EA eDIPTFT43-A
compiler manual

August 2012

Table of Contents

1 Overview

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 `' '`.
It is possible to use Hex-values between curly brackets `{ }`.
ASCII numbers can also be entered directly.
Example (output of "abc-def-xyz"): `#ZL0,0,"abc",45,'def',{2D78797A}`

KitEditor: double click within the curly brackets or quotation marks opens a EditBox, use the mouse to select special characters.
Please make sure that you have selected the correct font (right click on the font and 'Select Font for EditBox')

Commands

Command letters and parameters specified in the EA eDIPTFT43-A data sheet are valid. Two exceptions facilitate the creation of command lines:

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 numeric constants, which have to be defined first. The name of a constant can have be up to 60 characters and must begin with a letter followed by letters, numbers or underscores. Up to 2000 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.

String Constants

A string-constant is a constant name between two exclamation marks
Example1: `!NAME! = "example text"`
Example2: `!NAME! = "abc",45,'def',{2D78797A}`

Upper / lower case

No difference is made between upper case and lower case.

3 Compiler Functions

Calculating The 4 basic mathematical operations +, -, * and / can be applied to numeric 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.

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

Follwing functions are available:

LO(value)	returns the Low-Byte
HI(value)	returns the High-Byte
SET_RGB(r,g,b)	returns a color constant with the three r,g,b values
GET_R(c)	returns the Red, Green or Blue value from a color constant
GET_G(c)	e.g. `color = SET_RGB($1E,$20,$3F)`
GET_B(c)	`#FP 16, GET_R(color), GET_G(color), GET_B(color)`
MIN(value1,value2,...)	returns the minimum value
MAX(value1,value2,...)	returns the maximum value
AVG(value1,value2,...)	returns the average value
RANDOM(min,max)	returns a random value from the range min..max
RANDOM(min,max,delta)	delta = maximum difference to the last random value
MOD(v, d)	the modulo function returns the remainder of the division v/d
SIN(w, a)	w = angle in tenth of degree
COS(w, a)	a = amplitude
TAN(w, a)	

to calculate the bounding box of images following functions are available:

PICTURE_W(nr)	**PICTURE_H**(nr)	for Images 24
PICTURE_W(nr, page)	**PICTURE_H**(nr, page)	
BUTTON_W(nr)	**BUTTON_H**(nr)	for Touchbuttons 24
BUTTON_W(nr, page)	**BUTTON_H**(nr, page)	
ANIMATION_W(nr)	**ANIMATION_H**(nr)	for Animations 25
ANIMATION_W(nr, page)	**ANIMATION_H**(nr, page)	
INSTRUMENT_W(nr)	**INSTRUMENT_H**(nr)	for Instruments 26
INSTRUMENT_W(nr, page)	**INSTRUMENT_H**(nr, page)	

to calculate the bounding box of strings following functions are available:

STRING_W(nr, par, font) **STRING_H**(nr, par, font) for internal Strings 10

STRING_W(nr, par, font, page) **STRING_H**(nr, par, font, page)

STRING_W(!NAME!,par,font) **STRING_H**(!NAME!,par,font) for Stringconstants 5

STRING_W(!NAME!,par,font,page) **STRING_H**(!NAME!,par,font,page)

nr = internal string number (see compiler option STRING: 10)
font = font number (eDIP command #ZF 36)

par = **STRING_P**(zoomX, zoomY, width, height, space, code)
this values needs the compiler to calculate the correct outline in functions STRING_W and STRING_H

zoomX, zoomY = zoom factor 1..8 (eDIP command #ZZ 36)
width, height = additional width/height 0..15 (eDIP command #ZY 36)
space = spacewidth (eDIP command #ZJ) 36
code = stringcode (eDIP command #ST 50)

Example:

```
String: 1, "Hello World"

font       = SWISS30B
stringcode = 1
zoomX      = 1
zoomY      = 1
addwidth   = 3
addheight  = 5
spacewidth = 0

Makro: MnPowerOn
       #ST stringcode
       #ZY addwidth,addheight
       #ZJ spacewidth

       #ZF font
       #ZZ zoomX,zoomY
       #FZ YELLOW,TRANSPARENT

par = STRING_P(zoomX,zoomY,addwidth,addheight,spacewidth,stringcode)
w = STRING_W(1,par,font)
h = STRING_H(1,par,font)
x = (XPIXEL-w)/2
y = (YPIXEL-h)/2
       #RF x,y, x+w-1, y+h-1, BLUE
       #ZL x,y, stringcode,2
```

String Functions A string-function converts a value into a string constant the function is between two exclamation marks. Following functions are available:

```
!STR(value, digits)!       for decimal numbers
!HEXSTR(value, digits)!    for hexadecimal numbers
!BINSTR(value, digits)!    for binary numbers
```

digits = 0: variable length
digits > 0: fix numbers of digits with leading zeros
digits < 0: fix numbers of digits with leading spaces

4 Compiler Options

4.1 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. Optionally you can choose another path for the destination file.
`INCLUDE <file>` `INCLUDE <file>,number`	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. The optional parameter (number) defines how often the file will be included.
`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`

4.2 Transfer

`AUTOSCAN: n1`	Scan baudrate for connected eDIP on COM/USB before programming n1=0: autoscan off, use `baud` for connecting and programming n1=1: autoscan on, search baudrate automatically and programm with baudrate `baud`
`COMx: baud`	With this statement the COM port and baud rate is defined.
`USB: baud, "device"`	With this statement the USB device and baud rate is defined. If the EA EVALeDIPTFT43 is connected to the USB, "device" is `"eDIP Programmer"`.

`RS485ADR: adr`	Selects the eDIP with RS485 address "adr" before uploading the macros. "adr" can be a number from 0..255. (see example INIT_with_RS485_address.KMC 68)

`VERIFY`	Verifies the complete contents of the FLASH memory after upload.

`DISABLEBIGIMAGES: size, "types"`

With this statement it is possible to replace big images with a placeholder. This is usefull during developement to reduce transfertime.
`size`: Image witch are greater than size [in KB] will be replaced by an rectangle
`"types"`: replace only following types: `F`=font, `P`=picture, `B`=button, `A`=animation, `I`=instrument

Example:

```
DisableBigImages: 10,"FPBAI"  ; all fonts, pictures, buttons, animations and instruments
                              greater than 10 kB are replaced with an rectangle
```

4.3 String

```
STRING: nr "text..."
STRING: nr[page]"text..."
```

The statement STRING defines and stores internal strings nr (1..255 without 10+13 because this codes are end of string codes).
The internal strings can be used with any command that uses strings e.g (#ZL,#ZC,#ZR,#ZB 36, #AT,#AK,#AU,#AJ 43, #BX 39,#IX 40,#VE 48)
After the stringcode, defined with #ST n1 50, internal strings are used. Optionally different strings 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. strings in different languages.

You can use the Compiler Functions 6 STRING_W, STRING_H and STRING_P to get the outline in pixels of the string.

(see How-to-use example String tables - EXPERT 107)

Example 1:

```
StringCode = $01

STRING: 1, "Hello World"

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

Example 2:

```
StringCode = $01

STRING: 1,    "Hello World  "
STRING: 1[1], "Hallo Welt   "
STRING: 1[2], "Ciao a tutti "

MACRO: MnPowerOn
       #ST StringCode

       #MK 0
       #ZL 10,10, StringCode, 1

       #MK 1
       #ZL 10,30, StringCode, 1

       #MK 2
       #ZL 10,50, StringCode, 1
```

4.4 Fonts

4.4.1 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 62.
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.

(see How-to-use example Place Strings - BEGINNER 69)

---

predfined fonts (`include <..\default_font.kmi>`):

```
; default fonts
FONT4x6   = 1
FONT6x8   = 2
FONT7x12  = 3
GENEVA10  = 4
CHICAGO14 = 5
SWISS30B  = 6
BIGZIF50  = 7
BIGZIF100 = 8
```

```
Path: <..\Fonts>

Font: FONT4x6,    <4x6.FXT>
Font: FONT6x8,    <6x8.FXT>
Font: FONT7x12,   <7x12.FXT>

Font: GENEVA10,   <GENEVA10.FXT>
Font: CHICAGO14,  <CHICAG14.FXT>
Font: SWISS30B,   <SWISS30B.FXT>

Font: BIGZIF50,   <BIGZIF50.FXT>
Font: BIGZIF100,  <BIGZIF100.FXT>
```

see Character Table
Terminal 8x8 51
Terminal 8x16 52

Font 4x6 53
Font 6x8 54
Font 7x12 55

Geneva 10 56
Chicago 14 57
Swiss 30 58

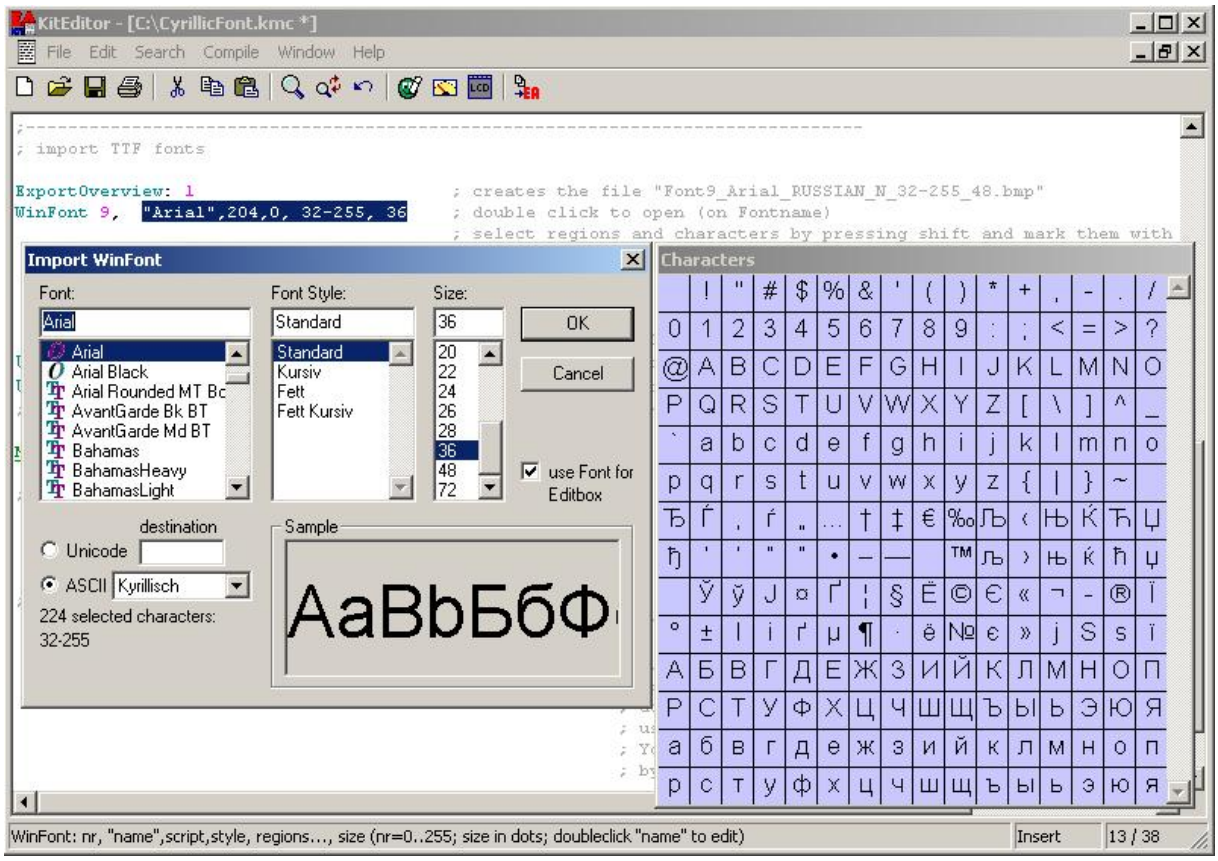
BigZif 50 59
BigZif 100 60

4.4.2 WinFont

```
WINFONT: nr, "name",script,style, regions.., size
WINFONT: nr[page], "name",script,style, regions.., 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.
Select the start-character by pressing the left mouse botton and move to the end-character.
Additonal regions can be selected with the SHIFT-key.
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.

