4	$\overline{CS}$	I	SPI: Chip Select	low active
5	MOSI	I	SPI: MOSI	
6	MISO	0	SPI: MISO	
7	CLK	I	SPI: Clock	
8	RxD	I	RS232: Receive Data	3.3V level, internal Pull-Up (1MΩ)
9	TxD	O	RS232: Transmit Data	3.3V level
10	DE	O	RS485: Transmit enable	3.3V level
11	SDA	I/O	I²C: Serial Data	internal pull-up: (4,7kΩ)
12	SCL	I	I²C: Serial CLK	internal pull-up: (4,7kΩ)
13	SBUF Test	O I	Low: sendbuffer contains data PowerOn Low: testmode enabled	internal pull-up (10kΩ)
14	DAC	O	Analog output	0~3.1V
15	AIN1	I	Analog input 1	0~3.1V
16	AIN2	I	Analog input 2	0~3.1V
17	I/O 1	I/O	Digital in- or output	High Power Output
18	I/O 2	I/O	Digital in- or output	High Power Output
19	I/O 3	I/O	Digital in- or output	High Power Output
20	I/O 4	I/O	Digital in- or output	High Power Output
21	I/O 5	I/O	Digital in- or output / PWM	Low Power Output
22	I/O 6	I/O	Digital in- or output	Low Power Output
23	I/O 7	I/O	Digital in- or output	Low Power Output
24	I/O 8	I/O	Digital in- or output	Low Power Output
25	V_{out}		Power output (max. 3.1V)	for external periphery (max. 100 mA)
26	GND		Ground 0V	
27	d.n.c		internally connected	
28	d.n.c		internally connected	
29	SPEAK	O	Speaker output	
30	SPEAK POW	O	Speaker output (Power)	
31	d.n.c		internally connected	
32	d.n.c		internally connected	
33	n.c.			
34	n.c.			
35	n.c.			
36	n.c.			
37	n.c.			
38	n.c.			
39	d.n.c		internally connected	
40	GND			

PIN ASSIGNMENT SCREW TERMINAL[1] (J2)

Pin	Symbol	I/O	Description	
1	GND		Ground 0V	
2	V_{in}		Power Supply 3.3~5V	
3	AIN1	I	Analog input 1	0~3.1V
4	AIN2	I	Analog input 2	0~3.1V
5	I/O 1	I/O	Digital in- or output	High Power Output
6	I/O 2	I/O	Digital in- or output	High Power Output
7	I/O 3	I/O	Digital in- or output	High Power Output
8	I/O 4	I/O	Digital in- or output	High Power Output
9	I/O 5	I/O	Digital in- or output / PWM output	Low Power Output
10	I/O 6	I/O	Digital in- or output	Low Power Output
11	I/O 7	I/O	Digital in- or output	Low Power Output
12	I/O 8	I/O	Digital in- or output	Low Power Output

1) Mounted in hardware version Z only (EA PLUGL128-6GTCZ or EA PLUGS102-6GTCZ)

PIN ASSIGNMENT HEADER CONNECTOR[1] (J4)

Pin	Symbol	I/O	Description		
1	GND		Ground 0V		
2	V_{in}		Power Supply 3.3~5V		
3	$\overline{RES}$	I	$\overline{Reset}$	internal pull-up (10kΩ), low active	
4	$\overline{CS}$	I	SPI: Chip Select	low active	
5	MOSI	I	SPI: MOSI		
6	MISO		SPI: MISO		
7	CLK	I	SPI: CLK		
8	RxD	I	RS232: Receive Data	3.3V level, internal pull-up (1MΩ)	
9	TxD	O	RS232: Transmit Data	3.3V level	
10	DE	O	RS485: Transmit enable	3.3V level	
11	SDA	I/O	I²C: Serial Data	internal pull-up: (4,7kΩ)	
12	SCL	I	I²C: Serial Clock	internal pull-up: (4,7kΩ)	
13	SBUF Test	O I	Low: sendbuffer contains data PowerOn Low: testmode enabled	internal pull-up (10kΩ)	
14	DAC	O	Analog output	0~3.1V	
15	AIN1	I	Analog input 1	0~3.1V	
16	AIN2	I	Analog input 2	0~3.1V	
17	I/O 1	I/O	Digital in- or output	High Power Output	
18	I/O 2	I/O	Digital in- or output	High Power Output	
19	I/O 3	I/O	Digital in- or output	High Power Output	
20	I/O 4	I/O	Digital in- or output	High Power Output	
21[2]	I/O 5	I/O	Digital in- or output / PWM output	Low Power Output	PLUGL128-6 only
22[2]	I/O 6	I/O	Digital in- or output	Low Power Output	
23[2]	I/O 7	I/O	Digital in- or output	Low Power Output	
24[2]	I/O 8	I/O	Digital in- or output	Low Power Output	
25[2]	V_{out}		Power output (max. 3.1V)	for external periphery (max. 100 mA)	
26[2]	GND		Ground 0V		

1) Mounted in hardware version Z only (EA PLUGL128-6GTCZ or EA PLUGS102-6GTCZ)
2) only for EA PLUGL128-6 version

SERIAL INTERFACES

The module provides 4 serial interfaces, including USB, RS232, SPI and I²C. All interfaces are enabled and received data will be put into one receive buffer. A protocol command is provided if an exclusive interface request is requested.

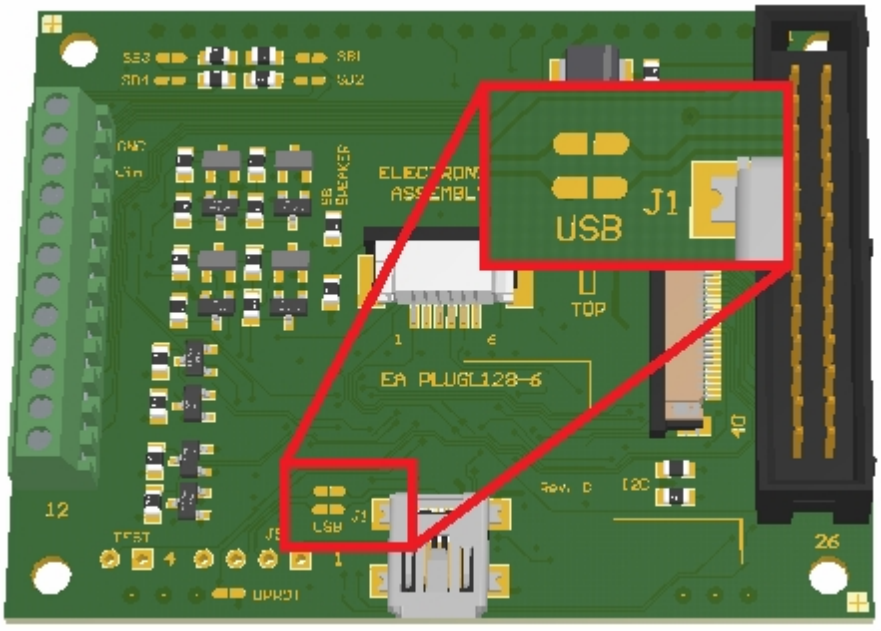
USB

The **U**niversal **S**erial **B**us is a serial bus system for interfacing a PC with other peripherals. It's based on differential data transfer. The bus topology is a strict master-slave communication. In the case of EA PLUG-Series the PC/Master needs to coordinate the communication. The module has a CDC (Communications Device Class) and is found by Windows PC's as a virtual COM-Port:

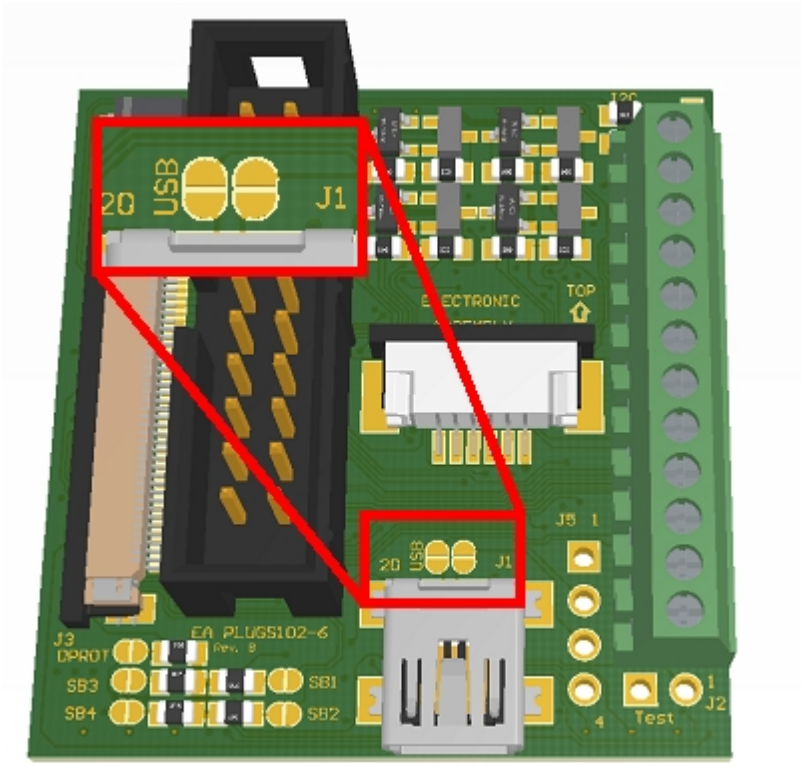
Description	Value
Device Class	2
USB Vendor ID	0x2DA9
USB Product ID	0x0CE4
Device description	EA PLUG

To program the module, adjust settings or to perform initial tests, we recommend using the USB interface. It's easy to connect, the transfer rate is fast and no interface parameters need to be specified. The driver for Windows can be downloaded on our web-page.

If the USB connection is not to be made via the mini-USB connector (J1) but via the primary connector (J5), two solder bridges must be closed:



Solder bridge to use the primary connector (EA PLUGL128-6)



Solder bridge to use the primary connector (EA PLUGS102-6)

The primary connector (J5) pinout is compatible with the PC mainboard pinout (Note polarity).

Pin	Symbol	Description
1	VBUS	+5V
2	D-	Data -
3	D+	Data +
4	GND	Ground 0V

Attention:
A protocol has to be used in USB CDC mode. It's impossible to use the USB interface without a protocol, which means the solder bridge (DPROT) must be kept open. Otherwise the high-speed connection of USB leads to buffer overflow.

RS232

RS232 is a standard for a serial interface. The transmission is serially asynchronous. Thus the data is converted into a bit stream and transmitted. There is no clock signal, so every bus user must work with the same transmission rate (so-called baud rate). RS232 is a voltage interface, such that data is transmitted using changing voltage levels. RS232 consists of "listening" and "talking" lines that are crossed between the two parties. The PLUG fits for direct connection to a RS-232 interface with CMOS level (in the case of EA PLUG-Series **3.1 V**). If you have an interface with another level, an external levelshifter is needed.
In the EA PLUG-Series, the data format is fixed to 8-N-1:

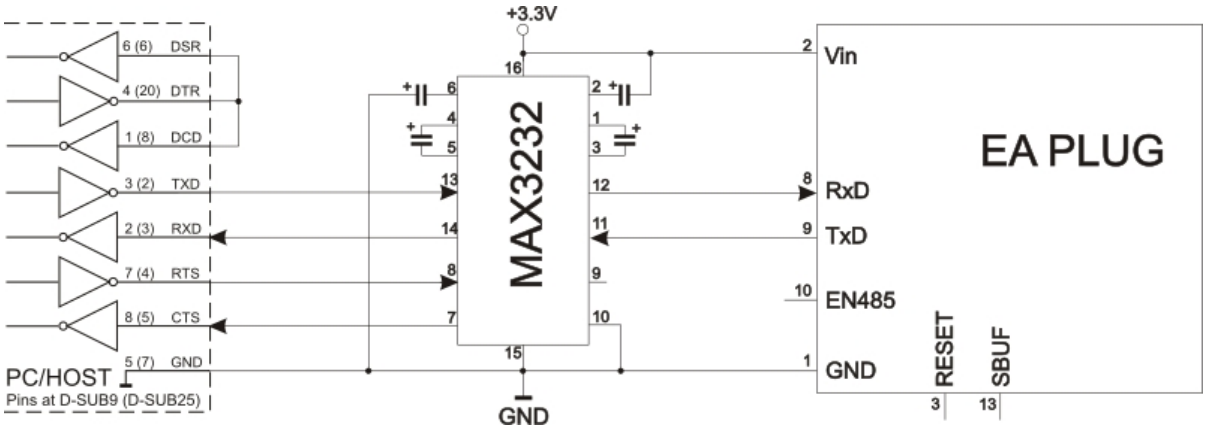
The EA PLUG-Series works with the folowing baud rates.

Baud	Error
2400	+0.16
4800	+0.16
9600	+0.16
19200	+0.16
38400	+0.16
57600	+0.16
115200	+0.16

The parameter (baud rate) is set using command #+R (higher-level control unit).

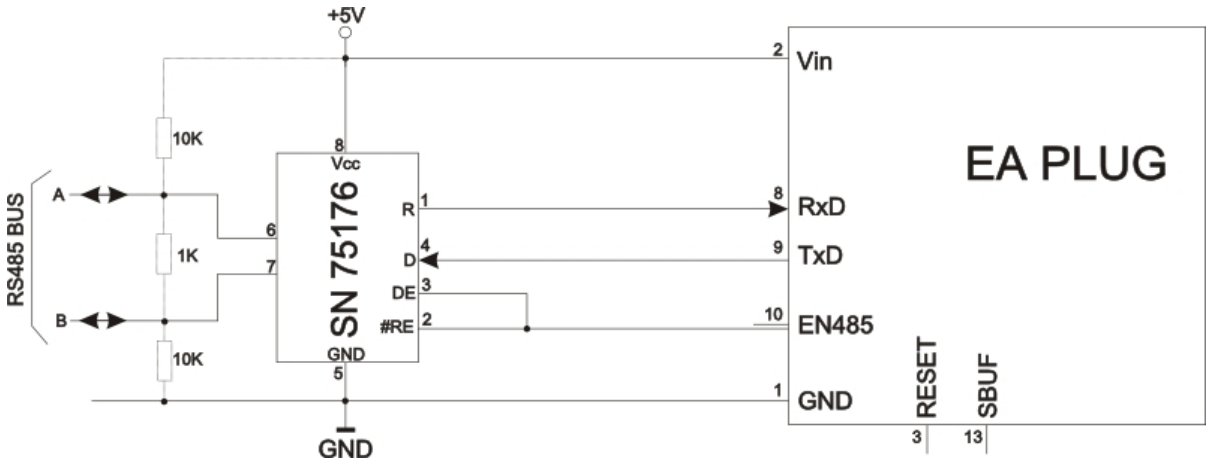
Application notes

RS232 with levelshifter:
In the PC world and industrial controls, levels of + 12V and - 12V are defined as standard. With boards or micro-controllers levels of 0V and VDD (in the case of EA PLUG-Series 3.1 V) are common. To adjust the signal levels, there are some possibilities in the form of level shifters (e.g., ICL3232, MAX3232).



