



EA eDIPTFT43-A compiler manual

Dezember 2008
© ELECTRONIC ASSEMBLY GmbH

Table of Contents

Part I eDIPTFTcompiler	2
Part II Syntax rules	3
Part III Compiler Options	5
1 General	5
2 Transfer	6
3 Fonts	7
LogFontWidth	7
ExportOverview	8
ExportWinfont	9
WinFont	10
Font	10
4 Bitmaps	11
MaxSize	11
MaxColorDepth	11
MakeTransparent	11
DoRLE	11
Pattern	12
Border	12
Button	12
Picture	13
Animation	13
5 Macros	14
SystemMacros	14
Macro	14
TouchMacro	14
BitMacro	14
PortMacro	14
MatrixMacro	15
ProcessMacro	15
AnalogMacro	15
Part IV EA eDIPTFT43-A commands	16
1 Terminal	16
2 Draw	17
3 Text	19
4 Bitmap	20
5 Animation	21
6 Bargraph	22
7 Macros	23
8 Touch	25
9 Backlight	28
10 Digital IO-Port	29
11 Analogue Input	30
12 Other commands	31

1 eDIPTFTcompiler

General

The EA eDIPTFT43-A is able to store many pictures, fonts and macros in internal FLASH memory. The EA KIT Editor is a powerful, free of charge software tool to create those macros and to store the pictures and fonts very easily.

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 eDIPTFT43-A.

2 Syntax rules

ESC	The ESC character (\$1B, 27d) is represented by the number sign '#'. The escape character must always be the first character in a line (except for tabs and spaces). This is followed by command letters and any parameters.
Comma	The comma is used to separate the parameters of a macro.
Numbers	All numbers are converted to binary values. Decimal, hexadecimal and binary numbers can be written. Example: 163(dez) = \$A3(hex) = %10100011(bin)
Comments	Comments must begin with a semicolon. Example: ; this is a comment
Text	Text (strings) must be enclosed within quotation marks "" or ''. ASCII numbers can also be entered directly. Example (output of "abc-def"): #ZL 0,0, "abc",45,'def'
Commands	Command letters and parameters specified in the EA eDIPTFT43-A data sheet are valid. Two exceptions facilitate the creation of command lines: <ol style="list-style-type: none">1. The <NUL> is appended automatically by the compiler. This means commands in which a string is output, the <NUL> no longer has to be entered as the end identifier. Example: #ZL 0,0,"Text"2. In the Send bytes command, the number of bytes to be sent is not specified; this number is calculated automatically by the compiler. Example: #SB 1,2,"Test"
Constants	Words without quotation marks are interpreted as constants, which have to be defined first. The name of a constant can have be up to 29 characters and must begin with a letter followed by letters, numbers or underscores. Up to 1024 constants can be defined. Please note that Compiler Options like e.g. INFO or MACRO can not be used. Example: CORNER_X=5; the word CORNER_X is replaced with immediate effect by the value 5.
Upper / lower case	No difference is made between upper case and lower case.

Calculating

The 4 basic mathematical operations +, -, * and / can be applied to constants and numbers. Round brackets can be used, and multiplication and division come before addition and subtraction.

Example: #RL X,Y, X+WIDTH, Y+HEIGHT

following C-style operations are also possible:

- pre/post increment and decrement: ++, --; e.g: ++a, b++, --c, d--
- shift and bit operations: <<, >>, &, |, ^
- combined operators: *=, /=, +=, -=, <<=, >>=, &=, |=, ^=

During compiling procedure all constants are calculated and transformed to fixed numbers.

3 Compiler Options

3.1 General

`eDIPTFT43-A "title"` Defines EA eDIPTFT43-A as target. "title" is a short description for the project. It is shown on the display when uploading the FLASH memory of the module. "title" can be read out by the command "ESC S J". Max. 32 character will be stored; more are allowed but will be suppressed.

`DESTINATION "new.df"` Specifies a new file name for the DATA-FLASH upload file.

`INCLUDE <file>` Includes the contents of the file <file> to be used in this actual file. This makes it possible to divide a project up into a number of source files. The file should have the extension *.kmi.

`PATH <path>` Sets a new path to find the following files.

`CODETABLE: nr` A code table is useful adapt different ASCII tables. With that, the ASCII code can be changed for some single character (e.g. "ä", "ß"). Up to 255 different code tables nr (1..255) can be defined. nr = 0 will disable all conversion.

Example:

```
CodeTable: 1 ; use codetable 1 for *.FXT fonts with
DOS-Code
'€' = 128
'äöüÄÖÜß' = $84,$94,$81, $8E,$99,$9A, $E1
```

3.2 Transfer

`COMx: baud`

With this statement the COM port and baud rate is defined when the EA eDIPTFT43-A is connected to the RS-232 (e.g. via EA 9777-1RS232),.

`USB: "device"`

If the EA STARTeDIPTFT4 is connected to the USB, "device" is "eDIP Programmer" (only Windows 2000/XP/Vista).

`RS485ADR: adr`

Selects the eDIP with RS485 address "adr" before uploading the macros.
"adr" can be a number from 0..255.

`VERIFY`

Verifies the complete contents of the FLASH memory after upload.