(see How-to-use example Cyrillic font - BEGINNER 73)



4.4.3 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
```

4.4.4 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

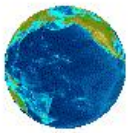
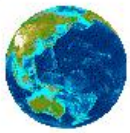
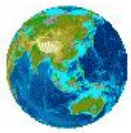
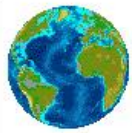
Example:

```
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	$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	{	|	}	~	□

Animation1_Erde_G16_1-8.bmp:

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

4.4.5 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'
..###..
.#...#.
#......
#......
#......
#......
.#...#.
..###..
```

4.5 Bitmaps

4.5.1 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.

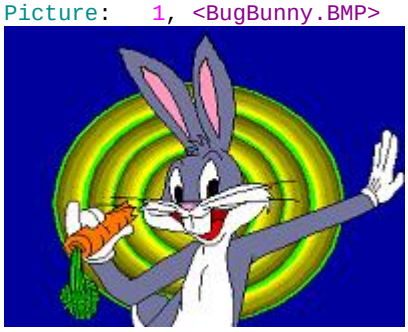
p = 1 reduce proportional
p = 0 reduce non proportional to exact "width" and "height", distortions are possible

---

Examples:

```
Picture:   1, <BugBunny.BMP>
```

```
MaxSize: 200,200,1
Picture:   1, <BugBunny.BMP>
```



```
MaxSize: 200,200,0
Picture:   1, <BugBunny.BMP>
```

4.5.2 MaxColorDepth

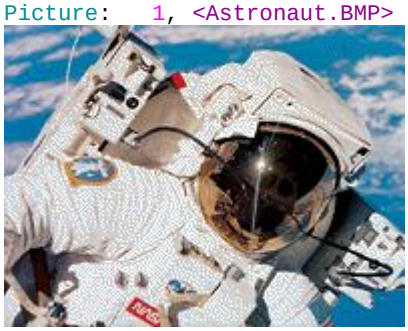
`MAXCOLORDEPTH: bitpixel` Reduces color depth of bitmaps. This saves memory space. Attention: This may effect to the quality of the image.

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

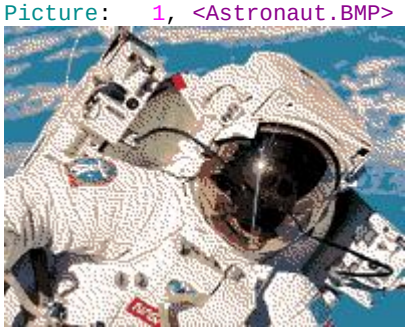
Examples:

```
MaxColorDepth: 16
Picture:   1, <Astronaut.BMP>
```

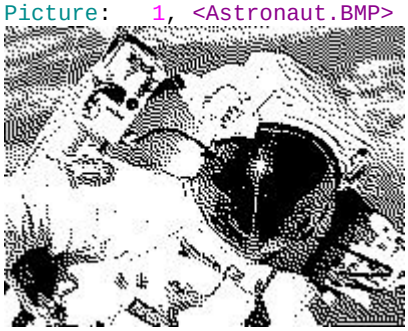
```
MaxColorDepth: 8
Picture:   1, <Astronaut.BMP>
```



```
MaxColorDepth: 4
Picture:   1, <Astronaut.BMP>
```



```
MaxColorDepth: 1
Picture:   1, <Astronaut.BMP>
```



4.5.3 Dithering

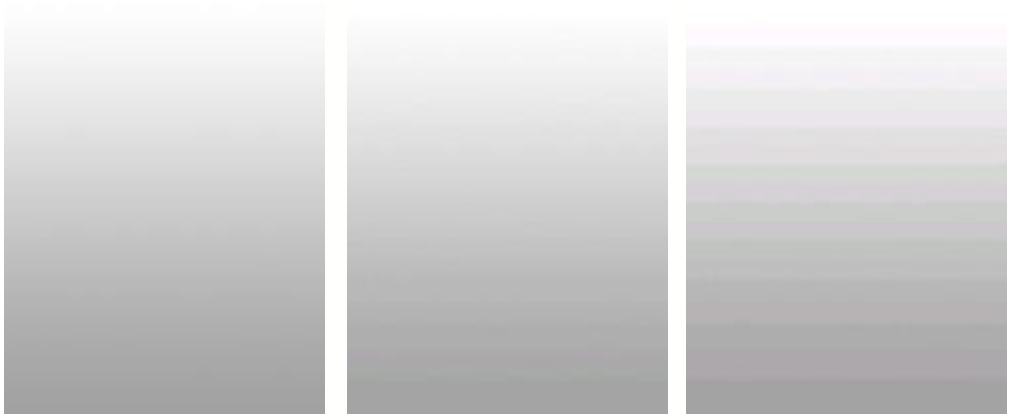
`DITHERING: n1`

The EA eDIPTFT43-A is a 16-Bit Hi-Color Display with 65536 colors. It is necessary to convert a 24-bit True-Color or 24-bit Color-Palette from RGB888 into RGB565 colorspace.

n1=0: no dithering, not used bits are truncated
n1=1: dithering is only on for 24-bit True-Color images (default)
n1=2: dithering is only on for 8-/4-bit images with 24-bit Color-Palette
n1=3: dithering is on for all 24-bit True-Color and 24-bit Color-Palette images

---

24-bit original BMP 115254 Byte	dithering = 1 DoRLE=0: 76830 Byte DoRLE=1: 49502 Byte	dithering = 0 DoRLE=0: 76830 Byte DoRLE=1: 12443 Byte



4.5.4 MakeTransparent

`MAKETRANSPARENT: type`

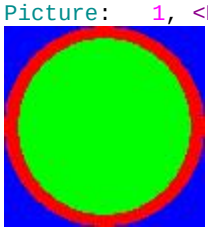
Parts of a picture can be switched to transparent for a nice overlie of a picture on the background. GIF, TGA, PNG and G16 files may already include any transparency information. If not (or a BMP / JPEG format is used) one color can be defined to become transparent. The color will be picked out from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

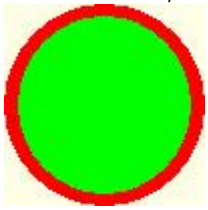
0 = no transparency (default)

Examples:

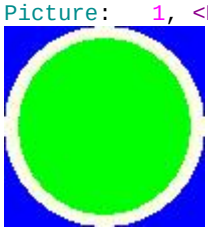
```
MakeTransparent: 0
Picture:   1, <Kreis.BMP>
```



```
MakeTransparent: 1
Picture:   1, <Kreis.BMP>
```



```
MakeTransparent: 2
Picture:   1, <Kreis.BMP>
```



```
MakeTransparent: 5
Picture:   1, <Kreis.BMP>
```

4.5.5 RemoveBorder

REMOVEBORDER: n1

With this compiler option it is possible to remove an additionally not used border
n1=0: off
n1=1: cut a single color boundary
n1=2: cut a single color boundary only before resize (see MAXSIZE 16)
n1=3: cut only a transparency boundary
n1=4: cut transparency boundary only before resize (default) (see MAXSIZE 16)

Examples:

```
MakeTransparent: 0
RemoveBorder:    0
Picture:   1, <toucan.BMP>
```

```
MakeTransparent: 0
RemoveBorder:    1
Picture:   2, <toucan.BMP>
```

```
MakeTransparent: 0
RemoveBorder:    3
Picture:   1, <toucan.BMP>
```

```
MakeTransparent: 1
RemoveBorder:    3
Picture:   2, <toucan.BMP>
```

```
MaxSize: 100,100,1
MakeTransparent: 0
RemoveBorder:    0
Picture:   1, <toucan.BMP>
```

```
MaxSize: 100,100,1
MakeTransparent: 0
RemoveBorder:    2
Picture:   2, <toucan.BMP>
```

```
MaxSize: 100,100,1
MakeTransparent: 0
RemoveBorder:    4
Picture:   1, <toucan.BMP>
```

```
MaxSize: 100,100,1
MakeTransparent: 1
RemoveBorder:    4
Picture:   2, <toucan.BMP>
```

4.5.6 AlphaThreshold

`ALPHATHRESHOLD: n1`

The image formats *.TGA and *.PNG may include an alpha channel for transparency

n1=0: alpha channel will be ignored
n1=1..255: threshold level for a single transparency color, like a mask (default=128)

4.5.7 DoGamma

`DOGAMMA: n1`

The image format *.PNG may include an gamma table

n1=0: ignore the image gamma information (default)
n1=1: use the image gamma information

4.5.8 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)

4.5.9 Compress

`COMPRESS: n1`

Compression for animation (generates difference images). Large animations need a lot of memory space. Compression may reduce data size to save memory space. The drawing time may also be reduced.
Compression is only useable for **cyclic** animations. It is not possible to use single sub images from a compressed animation (see commands #WI 38, #WF 38).

n1=0: compression off (default)
n1=1: compression on (only useable for cyclic animations)

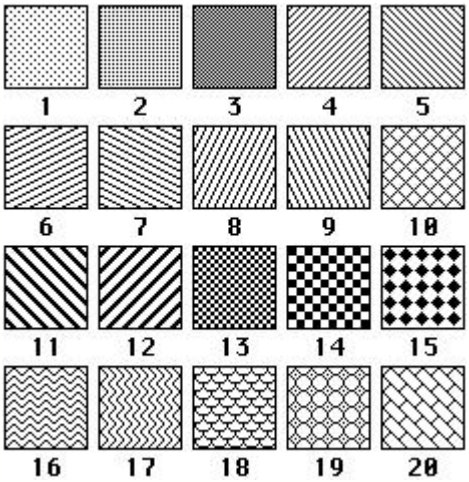
4.5.10 Pattern

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

Pattern are used to fill a box, a bargraph or to draw a line.
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, PNG 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.

---

predfined pattern (`include <..\default_pattern.kmi>`):




```
; default pattern (8x8 Bitmaps)
Path: <..\BITMAPS\monochrome\Pattern>

Pattern:  1  <pattern01.bmp>
Pattern:  2  <pattern02.bmp>
Pattern:  3  <pattern03.bmp>
Pattern:  4  <pattern04.bmp>
Pattern:  5  <pattern05.bmp>
Pattern:  6  <pattern06.bmp>
Pattern:  7  <pattern07.bmp>
Pattern:  8  <pattern08.bmp>
Pattern:  9  <pattern09.bmp>
Pattern: 10  <pattern10.bmp>
Pattern: 11  <pattern11.bmp>
Pattern: 12  <pattern12.bmp>
Pattern: 13  <pattern13.bmp>
Pattern: 14  <pattern14.bmp>
Pattern: 15  <pattern15.bmp>
Pattern: 16  <pattern16.bmp>
Pattern: 17  <pattern17.bmp>
Pattern: 18  <pattern18.bmp>
Pattern: 19  <pattern19.bmp>
Pattern: 20  <pattern20.bmp>
```

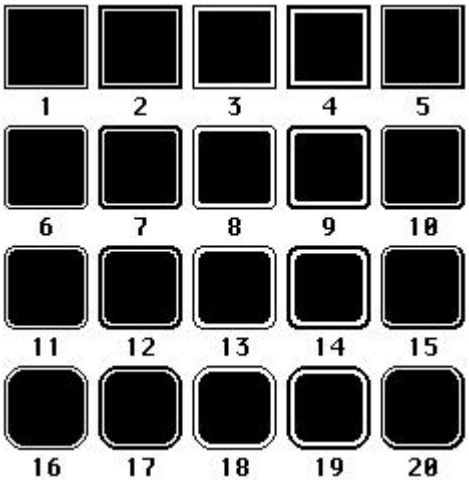
4.5.11 Border

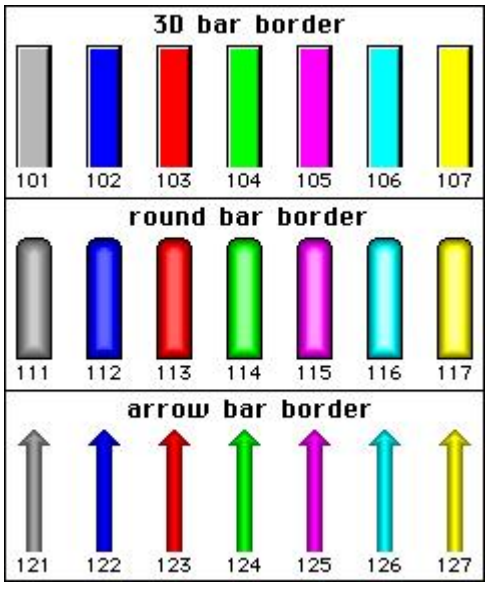
```
BORDER: nr <file>
BORDER: nr[page] <file>
BORDER: nr <file1>,<file2>
BORDER: nr[page] <file1>,<file2>
```

A border is used for rectangle 34, bargraph 39 and touch key/switch 43. 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, PNG 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.
(see How-to-use example Frame - BEGINNER 89)

---

predfined borders (`include <..\default_border.kmi>`):