RS232 V24 - Interface to a PC (EA PLUG-Series as slave)

RS485/RS422 interface:
With an external converter (e.g. SN75176), the EA PLUG-Series can be connected to a 2-wire RS-485 bus. Large distances of up to 1200m can thus be implemented (remote display). Several EA PLUG-Series displays can be operated on a single RS-485 bus by setting addresses (command #+R).



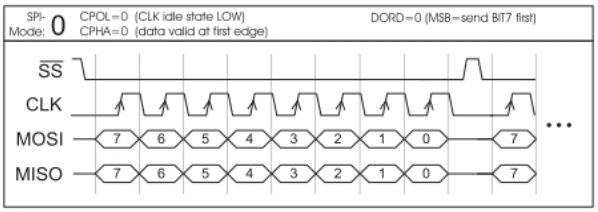
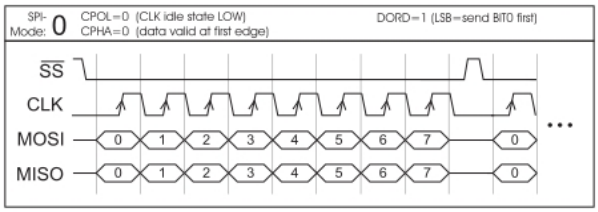
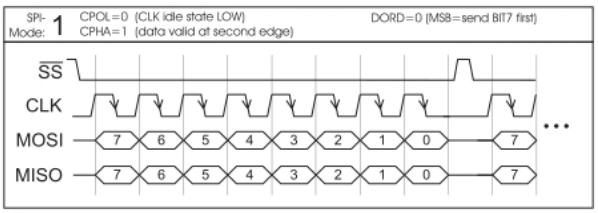
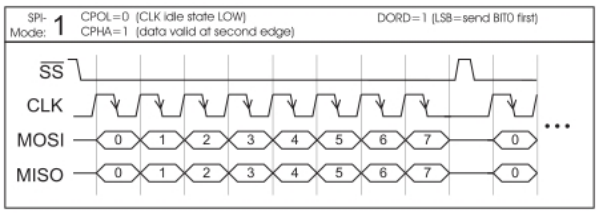
RS485 - Interface to a PLC (EA PLUG-Series as slave)

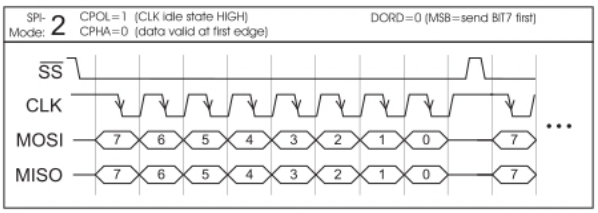
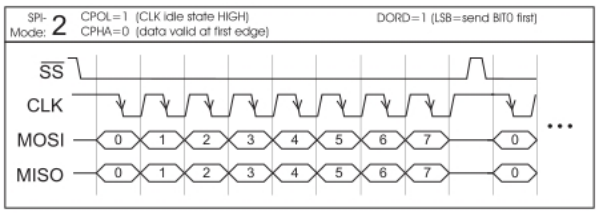
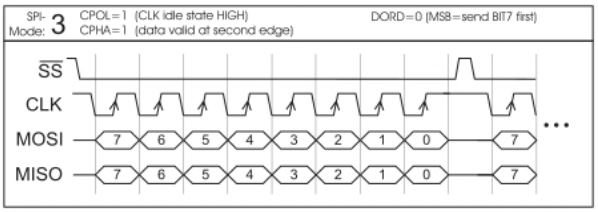
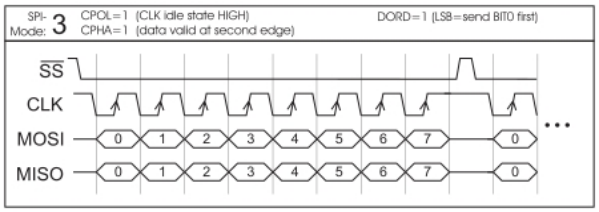
SPI

The **S**erial **P**eripheral **I**nterface is a bus system for serial synchronous data transfer, working with 4 lines:

- MOSI (**M**aster **O**ut → **S**lave **I**n) or SDO (Serial Data Out) or DO
- MISO (**M**aster **I**n ← **S**lave **O**ut) or SDI (Serial Data In) or DI
- SCK (**S**erial **C**lock) - Shift clock
- SS (**S**lave **S**elect → Addressing) or CS (Chip Select)

SPI works with a bidirectional transmission principle, meaning that data is exchanged between the connected devices at the same time. The communication is controlled by the master using the SCK line.
The protocol for data transfer is not defined in SPI, therefore there are different configuration possibilities, which are defined by the parameters Clock Polarity, Clock Phase and Data Order. The default setting is SPI mode 3 with DORD = 0. The command #+S set the mode 0..3. Alternatively the command can be stored directly into the PowerOn macro.

Mode	CPOL	CPHA	DORD (0) - MSB First	DORD (1) - LSB First
0	0	0		
1	0	1		

Mode	CPOL	CPHA	DORD (0) - MSB First	DORD (1) - LSB First
2	1	0		
3	1	1		

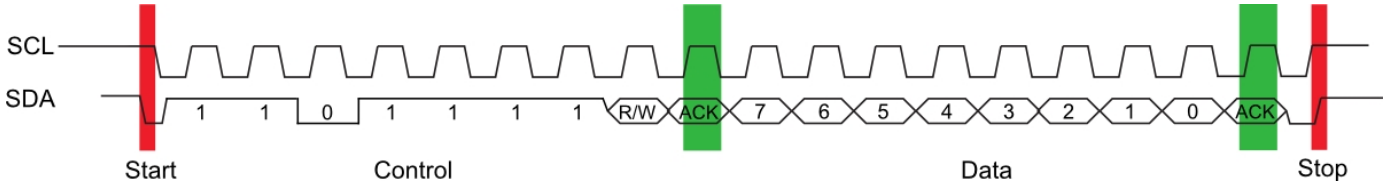
The maximum clock frequency is 1MHz. The module needs some time to prepare data for transfer. That means a wait cycle (no activity on the SCK-line) of at least **50µs** is required before reading data.

I²C

I²C stands for **I**nter-**I**ntegrated **C**ircuit and is a serial data-bus developed by Phillips. The bus is a Master-Slave implementation and needs 2 signal lines:

- SCL (**S**erial **C**lock **L**ine)
- SDA (**S**erial **D**ata **L**ine)

The electrical specification defines that both lines are terminated with a pull-up resistor at VDD, because all devices connected to the bus have open collector outputs. The bus clock is always given by the master, which controls the entire communication:



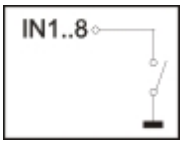
After the start condition, the slave address follows. In this case, bit 0 is the so-called R/W bit and determines whether the slave should be read (1) or data is transmitted (0). The data exchange takes place until the master executes the stop condition. More detailed information can be found in the I²C specification. The default I²C bus address is 0xDE (as 7-Bit address without R/W bit it's 0x6F) when writing to the slave unit. The command #+I can change the I²C write address to any other address. Alternatively the command can be written directly into the PowerOn macro.

The maximum frequency is 400kHz..The module needs some time to prepare data for transfer. That means a wait cycle (no activity on the SCL-line) of at least **50µs** is required before reading data.

IN- AND OUTPUTS

The EA PLUG-Series has 8 digital in- or outputs (CMOS level, non-floating). They can be redefined freely.
Note: I/O 5 can be used as PWM output.

Inputs (I/O 1-8)



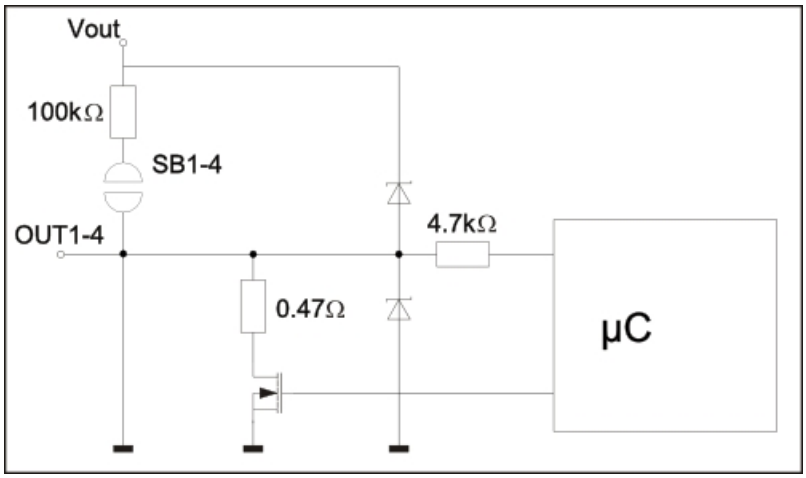
As status on delivery, all ports are defined as inputs. Each input provides an internal pull-up resistor, so it is possible to connect a key or switch directly between input and GND. The inputs can be queried and evaluated directly via the serial interface (#YR). In addition to that every port change may start an individual port - or bit- macro. The command #YA activates or deactivates automatic port query. Every alteration of inputs firstly calls bit macros and afterwards port macros. If there is no macro defined, the new status is transfered into the send buffer (<ESC>P).

Note: The logic circuitry is designed for slow operations; in other words, more than 3 changes per second cannot be executed.

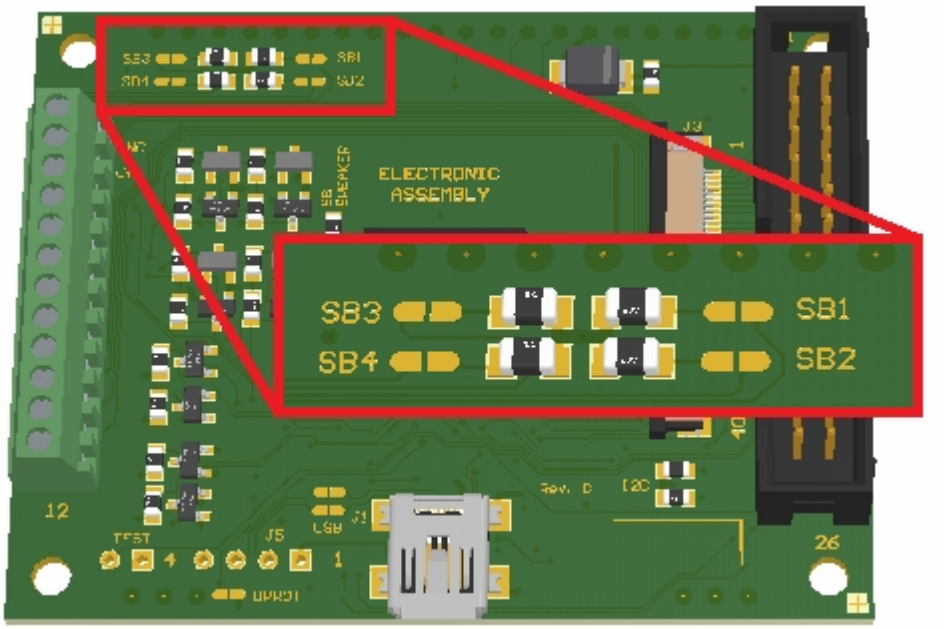
Output (I/O 1-4 / High power)

The command #YM redefines one or several inputs as outputs. Each line can be controlled individually using the #YW command. These port pins already hold an internal MOSFET (max. 360mA)

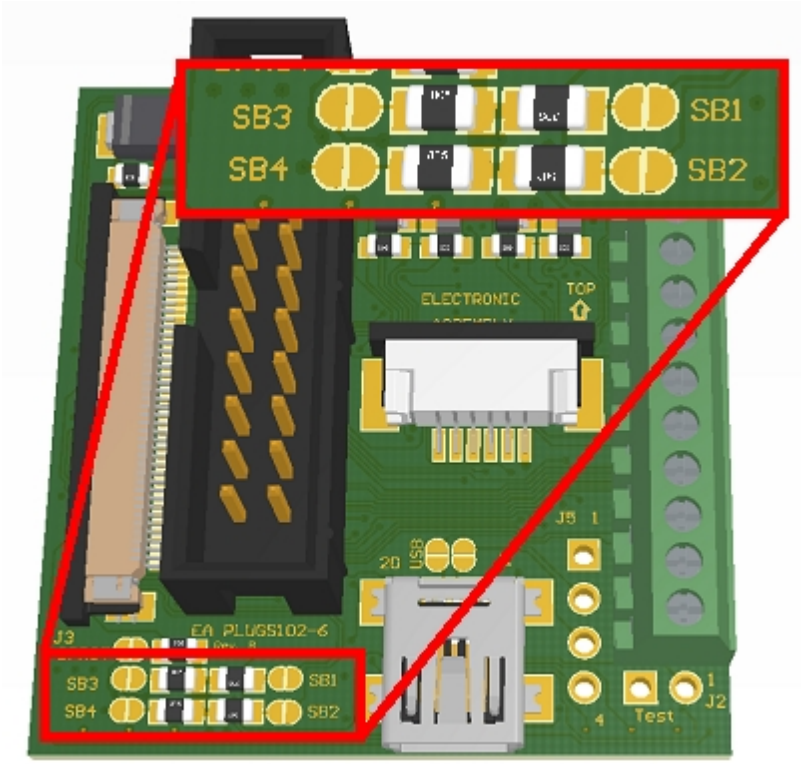
Note: Close the solder bridges (SB1-SB4) when you want to use the internal pull ups (100kΩ).



I/O port wiring (Port 1-4)



Solder bridges for I/O ports 1-4 (EA PLUGL128-6)

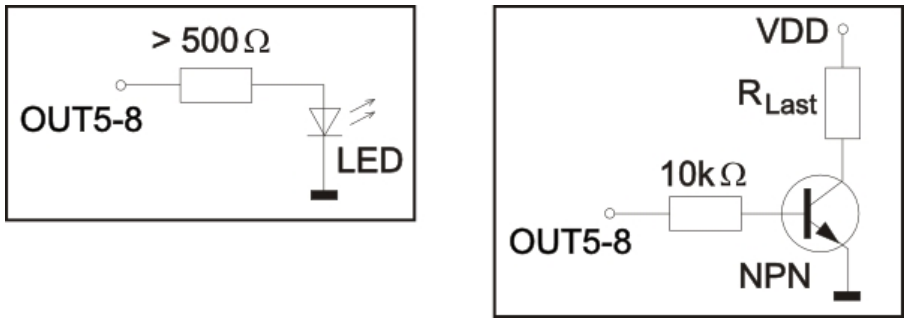


Solder bridges for I/O ports 1-4 (EA PLUGS102-6)

Output (I/O 5-8 / Low power)

The command #YM redefines one or several inputs as outputs. Each line can be controlled individually using the #YW command.