3.3 Fonts

3.3.1 LogFontWidth

LOGFONTWIDTH: n1

Each character in proportional font does have an individual width. The statement LOGFONTWIDTH provides the width for all characters in form of a table. The result is in LOG file (find it in project directory).
 n1 > 0: specifies the count of column
 n1 = 0: no table will be generated

Example:

```
LogFontWidth: 4
WinFont: 9, "Arial", 0, 0, 32, 127, 24
```

Output in Logfile:

```
Import WinFont "Arial", ANSI
height: 24 dots, used codes: 32..127, 5182 bytes
width: 32:' '= 7 33:'!' = 8 34:'"' = 9 35:'#' = 13
36:'$' = 13 37:'%' = 21 38:'&' = 16 39:''' = 5
40:'(' = 8 41:')' = 8 42:'*' = 9 43:'+' = 14
44:', ' = 7 45:'-' = 8 46:',' = 7 47: '/' = 7
48:'0' = 13 49:'1' = 13 50:'2' = 13 51:'3' = 13
52:'4' = 13 53:'5' = 13 54:'6' = 13 55:'7' = 13
56:'8' = 13 57:'9' = 13 58:':' = 7 59: ';' = 7
60:'<' = 14 61:'=' = 14 62:'>' = 14 63:'?' = 13
64:'@' = 24 65:'A' = 15 66:'B' = 16 67:'C' = 17
68:'D' = 17 69:'E' = 16 70:'F' = 15 71:'G' = 19
72:'H' = 17 73:'I' = 6 74:'J' = 12 75:'K' = 16
76:'L' = 13 77:'M' = 19 78:'N' = 17 79:'O' = 19
80:'P' = 16 81:'Q' = 19 82:'R' = 17 83:'S' = 16
84:'T' = 14 85:'U' = 17 86:'V' = 15 87:'W' = 23
88:'X' = 15 89:'Y' = 16 90:'Z' = 15 91:'[' = 7
92:'\' = 7 93:']' = 7 94:'^' = 12 95:'_' = 13
96:'`' = 8 97:'a' = 13 98:'b' = 14 99:'c' = 12
100:'d' = 14 101:'e' = 13 102:'f' = 7 103:'g' = 14
104:'h' = 14 105:'i' = 5 106:'j' = 6 107:'k' = 12
108:'l' = 6 109:'m' = 20 110:'n' = 14 111:'o' = 13
112:'p' = 14 113:'q' = 14 114:'r' = 8 115:'s' = 12
116:'t' = 7 117:'u' = 14 118:'v' = 11 119:'w' = 17
120:'x' = 11 121:'y' = 12 122:'z' = 12 123:'{' = 8
124:'|' = 6 125:'}' = 8 126:'~' = 14 127:'•' = 18
```

3.3.2 ExportOverview

`EXPORTOVERVIEW: n1`

This statement enables the generation of a BMP file for all following fonts and animations.
This is good to get an overview which character / pictures are available.

- n1= 1: only fonts will be exported
- n1= 2: only animations will be exported
- n1= 3: fonts and animations will be exported
- n1= 0: no export at all


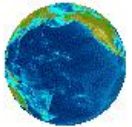
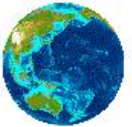



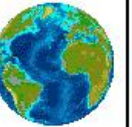
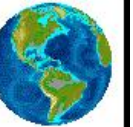
Example:

```
ExportOverview: 3
WinFont: 9, "Arial", 0, 0, 32, 127, 48 ; export "Font9_Arial_ANSI_N_32-127_48.bmp"
Animation: 1 <Erde.G16> ; export "Animation1_Erde_G16_1-8.bmp"
```

Font9_Arial_ANSI_N_32-127_48.bmp:

+ Lower Upper	S0 (0)	S1 (1)	S2 (2)	S3 (3)	S4 (4)	S5 (5)	S6 (6)	S7 (7)	S8 (8)	S9 (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	{		}	~	□

Animation1_Erde_G16_1-8.bmp:

\$1 (1)	\$2 (2)	\$3 (3)	\$4 (4)	\$5 (5)	\$6 (6)	\$7 (7)	\$8 (8)
							

3.3.3 ExportWinfont

EXPORTWINFONT: n1

n1= 1: Exports all following win fonts as a FXT-File. The file is stored in project path.

To change or add some character it can easily be edited with the "KitEditor.exe" or another simple text editor .

n1= 0: no FXT-export will be done.

ExportWinFont: 1

WinFont: 9, "Arial", 0, 0, 66, 67, 8 ; use only character 'B' and 'C'

Font9_Arial_ANSI_N_66-67_8.fxt:

```
; First Nr   : 66
; Last  Nr   : 67
; Typ       : monospaced
; width     : 7
; height    : 8
```