```
; default border (24x24 Bitmaps)
Border:  1   <border01.bmp>
Border:  2   <border02.bmp>
Border:  3   <border03.bmp>
Border:  4   <border04.bmp>
Border:  5   <border05.bmp>
Border:  6   <border06.bmp>
Border:  7   <border07.bmp>
Border:  8   <border08.bmp>
Border:  9   <border09.bmp>
Border: 10   <border10.bmp>
Border: 11   <border11.bmp>
Border: 12   <border12.bmp>
Border: 13   <border13.bmp>
Border: 14   <border14.bmp>
Border: 15   <border15.bmp>
Border: 16   <border16.bmp>
Border: 17   <border17.bmp>
Border: 18   <border18.bmp>
Border: 19   <border19.bmp>
Border: 20   <border20.bmp>
Border: 31   <Register_Normal.bmp>,<..Selected.bmp>
Border: 32   <3Dgrey_Normal.bmp>,<..Selected.bmp>
; default bars (24x24 Bitmaps)
Border: 101   <Bar3Dgrey.G16>
Border: 102   <Bar3Dblue.G16>
Border: 103   <Bar3Dred.G16>
Border: 104   <Bar3Dgreen.G16>
Border: 105   <Bar3Dmagenta.G16>
Border: 106   <Bar3Dcyan.G16>
Border: 107   <Bar3Dyellow.G16>
Border: 111   <BarRoundGrey.G16>
Border: 112   <BarRoundBlue.G16>
Border: 113   <BarRoundRed.G16>
Border: 114   <BarRoundGreen.G16>
Border: 115   <BarRoundMagenta.G16>
Border: 116   <BarRoundCyan.G16>
Border: 117   <BarRoundYellow.G16>
Border: 121   <BarArrowGrey.G16>
Border: 122   <BarArrowBlue.G16>
Border: 123   <BarArrowRed.G16>
Border: 124   <BarArrowGreen.G16>
Border: 125   <BarArrowMagenta.G16>
Border: 126   <BarArrowCyan.G16>
Border: 127   <BarArrowYellow.G16>
```

4.5.12 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 43). 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, PNG 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.

You can use the Compiler Functions 6 `BUTTON_W` and `BUTTON_H` to get the outline in pixels of the buttons.

(see How-to-use example Glass button - EXPERT 78)

4.5.13 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.
- PNG: Portable Network Graphics indexed-color / truecolor / greyscale and transparency.
- G16: internal eDIPTFT format, incl. RLE and transparency

All pictures are converted into internal G16 format with RLE encoding (Compileroption `DORLE` 21:).
Too big pictures are resized proportional (Compileroption `MAXSIZE` 16:).
It is also possible to reduce the colordepth (Compileroption `MAXCOLORDEPTH` 17:).
One Color can be defined as transparent for bitmaps without transparency (Compileroption `MAKETRANSPARENT` 19`:`)
It is possible to cut a single color or transparency border (Compileroption `REMOVEBORDER` 20`:`).
You can use the Compiler Functions 6 `PICTURE_W` and `PICTURE_H` to get the outline in pixels of the picture.
The pictures can be used with the Bitmap commands. 37

(see How-to-use example BMP file - BEGINNER 74)

4.5.14 Animation

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

A animation is a self-running picture. The statement ANIMATION defines an animation image with nr (0..255).
The compiler can create an animation from single images:
- <file1>,<file2>,... two or more single images BMP, GIF, JPG, TGA, PNG or G16
- <file??> the questionmarks are interpreted as placeholder for numbered single images.
Note that max. 4 animations can run at the same time.
The animation images can be used with the Animation commands 38.
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.

For **cyclic** animations, compression may reduce data size to save memory space (Compileroption COMPRESS 21 :).
You can use the Compiler Functions 6 ANIMATION_W and ANIMATION_H to get the outline in pixels of the animation.

(see How-to-use example Animated gif - BEGINNER 75)

4.5.15 Instrument

`INSTRUMENT: nr <instrument.I16>`

A instrument is a serie of pictures which shows e.g. a panelmeter. The statement INSTRUMENT defines an instrument image with nr (0..255). After DoubleClick the `<instrument.I16>` an DialogBox apears for edit/change the instrument.
The instrument images can be used with the Instrument commands 40. Note that max. 4 instruments can used at the same time.
You can use the Compiler Functions 6 `INSTRUMENT_W` and `INSTRUMENT_H` to get the outline in pixels of the instrument.

(see How-to-use example Instrument by touch - BEGINNER 99)

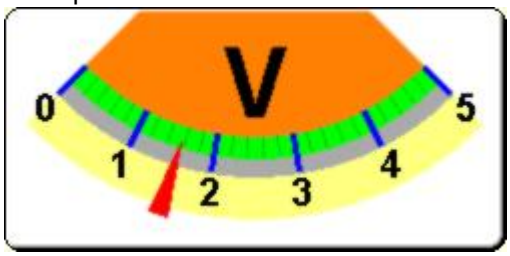
`INSTRUMENT: nr <file??>`

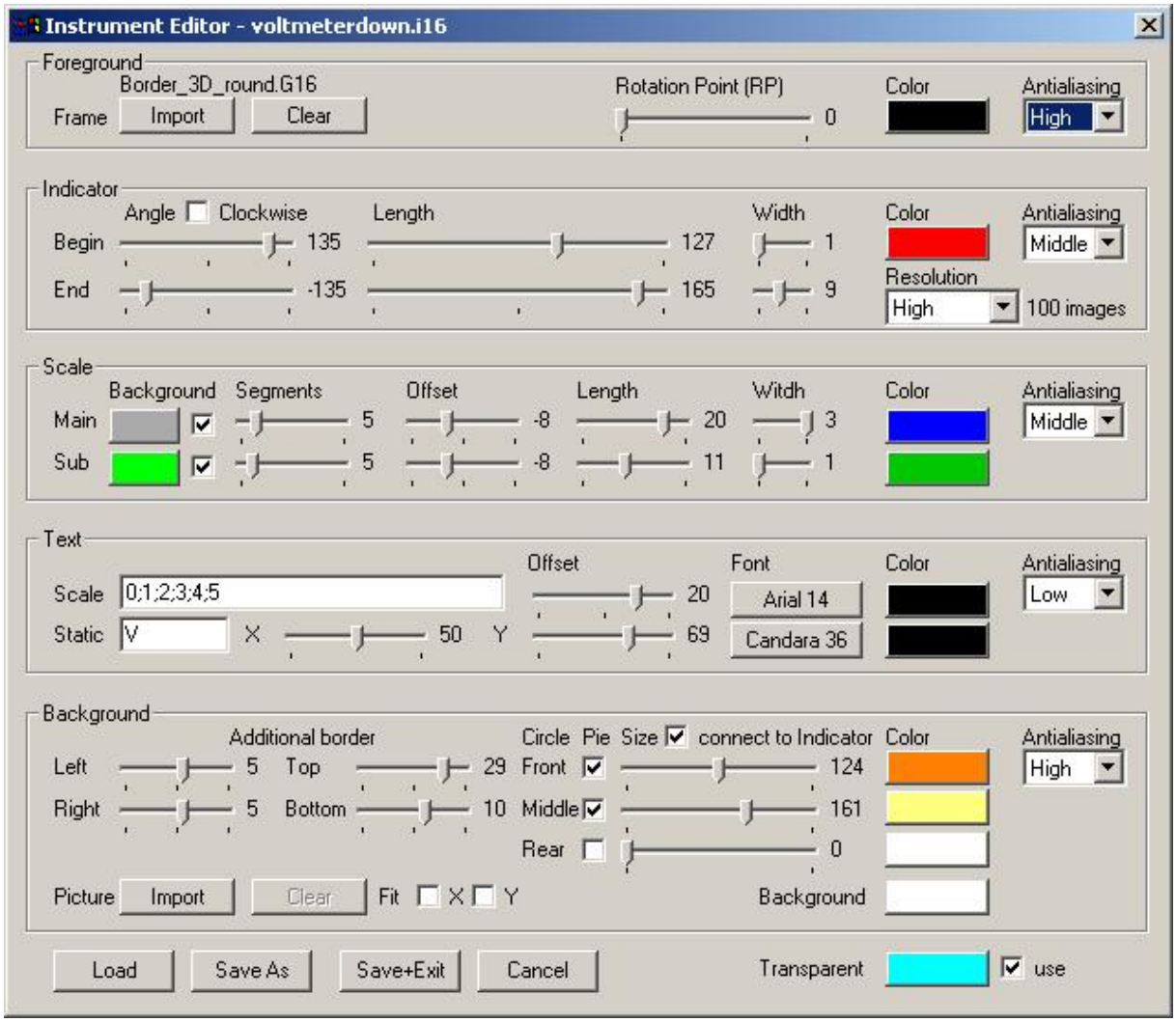
The compiler can create an instrument from single bitmaps, the questionmarks are interpreted as placeholder for numbered images (BMP, GIF, JPG, TGA, PNG or G16).
The image with number 00 is defined as background image only, without the indicator.
Image 1..n are the animations with indicator.
If the backgound image 00 is not available it is not possible to compress the animation and the execution time is longer.