A maximum current of 5mA 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.



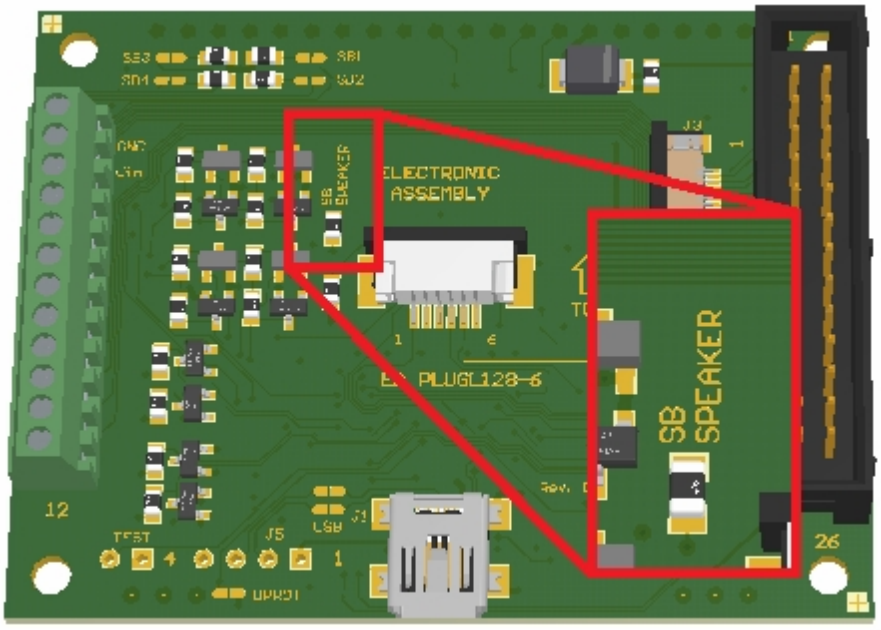
Application Example (Port 5-8)

PWM

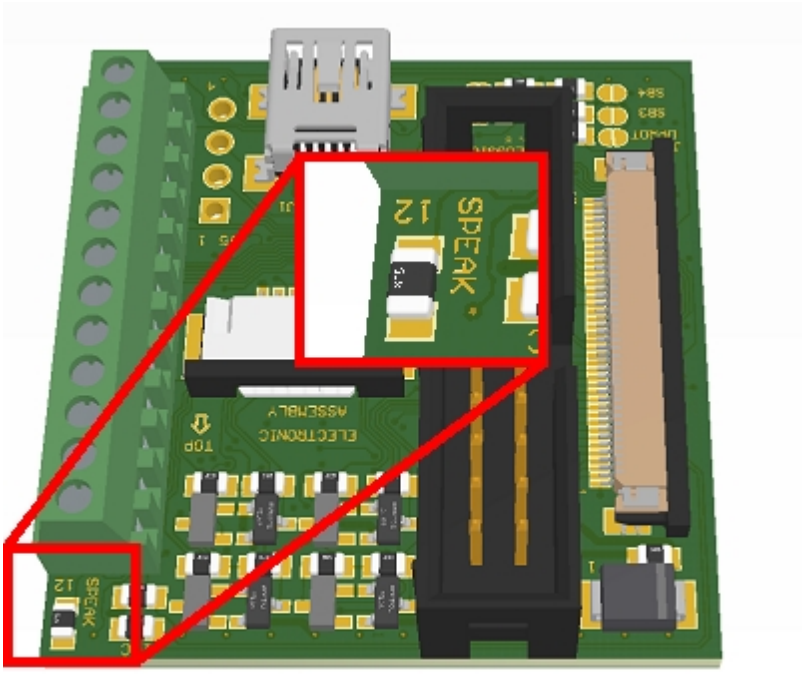
The module has the option of controlling external components via a PWM signal (pulse width modulation). Output Pin of the PWM signal is I/O Port 5. At constant frequency (adjustable from 2 Hz to 24 kHz #YO), the duty cycle of a rectangular pulse is changed. Modulation changes the ratio between the on- and off-time and thus the characteristics of the output signal. In this way, electromechanical components such as motors can be driven or even a quasi-analogue voltage can be generated. The variation of the duty cycles supports a low engine speed/voltage with a short start-up time or a high motor speed/voltage with a long start-up time. The output level is at 0V and Vin.

EXTERNAL SPEAKER

The EA PLUG-Series comes with a speaker. You can connect your own speaker if it is to silent. For this, pin 29 or pin 30 (power) are used (see following application examples). Internal speaker is deactivated by removing the 0Ω resistor (SB Speaker).



Solder bridge for external speaker (EA PLUGL128-6)



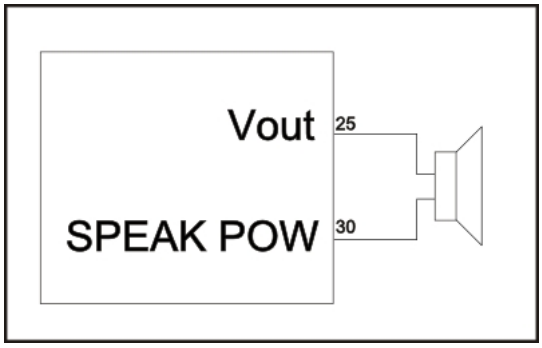
Solder bridge for external speaker (EA PLUGS102-6)

Application examples:

Connecting an external speaker:

An external speaker can be connected directly to the pins V_{out} and SPEAK POW. The following maximum values must not be exceeded.

Value	min.	typ.	max.	Unit
Current consumption	-	-	400	mA
Power consumption	-	-	1	W
Internal resistance speaker	6	8	32	Ω

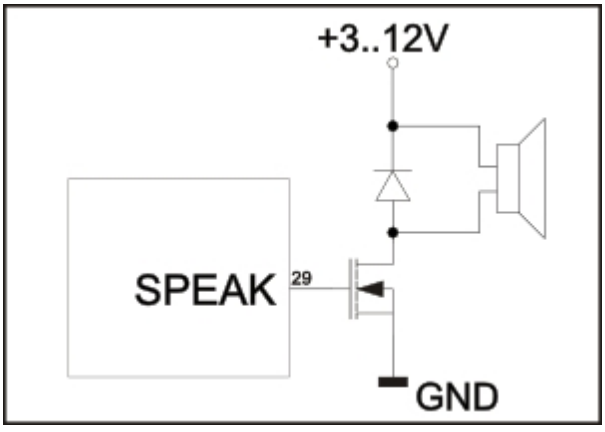


Connecting an external speaker

Connecting an external speaker with more power:

If you want to increase the volume further you need an additional control circuit with its own power supply.

Value	min.	typ.	max.	Unit
Output voltage V_{speak} (Pin 29)	-	-	3,1	V
Duty cycle	-	50	-	%
Frequency	-	4	-	kHz



Connecting an external speaker with its own control circuit

SOFTWARE

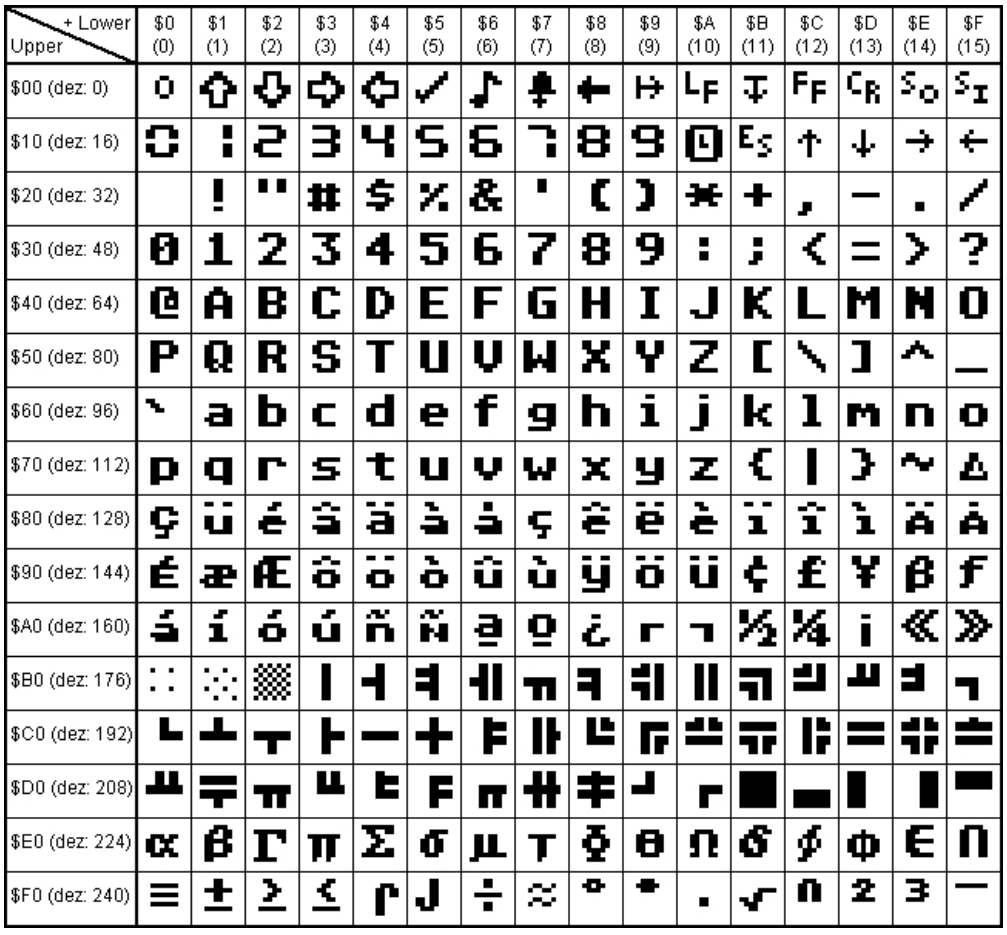
This display is programmed by means of commands, such as draw a rectangle from (0,0) to (64,15). No additional software or drivers are required. Strings and images can be placed with pixel accuracy. Text and graphics can be combined at any time. Different character sets can be used at same time. Each character set and the images can be zoomed from 2 to 4 times and rotated in 90° steps. With the largest character set, the words and numbers displayed will fill the screen.

LAYERS

The EA PLUG-Series has two different layers:

- **Terminal layer**
- **Graphic layer**

The terminal layer can be used for first steps and debugging. When you switch the unit on, the terminal layer is active and 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. 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 is entirely independent of the graphic outputs. If the graphics screen is deleted with #DL, for example, that does not affect the contents of the terminal window.
The terminal font is fixed in the ROM and can also be used for graphic outputs 'ESC Z...' (set FONT nr=0).



Terminal font

All other integrated commands are displayed in the graphic layer. Each command begins with ESCAPE or HASH followed by one or two command letters and then parameters. There are two ways to transmit commands:

ASCII mode

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

Binary mode

- Each command starts with the character ESC (hex: $1B, dec: 27).
- The command letters are transmitted directly.
- The coordinates x and y and all other parameters are transmitted as 8bit binary values.
- 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).).

BLINK MODE

After power on or the command 'ESC DG 0' the EA PLUG-Series is in blink mode. Two picture contents are alternatly shown in an adjustable period.

Blink attributes are set by the commands #ZB, #UB, #GB':

- n1=0: no blink
- n1=1: On/Off blink
- n1=2: blink inverted
- n1=3: Off/On blink (phase shifted)

Between strings (#ZL, #ZC, #ZR), flashing can be activated locally:

- Strings between two '~' ($7E) mean blink on/off.
- Strings between two '&' ($26) mean blink off/on phase shifted.
- Strings between two '@' ($40) mean blink inverted.

In addition you can assign or delete postly an rectangle area a blink mode, by using the command 'ESC Q...'

FONTS

Apart from the 8x8 terminal font (font no. 8), 3 additional monospaced fonts, 3 proportional fonts and 1 large numeric font are integrated as standard. The proportional fonts result in a more attractive appearance, and at the same time require less space on screen (e.g. the “i” is narrow and the “W” is wide). Each character can be positioned with pixel accuracy and the width and height can be scaled. Each text can be output left justified, right justified or centered. 90°/ 180° and 270° rotation is also possible. Macro programming permits additional fonts to be integrated (up to 15). This is done using the EA LCD-Tools).

Preloaded fonts

Upper + Lower	$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)					ö					Ö	Ü				β	

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	ā	ä	à	á	ç	ē	ë	è	ï	î	ì	Ä	Á
$90 (dez: 144)	É	æ	Æ	ō	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 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

Upper \ + Lower	$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)									°							

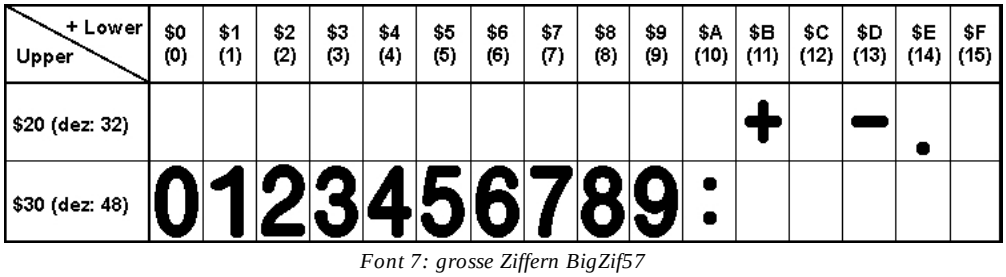
Font 4: GENEVA10 proportional

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

Font 5: CHICAGO14 proportional

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

Font 6: Swiss30 Bold proportional



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

Font 7: grosse Ziffern BigZif57

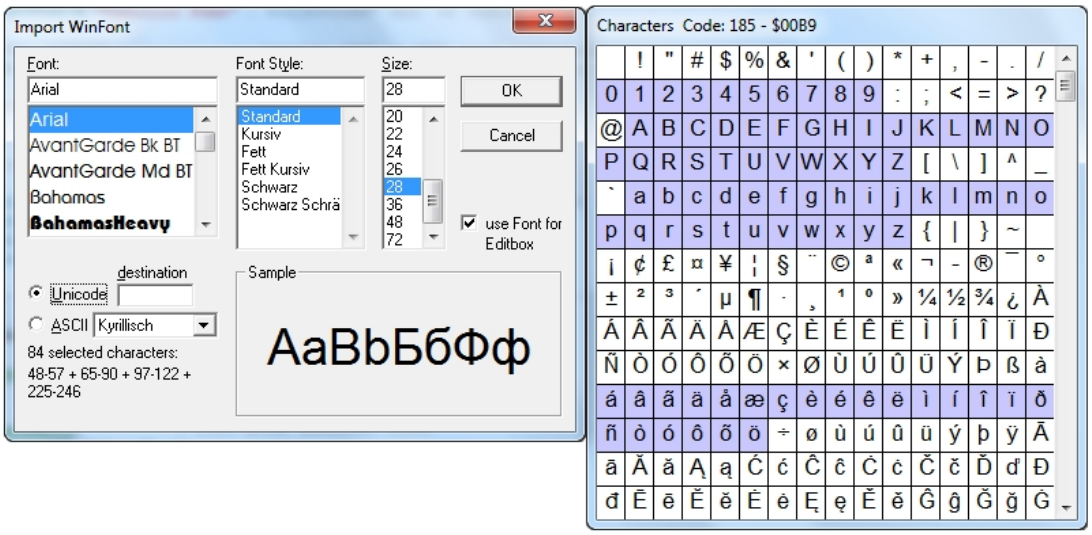
Additional fonts

The KitEditor can be used to include additional fonts.