66 \$42 'B'

```
#####..
#....#.
#....#.
#####.
#....#.
#....#.
#....#.
#....#.
#####..
```

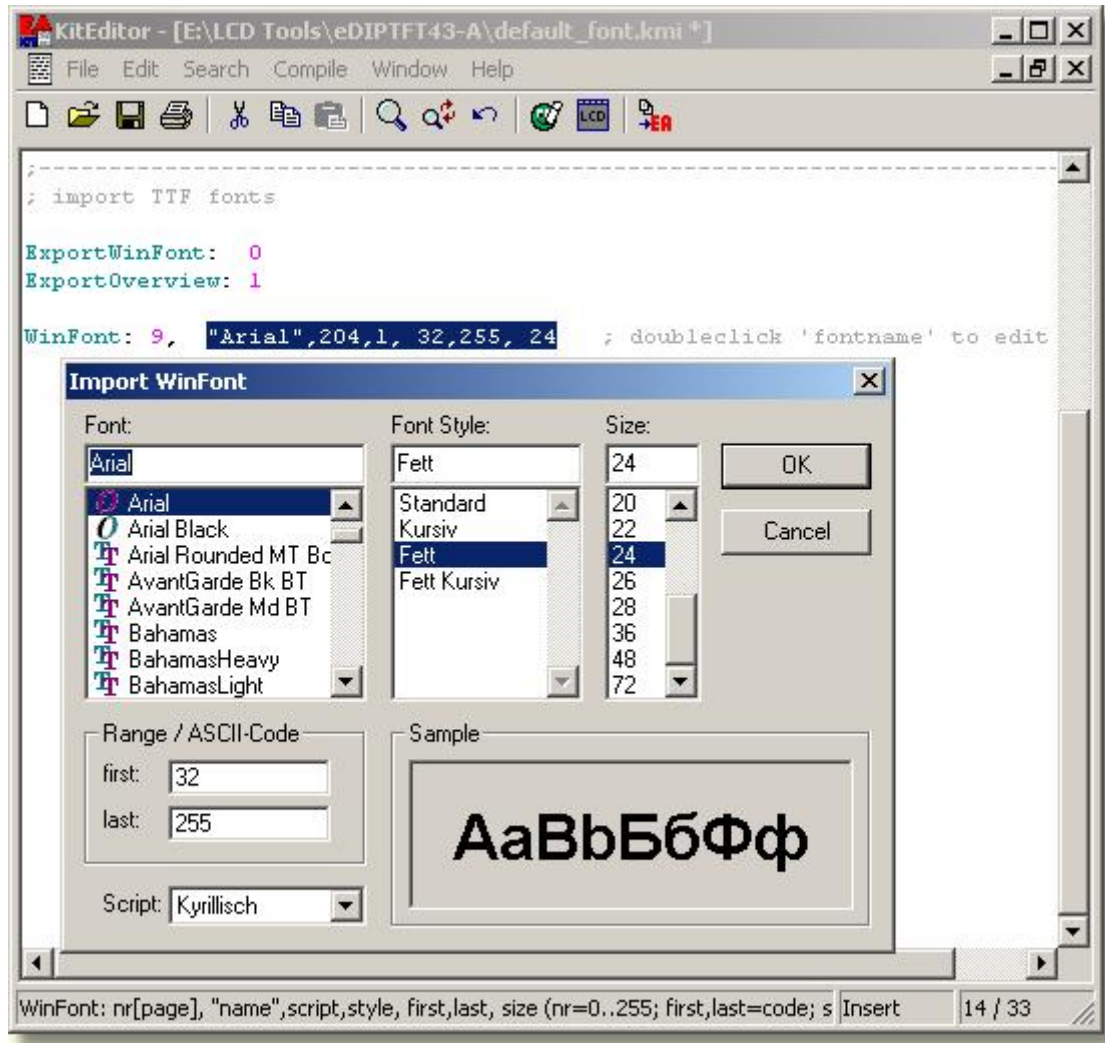
67 \$43 'C'

```
..###..
.#...#.
#.....
#.....
#.....
#.....
.#...#.
..###..
```

3.3.4 WinFont

```
WINFONT: nr, "name",script,style, first,last, size
WINFONT: nr[page], "name",script,style, first,last, size
```

Defines a Windows font and assigns to font number nr (0..255). The best is to double click on "name" to edit all parameter. Optionally different winfonts can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.



3.3.5 Font

```
FONT: nr,<file>
FONT: nr[page],<file>
```

Defines a font file which will be assigned to the number nr (0..255). <file> can be FXT, G16. Optionally different fonts can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.4 Bitmaps

3.4.1 MaxSize

`MAXSIZE: width,height,p` If a picture or bitmap is larger than `width x height` dots (default: 480x272) the size can be reduced automatically to fit to the display.

`p = 1` reduce proportional

`p = 0` reduce non proportional to exact "width" and "height", distortions are possible

3.4.2 MaxColorDepth

`MAXCOLORDEPTH: bitpixel` Reduces color depth of bitmaps. This saves memory space. Attention: This may effect to the quality of the image.

`bitpixel = 1`: black&white (monochrome)

`bitpixel = 4`: change to 4 bit color depth

`bitpixel = 8`: change to 8 bit color depth

`bitpixel = 16`: change to 16 bit color depth

3.4.3 MakeTransparent

`MAKETRANSSPARENT: type` Parts of a picture can be switched to transparent for a nice overlie of a picture on the background. GIF, TGA and G16 files may already include any transparent information. If not (or a BMP / JPEG format is used) one color can be defined to become transparent. The color will be picked out from 1 of 9 positions (type).

1 = Top Left

2 = Top Center

3 = Top Right

4 = Middle Left

5 = Middle Center

6 = Middle Right

7 = Bottom Left

8 = Bottom Center

9 = Bottom Right

0 = no transparency

3.4.4 DoRLE

`DORLE: type`

Large pictures and fonts need a lot of memory space. RLE compression may reduce data size to save memory space. RLE compression is loss-free.

`type = 0`: RLE compression is disabled

`type = 1`: RLE compression is always on

`type = 2`: RLE is made automatically when compressed file is smaller than the non-compressed one. (default)

3.4.5 Pattern

```
PATTERN: nr <file>
```

```
PATTERN: nr[page]<file>
```

Pattern are used to fill a [box](#)^[17], a [bargraph](#)^[22] or to draw a [line](#)^[17]. The statement PATTERN defines the pattern nr (0..255) as the bitmap <file>. The bitmap size need to be 8x8 dots exactly. <file> can be BMP, GIF, JPG, TGA or G16.