```
INSTRUMENTPARAM: bA eA
INSTRUMENTPARAM: bA eA rpPos
INSTRUMENTPARAM: bA eA rpX rpY
```

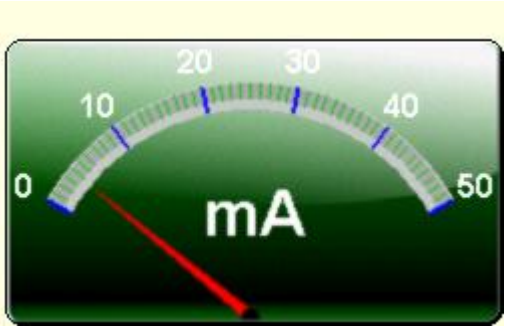
InstrumentParam is used to set the range and position of the indicator for imported animations.
bA, eA = Indicator begin- and end-Angle -180°..+180° (0°=to TOP, Y-axis, see "InstrumentEdit.exe")
rpPos = Rotationpoint position 1..9 (see MAKETRANSPARENT 19) (default=5=middle center of the image)
rpX, rpY = Rotationpoint coordinates in dots

Example "voltmeterdown.i16":





further examples:



4.6 Macros

4.6.1 ExportMacro

```
EXPORTMACRO: n1 [,"chartyp"] [,<filename>]
```

n1=0: no export
n1=1: export all following Macros as a include-File *.h for C;
n1=2: export all following Macros as a binary-File *.bin;
n1=3: export both a include-File *.h and a binary-File *.bin;
"chartyp": optionally another variable type for the byte-array (default is "unsigned char")
<filename>: optionally another filename (default is "macroname_macronumber")

Example:

```
ExportMacro: 1, "char flash"

Macro: 5
      #TA
      #DF BLUE

      #FZ WHITE,TRANSPARENT
      #ZF FONT4x6
      #ZL 12,10, "Font4x6: 0123456789"
      #ZF FONT6x8
      #ZL 12,20, "Font6x8: Schriftprobe"
      #ZF FONT7x12
      #ZL 12,30, "Font7x12: Schriftprobe"
```

Output in Logfile "Macro_5.h":