- Compile statement `WinFont`:
  It is possible to raster TrueType-Fonts in different sizes witch can be used. A doubleclick to the fontname within the KitEditor opens the font selection box. To simplify the use of fonts, there is the possibilty of an edit box. If you output a string with KitEditor (e.g. #ZL 5,5, "Hello"), you can perform a double click on the string to open it. Now you can select the desired characters. This is mainly recommended using cyrillic, asian or symbol fonts. In that way, the KitEditor automatically places the right ASCII-Code. Alternatively you can use instead of the quotation

mark curly brackets (e.g. #ZL 5,5, {48656C6C6F}).

- Compiler option `Font`:
  All *.FXT font files can be used



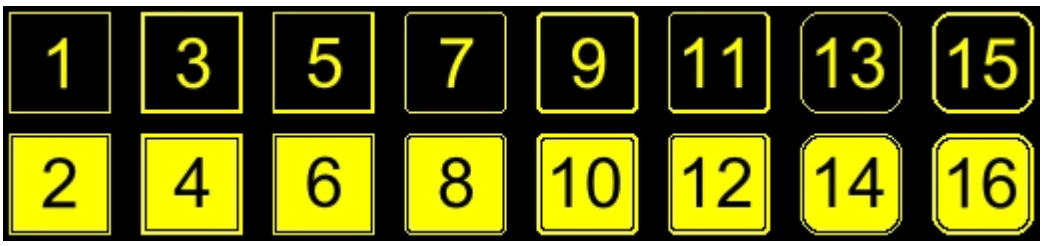
Import WinFonts

TOUCH PANEL

The -xxxTC version is shipped with a capacitive touch panel. Up to 40 touch areas (keys, switches, menus, bar graph inputs) can be defined simultaneously. The touch sensitive area can be defined by 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 29/30), 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 Macro programming).

Frames and key Shapes

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



Bitmap as keys

Apart from the frame types, which are infinitely scalable, it is also possible to use bitmaps as touch keys or touch switches. You can use the KitEditor 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 original size of the button bitmaps.

Switches in groups (Radio group)

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 (#AR 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.

MACRO PROGRAMMING

Single or multiple command sequences can be grouped together in macros and stored in the data flash memory. Then you can start them by using the Run macro commands. The KitEditor is used to program such macros. There are different types of macros (compiler directive marked in green letters):

- Normal macro **Macro**:
  These are started by means of an #MN 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 #MT command.
- Menu macro (1 to 255) **MenuMacro**:
  Started when you choose a menu item or issue an #MM command.
- Bit macro **BitMacro**:
  will be started by a single line IN 1..8 (bit) will change or by command #MB. 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 #YD.
- Port macro **PortMacro**:
  These are started when voltage (binary) is applied to IN 1..8 or by command #MP.
- Analog macro **AnalogMacro**:
  will start whenever voltage changes or limit exceeds or by command #MV.
- 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 3).
- Watchdog-macro **WatchdogMacro**:
  Started after a fault/error (e.g. failure).

Important: If a continuous loop is programmed in a power-on, reset or watchdog 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 using the protocol break command.

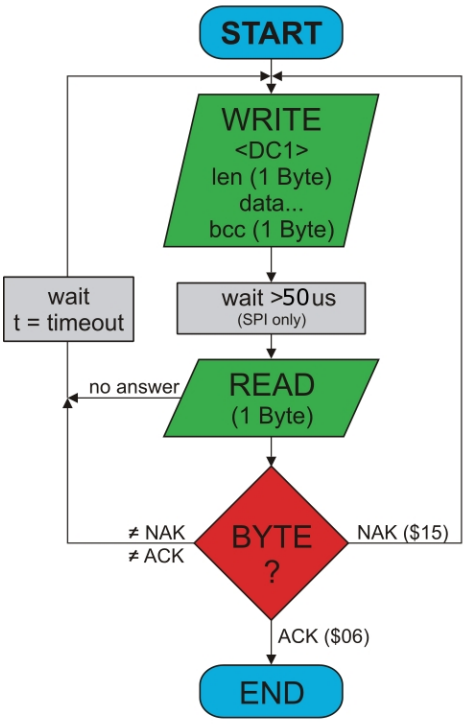
PROTOCOL / DATA TRANSFER

The transmission protocol is identical regardless which of the 4 serial interfaces is used to transfer data from the higher-level controller. The hardware circuit for each interface varies, which is described under the chapter serial interfaces.
The data transfer is embedded in a fixed frame with a checksum (protocol package). The EA PLUG-Series 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.

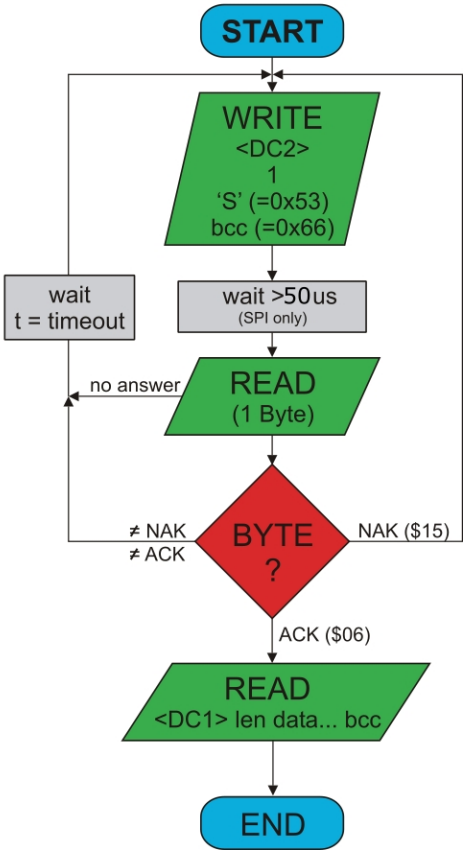
Note: It is neccessary to read the <ACK> byte in any case. If the host computer does not receive an acknowledgment, at least one byte is lost. In this case, the set timeout has to elapsed before the package is sent again.

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 split up between a number of packages. All data in the packages are compiled again after being correctly received by the EA PLUG.

Send

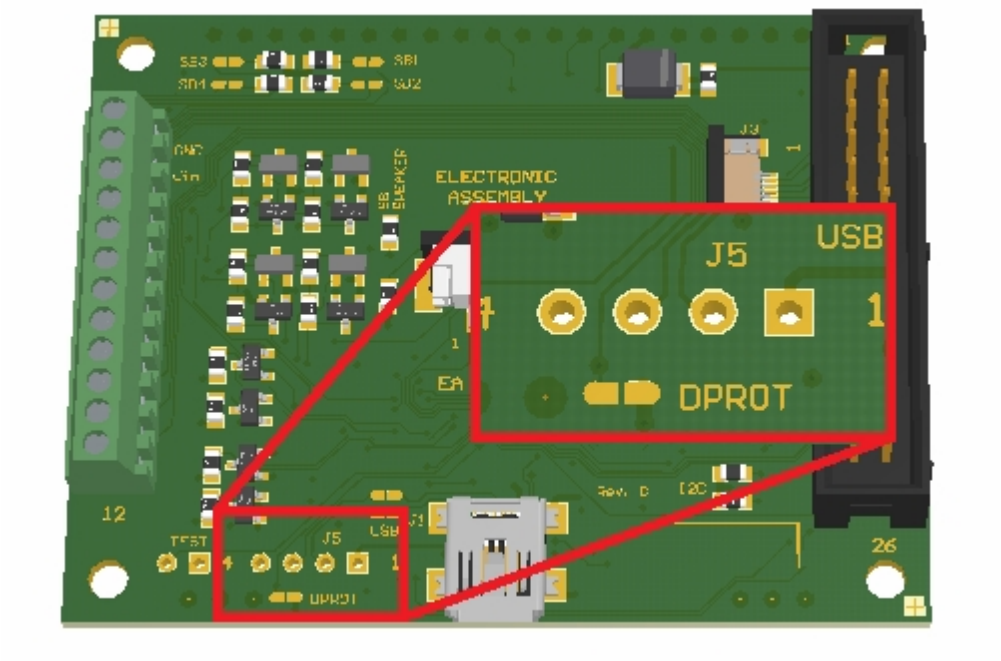


Receive

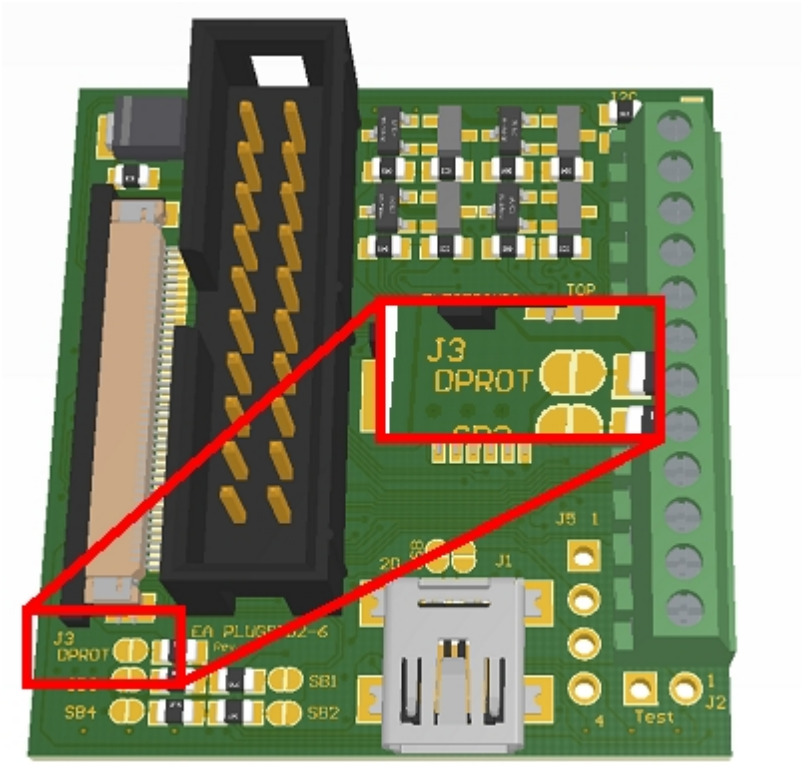


Deactivation the small protocol

For tests the protocol can be switched off by closing the solder bridge 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.



Solder bridge to deactivate small protocol (EA PLUGL128-6)



Solder bridge to deactivate small protocol (EA PLUGS102-6)

Note: Do not deactivate the protocol when using USB as communication interface.

Small Protocol commands

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

1. Transfer commands or data to EA PLUG-Series

This protocol command transfers data to the display. Several graphics commands can be packaged in a protocol package. If the data is larger than the maximum packet size, the data can be split into several packets. The module reassembles the individual data packets.

Module receives	DC1 0x11	length (8 Bit) 0xXX	Data..... 0x....	bcc (8 Bit) 0xXX
Module sends	ACK 0x06			

2. Request data of send buffer

If data is generated in the module, it is stored in the module's send buffer. The data can be requested via the serial interfaces. Whether data is available can be monitored via the pin 13 SBUF, or the higher-level controller can cyclically poll the data.

Module receives	DC2 0x12	length (8 Bit) 0x01	'S' 0x53	bcc (8 Bit) 0x66
Module sends	ACK 0x06			
Module sends	DC1 0x11	length (8 Bit) 0xXX	Data..... 0x....	bcc (8 Bit) 0xXX

3. Repeat last data packet

If a received packet of the module is faulty (wrong length or checksum) it can be requested again:

Module receives	DC2 0x12	length (8 Bit) 0x01	'R' 0x52	bcc (8 Bit) 0x65
Module sends	ACK 0x06			
Module sends	DC1 0x11	length (8 Bit) 0xXX	Data..... 0x....	bcc (8 Bit) 0xXX

4. Request buffer information

This command queries whether user data is ready (= Pin 13 SBUF) and also indicates how much free space is left in the device's receive buffer.

Module receives	DC2 0x12	length (8 Bit) 0x01	'I' 0x49		bcc (8 Bit) 0x5C
Module sends	ACK 0x06				
Module sends	DC2 0x12	length (8 Bit) 0x02	send buffer bytes ready (8 Bit) 0xXX	receive buffer bytes free (8 Bit) 0xXX	bcc (8 Bit) 0xXX

5. Protocol settings

This can be used to limit the maximum packet size that the display may send. As default a packet size with up to 64 bytes of user data is set. Furthermore, the time-out can be set in 1 / 100s. The time-out is activated when individual bytes have been lost. After the timeout, the entire packet will be

flashed and must be retransmitted.

Module receives Default values	DC2 0x12	length (8 Bit) 0x03	'D' 0x44	packet size send buffer (8 Bit) 0x80	Time-out (8 Bit) in 1/100s 0xC8 (=2 seconds)	bcc (8 Bit) 0x20
Module sends	ACK 0x06					

6. Protocol information

Request protocol settings (see 5.).

Module receives	DC2 0x12	length (8 Bit) 0x01	'P' 0x50			bcc (8 Bit) 0x63
Module sends	ACK 0x06					
Module sends	DC2 0x12	length (8 Bit) 0x03	maximum packet size send buffer (8 Bit) 0x80	packet size send buffer (8 Bit) 0xXX	Time-out (8 Bit) in ms 0xXX	bcc (8 Bit) 0xXX

7. RS485 address select / deselect

With this command, a module can be selected or deselected on the RS485 bus. By default, the module with address 0x7 is always active.