Optionally different pattern can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.4.6 Border

```
BORDER: nr <file>
```

```
BORDER: nr[page] <file>
```

```
BORDER: nr <file1>,<file2>
```

```
BORDER: nr[page] <file1>,<file2>
```

A border is used for [rectangle](#)^[17], [bargraph](#)^[22] and [touch key/switch](#)^[25]. A border can be scaled. The statement BORDER defines a bitmap <file> for a border nr (0..255). <file> can be BMP, GIF, JPG, TGA or G16. The bitmap size need to be 24x24 dots exactly.

When used for a touch key or a switch, 2 different bitmaps can be defined as <file1> and <file2>. <file1> is for touch key/ switch and <file2> will be used if the touch key/ switch is pressed.

Optionally different border can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.4.7 Button

```
BUTTON: nr <file>
```

```
BUTTON: nr[page] <file>
```

```
BUTTON: nr <file1>,<file2>
```

```
BUTTON: nr[page] <file1>,<file2>
```

A button is used for a touch key or a switch ([see Touch commands](#)^[25]). Note that using a button for touch key/ switch is not so flexible as a border (width and height is fix). The statement BUTTON defines a bitmap <file> for a button nr (0..255). <file> can be BMP, GIF, JPG, TGA or G16.

Optionally 2 different buttons can be defined as <file1> and <file2>. <file1> is for touch key/ switch and <file2> will be used if the touch key/ switch is pressed.

Optionally different buttons can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.4.8 Picture

```
PICTURE: nr,<file>
PICTURE: nr[page],<file>
```

It is convenient to store all bitmap in FLASH; this will save transfer time via serial interface. The statement PICTURE defines a bitmap <file> with nr (0..255). Optionally different pictures can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

Following image 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.
- G16: internal eDIPTFT format, incl. RLE and transparency

All pictures are converted into internal G16 format with RLE encoding (Compileroption [DORLE](#)^[11]).

Too big pictures are resized proportional (Compileroption [MAXSIZE](#)^[11]).

It is also possible to reduce the colordepth (Compileroption [MAXCOLORDEPTH](#)^[11]).

One Color can be defined as transparent for bitmaps without transparency (Compileroption [MAKETRSPARENT](#)^[11] :)

The pictures can be used with the [Bitmap commands](#)^[20].

3.4.9 Animation

```
ANIMATION: nr <file>
ANIMATION: nr[page] <file>
ANIMATION: nr <file1>,<file2>,...
ANIMATION: nr[page] <file1>,<file2>,...
```

An animation is a self-running picture. The statement ANIMATION defines an animation image with nr (0..255).

<file> can be an animated GIF, G16.

<file1>,<file2>,... two or more single bitmaps BMP, GIF, JPG, TGA or G16

Note that max. 4 animations can run at the same time.

The animation images can be used with the [Animation commands](#)^[21].

Optionally different animations can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5 Macros

3.5.1 SystemMacros

`POWERONMACRO:` All commands defined in this macro will be automatically executed when the power supply is switched on.

`RESETMACRO:` All commands defined in this macro will be automatically executed when an external reset on Pin 5 is done.

`WATCHDOGMACRO:` All commands defined in this macro will be automatically executed when the display hangs up.

`BROWNOUTMACRO:` All commands defined in this macro will be automatically executed when VDD brakes down to 4,3V or lower.

3.5.2 Macro

`MACRO: nr`
`MACRO: nr[page]`

Defines a normal macro with number nr (0..255). This macro will be executed with the command `#MN nr`^[23]. Optionally different normal macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.3 TouchMacro

`TOUCHMACRO: nr`
`TOUCHMACRO: nr[page]`

Defines a touch macro with number nr (0..255). This macro will be executed if a touch key / switch with the return code nr is defined and the touch key /switch is pressed or by command `#MT nr`^[23]. Optionally different touch macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.4 BitMacro

`BITMACRO: nr`
`BITMACRO: nr[page]`

Defines a bit macro with number nr (0..255). bitmacros will start voltage at a single line IN 1..8 (bit) will change or by command `#MB nr`^[23]. BitMacro 1..8 are good for falling edge at input 1..8. BitMacro 9..16 are good for rising edge at input 1..8. Since firmware V1.1 it is possible to change the assignment between BitMacro and input with command `#YD n1,n2,n3`^[29]. Optionally different bit macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.5 PortMacro

`PORTMACRO: nr`
`PORTMACRO: nr[page]`

Defines a port macro with number nr (0..255). This macro will be executed if the matching binary bit code is put on the pins IN1..IN8 or by command `#MP nr`^[23].

Optionally different port macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.6 MatrixMacro

MATRIXMACRO: nr
MATRIXMACRO: nr[page]

Defines a matrix macro with number nr (0..255). This macro will be executed if the one of the connected key pad is pressed or by command `#MX nr`^[23].

Matrix Macro 1..64: start when keypressed.

Matrix Macro 0: start after release of key.

Since firmware V1.1 it is possible to change the assignment between keynumber and matrixmacro with command `#YX n1,n2,n3`^[29].

The relating pins for matrix keyboard need to be defined with the command `#YM n1,n2,n3`^[29].