```
/* Macro 5 as include */

#define MACRO_5_LEN  110

char flash MACRO_5[MACRO_5_LEN] =
{
   27, 84, 65, 27, 68, 70,  2, 27, 70, 90,  8,  0, 27, 90, 70,  1, 27, 90, 76, 12,
    0, 10,  0, 70,111,110,116, 52,120, 54, 58, 32, 48, 49, 50, 51, 52, 53, 54, 55,
   56, 57,  0, 27, 90, 70,  2, 27, 90, 76, 12,  0, 20,  0, 70,111,110,116, 54,120,
   56, 58, 32, 83, 99,104,114,105,102,116,112,114,111, 98,101,  0, 27, 90, 70,  3,
   27, 90, 76, 12,  0, 30,  0, 70,111,110,116, 55,120, 49, 50, 58, 32, 83, 99,104,
  114,105,102,116,112,114,111, 98,101,  0
};
```

4.6.2 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 4,6V or lower.

4.6.3 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 41.
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.

4.6.4 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 41.
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.

(see How-to-use example 3 simple touch buttons - BEGINNER 76)

4.6.5 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 41.
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 intput with command #YD n1,n2,n3 47.
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.

(see How-to-use example Bit Macro - BEGINNER 111)

4.6.6 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 41.
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.

(see How-to-use example Port Macro - BEGINNER 113)

4.6.7 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 41.
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 47.
The relating pins for matrix keyboard need to be defined with the command #YM n1,n2,n3 47.
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.

4.6.8 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 41) or by command #MC nr 41.
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.

(see How-to-use example Process Macro - BEGINNER 117)

4.6.9 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 41.
Since firmware V1.1 it is possible to change the assignment between analogurmacrofunction 0..19 and analogmacronumber with command #VM n1,n2 48.
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.

(see How-to-use example Analogue Macro - Beginner 109)

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

5 EA eDIPTFT43-A commands

5.1 Terminal

Terminal definition:

Set terminal color	#FT fg,bg	Preset color 1..32 for terminal mode (see default colors 61) 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 initialisized and cleared; the software version, hardware revision, macrofile-projectname and CRC-checksum are output in the terminal

(see How-to-use example Keypad - EXPERT 82)

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

5.2 Display

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; (see default colors 61)
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 (see default colors 61)
Invert display	#DI	Invert display content

(see How-to-use example Change display orientation - BEGINNER 119)

5.3 Draw

Draw straight lines and points:

Set color for lines	#FG fg,bg	Colors 1..32 (0=transparent) fg=color for line; bg=pattern background; (see default colors 61)
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 22 :)
Draw point	#GP x1,y1	Set a point at coordinates x1, y1
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
Set start point (firmware V1.5)	#GS x1,y1	Set the last end point at coordinates x1,y1 for commands #GW, #GX and #GY
Draw X-graph (firmware V1.5)	#GX step,y1,...	Draw lines with fix X-steps=0..127 (add 128 for neg.steps) and variable count of Y-values 1..255
Draw Y-graph (firmware V1.5)	#GY step,x1,...	Draw lines with fix Y-steps=0..127 (add 128 for neg.steps) and variable count of X-values 1..255

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 (see default colors 61)
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; (see default colors 61)
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 22 :)
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 22 :)

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; (see default colors 61)
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 23 :)
Draw border box	#RR x1,y1,x2,y2	Draw a border box from x1,y1 to x2,y2

(see How-to-use example Frame - BEGINNER 89)

5.4 Text

Text settings:

Set text color	#FZ fg,bg	Color 1..32 (0=transparent) for string and character fg=text color; bg=background color; (see default colors 61)
Set font	#ZF n1	Set font with the number nr (see compiler option FONT 11: or WINFONT 12:)
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 witdh 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) canceles 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) canceles 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) canceles the special function of '|' and '\'
Output string in an area (firmware V1.2)	#ZB x1,y1,x2,y2, n1,"text..."	Output a string (...) inside area from x1,y1 to x2,y2 at position n1=1..9; the area will be filled with background color; n1=1: Top Left; n1=2: Top Center; n1=3: Top Right n1=4: Middle Left; n1=5: Middle Center; n1=6: Middle Right n1=7: Bottom Left; n1=8: Bottom Center; n1=9: Bottom Right
String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal

(see How-to-use example Place Strings - BEGINNER 69)
(see How-to-use example Cyrillic font - BEGINNER 73)

5.5 Bitmap

Bitmap settings:

Set bitmap colors	#FU fg,bg	painting color 1..32 (0=transparent) for monchrome bitmaps fg=foreground color; bg=background color; (see default colors 61)
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,PNG,G16) n1=3: replace transparent color from bitmap-file (GIF,TGA,PNG,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 24 :)
Load image	#UL x1,y1,data...	Load an image to x1,y1; data... = image in G16 62 format This command is only for serial interface, do not use this command in a macro !

(see How-to-use example BMP file - BEGINNER 74)

Hardcopy:

RLE compression (firmware V1.5)	#UR	the next hardcopy (#UH xx1,yy1,xx2,yy2) will be sent with RLE compression
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

5.6 Animation

Animation settings:

Set animation colors	#FW fg,bg	color for 1..32 monchrome animation images fg=foreground color; bg=background color; (see default colors 61)
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 the sub image n2 to x1,y1(see ANIMATION 25 :)

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 ANIMATION 25 :) 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

(see How-to-use example Animated gif - BEGINNER 75)

5.7 Bargraph

Bargraph settings:

Set color for bargraph	#FB fg,bg,fc	Set colors 1..32 for bargraph (see default colors 61) 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 PATTERN 22 :)
Bargraph border	#BE nr	Border for bargraph nr=0..255 (valid for type=4..7) (see compiler option BORDER 23 :)
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(eft), 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 bar no=1..20 to the new value
Draw bargraph	#BN no	Entirely redraw the bar with the number no=1..20
Send bargraph value	#BS no	Send the current value of bar no=1..20 to sendbuffer
Delete bargraph	#BD no,n2	The definition of the bar 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 remains visible; n2=1: Bar is deleted

(see How-to-use example Bargraph by touch - BEGINNER 95)

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; (see default colors 61)
User value font	#BF nr	Set font nr for bargraph user value (see compiler option FONT 11 : or WINFONT 12 :)
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 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"

5.8 Instrument

Define/use instruments:

Define instrument	# IP no, x1,y1, n2,n3, sv,ev	Define instrument no=1..4 at position x1,y1 (=left top edge) with instrument image n2=0..255. Instrument angle: n3=0: 0°; n3=1: 90°; n3=2: 180°; n3=3: 270°; sv, ev are the values for 0% and 100% (see compiler option INSTRUMENT 26 :)
Update instrument	# IA no,value	Set and draw the instrument no=1..4 to the new value
Draw instrument	# IN no	Entirely redraw the instrument with the number no=1..4
Send instrument value	# IS no	Send the current value of instrument no=1..4 to sendbuffer
Delete instrument	# ID no,n2	The definition of the instrument with the number no=1..4 becomes invalid. n2=0: Instrument remains visible; n2=1: Instrument is deleted

(see How-to-use example Instrument by touch - BEGINNER 99)

Instrument user values - Format text output:

User value color	# FI fg,bg	Set color 1..32 for instrument user value fg=foreground; bg=background color; (see default colors 61)
User value font	# IF nr	Set font nr for instrument user value (see compiler option FONT 11 : or WINFONT 12 :)
User value zoom	# IZ n1,n2	Set zoom factor for instrument user value n1=X-Zoom 1x..8x; n2=Y-Zoom 1x..8x
User value additional width/height	# IY n1,n2	n1=0..15: additional width left/right; n2=0..15: additional height top/bottom for instrument user value;
User value angle	# IW n1	Set writing angle for instrument user value n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°;
User values / scaling	# IX no,x1,y1, "FormatString"	Define user value for instrument no=1..4. Output is always right justified to x1,y1 Format String: "iv1=uservalue1;iv2=uservalue2" Assign two instrument values (iv1,iv2 =0..254) to user defined values max. range: 4 1/2 digits 19999 + decimal point ('.' or ',') + sign e.g. display "-123.4" for iv1=0 and "567.8" for iv2=100 Format String: "0=-123.4;100=567.8"

5.9 Macros

Run macros:

Run macro	#MN nr	Call the (normal) macro with the number nr (max. 7 levels) (see compiler option MACRO 29 :)
Run touch macros	#MT nr	Call the touch macro with the number nr (max. 7 levels) (see compiler option TOUCHMACRO 29 :)
Run port macro	#MP nr	Call the port macro with the number nr (max. 7 levels) (see compiler option PORTMACRO 29 :)
Run bit macro	#MB nr	Call the bit macro with the number nr (max. 7 levels) (see compiler option BITMACRO 29 :)
Run matrix macro	#MX nr	Call the matrix macro with the number nr (max. 7 levels) (see compiler option MATRIXMACRO 30 :)
Run process macro	#MC nr	Call the process macro with the number nr (max. 7 levels) (see compiler option PROCESSMACRO 30 :)
Run analogue macro	#MV nr	Call the analogue macro with the number nr (max. 7 levels) (see compiler option ANALOGMACRO 31 :)

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

(see How-to-use example Languages/Macro Pages - BEGINNER 105)

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

(see How-to-use example Automatic Macro - EXPERT 115)

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

(see How-to-use example Process Macro - BEGINNER 117)

5.10 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
		(see default colors 61)

Touch border form	#AE nr,n2	nr=1..255 border number (see compiler option BORDER 23 :) 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
		(see default colors 61)

Touch button number	#AC nr,n2,n3,n4	nr=0..255 button number (see compiler option BUTTON 24 :) 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.

(see How-to-use example Radio group - BEGINNER 80)

Label font presets:

Font color	#FA nf,sf	Color 1..32 for touch labeling (see default colors 61) 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 11 : or WINFONT 12 :)
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"		

(see How-to-use example 3 simple touch buttons - BEGINNER 76)
(see How-to-use example Glass button - EXPERT 78)

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. (see default colors 61)
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 no	The bargraph with no=1..20 is defined for input by touch panel.
Set instrument by touch (firmware V1.4)	#A+ no	The instrument with no=1..4 is defined for input by touch panel.

(see How-to-use example Free draw area - BEGINNER 86)
(see How-to-use example Bargraph by touch - BEGINNER 95)
(see How-to-use example Instrument by touch - BEGINNER 99)

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/instrument value on/off	#AQ n1	Automatic transmission of a new bargraph or instrument 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 continous 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

5.11 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
Assign instrument with backlight (firmware V1.4)	#Y+ no	Assign instrument no=1..4 for changing brightness of the backlight
Save parameter	#Y@	Save the actual brightness and changetime for poweron to EEPROM

(see How-to-use example Bargraph by touch - BEGINNER 95)

5.12 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	n1=1..8: input port n2=0: falling-edge or n2=1: rising-edge n3=0..255: new BitMacro number

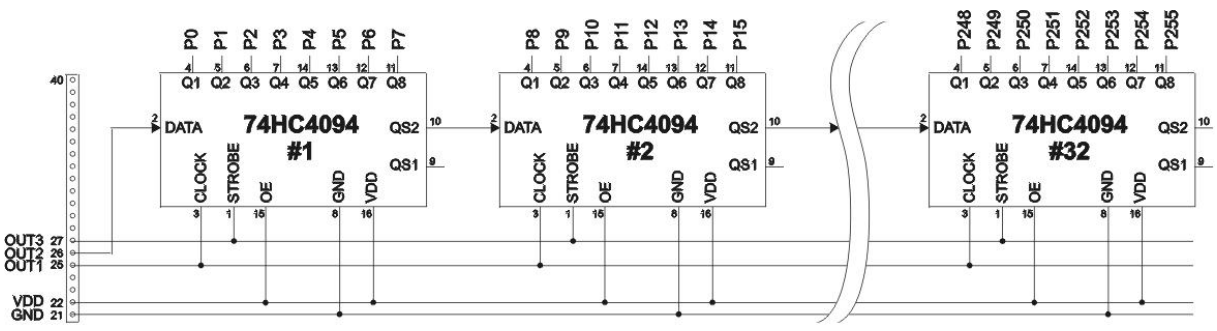
(see How-to-use example Bit Macro - BEGINNER 111)
(see How-to-use example Port Macro - BEGINNER 113)

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 matrixtmacro n2=0..255 After release the key n1=0 run matrixtmacro n2=0..255

Additional Outputs:

Write extended ports (firmware V1.4 with 74HC4094)	#YE n1,n2,n3	write from output port n1=0..255 to port n2=0..255 n3=0 Reset ports n3=1 Set ports n3=2 Invert ports



5.13 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 serveral analogmacros can be started automatically.
Redefine analogmacro (firmware V1.1)	#VM n1,n2	Assign analogmacrofunction n1=0..19 with analogmacronumber n2=0..255
Bargraph for AIN1/AIN2	#VB ch,no	Assigns bargraph no=1..20 to analogue input ch=1..2 (it is possible to assign more than one bargraph to an anlogue input). Define start- endvalues for bargraph (see #B RLOU 39) in [mV/20]
Instrument for AIN1/AIN2 (firmware V1.4)	#V+ ch,no	Assigns instrument no=1..4 to analogue input ch=1..2 Define start- endvalues for instrument (see # IP 40) in [mV/20]
Redraw bar/instrument	#VR ch	Redraw all bargraphs and instruments defined for channel ch=1..2

(see How-to-use example Analogue Macro - Beginner 109)
(see How-to-use example Instrument by analogue input - BEGINNER 106)

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; (see default colors 61)
User value Font	#VF ch,nr	Set font nr for channel ch=1..2 (see compiler option FONT 11: or WINFONT 12:)
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 formated string for channel ch=1..2 to sendbuffer
Display on terminal	#VT ch	Show formated string of channel ch=1..2 on termial window
Display user value	#VG ch,x1,y1	Show formated string of channel ch=1..2 at coordinate x1,y1

5.14 Other commands

Use string table:

String table code (firmware V1.5)	#ST n1	n1=0: no use of internal strings n1>0: after code n1 appears following codes are internal string numbers (see compiler option STRING: 10)

(see How-to-use example String tables - EXPERT 107)

Send functions:

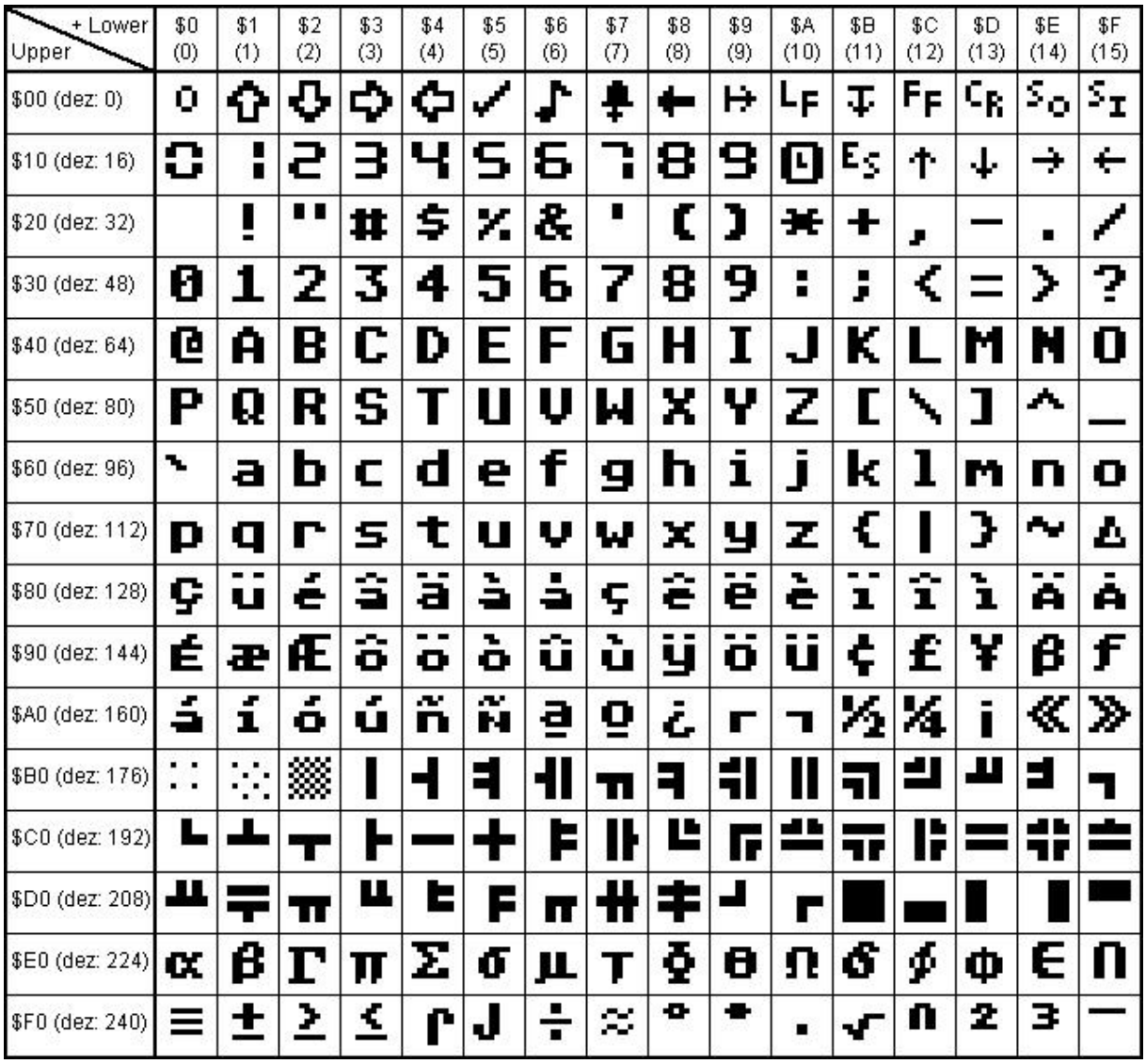
Send bytes	#SB data...	bytes are sent to the sendbuffer data... can be numbers or strings e.g #SB "Test",10,13
Send version	#SV	The version is sent as a string to sendbuffer e.g. "EA eDIPTFT43-A V1.0 Rev.A TP+"
Send projectname	#SJ	The macro-projectname is sent as a string to the sendbuffer e.g. "init / 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; (see default colors 61)
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). (see compile option RS485ADR 9) (see example INIT_with_RS485_address.KMC 68)
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

6 Default Fonts

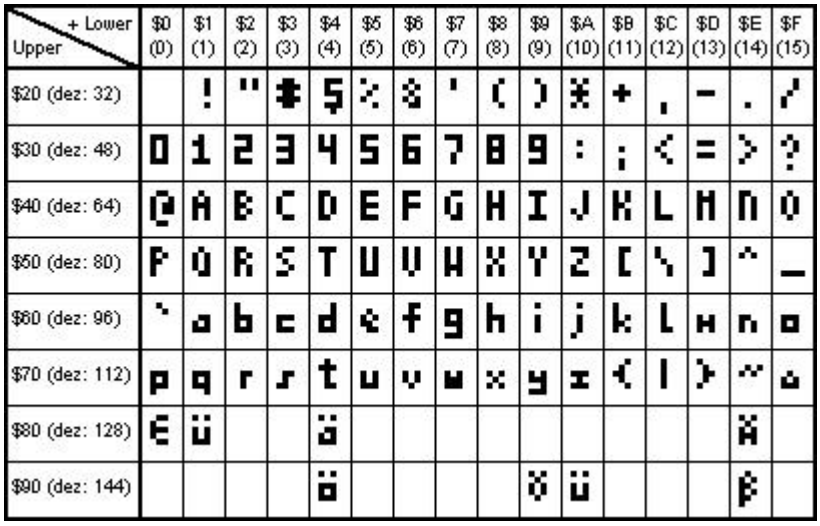
6.1 Terminal 8x8



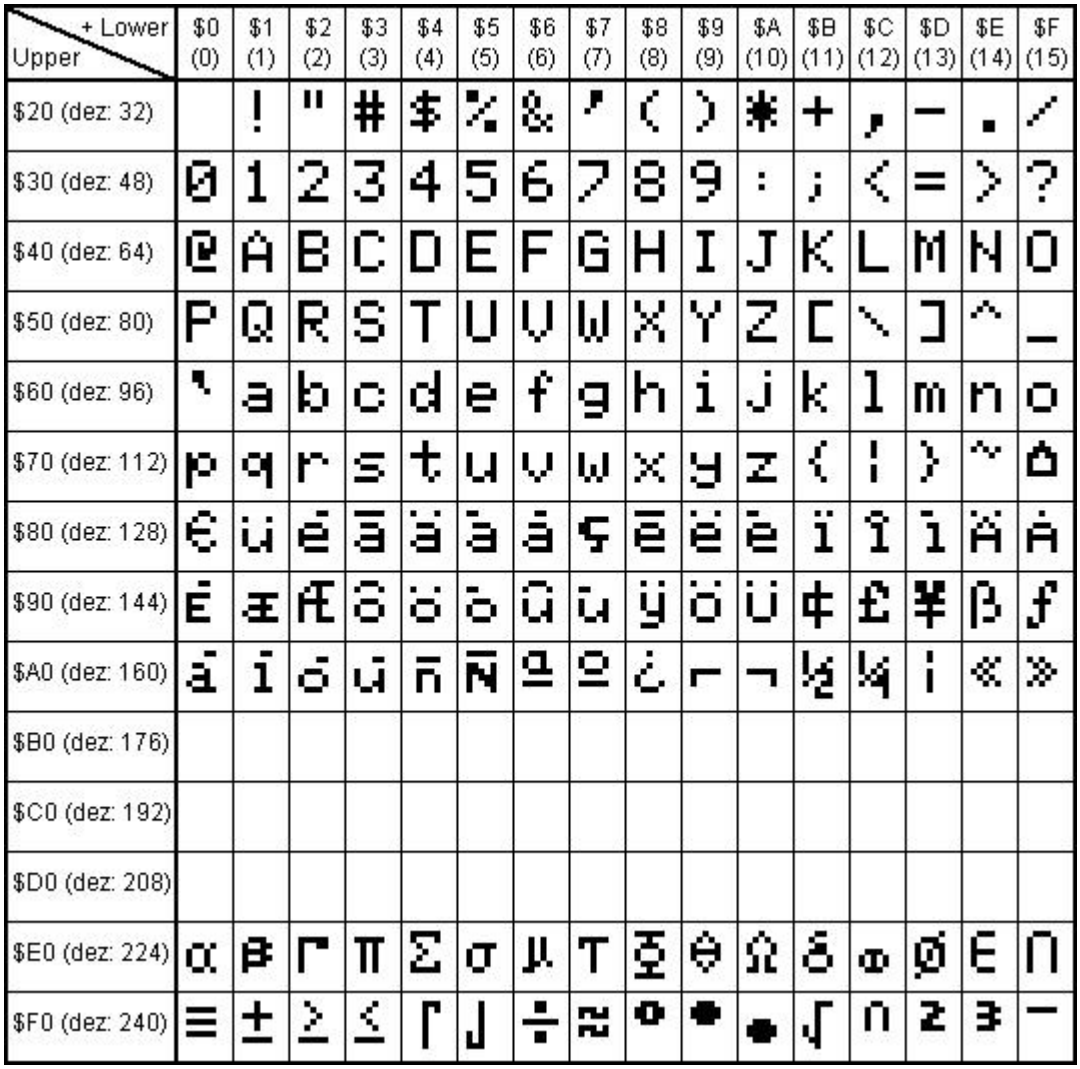
6.2 Terminal 8x16

+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$00 (dez: 0)	NUL	⇧	⇩	⇨	⇦	✓	♪	🔔	←	↦	LF	↧	FF	CR	SO	SI
$10 (dez: 16)	0	!	2	3	4	5	6	7	8	9	🕒	ESC	↑	↓	→	←
$20 (dez: 32)		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	ß	ƒ
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
$B0 (dez: 176)	⁞	⁞	▒	┃	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
$C0 (dez: 192)	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
$D0 (dez: 208)	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▉	▄	▌	▐	▀
$E0 (dez: 224)	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	ϕ	φ	∈	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	•	·	√	n	2	3	¯

6.3 Font 4x6


+ 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)
$40 (dez: 64)
$50 (dez: 80)
$60 (dez: 96)
$70 (dez: 112)
$80 (dez: 128)
$90 (dez: 144)

6.4 Font 6x8



+ 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)	ā	ī	ó	ū	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	¯

6.5 Font 7x12

+ 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	¯

6.6 Geneva 10

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

6.7 Chicago 14

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

6.8 Swiss 30

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

6.9 BigZif 50

+ 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	:					

6.10 BigZif 100

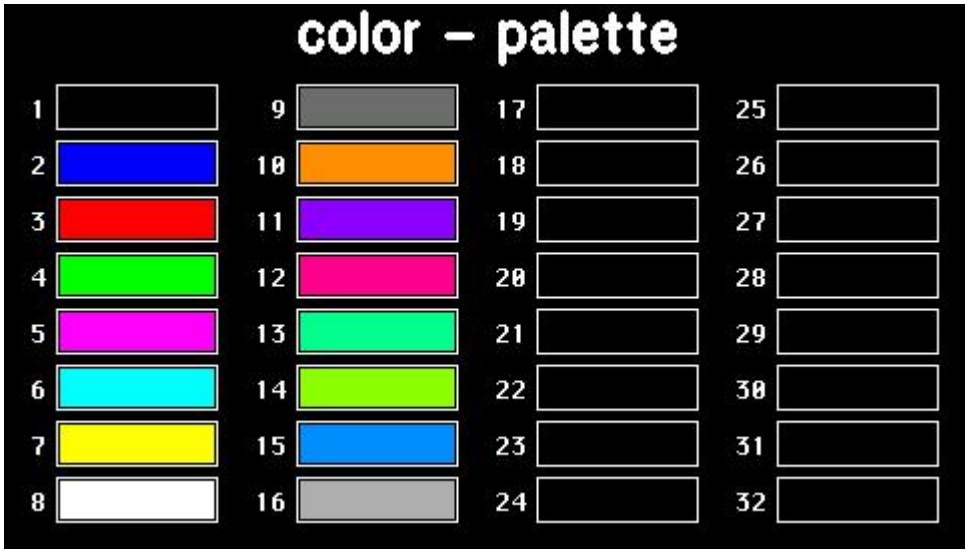
+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)												+		-	.	
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:					

7 Default Colors

EA eDIPTFT43-A is able to work with 65,536 colors. For an easy use there exists a color palette with 32 entrys (16 colors are predefined after PowerOn). This color palette can be redefined at any time without changing the content of the display (command: #FPnoRGB 50). To use a color for text and graphic functions you set only a number between 1..32. The dummy color number 255 means that the actual color is not changed e.g you want only to change the foreground- and not the background color. The color number 0=transparent is special and can be used for background of character e.g. that means placing a character no rectangular field will be deleted around the character itself.

predfined constants (include <..\default_constant.kmi>)