Module receives Default values	DC2 0x12	length (8 Bit) 0x03	'A' 0x41	'S' (=select) 'D' (=deselect) 0x53 or 0x44	RS485-address 0xXX	bcc (8 Bit) 0xXX
Module sends	ACK 0x06 ----	→ select → deselect				

8. RS485 enable signal - delay

Some RS485 masters take some time to change the enable signal, e.g. to switch from write to read mode. In order to enable successful communication with these devices, this command can be used to delay switching to write mode.

Module receives Default values	DC2 0x12	length (8 Bit) 0x03	'T' 0x54	Delay in 10 us 0x00 0x00	bcc (8 Bit) 0x69
Module sends	ACK 0x06				

9. Request interface exclusively

All 4 serial ports are handled in parallel and equivalently after reset. To ensure that a sequence of protocol packets is executed without interruption, the other serial interfaces can be disabled so the active interface can communicate with the module exclusively. This is useful, for example, for a project update via USB to prevent any interruption.

Module receives	DC2 0x12	length (8 Bit) 0x02	'G' 0x47	0x00 = Release 0x01 = Request	bcc (8 Bit) 0xXX
Module sends	ACK 0x06				
Module sends	DC2 0x12	length (8 Bit) 0x01	active (8 Bit) 0x00 = all 0x01 = RS232 0x02 = SPI 0x03 = IIC 0x04 = USB		bcc (8 Bit) 0xXX

10. Break-Command, Break / Stop execution

If a continuous loop has been programmed in a macro or if a normal process flow is blocked, this command can be used to interrupt and quit. This command is also suitable for update processes.

Module receives Default values	DC2 0x12	length (8 Bit) 0x02	'C' 0x43	break 0x01 = Wait command 0x02 = actual macro file 0x04 = Clear send buffer 0x08 = Clear receive buffer 0x10 = Stops automatic port and analog scan 0xFF = Stops everything	bcc (8 Bit) 0xXX
Module sends	ACK 0x06				

11. Software Reset

The module is restarted with this protocol command. Depending on the parameter, various start options can be selected to automatically run after the reset.

Module receives Default values	DC2 0x12	length (8 Bit) 0x02	'B' 0x42	Option 0x00 = normal restart 0x01 = Restart with test mode 0x02 = Restart without running PowerOnMacro macro	bcc (8 Bit) 0xXX
Module sends	ACK 0x06				

BCC-Calculation

The calculation of the checksum requires a simple 8-bit sum test (modulo 256). The following is a typical C implementation.

```
//---------------------------------------------------------------------------
//function: buffer2bcc()
//input:    ptr data, block length
//output:   Byte bcc
//descr:    calculate bcc for a buffer
//---------------------------------------------------------------------------
UBYTE buffer2bcc(UBYTE *dat, UBYTE len)
{
  UBYTE bcc = 0;
  while(len--)
    bcc += *dat++;
  return bcc;
}
```

COMMAND SUMMARY

The commands can be transmitted at runtime via the serial interfaces or stored in so-called macro files on the module's internal FLASH memory.
The following tables describe all commands.

All commands at a glance

Terminal #T

#TP	Position cursor
#TC	Cursor on/off
#TS	Save cursor position
#TR	Restore cursor position
#TA	Terminal off
#TE	Terminal on
#TV	Output version
#TJ	Output project name
#TI	Output information

Display #D

#DO	Set display orientation
#DR	Reset display
#DL	Delete display
#DI	Invert display
#DS	Fill display
#DA	Switch display off
#DE	Switch display on
#DC	Show clipboard
#DN	Show normal display content
#DZ	Set screensaver
#DW	Settings for mode 1 (Brightness)
#DX	Settings for mode 2 (Animated image/ Random pattern)
#DV	Settings for mode 3 (Invert mode)
#DY	Screensaver (Re)-Trigger

Clipboard #C

#CB	Save display contents
#CS	Save area
#CR	Restore area
#CK	Copy area

Line/ Point #G

#GZ	Point size / line thickness
#GV	Graphic link mode
#GB	Blink attribute
#GP	Draw point
#GD	Draw straight line
#GW	Continue straight line
#GR	Draw rectangle

Box #R

#RL	Delete area
#RI	Invert area
#RS	Fill area
#RM	Area with fill pattern
#RO	Draw box
#RR	Draw frame
#RT	Draw frame box

Text/ String/ Character #Z

#ZF	Set font
#ZZ	Set font zoom factor
#ZY	Additional line spacing
#ZJ	Set space width
#ZW	Text angle
#ZV	Text link mode
#ZB	Text flashing attribute
#ZL	Output string (left justified)
#ZC	Output string (centered)
#ZR	Output string (right justified)
#ZT	String for terminal

Bitmap/ Picture #U

#UZ	Image zoom factor
#UW	Image angle
#UV	Image link mode
#UB	Image flashing attribute
#UC	Image from clipboard
#UI	Load internal image
#UL	Load image
#UH	Send hardcopy

Bargraph/ Slider #B

#BR	Define bargraph (right)
#BL	Define bargraph (left)
#BO	Define bargraph (up)
#BU	Define bargraph (down)
#BD	Delete bargraph
#BA	Update bargraph
#BZ	Redraw bargraph
#BS	Send bargraph value

Blink area #Q

#QZ	Set flashing time
#QL	Delete flashing attribute
#QI	Flashing inversely
#QM	Flashing area pattern
#QR	Restore phase shifting area
#QE	Inverted phase shifted area
#QP	Phase shifted flashing pattern

Menu/ touchable #N

#NF	Set menu font
#NZ	Menu font zoom factor
#NY	Additional line spacing
#NW	Menu angle
#NT	Touch menu automation
#ND	Define and display menu
#NN	Next item
#NP	Previous item
#NS	End of menu/ send
#NM	End of menu /macro
#NA	End of menu/ cancel

Macro #M

#MN	Run normal macro
#MT	Run touch macro

#AE	Touch frame
#AR	Radio group for switches
#AF	Label font
#AZ	Label zoom factor
#AY	Additional line spacing
#AW	Label angle
#AT	Define touch key
#AU	Define touch key (image)
#AK	Define touch switch
#AJ	Define touch switch (image)
#AM	Define touch key with menu function
#AD	Define drawing area
#AH	Define free touch area
#AB	Set bar by touch
#AA	Touch query on/off
#AI	Touch key response (Automatic inversion)
#AS	Touch key response (sound)
#AQ	Send bar value automatically
#AN	Invert touch key
#AP	Set touch switch
#AX	Query touch switch
#AG	Query radio group
#AL	Delete touch area (by return code)
#AV	Delete touch area (by coordinates)

Replies

<ESC>A	Button / switch state (value changed)
<ESC>B	Bargraph value (value changed)
<ESC>N	Touch menu (value changed)
<ESC>T	Menu (value changed)
<ESC>P	Port read (value changed)
<ESC>H	Free touch area (value changed)
<ESC>N	Menu
<ESC>B	Bargraph value
<ESC>X	Button / switch
<ESC>G	Radiogroup
<ESC>Y	Port state
<ESC>D	Analog value
<ESC>V	Firmware version
<ESC>J	Project Name
<ESC>I	Internal information
<ESC>U	Hardcopy data
<ESC>W	Formated string (ADC/DAC)

TERMINAL

In the terminal layer, all received data is displayed directly. This layer is useful for quickly creating simple outputs or receiving error messages during development time.

Command	Codes		Remarks	
Form feed FF (dec: 12)	^L		The content of the screen are deleted and the cursor is placed at pos (1,1)	
Carriage return CR (dec: 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	

Terminal layer settings

Command	Codes				Remarks	
Position cursor	ESC	T	P	Column, Line	Origin upper-left corner (1,1)	1,1
Cursor on/off			C	Visibility	**Visibility** =0 (Terminal invisible); =1 (Terminal visible)	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 is switched off; outputs are rejected	
Terminal on			E		Terminal is switched on	On

Output informations

Command	Codes				Remarks	
Display version	ESC	T	V		The version no. is output in the terminal (e.g. "EAPLUG128-6 V1.0 T+")	
Display project name			J		The macro project name is output to the terminal (e.g. "init / delivery state")	
Display information			I		The terminal is initialized and deleted; software version, the macro project name and the CRC-checksum is ouput to the terminal	

DISPLAY (EFFECT THE ENTIRE DISPLAY)

Command	Codes				Remarks	
Set display orientation	ESC	D	O	Orientation	Orientation =0 (0°); =1 (90°); =2 (180°); =3 (270°)	0
Reset display			R		Reset the display	

Display content

Command	Codes				Remarks	
Delete display	ESC	D	L		Delete display contents (all pixels off)	
Invert display			I		Invert display contents (invert all pixels)	
Fill display			S		Fill display contents (all pixels on)	
Switch display off			A		Display content becomes invisible but are retained, commands are still possible	
Switch display on			E		Display content becomes visible again	On
Show clipboard			C		Show content of clipboard; Standard display outputs are no longer visible	
Show normal display content			N		Normal operation, standard display outputs are visible	

Screensaver

Command	Codes				Remarks	
Set screensaver	ESC	D	Z	Mode, Mask	Set the screensaver **Mode** =0 (No screensaver); = 1 (change display brightness); = 2 (Animated images/ Random pattern); =3 (invert display) and the retrigger **Mask** = $01 (Touch); =$2 (USB); =$4 (RS232); =$8 (SPI); =$16 (I²C)	0
Settings for mode 1 (Brightness)			W	ttTime1[1)], Bright1, ttTime2[1)], Bright2	If screensaver mode 1 is active (#DZ1,...) **ttTime1** and **TtTime2** (16-Bit) sets the time in seconds, when the brightness (**Bright1** and **Bright2**) is active. The value of brightness is relative to the actual brightness (0...150%).	
Settings for mode 2 (Animated image/Random pattern)			X	Type, ttTime[1)]	If screensaver mode 2 is active (#DZ2,...) **Type** sets the kind of animation **Type** =0 (Random pattern/starlit sky); =1...255 (Animated image 1...255) **ttTime** (16-Bit) sets the start time in seconds	
Settings for mode 3 (Inverse mode)			V	ttTime1[1)], ttTime2[1)]	If screensaver mode 3 is active (#DZ3,...) **ttTime1** (16Bit) sets the start time in seconds and **ttTime2** (16Bit) the inverting time in seconds	
Screensaver (Re)-Trigger			Y	Option	**Option** =0 (Trigger/Start screensaver); =1 (Retrigger screensaver)	

1) 16-bit value range (for binary transmission first low then high byte)

CLIPBOARD

Command	Codes				Remarks	
Save display contents	ESC	C	B		The entire contents of the display are copied to the clipboard as an image area	
Save area			S	x1, y1, x2, y2	The image area from **x1**, **y1** to **x2**, **y2** is copied to the clipboard	
Restore area			R		The image area on the clipboard is copied back to the display	
Copy area			K	x, y	The image area on the clipboard is copied to **x**, **y** in the display	

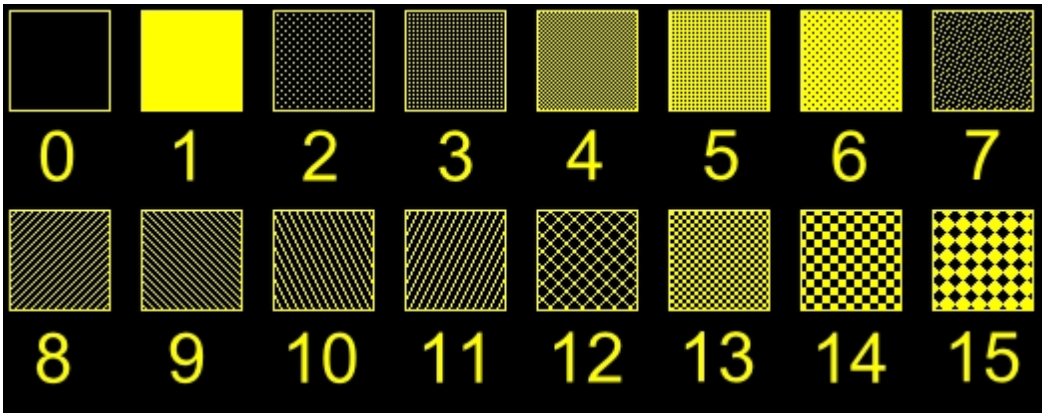
LINE/ POINT/ BOX

Straight lines and points

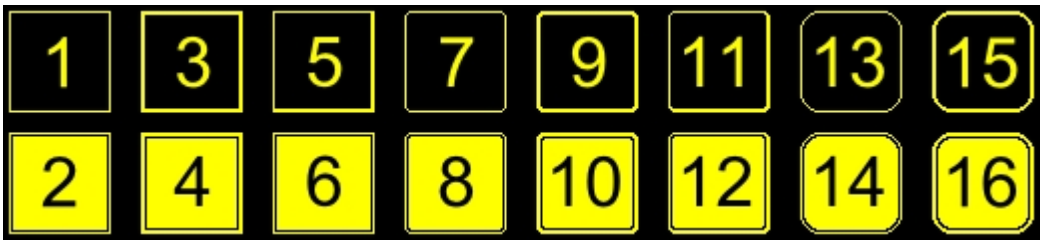
Command	Codes				Remarks	
Point size / line thickness	ESC	G	Z	n1, n2	**n1**= x-point size (1...15); **n2** = y-point size (1...15)	1,1
Graphic mode			V	Mode	Set drawing **Mode** =1 (set); =2 (delete); =3 (inverse)	1
Blink attribute			B	Mode	Set blink **Mode** =0 (no blink); =1 (on/off); =2 (blink inverted); =3 (off/on phase shifted)	0
Draw point	ESC	G	P	x, y	Set a point at coordinates **x**, **y**	
Draw straight line			D	x1, y1, x2, y2	Draw a straight line from **x1**, **y1** to **x2**, **y2**	
Continue straight line			W	x, y	Draw a straight line from the last end point to **x**, **y**	0,0
Draw rectangle			R	x1, y1, x2, y2	Draw four straight lines as a rectangle from **x1**, **y1** to **x2**, **y2**	