Optionally different matrix macro can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.7 ProcessMacro

PROCESSMACRO: nr
PROCESSMACRO: nr[page]

Defines a process macro with number nr (0..255). This macro will be executed automatically (see command `#MD no,type,n3,n4,zs`^[23]) or by command `#MC nr`^[23].

Optionally different process macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages.

3.5.8 AnalogMacro

ANALOGMACRO: nr
ANALOGMACRO: nr[page]

Defines an analogue macro with number nr (0..255). The macro will be executed automatically when the relating voltage is on the pins AIN1 and AIN2 (see table below) or by command `#MV nr`^[23].

Since firmware V1.1 it is possible to change the assignment between analogurmacrofunction 0..19 and analogmacro number with command `#VM n1,n2`^[30].

Optionally different analog macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. reens in different languages.

Macro nr		
AIN1	AIN2	Macro starts at
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

4 EA eDIPTFT43-A commands

4.1 Terminal

Terminal definition:

Set terminal color	#FT fg,bg	Preset color 1..32 for terminal mode fg=foreground color; bg=background color
Define window	#TW 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
Terminal off	#TA	Terminal display is switched off; outputs are rejected
Terminal on	#TE	Terminal display is switched on;

Cursor commands:

Position cursor	#TP C,L	C=column; L=line; origin upper-left corner (1,1)
Cursor on/off	#TC n1	n1=0: Cursor is invisible; n1=1: Cursor flashes;
Save cursor position	#TS	The current cursor position is saved
Restore cursor position	#TR	The last saved cursor position is restored

Terminal output:

String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal
Output version	#TV	The version no. is output in the terminal e.g. "EA eDIPTFT43-A V1.0 Rev.A"
Output projectname	#TJ	The macrofile-projectname is output in the terminal e.g. "init / delivery state"
Output interface	#TQ	The used interface is output in the terminal e.g "RS232,115200 baud, ADR: \$07"
Output informationen	#TI	The terminal is initialized and cleared; the software version, hardware revision, macrofile-projectname and CRC-checksum are output in the terminal

Special ASCII-characters:

Form feed	FF (dec:12)	The contents of the screen are deleted and the cursor is placed at pos. (1,1)
Carriage return	CR (dec:13)	Cursor to the beginning of the line on the extreme left
Line feed	LF (dec:10)	Cursor 1 line lower, if cursor in last line then scroll

4.2 Draw

Display commands (effect on the entire display):

Set display color	#FD fg,bg	Defines color 1..32 for display and areas fg=foreground color; bg=background color
Set display orientation	#DO n1	n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270° (0°+180°=480x272; 90°+270°=272x480)
Delete display	#DL	Delete display contents (all pixels to background color)
Fill display	#DS	Fill display contents (all pixels to foreground color)
Fill display with color	#DF n1	Fill complete display content with color n1=1..32
Invert display	#DI	Invert display content

Draw straight lines and points:

Set color for lines	#FG fg,bg	Colors 1..32 (0=transparent) fg=color for line; bg=pattern background
Draw rectangle	#GR x1,y1,x2,y2	Draw four straight lines as a rectangle from x1,y1 to x2,y2
Draw straight line	#GD x1,y1,x2,y2	Draw straight line from x1,y1 to x2,y2
Continue straight line	#GW x1,y1	Draw a straight line from last end point to x1, y1
Draw point	#GP x1,y1	Set a point at coordinates x1, y1
Point size/line thickness	#GZ n1,n2	n1=X-point size (1 to 15) n2=Y-point size (1 to 15)
Pattern	#GM n1	Set line/point pattern no n1=1..255; 0=do not use pattern (see compiler option PATTERN ↗ :)

Change/draw rectangular areas:

Delete area	#RL x1,y1,x2,y2	Delete an area from x1,y1 to x2,y2 (fill with background color)
Fill area	#RS x1,y1,x2,y2	Fill an area from x1,y1 to x2,y2 (fill with foreground color)
Fill area with color	#RF x1,y1,x2,y2,n1	Fill an area from x1,y1 to x2,y2 with color n1=1..32
Invert area	#RI x1,y1,x2,y2	Invert an area from x1,y1 to x2,y2
Copy area	#RC x1,y1,x2,y2,x3,y3	Copy an area from x1,y1 to x2,y2 to new position x3,y3

Draw rectangular areas with pattern:

Pattern color	#FM fg,bg	Color 1..32 (0=transparent) for monochrome pattern fg=foreground; bg=background color
Area with fill pattern	#RM x1,y1,x2,y2,nr	Draw an area from x1,y1 to x2,y2 with pattern nr (see compiler option PATTERN ^[12] :)
Draw box	#RO x1,y1,x2,y2,nr	Draw a rectangle x1,y1 to x2,y2 and fill with pattern nr (see compiler option PATTERN ^[12] :)

Draw frames/borders:

Set color for border	#FR c1,c2,c3	Set color 1..32 (0=transparent) for border segments c1=frame outside; c2=frame inside; c3=filling
Set border type	#RE nr,n2	Set border type nr=1..255 border angle: n2=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270° (see compiler option BORDER ^[12] :)
Draw border box	#RR x1,y1,x2,y2	Draw a border box from x1,y1 to x2,y2

4.3 Text

Text settings:

Set text color	#FZ fg,bg	Color 1..32 (0=transparent) for string and character fg=text color; bg=background color
Set font	#ZF n1	Set font with the number nr (see compiler option FONT : or WINFONT :)
Font zoom factor	#ZZ n1,n2	n1 = X-zoom factor (1x to 8x) n2 = Y-zoom factor (1x to 8x)
Additional width/height	#ZY n1,n2	n1=0..15: additional width left/right n2=0..15: additional height top/bottom
Spacewidth	#ZJ n1	n1=0: use spacewidth from font n1=1: same width as a number n1>=2: width in dot
Text angle	#ZW n1	Text output angle n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°

Text output:

Output string left justified	#ZL x,y,"text..."	A string (text...) is output left justified to x,y several lines are separated by the character ' ' (\$7C, pipe) character '\' (\$5C, backslash) cancels the special function of ' ' and '\'
Output string centered	#ZC x,y,"text..."	A string (text...) is output centered to x,y several lines are separated by the character ' ' (\$7C, pipe) character '\' (\$5C, backslash) cancels the special function of ' ' and '\'
Output string right justified	#ZR x,y,"text..."	A string (text...) is output right justified to x,y several lines are separated by the character ' ' (\$7C, pipe) character '\' (\$5C, backslash) cancels the special function of ' ' and '\'
String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal

4.4 Bitmap

Bitmap settings:

Set bitmap colors	#FU fg,bg	painting color 1..32 (0=transparent) for monochrome bitmaps fg=foreground color; bg=background color
Image zoom factor	#UZ n1,n2	n1 = X-zoom factor (1x to 8x) n2 = Y-zoom factor (1x to 8x)
Image angle	#UW n1	output angle of the image n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Mirror Image	#UX n1	n1=0: normal display n1=1: the image is mirrored horizontally
Transparency for color bitmaps	#UT 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 (GIF,TGA,G16) with actually background color

Output bitmaps:

Load internal image	#UI x1,y1,nr	Load internal image with the no (0 to 255) from the data flash memory to x1,y1 (see compiler option PICTURE ₁₃ :)
Load image	#UL x1,y1,data...	Load an image to x1,y1; data... = image in G16 format This command is only for serial interface, do not use this command in a macro !

Hardcopy:

Send hardcopy	#UH x1,y1,x2,y2	After this command, the image extract is sent (to sendbuffer) in G16 format. With the program "BitmapEdit.exe" from the LCD-Tools you can convert the G16 format to a Windows *.BMP image
---------------	-----------------	--

4.5 Animation

Animation settings:

Set animation colors	#FW fg,bg	color for 1..32 monochrome animation images fg=foreground color; bg=background color
Animation zoom factor	#WZ n1,n2	n1 = X-zoom factor (1x to 8x) n2 = Y-zoom factor (1x to 8x)
Animation angle	#WW n1	output angle of the animation image n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Mirror animation	#WX n1	n1=0: normal display n1=1: the animation image is mirrored horizontally
Transparency for color animation	#WT 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

Animation bitmap:

Load single image	#WI x1,y1,nr,n2	Load from animation image nr=0..255 (see compiler option ANIMATION ^[13] :) the single image n2 to x1,y1
-------------------	-----------------	---

Animation process:

Define animation process	#WD no,x1,y1,nr,type,time	Define an animationprocess no=1..4 at position x1,y1 (=left top edge) with animation image nr=0..255. (see compiler option ANIMATION ^[13] :) 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	#WY no,type	Assign a new type=1..7 to animationprocess no=1..4
Change animation time	#WC no,time	Assign a new time=0..255 to animationprocess no=1..4
Next animation image	#WN no	Show the next image from animationprocess no=1..4
Previous animation image	#WP no	Show the previous image from animationprocess no=1..4
Show animation image	#WF no,n2	Show image n2 from animationprocess no=1..4
Run to animation image	#WM no,n2	Run animationprocess no=1..4 from actually image to image n2
Stop animationprocess	#WL no	Stop animationprocess no=1..4 and clear last image with actually background color

4.6 Bargraph

Bargraph settings:

Set color for bargraph	#FB fg,bg,fc	Set colors 1..32 for bargraph fg=foreground; bg=background; fc=color for frame
Bargraph pattern	#BM nr	Pattern for bargraph nr=1..255; nr=0 no pattern/solid (valid for type=0..3) (see compiler option PATTERN [12] :)
Bargraph border	#BE nr	Border for bargraph nr=0..255 (valid for type=4..7) (see compiler option BORDER [12] :)
Bargraph linewidth	#BB n1	Linewidth for bargraph n1=1..255; n1=0 automatic (valid for type=2,3,6,7)

Define/use bargraphs:

Define bargraph	#BR #BL #BO #BU no,x1,y1,x2,y2, sv,ev,type	Define bargraph no=1..20 to L(ef), R(ight), O(up), U(down) x1,y1,x2,y2 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
Update bargraph	#BA no,value	Set and draw the bargraph no=1..20 to the new value
Draw bargraph	#BN no	Entirely redraw the bargraph with the number no=1..20
Send bargraph value	#BS no	Send the current value of bargraph number no=1..20 to sendbuffer
Delete bargraph	#BD 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

Bargraph user values - Format text output:

User value color	#FX fg,bg	Set color 1..32 for bargraph user value fg=foreground; bg=background color
User value font	#BF nr	Set font nr for bargraph user value (see compiler option FONT [10] : or WINFONT [10] :)
User value zoom	#BZ n1,n2	Set zoom factor for bargraph user value n1=X-Zoom 1x..8x; n2=Y-Zoom 1x..8x
User value additional width/height	#BY n1,n2	n1=0..15: additional width left/right; n2=0..15: additional height top/bottom for bargraph user value;
User value angle	#BW n1	Set writing angle for bargraph user value n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°;
User values / scaling	#BX no,x1,y1, "FormatString"	Define user value for bargraph no=1..20. Output is always right justified to x1,y1 Format String: "bv1=uservalue1;bv2=uservalue2" 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"

4.7 Macros

Run macros:

Run macro	#MN nr	Call the (normal) macro with the number nr (max. 7 levels) (see compiler option MACRO [14] :)
Run touch macros	#MT nr	Call the touch macro with the number nr (max. 7 levels) (see compiler option TOUCHMACRO [14] :)
Run port macro	#MP nr	Call the port macro with the number nr (max. 7 levels) (see compiler option PORTMACRO [14] :)
Run bit macro	#MB nr	Call the bit macro with the number nr (max. 7 levels) (see compiler option BITMACRO [14] :)
Run matrix macro	#MX nr	Call the matrix macro with the number nr (max. 7 levels) (see compiler option MATRIXMACRO [15] :)
Run process macro	#MC nr	Call the process macro with the number nr (max. 7 levels) (see compiler option PROCESSMACRO [15] :)
Run analogue macro	#MV nr	Call the analogue macro with the number nr (max. 7 levels) (see compiler option ANALOGMACRO [15] :)

Macro settings:

Disable macros	#ML 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	#MU 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	#MK 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	#MW	the current macro/image page is saved (when used in process macros)
Restore macro/image page	#MR	the last saved macro/image page is restored

Automatic (normal-) macros:

Macro with delay	#MG 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	#ME 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	#MA 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	#MJ 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	#MD 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	#MZ 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	#MS 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

4.8 Touch

Touch presets:

Touch border colors	#FE n1,n2,n3, s1,s2,s3	Set the colors 1..32 (0=transparent) for touch border (#AT #AK) n=normal; s=selected; 1=frame outside; 2=frame inside; 3=filling
Touch border form	#AE nr,n2	nr=1..255 border number (see compiler option BORDER [12] :) nr=0 no border n2=angle 0=0°; 1=90°; 2=180°; 3=270°
Touch button colors	#FC nf,nb, sf,sb	Set the colors 1..32 for monochrome touch buttons (#AU #AJ) n=normal; s=selected; f=foreground; b=background
Touch button number	#AC nr,n2,n3,n4	nr=0..255 button number (see compiler option BUTTON [12] :) n2=button angle 0=0°; 1=90°; 2=180°; 3=270° n3=X-Zoom 1..8; n4=Y-Zoom 1..8
Radio group for switches	#AR n1	n1=0: newly defined switches do not belong to a group n1=1..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.

Label font presets:

Font color	#FA nf,sf	Color 1..32 for touch labeling nf=normal fontcolor; sf= fontcolor for selection
Label font	#AF nr	Set font with the number nr for touch key label (see compiler option FONT [10] : or WINFONT [10] :)
Label zoom factor	#AZ n1,n2	n1=X-zoom factor (1x to 8x); n2=Y-zoom factor (1x to 8x)
Additional width/height	#AY n1,n2	n1=0..15: additional width left/right n2=0..15: additional height top/bottom
Label angle	#AW n1	Label output angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Offset for selected label	#AO n1,n2	n1=X-offset; n2=Y-offset n1,n2=0..7 (add +8 for negative direction)

Define touch key/switches:

Define touch key	#AT x1,y1,x2,y2, downCode,upCode, "text..." #AU x,y, downCode,upCode, "text..."	key remains depressed as long as there is contact
Define touch switch	#AK x1,y1,x2,y2, downCode,upCode, "text..." #AJ x,y, downCode,upCode, "text..."	status of the switch toggles after each contact

#AT: The area from x1,y1 to x2,y2 is drawn with actual border and defined as a key
 #AK: The area from x1,y1 to x2,y2 is drawn with actual border and defined as a switch
 #AU: The actual button is loaded to x,y and defined as a key
 #AJ: The actual button is loaded to x,y and defined as a switch
 'downCode': (1-255) return/touchmacro when key pressed
 'upCode': (1-255) return/touchmacro when key released
 (downCode/upCode = 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"

Define touch areas:

Define drawing area	#AD x1,y1,x2,y2, n1,fg	A drawing area is defined. You can then draw with a line width of n1 and color fg within the corner coordinates x1,y1 and x2,y2.
Define free touch area	#AH 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.
Set bar by touch	#AB n1	The bargraph with number n1 is defined for input by touch panel.

Global settings:

Touch query on/off	#AA n1	Touch query is n1=0: deactivated n1=1: activated
Touch key response	#AI n1	Automatic inversion when touch key touched n1=0: OFF n1=1: ON
Touch key response buzzer	#AS n1	Tone sounds briefly when a touch key is touched n1=0: OFF n1=1: ON
Send bar value on/off	#AQ n1	Automatic transmission of a new bar graph value by touch input n1=0: deactivated n1=1: is placed in the sendbuffer once at the end of input n1=2: changes are placed continuous into sendbuffer during input

Other touch functions:

Invert touch key	#AN code	The touch key with the assigned return code is inverted manually
Set touch switch	#AP code,n1	The status of the switch with the return code is changed to n1=0: OFF n1=1: ON
Query touch switch	#AX code	The status of the switch with the return code is placed in the sendbuffer (off=0; on=1)
Query radio group	#AG n1	down code of the activated switch from the radio group n1 is placed in the sendbuffer
Delete touch area by up- or down-code	#AL code, n1	The touch area with the return code is removed from the touch query (code=0: all touch areas). When n1=0, the area remains visible on the display When n1=1, the area is deleted
Delete touch area by coordinates	#AV x,y,n1	Remove the Touch area that includes the coordinates x1,y1 from the touch query. When n1=0, the area remains visible on the display When n1=1, the area is deleted

4.9 Backlight

LED backlight:

Illumination brightness	#YH n1	Set brightness of the LED illumination n1=0 to 100%.
Increase brightness	#YN	Increase brightness of the LED illumination (one step=1%)
Decrease brightness	#YP	Decrease brightness of the LED illumination (one step=1%)
Brightness changetime	#YZ n1	Time n1=0..31 in 1/10sec for changing brightness from 0 to 100%
Illumination on/off	#YL n1	LED n1=0: OFF; n1=1: ON n1=2 to 255: LED switched ON for n1/10sec
Assign bar with backlight	#YB no	Assign bar no=1..20 for changing brightness of the backlight
Save parameter	#Y@	Save the actual brightness and changetime for poweron to EEPROM

4.10 Digital IO-Port

I/O-ports:

Write output port	#YW n1,n2	n1=0: Set all 8 output ports in accordance with n2 (=8-bit binary) n1=1..8: Reset output port n1 (n2=0); set (n2=1); invert (n2=2)
Read input port	#YR 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=5V, 0=L-level=0V)
Port scan on/off	#YA n1	The automatic scan of the input port is n1=0: deactivated n1=1: activated
Invert input port	#YI n1	The input port is n1=0: evaluated normal n1=1: evaluated inverted
Redefine input bitmacro (firmware V1.1)	#YD 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)

External matrix keyboard:

Matrix keyboard	#YM 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)
Redefine matrixmacro for keys (firmware V1.1)	#YX n1,n2	Assign keynumber n1=1..65 with matrixmacro n2=0..255 After release the key n1=0 run matrixmacro n2=0..255

4.11 Analogue Input

Analogue inputs:

Calibration	#V@ ch,x1	Calibration procedure is as follows: 1.) Apply defined voltage (3..5V) to AIN1 or AIN2 2.) Run this command with channel information ch=1..2 and x1=voltage value in [mV] e.g. 4.0V on AIN1 command: #V@1,4000
Enable/disable AIN scan	#VA n1	n1=0 disables input scan for AIN1 and AIN2 n1=1 enable input scan
Send analog value	#VD ch	Voltage in [mV] will be sent (to sendbuffer) for channel ch=1..2
Limit for analog macro	#VK 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 several analogmacros can be started automatically.
Redefine analogmacro (firmware V1.1)	#VM n1,n2	Assign analogmacrofunction n1=0..19 with analogmacro number n2=0..255
Bargraph for AIN1/AIN2	#VB ch,bn	Assigns bargraph bn=1..20 to analogue input ch=1..2 (it is possible to assign more than one bargraph to an analogue input). Define start- endvalues for bargraph in [mV/20]
Redraw bargraph	#VR ch	Redraw all bar graphs defined for channel ch=1..2

Analogue in user values - Format text output:

User value color	#FV ch,fg,bg	Set color 1..32 for string output of channel ch=1..2 fg=foreground, bg=background color
User value Font	#VF ch,nr	Set font nr for channel ch=1..2 (see compiler option FONT ¹⁰ : or WINFONT ¹⁰ :)
User value zoom	#VZ ch,n1,n2	Set zoom factor for channel ch=1..2 n1=X-Zoom 1x..8x n2=Y-Zoom 1x..8x
User value additional width/height	#VY ch,n1,n2	n1=0..15: additional width left/right n2=0..15: additional height top/bottom for channel ch=1..2;
User value angle	#VW ch,n1	Set writing angle for channel ch=1..2 n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°;
User values / scaling	#VE ch,"FmtStr"	Set user value for channel ch=1..2 Format String: "mV1=uservalue1;mV2=uservalue2" Assign two voltages (0..5000mV) to user defined values max. range: 4 1/2 digits 19999 + decimal point ('.' oder ',') + sign e.g. display for 2000mV should be "-123.45" and "0.00" for 1000mV Format String: "2000=-123.45;1000=0"
Send user value	#VS ch	This will send current voltage as formatted string for channel ch=1..2 to sendbuffer
Display on terminal	#VT ch	Show formatted string of channel ch=1..2 on terminal window
Display user value	#VG ch,x1,y1	Show formatted string of channel ch=1..2 at coordinate x1,y1

4.12 Other commands

Send functions:

Send bytes	#SB data...	bytes are sent to the sendbuffer data... can be numbers or strings e.g #SB "Test",10,13
Send version	#SV	The version is sent as a string to sendbuffer e.g. "EA eDIPTFT43-A V1.0 Rev.A TP+"
Send projectname	#SJ	The macro-projectname is sent as a string to the sendbuffer e.g. "init / delivery state"
Send internal infos	#SI	Internal information about the edip is sent to the sendbuffer.

Other functions:

Define color	#FP 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)	#X n1	Wait n1/10sec before the next command is executed.
Set RS485 address	#KA 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	#YS n1	The tone output (pin 16) becomes n1=0:OFF; n1=1:ON; n1=2 to 255:ON for n1/10s