```
;-------------------------------------------------------------------------------
; color constants

TRANSPARENT =  0

BLACK       =  1      ; RGB:   0   0   0
BLUE        =  2      ; RGB:   0   0 255
RED         =  3      ; RGB: 255   0   0
GREEN       =  4      ; RGB:   0 255   0
MAGENTA     =  5      ; RGB: 255   0 255
CYAN        =  6      ; RGB:   0 255 255
YELLOW      =  7      ; RGB: 255 255   0
WHITE       =  8      ; RGB: 255 255 255
DARKGREY    =  9      ; RGB: 111 111 111
ORANGE      = 10      ; RGB: 255 143   0
PURPLE      = 11      ; RGB: 143   0 255
DEEPPINK    = 12      ; RGB: 255   0 143
MINT        = 13      ; RGB:   0 255 143
LAWNGREEN   = 14      ; RGB: 143 255   0
SKYBLUE     = 15      ; RGB:   0 143 255
GREY        = 16      ; RGB: 175 175 175

NOCHANGE    =255
```



8 G16FORMAT

Use 'BitmapEdit.exe' from the LCD-Tools package to convert
*.BMP, *.JPG, *.GIF, *.TGA or *.PNG into internal G16-format.

Structure of an image file in the G16 format:

This format handles both a single picture, and several pictures (e.g. containing fonts or animations).
A transparency color is supported.

Structure of the picture header:

Byte	value	descripion	
1.	$1b	ESC An image file always begins with	
2.	$55	' U ' the imag load instruction	
3.	$4c	' L '	
4.	$00	X-coordinate LOW byte	
5.	$00	X-coordinate HIGH byte	
6.	$00	Y-coordinate LOW byte	
7.	$00	Y-coordinate HIGH byte	
8.	$47	'G' identification for a picture -, Font-,	
9.	$31	'1' or animation-file in the G16 format	
10.	$36	'6'	
11.	Bits per pixel	1=Monochrome; 4=16 colors; 8=256 colors; 16=65536 colors High Color RGB565	
12.	Transparency	0=none, 1=the following vaild transparency color	
		4/8-bit:	16-Bit:
13.		Pallet No. of the transparency color	\ RGB565-WORD that
14.		Number of existing color palette (0=256)	/ transparency color
15.		reserved	
16.	Size of the pictures	0=different dimensions 1=equal width 2=qual height (e.g. proportional Fonts) 3=equal dimensions	
17.	First	First picture / characters number	
18.	Last	Last picture / characters number	
19.	Width	\ Width of the broadest picture/character Low byte	
20.		/ High byte	
21.	Height	\ Height of the highest picture/character Low byte	
22.		/ High byte	

After the header, the color pallete entries follow (only for 4 or 8-bits of pictures)

Palette entry: 16-Bit RGB565-WORD
1. Byte: \ Low byte
2. Byte: / High byte

Palette entry: RGB565-WORD

Bit NR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	R4	R3	R2	G1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

After the color palette, the picture table follows, 8 bytes per picture

Byte	value	description
1.	Width	\ Width of the picture in pixel Low byte
2.		/ High byte
3.	Height	\ Height of the picture in pixel Low byte
4.		/ High byte
5.	Offset	\ offset of the graphic data Low byte
6.		> Mid byte
7.		/ High byte (starting from file beginning)
8.	BIT D6..D0 = 0..127 waiting period for animations (in 0.1 secs)	
	BIT D7	= 0: uncompressed graphic data = 1: RLE compressed character rows

---

According to the picture table, the actual graphic data follows
The lines are always put down from above downward
The arrangement of the pixels in a line is from left to the right

---

Structure of a line of the picture (uncompressed):

1 bit per pixel: (monochrome)	the first pixel is the data bit D7 Number of Bytes per Line = (Width+7) / 8
4 bits per pixel: (16-color)	the first pixel is the HI Nibble (D7..D4) the second pixel is the LO Nibble (D3..D0) Number of Bytes per Line = (Width+1) / 2
8 bits per pixel: (256-color)	the first pixel is the first byte Number of Bytes per Line = width
16 bits per pixel: (65636-color)	the first pixel is the first RGB565-WORD Number of Bytes per Line = width * 2

Structure of a RLE compressed line:

first the Number-byte is read
BIT D6..D0 = 0..127; +1 = number of 1..128
BIT D7 = 0 the following are uncompressed bytes/pixels
BIT D7 = 1 the following byte/pixel is repeated this number of times

Next, the repeating byte/repeating pixel or the uncompressed bytes/pixels follow.
For pictures with 1 -, 4 and 8-bits per pixel, the data is treated byte by byte.
For High Color 16-Bit picture, the data are treated pixel-wise.

Example monochrome line with 128 pixels:
00 00 00 00 00 12 34 56 78 FF FF FF FF FF FF FF
compresses to
84 00 03 12 34 56 78 86 ff

Example 16-Bit RGB565 line with 16 pixels:
0000 0000 0000 0000 0000 1234 5678 ABCD FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
compresses to
84 0000 02 1234 5678 ABCD 87 FFFF

9 How-to-use

To find an easy start, you will find a project under "..\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\My first project\my_first_project.KMC". In that example all main commands are used.
There are two different classes of examples. The ones starting with "BEGINNER.." are good to get an easy start. The ones starting with "EXPERT" describe special functions, such as using constants, definitions and compiler functions.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\My first project

File:
my_first_project.kmc

Commands:
#AT, #BR, #ZL, #UI, #WD

Open file in KitEditor

```
eDIPTFT43-A   "First project"
...
...
...
;-------------------------------------------------------------------------------
;Include picture
Picture: 1, <..\..\BITMAPS\color\ESCHER_9_15.bmp> ;store as picture 1
Animation: 1, <..\..\BITMAPS\color\Animation\horse.gif> ;store as animation 1

;-------------------------------------------------------------------------------
;start of macro programming
;Normal Macros:

Macro: 0 ;define macro 0, called after power on, reset, watchdog reset
       #TA                  ; terminal off
       #AF CHICAGO14  ; set touch label font, the font is defined in include file
"default_font.kmi"
       #AT 5,10,150,35,1,0, "Picture"      ; place 3 touchbuttons at x1,y1 to x2,y2,
Touchmacro 1 is called
       #AT 5,45,150,70,2,0, "String"       ; touchmacro 2 is called
       #AT 5,80,150,105,3,0, "Bargraph"    ; touchmacro 3 is called
       #AT 5,115,150,140,4,0,"Animation"   ; touchmacro 3 is called


;Touch Macros:
TouchMacro: 1 ;Picture
       #BD 1, 0             ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                            ; because pixels are deleted with next command
       #WL 1                ;stop animation process
       #RL 220,0,479,239    ; delete area on the right (to delete pixels of other
touchmacros)
       #UI 220,10, 1  ;load internal picture 1

TouchMacro: 2 ;String
       #BD 1, 0             ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                            ; because pixels are deleted with next command
       #WL 1                ;stop animation process
       #RL 220,0,479,239    ; delete area on the right (to delete pixels of other
touchmacros)
```

```
        #ZF SWISS30B          ;set font for strings (font is defined in "default_font.kmi")
        #ZC 260,40, "Hello|World" ;write string centered, '|' means next line

TouchMacro: 3 ;Bargraph
        #BD 1, 0              ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                              ; because pixels are deleted with next command
        #WL 1                 ;stop animation process
        #RL 220,0,479,239     ; delete area on the right (to delete pixels of other
touchmacros)
        #AQ 0                 ; deactivate sending barvalues into sendbuffer
        #FB RED, BLACK, WHITE ; set colors for bargraph
        #BM 2                 ; set pattern
        #BO 1,220,190,270,10,0,100,1 ; define bargraph and show it
        #BA 1,75              ; set bar 1 to new val of 75
        #AB 1                 ; set bar 1 with touch