Change/ draw rectangular areas

Command	Codes				Remarks	
Delete area	ESC	R	L	x1, y1, x2, y2	Delete an area from **x1**, **y1** to **x2**, **y2** (all pixels off)	
Invert area			I	x1, y1, x2, y2	Invert an area from **x1**, **y1** to **x2**, **y2** (invert all pixels)	
Fill area			S	x1, y1, x2, y2	Fill an area from **x1**, **y1** to **x2**, **y2** (all pixels on)	
Area with fill pattern			M	x1, y1, x2, y2, Pattern	Fill an area from **x1**, **y1** to **x2**, **y2** with pattern **Pattern** (always set)	
Draw box			O	x1, y1, x2, y2, Pattern	Draw rectangle from **x1**, **y1** to **x2**, **y2** with pattern **Pattern** (always replace)	
Draw frame			R	x1, y1, x2, y2, Frame	Draw frame of type **Frame** from **x1**, **y1** to **x2**, **y2** (always set)	
Draw framed box			T	x1, y1, x2, y2, Frame	Draw frame box of type **Frame** from **x1**, **y1** to **x2**, **y2** (always replaye)	



Integrated pattern



Integrated frame and key shapes

TEXT/ STRING/ CHARACTER

Settings

Command	Codes				Remarks	
Set font	ESC	Z	F	n1	Set font with the number **n1** = 0...15	0
Set font zoom factor			Z	x, y	**x** = x-zoom factor (1...4); **y** = y-zoom factor (1...4)	1,1
Additional line spacing			Y	Spacing	Insert **Spacing** = 0...15 dots between two lines as additional spacing	0
Set space width			J	Spacing	Spacing =0 (Use original space from font); =1 (same width as numbers); >=2 (width in dots)	0
Text angle			W	Angle	Set text **Angle** = 0 (0°); = 1 (90°); = 2 (180°); = 3 (270°)	0
Text mode			V	Mode	Set link **Mode** = 1 (set); =2 (delete); = 3 (inverse); =4 (replace); =5 (inverse replace)	4
Text flashing attribute			B	Mode	Set flashing **Mode**=0 (no blink); =1 (on/off); =2 (blink inverted); =3 (off/on phase shifted)	0

Strings

Command	Codes				Remarks	
Display / place string L: left justified C: centered R: right justified	ESC	Z	L	x, y, Text ..., NUL	A string is placed to **x**, **y**; string termination is: **'NUL'** ($00), 'LF' ($0A) or 'CR' ($0D); several lines are seperated by the character '|' ($7C); Text between two '~' ($7E): characters blink on/off; Text between two '&' ($26): characters blink phase shifted; Text between two '@' ($40): characters blink inverse; The character '\' ($5C, backslash) cancels the special function of characters '|~@\'; e.g. "name\@test.de" => "name@test.de"	
			C			
			R			
String for terminal	ESC	Z	T	Text ...	Command to output a string (**Text ...**) from a macro to the terminal	

BITMAP/ PICTURE

Settings

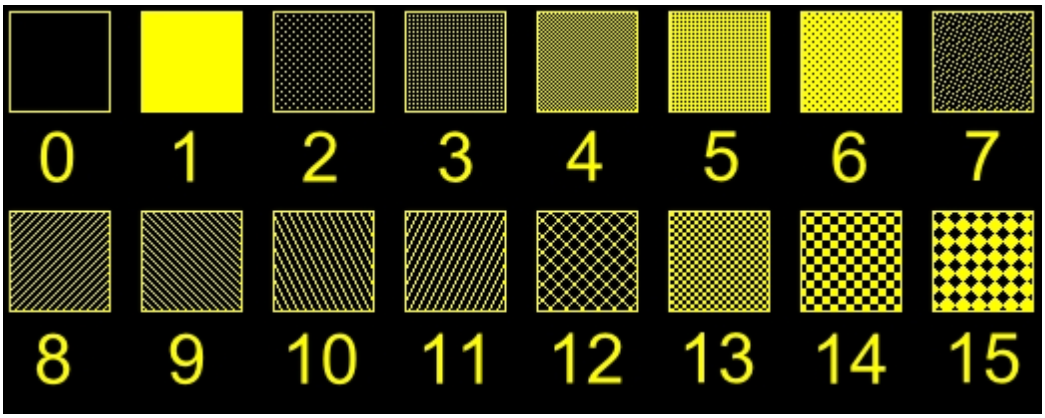
Command	Codes				Remarks	
Image zoom factor	ESC	U	Z	x, y	**x** = x-zoom factor (1...4); **y** = y-zoom factor (1...4)	1,1
Image angle			W	Angle	Set image **Angle** = 0 (0°); = 1 (90°); = 2 (180°); = 3 (270°)	0
Image link mode			V	Mode	Set link **Mode** = 1 (set); =2 (delete); = 3 (inverse); =4 (replace); =5 (inverse replace)	4
Image flashing attribute			B	Mode	Set flashing **Mode**=0 (no blink); =1 (on/off); =2 (blink inverted); =3 (off/on phase shifted)	0

Images

Command	Codes				Remarks	
Image from clipboard	ESC	U	C	x, y	The current content of the clipboard are laded to **x**, **y** with all the image attributes	
Load internal image			I	x, y, n1	Load internal image with number **n1** =0...255 from FLASH to **x**, **y**	
Load image			L	x, y, BLH data ...	Load an image to **x**, **y**; **data...** = image in BLH-format	
Send hardcopy			H	x1, y1, x2, y2	An image area **x1**, **y1**, **x2**, **y2** is put into the sendbuffer. The image is end in BLH-format	

BARGRAPH/ SLIDER

Command	Codes				Remarks	
Define bargraph	ESC	B	R L O U	n1, x1, y1, x2, y2, aw ew Type, Pattern	Define bargraph with number **n1**=1..32 to l(eft), r(ight), o(up), u(down). **x1**, **y1**, **x2**, **y2** are the surrounding rectangle of the bar. **aw** and **ew** are the values for 0% and 100%. **Type** =0 (pattern bar); =1 (pattern bar in rectangle); **Pattern** = bar pattern **Type** =2 (line bar); =3 (line bar in rectangle); **Pattern** = line width	-
Delete bargraph		B	D	n1, Visibility	The definition of bargraph **n1** becomes invalid. If the bargraph was defined as an input by touch, the touch field will also be deleted **Visibility** =0 (Bargraph remains visible) =1 (Bargraph is deleted)	
Set bargraph		B	A	n1, Value	Set and draw the bargraph number **n1** to new **Value**	
Redraw bargraph			Z	n1	Entirely redraw the bargraph number **n1**	
Send bargraph value			S	n1	Send the current value of the bargraph number **n1**	



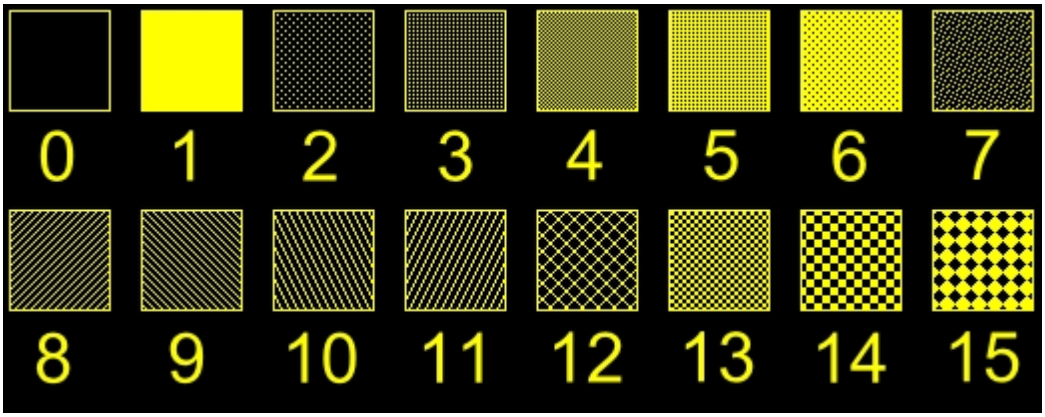
Integrated pattern

Additional Bargraph/Slider commands (touchable)

Command	Codes				Remarks	
Set bar by touch	ESC	A	B	nr	The bargraph with the number **nr** is defined for input by touch panel.	
Send bar value automatically			Q	n1	The Automatic transmission of a new bargraph 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
Set brightness by touch	ESC	Y	B	nr	Change brightness by bargraph number **nr**	
Bargraph for analogue output	ESC	V	B	128, no	Assigns bargraph **no**=1…20 to analogue output Define start- end values (aw, ew) forbargraph in [mV/20]	

BLINK AREA

Command	Codes				Remarks	
Set blink time	ESC	Q	Z	Time	Set flashing **Time** =1...15 in 1/10s; = 0 (flashing deactivated)	6
Delete blink attribute			L	x1, y1, x2, y2	Delete the flashing attribute from **x1**, **y1** to **x2**, **y2**. Do not use this command for phase shifted areas! (Copies the area from graphiclayer to blinklayer)	
Blink inversely			I	x1, y1, x2, y2	Define an inverted flashing area from **x1**, **y1** to **x2**, **y2**. (Copies the inverted area from graphiclayer to blinklayer)	
Blink area pattern			M	x1, y1, x2, y2, Pattern	Define a flashing area (on/off) with pattern **Pattern** from **x1**, **y1** to **x2**, **y2** (Draw the pattern into blinklayer)	
Restore phase shifted area			R	x1, y1, x2, y2	Delete the phase shifted flashing area from **x1**, **y1** to **x2**, **y2**. Do not use this command for other flashing attributes! (Copies the area from blinklayer to graphiclayer)	
Inverted phase shifted area			E	x1, y1, x2, y2	Define a phase shifted inverted flashing area from **x1**, **y1** to **x2**, **y2**. (Copies the inverted are from blinklayer to graphiclayer)	
Phase shifted blinking pattern			P	x1, y1, x2, y2, Pattern	Define a phase shifted flashing area (off/on) with pattern **Pattern** from **x1**, **y1** to **x2**, **y2**. (Draw the pattern into graphiclayer)	



Integrated pattern

MENU/ TOUCHABLE

Settings for menu box/touch menu

Command	Codes				Remarks	
Set menu font	ESC	N	F	n1	Set font with the number **n1** = 0...15 for menu display	0
Menu font zoom factor			Z	x, y	**x** = x-zoom factor (1...4); **y** = y-zoom factor (1...4)	1,1
Additional line spacing			Y	Spacing	Insert **Spacing** = 0...15 dots between two lines as additional spacing	0
Menu angle			W	Angle	Menu display **Angle** =0 (0°); =1 (90°)	0
Touch menu automation			T	Type	**Type** =1: (Touchmenu opens automatically); =0 (Touchmenu doesn't open automatically, instead the request 'ESC T0' is sent to the host,which can then open the touch menu with 'ESC NT2')	1

Menu box commands (control not by touch)

Command	Codes				Remarks	
Define and display menu	ESC	N	D	x, y, n1, Text ..., NUL	A menu is drawn at the corner **x**, **y** witch the current font of menu **n1** = currently inverted entry (e.g. 1 = first entry) **Text...** = string with menu items. The different items are seperated by the character '|' ($7C,dez:124) e.g. "Item1|Item2|Item3" The background of the menu is automatically saved into the clipboard (previous contend will be overwritten). If a menu is already defined, it is automatically canceled and deleted	
Next item			N		The next item is inverted or remains at the end	
Previous item			P		The previous item is inverted or remains at the beginning	
End of menu/ send			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			M	n1	The menu is removed and replaced with the original background. Menu macro n1 is called for item 1, menu macro **n1**+1 for item 2 and so on... .	
End of menu/ cancel	ESC	N	A		The menu is removed and replaced with the original background	

Additional Menu commands (touchable)

Command	Codes				Remarks	
Define touch key with menu function	ESC	A	M	x1, y1, x2, y2, Down Code, up Code, MNU Code, Text ..., NUL	The area from **x1**, **y1** to **x2**, **y2** 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.	

MACRO

Single or multiple command sequences can be collected as so-called macros. The following commands describe how to work with macros:

Call macros

Command	Codes				Remarks	
Run normal macro	ESC	M	N	n1	Call the (normal) macro with the number **n1** (max. 7 levels)	
Run touch macro			T	n1	Call the touch macro with the number **n1** (max. 7 levels)	
Run menu macro			M	n1	Call the menu macro with the number **n1** (max. 7 levels)	
Run port macro			P	n1	Call the port macro with the number **n1** (max. 7 levels)	
Run bit macro			B	bittype	Call the bit macro with the **bittype** (max. 7 levels)	
Run analog macro			V	analogtype	Call the analog macro with the **analogtype** (max. 7 levels)	

Bit macros	
bittype	Start the macro at...
1...8	falling edge (I/O 1..8)
9...16	rising edge (I/O 1..8)

Analog macros		
analogtype		Start the macro at...
AIN1	AIN2	
0	10	every change of the analog value
1	11	falling analog value
2	12	rising analog value
3	13	smaller than lower limit
4	14	larger than lower limit
5	15	smaller than upper limit
6	16	larger than upper limit
7	17	outside both limits
8	18	within both limits
9	19	smaller than other analog channel

Automatically running macros

Command	Codes				Remarks	
Macro with delay	ESC	M	G	n1, Delay	Call the (normal) macro with the number **n1** in **Delay**/10s. Execution is stopped by commands (e.g. receipt or touch macros)	
Automatic macros once only			E	n1, n2, Time	Automatically run macros **n1** to **n2** once only; **Time** = pause in 1/10s. Execution is stopped by commands (e.g. receipt or touch macros)	
Automatic macros cyclically			A	n1, n2, Time	Automatically run macros **n1** to **n2** cyclically; **Time** = pause in 1/10s. Execution is stopped by commands (e.g. receipt or touch macros)	
Automatic macros ping pong			J	n1, n2, Time	Automatically run macros **n1** to **n2** to n1 (ping pong); **Time** = pause in 1/10s. Execution is stopped by commands (e.g. receipt or touch macros)	

Macro processes (automatically)

Command	Codes				Remarks	
Define macro process	ESC	M	D	no, Type, n1, n2, Time	A macro process with the number **no** (1 to 4) is defined (1=highest priority). The macros **n1** to **n2** are run successuvely every **Time**/10s. **Type =**1 (once only); =2 (cyclical); =3 (ping pong n1 to n2 to n1)	
Macro process interval			Z	no, Time	A new time **Time**/10s is assigned to the macro process **no** (1 to4). If the time **Time** is set to 0, the execution is stopped.	
Stop macro processes			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

GENERAL

Brightness

Command	Codes				Remarks	
Save brightness	ESC	Y	@		Save actual brightness on to FLASH	
Set brightness			H	Bright	Set brightness to **Bright** =0%...150%	100
Increase brightness			N		Increase current brightness	
Reduce brightness			P		Reduce current brightness	
Brightness on/off			L	n1	Brightness **n1** =0 (off); =1 (on); =2...255: The brightness is switched on for **n1**/10s.	1
Set brightness by bargraph			B	nr	Change brightness by bargraph number **nr**	

Send commands

Command	Codes				Remarks	
Send bytes	ESC	S	B	Length, data ...	**Length** (=1 to 255) bytes are sent to the sendbuffer **data...** = data to send. In the source text of the macro programming, the number **Length** must not be specified. This is counted by the PLUG-compiler and entered.	
Send version			V		The version is sent as a string to sendbuffer (e.g. "EAPLUG128-6 V1.0 T+")	
Send projectname			J		The macro project name is send as a string to sendbuffer (e.g. "init / delivery state")	
Send internal infos			I		Internal information about the PLUG is send to the sendbuffer	

Other commands

Command	Codes				Remarks	
Buzzer output	ESC	Y	S	n1	The buzzer output becomes **n1** =0 (off); =1 (on); =2...255 (on for **n1**/10s)	0
Wait (pause)	ESC	X		Time	Wait **Time**/10s before next command is executed	
RS232 settings	ESC	+	R	bbBaudrate[1], RS485, Flash	RS232: Default: **Baudrate =** 115200; **RS485** = 0-255 (7 is default); **Flash** =0 (settings valid until next reset); =1 (save settings permanently)	
SPI settings			S	Mode, Data Order, Flash	Set SPI-**Mode**[0...3] and **Data Order** (=0 MSB first; =1 LSB first).**Flash** =0 settings valid until next reset, =1 save permanently. The default SPI interface is set to Mode 3 MSB.	