TouchMacro: 4 ;Animation
        #BD 1, 0              ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                              ; because pixels are deleted with next command
        #RL 220,0,479,239     ; delete area on the right (to delete pixels of other
touchmacros)
        #WD 1, 220,30, 1,2,255 ; show animation 1, with picture 1 (see definition above,
cyclically with the time stored within the gif-fle
```

9.1 Factory Setting

This macrofile sets the display back to factroy setting.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\Init\

File:
Init.kmc

Commands:

Open file in KitEditor

```
eDIPTFT43-A "Status on delivery" ; define eDIP, "Projectname" max. 32 character
;brings the display back to ex-works condition with it's standard-fonts 1..8, standard-
pattern and standard-border

AutoScan: 1                  ; autoscan for correct baud rate to connect to eDIP on
COM/USB

;COM1: 115200                        ; program eDIP on COMx with 115200 Baud
USB: 230400, "eDIP Programmer"       ; use EA 9777-USB eDIP Programmer and program eDIP
with 230400 baud

;VERIFY                      ; verify after program

;-------------------------------------------------------------------------------
; load defaults

include <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>
include <..\default_pattern.kmi>
include <..\default_border.kmi>

;-------------------------------------------------------------------------------

MnAutoStart = 0

PowerOnMakro:         ; runs after power-on
       #MN MnAutoStart

ResetMakro:           ; runs after external reset
       #MN MnAutoStart

WatchdogMakro:; runs after a crash (>500ms)
       #MN MnAutoStart

BrownOutMakro:; runs when supply voltage drops <3V
       #MN MnAutoStart

;-------------------------------------------------------------------------------
Makro: MnAutoStart
```

9.2 RS485 - Factory Setting

This macrofile uses RS485 addressing and sets the display back to factory setting.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\Init\

File:
INIT_with_RS485_address.KMC

Commands:

Open file in KitEditor

```
eDIPTFT43-A "Status on delivery" ; define eDIP, "Projectname" max. 32 character
;brings the display back to ex-works condition with it's standard-fonts 1..8, standard-
pattern and standard-border

AutoScan: 1                  ; autoscan for correct baud rate to connect to eDIP on
COM/USB

;COM1: 115200                        ; program eDIP on COMx with 115200 Baud
USB: 230400, "eDIP Programmer"       ; use EA 9777-USB eDIP Programmer and program eDIP
with 230400 baud

;VERIFY                      ; verify after program

progadr = 0          ; Constant for program address
RS485ADR: progadr    ; program only eDIP with address xx (possible addresses: 0..255)

;newadr = 10         ; Constant for new software address, see Makro 0 (#KA newadr)
                     ; (software addres only possible for hardware address 0)
newadr = progadr     ; do not change the address


;-------------------------------------------------------------------------------
; load defaults

include <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>
include <..\default_pattern.kmi>
include <..\default_border.kmi>

;-------------------------------------------------------------------------------

MnAutoStart = 0

PowerOnMakro:        ; runs after power-on
       #MN MnAutoStart

ResetMakro:          ; runs after external reset
       #MN MnAutoStart

WatchdogMakro:; runs after a crash (>500ms)
       #MN MnAutoStart

BrownOutMakro:; runs when supply voltage drops <3V
       #MN MnAutoStart

;-------------------------------------------------------------------------------
Makro: MnAutoStart
       #KA newadr
```

9.3 Place Strings - BEGINNER

Place different strings with different fonts and orientation. This example is available as an EXPERT-version (EXPERT - Place text.kmc 71). In addition you will find help to include WinFonts under BEGINNER - Cyrillic Font.kmc 73.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Font\

File:
BEGINNER - Place text.kmc

Commands:
WinFont, #ZL, #ZF,#ZW

Open file in KitEditor

```
eDIPTFT43-A "Place text
...
...
...
;-------------------------------------------------------------------------------
;include pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                                ; Cursor invisible
       #UI 122,20,1                         ; place picture no. 1
       #GD 40,120,439,120                   ; draw a Line

;---- left justified text ----
       #FZ MAGENTA,BLACK                    ; set text colour no. 5 and background no. 1
                                            ; same as "#FZ 5,1"
       #ZF 3                                ; set font no. 3
       #ZZ 2,2                         ; set zoom factor 2 in x and y direction
       #ZL 0,140,"left justified"           ; place text "left justified" at the left frame
in line 140

;---- centered text ----
       #FZ GREEN,WHITE                      ; set text colour
       #ZF 6                                ; set font no. 6
       #ZZ 2,1                         ; set zoom factor 2 in x and 1 in y direction
       #ZC 240,195,"centered text"          ; place text "centered text" in the center in
line 195

;---- right justified text ----
       #FZ RED,BLUE                         ; set text colour
       #ZF 2                                ; set font no. 2
       #ZZ 2,2                         ; set zoom factor 2 in x and y direction
       #ZR 479,256,"right justified"        ; place text "right justified" at the left
frame in line 256

;---- vertical centered text ----
       #FZ GREEN,BLACK                      ; set text colour
       #ZF 6                                ; set font no. 6
       #ZZ 1,1                         ; set zoom factor 1 in x and 1 in y direction
       #ZW 1                                ; rotate text 90°
       #ZC 450,136,"vertical centered"      ; place text "vertical centered" in the center
```

in row 450

9.4 Place Strings - EXPERT

Place different strings with different fonts and orientation.This example is available as a BEGINNER-version (BEGINNER - Place text.kmc 69). In addition you will find help to include WinFonts under BEGINNER - Cyrillic Font.kmc 73.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Font\

File:
EXPERT - Place text.kmc

Commands:
WinFont, #ZL, #ZF,#ZW

Open file in KitEditor

```
eDIPTFT43-A "Place text"
...
...
...
;-------------------------------------------------------------------------------
;include pictures
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open

;-------------------------------------------------------------------------------
;define string constants
!TEXT1! = "left justified"
!TEXT2! = "centered text"
!TEXT3! = "right justified"
!TEXT4! = "vertical centered"
;-------------------------------------------------------------------------------
Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)

;---- left justified text ----
       #FZ MAGENTA,BLACK                    ; set text colour no. 5 and background no. 1
                                            ; same as "#FZ 5,1"
       #ZF 3                                ; set font no. 3
       #ZZ 2,2                          ; set zoom factor 2 in x and y direction
       #ZL 0,140,!TEXT1!                ; place text "left justified" at the left frame in
line 140

;---- centered text ----
       #FZ GREEN,WHITE                      ; set text colour
       #ZF 6                                ; set font no. 6
       #ZZ 2,1                          ; set zoom factor 2 in x and 1 in y direction
       #ZC 240,195,!TEXT2!              ; place text "centered text" in the center in line 195

;---- right justified text ----
       #FZ RED,BLUE                         ; set text colour
       #ZF 2                                ; set font no. 2
       #ZZ 2,2                          ; set zoom factor 2 in x and y direction
```

```
       #ZR 479,256,!TEXT3!                ; place text "right justified" at the left
frame in line 256

;---- vertical centered text ----
       #FZ GREEN,BLACK                    ; set text colour
       #ZF 6                              ; set font no. 6
       #ZZ 1,1                      ; set zoom factor 1 in x and 1 in y direction
       #ZW 1                              ; rotate text 90°
       #ZC 450,136,!TEXT4!                ; place text "vertical centered" in the center
in row 450
```

9.5 Cyrillic font - BEGINNER

Show the use of Windows fonts, in this case Cyrillic font. Two examples of placing strings are available (see BEGINNER - Place text.kmc 69, EXPERT - Place text.kmc 71).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Font\

File:
BEGINNER - Cyrillic Font.kmc

Commands:
WinFont, #ZL, #ZF,#ZW

Open file in KitEditor

```
eDIPTFT43-A "Cyrillic Font"
...
...
...
;-------------------------------------------------------------------------------
;include Picture (Logo)
Picture 1 <..\..\..\bitmaps\color\ea logo making things easy black.bmp> ; double click to
open Bitmap Editor
;-------------------------------------------------------------------------------
;include an internal Windows font
ExportOverview: 1                          ; creates the file "Font9_Arial_RUSSIAN_N_32-
255_48.bmp"
WinFont 9,  "Arial",204,0, 32,255, 36      ; double click to open (on Fontname)
                                           ; select regions and characters by pressing
shift and mark them with the mouse
                                           ; be sure that the box "use Font for EditBox"
is activated
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                       ; Cursor invisible
       #UI 122,20,1                ; place Picture no. 1
       #GD 30,120,449,120          ; draw a Line

;---- Place cyrillic text ----
       #FZ YELLOW,BLACK            ; set text color and background color
                                   ; YELLOW is defined as 7 (compare with line 12:
"include <..\..\default_constant.kmi>"
       #ZF 9                       ; set font to no. 9 (the above, line 36, defined Font)
       #ZC 240,170, {CFD0C8C2C5D28220CAC0CA20C4C5CBC03F} ; character table: see file
"Font9_Arial_RUSSIAN_N_32-255_48.bmp"
                                                         ; double click between the
curly brackets to open EditBox for fonts
                                                         ; use mouse to select
characters
                                                         ; You have to select Font no.9
for EditBox to see the characters correctly
                                                         ; by clicking right on the
Fontname and "Select Font for EditBox"
```

9.6 BMP file - BEGINNER

Show simple pictures invertered and normal. If you want to show an animation, plese refer to BEGINNER - Show an animated giff file.kmc 75.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Picture\

File:
BEGINNER – show a bmp file.kmc

Commands:
#UI

Open file in KitEditor

```
eDIPTFT43-A "Show a bmp File"
...
...
...
;-------------------------------------------------------------------------------
Picture: 1 <..\..\..\bitmaps\color\ea logo making things easy black.bmp>       ; double
click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
       #UI 122,20,1                   ; place Picture no. 1
       #GD 30,120,449,120             ; draw a Line
```

9.7 Animated gif - BEGINNER

Example of a little animation. The animation is a gif-file. If you want to show a simple picture, please refer to BEGINNER – show a bmp_file.kmc 74.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Picture\

File:
BEGINNER - Show an animated giff file.kmc

Commands:
#WD

Open file in KitEditor

```
eDIPTFT43-A "Show an animated giff file"
...
...
...
;-------------------------------------------------------------------------------
Picture: 1 <..\..\..\bitmaps\color\ea logo making things easy black.bmp>       ; double
click to open
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
Animation: 5 <..\..\..\BITMAPS\color\Animation\horse.gif>; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,20,1                 ; place Picture no. 1
       #GD 30,120,449,120           ; draw a Line

;---- Place Information logo ----
       #WD 1,193,140,5,2,1          ; place Animation no. 5 (process no. 1)
```

9.8 3 simple touch buttons - BEGINNER

Explanation of general use of TouchButtons and TouchMacros. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 95), Radiogroups (see BEGINNER - radiogroup.kmc 80) and another Example with touch buttons (see EXPERT - numbers to terminal with autorepeat.kmc 82).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Touch\

File:
BEGINNER - 3 simple touch areas.kmc

Commands:
#AU, #AT

Open file in KitEditor

```
eDIPTFT43-A "3 simple touch areas"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
Button 1 <button\Button34x34_0.bmp>,<button\Button34x34_1.bmp>
Button 2 <button\andromeda0.gif>,<button\andromeda1.gif>
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,20,1                 ; place Picture no. 1
       #GD 30,120,449,120           ; draw a Line

;---- Place the left touch ----
       #AE 14,0                     ; set Frame style no. 14 and rotation for Touch
(1..20)
       #AF 6                        ; set font no. 6 for Touch area
       #AT 50,170,100,220,65,0 "A" ;       draw Touch area - this will put a $41 (65 dec.)
into send buffer