Baud	Error
2400	+0.16
4800	+0.16
9600	+0.16
19200	+0.16
38400	+0.16
57600	+0.16
115200	+0.16

Command	Codes				Remarks	
I²C settings			I	Address, Flash	Set I²C address of module. **Flash** =0 settings valid until next reset, =1 save permanently. The default address is set to 0xDE (0x6F)	

1) 32-bit value range (for binary transmission first low then high byte)

I/O/ DIGITAL/ PWM

Command	Codes				Remarks	
Read input port	ESC	Y	R	n1	n1 =0 (Read all input ports as binary value to sendbuffer); =1..8 (Read input port n1)	
Port scan on/off			A	n1	The automatic scan of the input port is n1 =0 (deactivated); =1 (activated)	1
Invert input port			I	n1	The input port is n1 = 0 (evaluated normal); =1 (evaluated inverted)	0
Redefine input bitmacro			D	n1, n2, n3	Input port **n1**=1..8 is assigned by falling edge **n2**=0 to BitMacro **n3**=0..255 Input port **n1**=1..8 is assigned by rising edge **n2**=1 to BitMacro **n3**=0..255	
Define output port			M	Mask	Define output ports according binary **Mask** (1=output; 0=input)	0
Write output port			W	n1,n2	**n1**=0: Set all defined output ports in accordance with **n2** (=binary value) **n1**=1..8: Reset output port **n1** (**n2**=0); set (**n2**=1); invert (**n2**=2)	
PWM settings	ESC	Y	O	0, ffFrequency*, Highphase, Lowphase	Set PWM **ffFrequency** (16-Bit; 2...24kHz) **Highphase** / **Lowphase** = Relation high-time to low-time e.g. 2,1 to get 66% high and 33% low	

*16-bit value range (for binary transmission first low then high byte)

Port mask (binary):

	B7 I/O 8	B6 I/O 7	B5 I/O 6	B4 I/O 5	B3 I/O 4	B2 I/O 3	B1 I/O 2	B0 I/O 1
Value after PowerOn	0	0	0	0	0	0	0	0

ANALOG INPUT/ ANALOG OUTPUT

Command group to parametrize and read the analog input and output of the module. The module has two 12-bit analog inputs and one 8-bit analog output

Analogue input

Command	Codes				Remarks	
Calibration	ESC	V	@	Channel, xx1[1)]	Calibration procedure is as follows: 1.) Apply defined voltage (1..3.1V) to AIN1 (channel1) or AIN2 (channel2) 2.) Run this command with channel information **Channel**=1...2 and **xx1**=voltage value [mV] (16-Bit) e.g. 3.1V on AIN1; Command: '#V@1,3100;	
Enable/disable AIN scan			A	n1	**n1** =0 (disables input scan for AIN1 and AIN2); =1 (enable input scan)	0
Send analog value			D	Channel	Voltage in [mV] will be sent (to sendbuffer) for **Channel**=1...2	
Limit for analog macro			K	Channel, Lower limit, Upper limit, Hysteresis	Sets two limits for **Channel** =1...2. **Lower limit** [mV/20]; **Upper limit** [mV/20]; **Hysteresis** [mV] Related to this limits several analog macros can be started automatically.	0,0,0
Bargraph for AIN1/AIN2			B	Channel, no	Assigns bargraph **no**=1...20 to analogue input channel **Channel**=1...2 (It is possible to assign more than one bargraph to an analogue input). Define start- end values (aw, ew) forbargraph in [mV/20]	
Redraw bargraph			R	Channel	Redraw all bargraphs defined for **Channel**=1...2	
Digital value font	ESC	V	F	Channel, n1	Set font **n1** for channel **Channel**=1...2	
Digital value zoom			Z	Channel, x, y	Set zoom factor for **Channel**=1...2; **x** = x-factor (1...4); **y** = y- factor (1...4)	1,1
Digital value angle			W	Channel, n1	Set writing angle for **Channel**; **n1** =0 (0°); =1 (90°)	0
Digital values / scaling			E	Channel, Format string ..., NUL	Set user value for **Channel**=1..2. **Format string**: "mV1=uservalue1;mV2=uservalue2". 'NUL' ($00) = termination Assign two voltages (0..3100mV) 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"	
Send digital value			S	Channel	This will send current voltage as formated string for **Channel**=1...2 to sendbuffer	
Display on terminal			T	Channel	Show formated string of **Channel**=1...2 on terminal layer	
Show user value			G	Channel, x, y	Show formated string of **Channel**=1...2 at coordinate **x**, **y**	

Analogue output

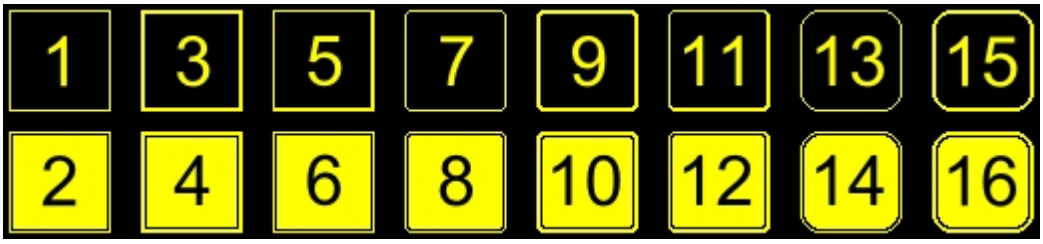
Command	Codes				Remarks	
User value font	ESC	V	F	128, n1	Set font **n1** for analogue output	
User value zoom			Z	128, x, y	Set zoom factor for analogue output; **x** = x-factor (1...4); **y** = y- factor (1...4)	
User value angle			W	128, n1	Set writing angle for analogue output; **n1** =0 (0°); =1 (90°)	
Digital values / scaling			E	128, Format string ..., NUL	Set user value for analogue output. **Format string**: "mV1=uservalue1;mV2=uservalue2". 'NUL' ($00) = termination Assign two voltages (0..3100mV) 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"	
Set analogue value	ESC	V	O	128, Value, Time, Ramp	Output 8-Bit analog (Pin 14) **Value**. Current analog output increases or decreases according to chosen **Ramp** (=0 linear; =1 accelerating → linear; =2 linear → slowing; =3 acceleration → linear → slowing; =4 User (#VU...) in **Time**/10s	
Define Lookup table (ramp)			U	128, n1,...,n100	Set 100 values for ramp (range 0...100)	
Bargraph for analogue output			B	128, no	Assigns bargraph **no**=1...20 to analogue output Define start- end values (aw, ew) forbargraph in [mV/20]	
Send user value			S	128	This will send current output voltage as formated string for to sendbuffer	
Display on terminal			T	128	Show formated string of analogue output on terminal layer	
Show user value			G	128, x, y	Show formated string of analogue output at coordinate **x**, **y**	

1) 16-bit value range (for binary transmission first low then high byte)

TOUCH

Settings

Command	Codes				Remarks	
Touch frame	ESC	A	E	Frame	The frame type for the display of touch keys/switches is set with **Frame**	1
Radio group for switches			R	nr	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.	



Integrated frame and key shapes

Touch label font

Command	Codes				Remarks	
Label font	ESC	A	F	nr	Set font with the number **nr** (0...15) for touch key label	0
Label zoom factor			Z	x, y	**x** = x-zoom factor (1...4); **y** = y-zoom factor (1...4)	1,1
Additional line spacing			Y	n1	Insert n1 pixel (0...15) between two lines of text as additional line spacing	0
Label angle			W	n1	Text output angle; n1 =0 (0°); =1 (90°)	0

Keys/ switches

Command	Codes				Remarks	
Define touch key (key remains depressed as long as there is contact)	ESC	A	T	x1, y1, x2, y2, Down Code, Up Code, Text ..., NUL	'T': The area from **x1**, **y1** to **x2**, **y2** is defined as a key. 'U': Image no. **n1** isl loaded to **x1**, **y1** 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	x1, y1, n1, Down Code, Up Code, Text ..., NUL		

Command	Codes				Remarks	
Invert touch key			N	Code	The touch key with the assigned return **Code** is inverted manually	
Define touch switch (status of the switch toggles after each contact)			K	x1, y1, x2, y2, Down Code, Up Code, Text ..., NUL	'K': The area from **x1**, **y1** to **x2**, **y2** is defined as a switch. 'J': Image no. **n1** is loaded to **x1**,**y1** 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	x1, y1, n1, Down Code, Up Code, Text ..., NUL		
Set touch switch			P	Code, n1	The status of the switch is changed by means of a command (**n1**=0=off; **n1**=1=on).	

Touch/drawing areas

Command	Codes				Remarks	
Define drawing area	ESC	A	D	x1, y1, x2, y2, n1	A drawing area is defined. You can then draw with a line width of **n1** within the corner coordinates **x1**, **y1** and **x2**, **y2**	
Define free touch area			H	x1, y1, x2, y2	A freely usable touch area is defined. Touch actions (down, up and drag) within the corner coordinates **x1**, **y1** and **x2**, **y2** are sent.	

Menu function

Command	Codes				Remarks	
Define touch key with menu function	ESC	A	M	x1, y1, x2, y2, Down Code, up Code, MNU Code, Text ..., NUL	The area from **x1**, **y1** to **x2**, **y2** 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.	

Global settings

Command	Codes				Remarks	
Touch query on/off	ESC	A	A	n1	Touch query is deactivated (**n1**=0 or activated (**n1**=1)	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

Bargraph/ Slider

Command	Codes				Remarks	
Set bar by touch	ESC	A	B	nr	The bargraph with the number **nr** is defined for input by touch panel.	
Send bar value automatically			Q	n1	The Automatic transmission of a new bargraph 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

Replies

Command	Codes				Remarks	
Query radio group	ESC	A	G	nr	The down code of the activated switch from radio group **nr** is placed in the sendbuffer	
Query touch switch			X	nr	The status of the switch (off=0; on=1) is placed in the send buffer.	

Delete touch area

Command	Codes				Remarks	
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. When **n1**=0, the area remains visible on the display; when **n1**=1, the area is deleted.	
			V	x, y, n1	Remove the Touch area that includes the coordinates **x**, **y** from the touch query. **n1**=0: area remains visible; **n1**=1: Delete area	

REPLIES

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' the host can read out the sendbuffer. This can be done per polling, alternatively pin 13 'SBUF' shows with Low-level that data is ready to transmit.

Automatic responses (placed into sendbuffer)

ESC	A	1	Code	Response from the 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 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 **Value** must be activated (see the '#AQ' command).
				After the '#BS' command, the current **Value** of the bar with the number **no** is transmitted.
ESC	N	1	Code	After a menu item is selcted by touch, the selected menu item **Code** is transmitted. It is only transmitted if no touch macro is defined with the number **Code**.
				After the '#NS' command, the currently selected menu item is transmitted. **Code**=0: no menu item is selected.
ESC	T	0		If automatic opening of a touch menu is disabled (see '#NT'), this request is sent to the host computer. The host can then open the touch menu with the 'ESC N T 2' 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 '#YA command. It is only transmitted when there is no corresponding port/bit macro defined!
ESC	H	3	Type, x, y	The following is transmitted in the case of a free touch area event: **Type** =0: is release; =1: is touch; =2: is drag within the free touch area at the coordinates x, y
ESC	X	2	Code, Value	After the '#AX' command, the current status (**Value**=0 or 1) of the touch switch **Code** is transmitted.
ESC	G	2	no, Code	After the '#AG' command, the **Code** of the active touch switch in the radio group **no** is sent.
ESC	Y	2	no, Value	After the '#YR' 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
ÊSC	D	3	no, Lo-Value, High-Value	After the '#VD' command, the currently analogue value of the channel **no**=1...2 is sent. (**Value** = 0..3100 mV)
ESC	V	Num	Version string ...	After the '#SV' command, the version of the PLUG firmware is transmitted as a string e.g. "EA PLUGL128-6 V1.0 T+"
ESC	J	Num	Projectname string ...	After the '#SJ' 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 '#SI' command, internal information is sent by PLUG (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)

ESC	W	Num	Channel, Format string	After the '#VS' command, the formated string of the analog **Channel** (ADC: 1..2/ DAC: 128) is transmitted

Response without length specification

ESC	U	L	x, y, Image data...	after the '#UH' command, a hard copy is sent in BLH Format. **x**, **y** = Start coordinates of the hard copy (upper corner) BLH-Data: 2 Byte: Width, height (in Pixel)+ amount of bytes of image data amount = ((width+7)/8*height

COMMAND EXAMPLES

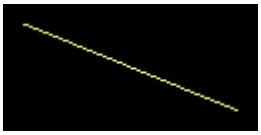
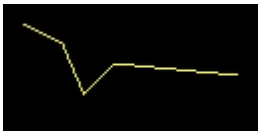
DISPLAY

Screensaver

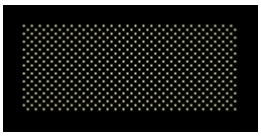
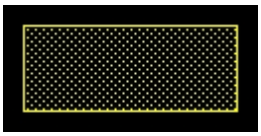
#DZ2,1, #DX0,5,	
#DZ2,1, #DX1,10,  PICTURE: nr,<file> Image was saved using the Kit Editor.(Image number.: 1)	
#DZ1,1, #DW10,50,5,50,	
	
	
#DZ3,1, #DV10,5,	
	

LINE/ POINT/ BOX

Draw line/ rectangle

#GD10,10,117,53	
#GD10,10,30,20, #GW40,45, #GW55,30, #GW117,35,	
#GR10,10,117,53	

Draw area/ box

#RM10,10,117,53,2	
#RO10,10,117,53,2	
#RR10,10,117,53,15	

TEXT/ STRING/ CHARACTER

Settings

#ZF6,	Font number 6 (Swiss 30 Bold proportional)

Place string

#ZL10,20,"Left"	
#ZC63,20,"Center"	
#ZR117,20,"Right"	

BITMAP/ PICTURE

Settings

#UZ2,2	Zoom: x=2; y=2

Load image

#UI17,0,1  PICTURE: nr,<file> Image was saved using the Kit Editor.(Image number.: 1)	
#UL39,7,BLH data	

BLH image data:

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1B	55	4C	27	07	18	18	00	00	00	00	00	00	00	00	00
00	18	00	01	3C	00	03	7E	00	03	E7	00	03	DB	80	03
BD	C0	07	7E	E0	0E	FF	70	1D	FF	B8	0B	FF	D0	07	FF
E0	07	FF	E0	06	18	60	06	18	60	06	18	60	07	F8	60
07	F8	60	07	F8	60	00	00	00	00	00	00	00	00	00	

BLH format

The BirmapEdit from the LCD-Tools package to edit/convert images into/from BLH-format.

Structure of an image file in BLH-format:

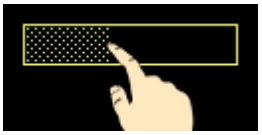
Description	Number of bytes
Image width	1
Image height	1
Image data	((width+7/8) · height The image is stored from top to down. One byte stands for 8 horizontal pixel on the screen (MSB left; LSB right; 0=black; 1=yellow)

	Bit Nr.								Bit Nr.			
	7	6	5	4	3	2	1	0	7	6	5	4
Byte 1	■	■	■	■					■	■	■	■
Byte 3	■	■									■	■
Byte 5	■											■
Byte 7	■				■			■				■
Byte 9												
Byte 11												
Byte 13						■	■					
Byte 15												
Byte 17	■			■					■			■
Byte 19	■				■	■	■	■				■
Byte 21	■	■									■	■
Byte 23	■	■	■	■					■	■	■	■
Byte 2												
Byte 4												
Byte 6												
Byte 8												
Byte 10												
Byte 12												
Byte 14												
Byte 16												
Byte 18												
Byte 20												
Byte 22												
Byte 24												

The complete BLH-file:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0x00	0C	0C	0F	00	3F	C0	7F	E0	76	E0	FF	F0	FF	F0	F9	F0
0x10	FF	F0	6F	60	70	E0	3F	C0	0F	00						

BARGRAPH/ SLIDER

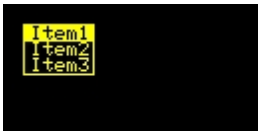
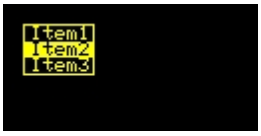
#BR1,10,10,117,30,0,100,1,2,	
#BA1,60,	
#AB1, Response of the display (sendbuffer): <ESC>B	

MENU/ TOUCHABLE

Settings

#NF2, (menu box)	Font number 2 (6x8 monospaced)
#AF2, (touch menu)	Font number 2 (6x8 monospaced)

Menu box commands (control not by touch)

#ND10,10,1,"Item1|Item2|Item3"	
#NN	

Touch menu

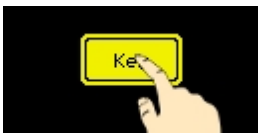
#AM10,10,50,25,0,0,1,"UCMenu|Item1|Item2|Item3"	
Response of the display (sendbuffer): <ESC>N	

TOUCH

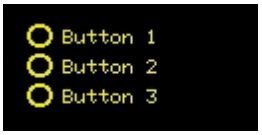
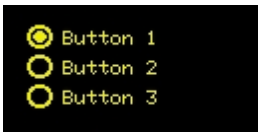
Settings

#AE9,	Frame type: 9
#AF2,	Font number: 2 (6x8 monospaced)

Key

#AT39,14,89,39,1,0,"Key"	
Response of the display (sendbuffer): <ESC>A	

Switch (Radiogroup)

#AR1, #AJ10,10,1,1,0,"RButton 1" #AJ10,25,1,2,0,"RButton 2" #AJ10,40,1,3,0,"RButton 3" #AR0,  PICTURE: nr,<file1>,<file2> Die beiden Bilder wurden mit dem Kit Editor im internen Speicher des Display abgespeichert (Bildnr.: 1)	
#AP1,1,	
Antwort des Displays (Sendepuffer): <ESC>A	

Touch/Drawing area

#AD0,0,127,63,2,	

KITEDITOR

The EA KIT Editor combines 3 functions:

- The editor itself which allows a simple definition of the macros, pictures and fonts like a standard text editor.
- The compiler which translates the text into the uploading code and shows up syntax error.
- The transmitter which search the right connection and uploads the data into the EA PLUG-Series.

The KitEditor is part of the EA LCD-Tools. These also include the necessary compiler and other tools like BitmapEdit or LCDterminal. You can download LCD-Tools from our website.

ELECTRICAL CHARACTERISTICS GENERAL (T_A=20°C; V_{in}=3,3V)

Value	Condition	min.	typ.	max.	Unit
Operating temperature	without touch panel	-40	-	80	°C
Operating temperature	with touch panel	-20	-	70	°C
Storage temperature		-30	-	80	°C
Storage humidity	<40°C	-	-	80	% RH
Operating voltage V_{in}		3,1	3,3	5,25	V
Output voltage V_{out}		-	-	3,1	V
		-	-	100	mA
Input low voltage		-0,3	-	0,93	V
Input high voltage (except AIN1 / AIN2)		2,32	-	3,4	V
Input high voltage AIN1 / AIN2		2,03	-	3,4	V
Output low voltage		-	-	0,6	V
Output high voltage		2,4	-	-	V
Output (I/O1...I/O4)	low level	-	-	360	mA
	high level[1)]	-	100	-	kΩ
Output current (I/O5...I/O8)		-	-	5	mA
I/O pull-up resistor	built-in	10	20	100	kΩ
I²C-bus pull-up	built-in	-	4,7	-	kΩ

1) close solder bridge (SB1...SB4) to activate pull-up

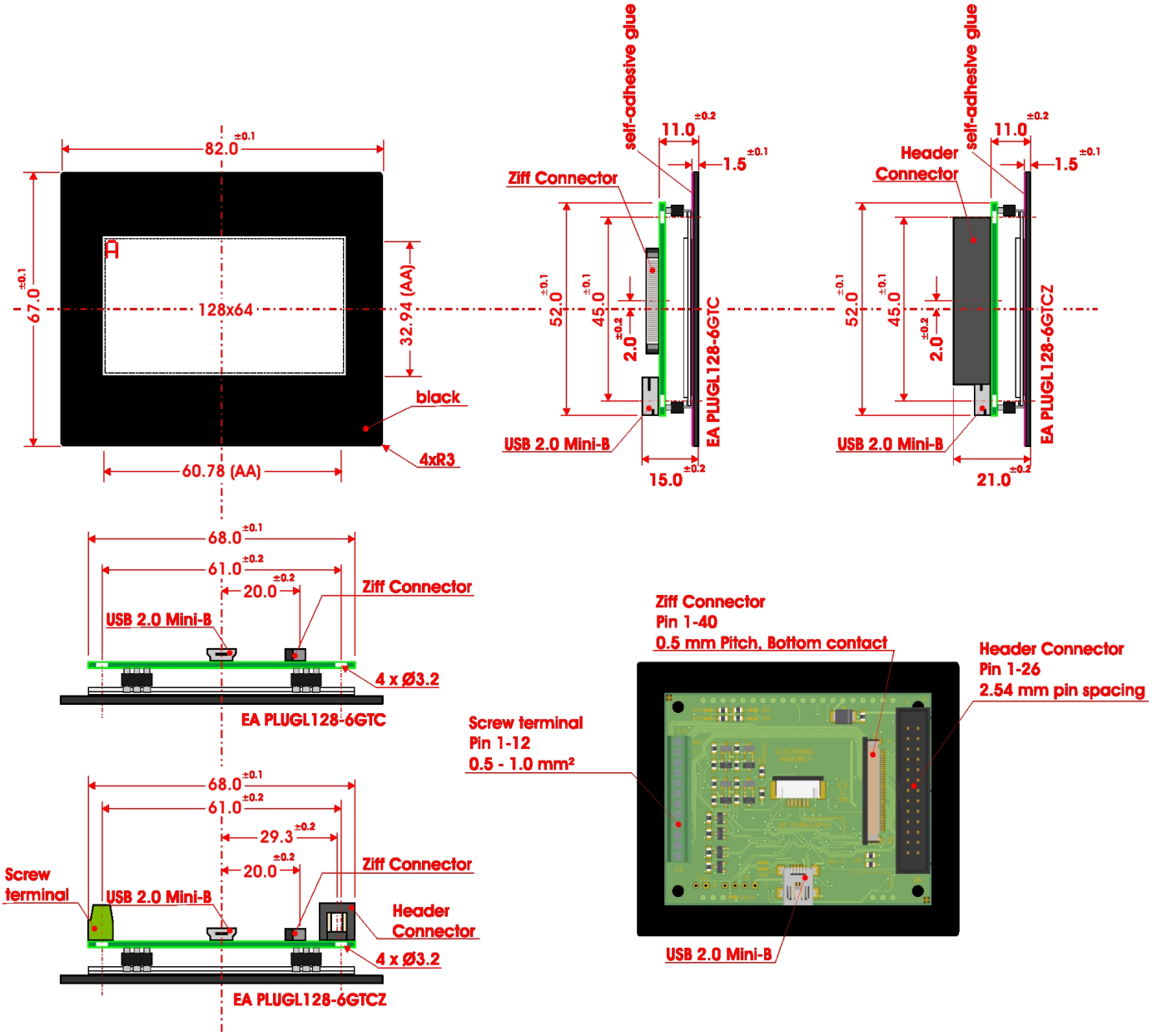
ELECTRICAL CHARACTERISTICS PLUGL128-6

Value	Condition	min.	typ.	max.	Unit
Supply current @3,3V	all pixel on	-	176	-	mA
	all pixel off	-	20	-	mA
	display off	-	15	-	mA
Supply current @5V	all pixel on	-	114	-	mA
	all pixel off	-	19	-	mA
	display off	-	15	-	mA

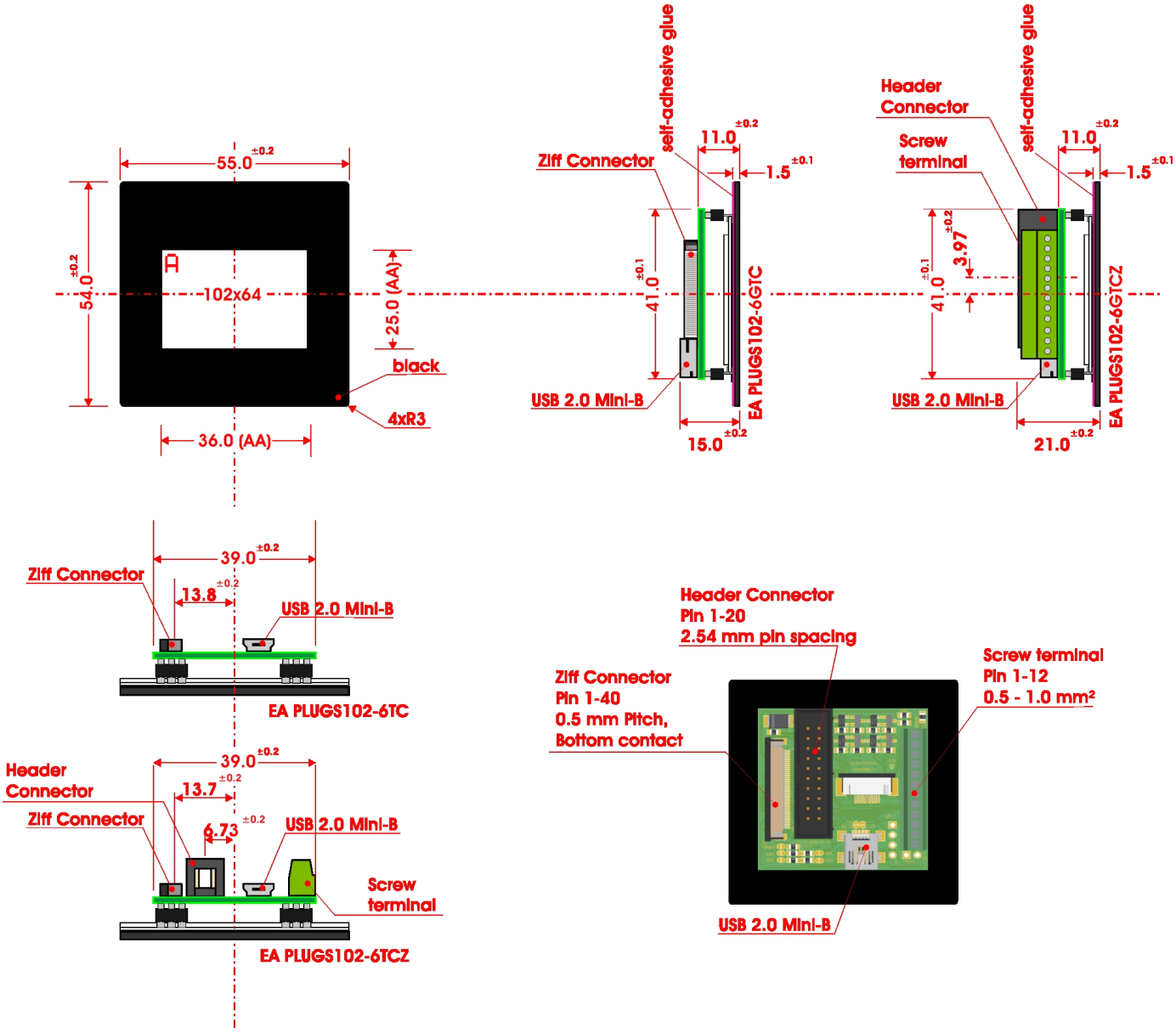
ELECTRICAL CHARACTERISTICS PLUGS102-6

Value	Condition	min.	typ.	max.	Unit
Supply current @3,3V	all pixel on	-	137	-	mA
	all pixel off	-	25	-	mA
	display off	-	15	-	mA
Supply current @5V	all pixel on	-	97	-	mA
	all pixel off	-	23	-	mA
	display off	-	15	-	mA

DIMENSIONS EA PLUGL128-6



DIMENSIONS EA PLUGS102-6



Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View EA PLUGL128-6GTC on WIN SOURCE
- Display Visions Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management