;---- Place the middle touch as a bitmap ----
       #AF 6                        ; set font no. 6for Touch area
       #AC 1,0,2,2                  ; set Button no.1 , rotation and size
       #AU 210,162,66,0 "B" ;       draw Touch area - this will put a $42 (66 dec.) into
send buffer

;---- Place the right touch as a bitmap ----
       #AF 6                        ; set font no. 6for Touch area
       #AC 2,0,1,1                  ; set Button no.2 , rotation and size
       #AU 350,160,67,68 "CC"       ; draw Touch area - this will put a $43 (67 dec.) into
send buffer
                                    ; the first "C" means Center

;---- Touch Macro for the right touch ----
Touch: 67
       #FZ 3,0              ; set color for text
       #ZF 6                        ; set font no. 6
       #ZC 240,245 "Macro #67"      ; place text
```

```
;---- Release the right touch ----
Touch: 68
       #RL 0,245,479,271            ; delete area (text)
```

9.9 Glass button - EXPERT

Example of using pictures as buttons. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 95) and another Example with touch buttons (see BEGINNER - 3 simple buttons.kmc 76)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Touch\

File:
EXPERT - Glass buttons.kmc

Commands:
#AU

Open file in KitEditor

```
eDIPTFT43-A "Glass buttons"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\glas-button\>
LOGO = 1
BACKGROUND = 2
Picture: LOGO, <ea_transparent.g16> ; double click to open
Picture: BACKGROUND <background.g16>

;--- define picture button ---
GLASBUT = 1
Button: GLASBUT <circle-green_40x40-trans.g16>,<circle-green_40x40-trans2.g16> ; double
click to open Bitmap Editor


;--- define Unicode Font ---
; double click to open (on Fontname)
; select regions and characters by pressing shift and mark them with the mouse
; be sure that the box "use Font for EditBox" is activated, if you want to edit strings
(refer to l.20)
WEB = 10
DAUPHIN = 11
WinFont: WEB "Webdings",-52,0, 52 + 55-60 + $F058 + $F0AF + $F0B2-$F0B3, 22
WinFont: DAUPHIN "Dauphin",-32,1, 32-125, 36

;-------------------------------------------------------------------------------
;define string-constants for webdings
; double click between the curly brackets to open EditBox for fonts
                                    ; use mouse to select characters
                                    ; You have to select Font no.9 for EditBox to see the
characters correctly
                                    ; by clicking right on the Fontname and "Select Font
for EditBox"
!PLAY! = {34}
!PAUS! = {39}
!REW! = {37}
!FOW! = {38}
!STOP! = {3A}
!EARCD! = {3D3C3E}

;-------------------------------------------------------------------------------
;define coinstants for touch-macros
play = 1
```

```
paus = 2
rew = 3
fow = 4
stop = 5

;define constants for normal-macros
delete = 10

;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo with background ----
       #TC 0                        ; Cursor invisible
       #FD BLACK,BLACK              ; set display colors
       #UI 0,0,BACKGROUND                   ; place background
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)

;---- Place buttons ----
xs = 250
pitch = 15
xw = BUTTON_W(GLASBUT)
ys = 175

       #AC GLASBUT,0,1,1            ; use Picturebutton 1
       #AF WEB               ; set Fontlabel for Touchbuttons (refer to l.16Webdings)
x=xs
       #AU x,ys,0,play,!PLAY!              ; define Picture Button with call touchmacro 1
x+=pitch+xw
       #AU x,ys,0,rew,!REW!
x+=pitch+xw
       #AU x,ys,0,fow,!FOW!
x+=pitch+xw
       #AU x,ys,0,stop,!STOP!

       #ZF WEB               ; select Font 1 (Webdings)
       #FZ WHITE,transparent ; set fontcolors
       #ZL 5,ys,!EARCD!             ; place earphone, cd...


x=xs
st_x=x
st_y=230
TouchMacro: play ;called by Button Play
       #MN DELETE ; dedlete old string and set font
       #ZL st_x,st_y, "Pressed play"       ; place description
       #AU x,ys,0,5,!PAUS!   ; replace play button as pause button

TouchMacro: rew ;called by Button rewind
       #MN DELETE ; dedlete old string and set font
       #ZL st_x,st_y, "Pressed rewind"     ; place description
       #AU x,ys,0,1,!PLAY!; replace pause button as play button

TouchMacro: fow ;called by Button forward
       #MN DELETE ; dedlete old string and set font
       #ZL st_x,st_y, "Pressed forward"    ; place description
       #AU x,ys,0,1,!PLAY!; replace pause button as play button

TouchMacro: stop ;called by Button stop
       #MN DELETE ; dedlete old string and set font
       #ZL st_x,st_y, "Pressed stop"       ; place description
       #AU x,ys,0,1,!PLAY!; replace pause button as play button

TouchMacro: paus ;called by Button rewind
       #MN DELETE ; dedlete old string and set font
       #ZL st_x,st_y,"Pressed pause"       ; place description
       #AU x,ys,0,1,!PLAY!; replace pause button as play button


Macro: DELETE
       #RL st_x,st_y,XMAX,YMAX      ; delete area to have clear background
       #ZF DAUPHIN                  ; set font to Dauphin
       #FZ WHITE,transparent ; set fontcolors
```

9.10 Radio group - BEGINNER

Explanation of general use of TouchButtons and TouchMacros. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 95), Buttons (see BEGINNER - 3 simple buttons.kmc 76) and another Example with touch buttons (see EXPERT - numbers to terminal with autorepeat.kmc 82)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Touch\

File:
BEGINNER - touch as radio button.kmc

Commands:
#AR, #AJ

Open file in KitEditor

```
eDIPTFT43-A "Touch as Radio Button"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
Button 1 <button\RadioButton75x16_0.bmp>,<button\RadioButton75x16_1.bmp>

;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,20,1                 ; place Picture no. 1
       #GD 30,120,449,120           ; draw a Line


;---- Place some buttons as a radio group ----

OriginX = 200                       ; using a constant makes it more easy to move the
group later
OriginY = 160
PitchY = 25

       #AF 4                                                  ; set font no. 4 for
Touch area
       #FA 8,1                                          ; define color for font
       #AR 1                                                  ; put the following
defined touch switches into the group no. 1
       #AC 1,0,1,1                                            ; use Button 1, no Zoom

       #AJ OriginX,OriginY+0*PitchY,'1',0 "L   Number 1" ; draw Touch switch - this will
put a "1" ($31, 49 dec.) into send buffer
                                                              ; the first "L" means
"left aligned text"
       #AJ OriginX,OriginY+1*PitchY,'2',0 "L   Number 2" ; draw Touch switch - this will
put a "2" ($32, 50 dec.) into send buffer
       #AJ OriginX,OriginY+2*PitchY,'3',0 "L   Number 3" ; draw Touch switch - this will
put a "3" ($33, 51 dec.) into send buffer
       #AJ OriginX,OriginY+3*PitchY,'4',0 "L   Number 4" ; draw Touch switch - this will
put a "4" ($34, 52 dec.) into send buffer
```

```
        #AP '1',1                                   ; preset switch no. 1 to
ON
```

9.11 Keypad - EXPERT

Place a keypad (0..9) and send the numbers to the terminal. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 95), Buttons (see BEGINNER - 3 simple buttons.kmc 76) and Radio groups (see BEGINNER - radiogroup.kmc 80).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Touch\

File:
EXPERT - numbers to terminal with autorepeat.kmc

Commands:
#AT

Open file in KitEditor

```
eDIPTFT43-A "numbers to terminal with autorepeat"
...
...
...
;--------------------------------------------------------------------------------
; load bitmaps
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open Bitmap
Editor

;--------------------------------------------------------------------------------
; Information about macro definition
; Content:
; 1. Place ELECTRONIC ASSEMBLY Logo
; 2. Define the small blue terminal window
; 3. Define a keypad with numbers 0..9
; 4. Define 10 touch macros for the touch keys 0..9 (key pressed)
; 5. Define one touch macro if any touch key 0..9 is released
; 6. Define 10 process macros for repeat function
;--------------------------------------------------------------------------------

; define constants for touchmacros
Nb1 = 1
Nb2 = Nb1+1
Nb3 = Nb2+1
Nb4 = Nb3+1
Nb5 = Nb4+1
Nb6 = Nb5+1
Nb7 = Nb6+1
Nb8 = Nb7+1
Nb9 = Nb8+1
Nb0 = Nb9+1

Carriage = NB0+1
FormFeed = Carriage+1

Stop = 20

; define constants for processmacros
Pm1 = 1
Pm2 = Pm1+1
Pm3 = Pm2+1
Pm4 = Pm3+1
Pm5 = Pm4+1
```

```
Pm6 = Pm5+1
Pm7 = Pm6+1
Pm8 = Pm7+1
Pm9 = Pm8+1
Pm0 = Pm9+1

;-------------------------------------------------------------------------------

Macro: MnAutoStart

;-----------------------------------------
;---- 1. Place ELECTRONIC ASSEMBLY Logo ----
;-----------------------------------------

       #TC 0                        ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)


;-------------------------------------------------
;---- 2. Define the small blue terminal window ----
;-------------------------------------------------

       #FT YELLOW,BLUE              ; define terminal colors
t_y = 8
ft = 2
t_x = 4
lines=8
column=25
       #TW ft,t_x,t_y,column,lines         ; define the font (8x16), origin
(5x8=40,7x16=112), width (20 character) and height (7 lines)
                                    ; note: origin is defined as column no.7 and line
no.10 (not dots)
       #TC 1                        ; Terminal on

;-----------------------------------------------
;---- 3. Define a keypad with numbers 0..9 ----
;-----------------------------------------------

       #FA YELLOW,BLACK                              ; define color for touch font
       #AF SWISS30B                          ; set font for touch area
       #AE 20,0                                      ; set border no. 20
       #FE WHITE,BLACK,BLUE, WHITE,BLACK,YELLOW   ; set border colors normal and for
selection

; using a constant for (touch)size and (touch)pitch makes it more easy to move the whole
key group later
; XPIXEL=320 and YPIXEL=240 are defined in the file <..\..\default_constant.kmi>, see line
12
; YPIXEL-80 to have enough space for the header
pitch = 3
ys=(t_y-1)*ft*8                     ;terminal is organised in lines 1..30/15, depending on
selcted font -> to calculate pixelpostion, decrement line and multiplay with font height
yh=(YPIXEL-ys-3*pitch)/4
xw = yh+8
xs = XPIXEL-(3*xw+2*pitch)



x=xs
y=ys
       #AT x,y,x+xw,y+yh, Nb1,Stop, "1"           ; define a touchkey with number "1".
When pressed, touchmacro no. 1 will be executed
x+=pitch+xw                                       ; while release, touchmacro no. 20 will
be executed
       #AT x,y,x+xw,y+yh, Nb2,Stop, "2"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb3,Stop, "3"

x=xs
y+=pitch+yh
       #AT x,y,x+xw,y+yh, Nb4,Stop, "4"
x+=pitch+xw
```

```
       #AT x,y,x+xw,y+yh, Nb5,Stop, "5"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb6,Stop, "6"

x=xs
y+=pitch+yh
       #AT x,y,x+xw,y+yh, Nb7,Stop, "7"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb8,Stop, "8"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb9,Stop, "9"

x=xs
y+=pitch+yh
       #AT x,y,x+xw,y+yh, FormFeed,0, "FF"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb0,Stop, "0"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Carriage,0, "CCR"


;-------------------------------------------------------------------------
;---- 4. Define 10 touch macros for the touch keys 0..9 (key pressed) ----
;-------------------------------------------------------------------------

DelayTime  = 7              ; define a constant for DelayTime (autorepeat process)
RepeatTime = 1              ; define a constant for RepeatTime (autorepeat process)
Pronumber = 1

TouchMacro: Nb1
       #ZT "1"              ; sends number "1" to terminal window
       #MD Pronumber,2, Pm1,Pm1, DelayTime        ; defines a process for autorepeating
touchkey "1",
                                  ; macro process no.1, type is "run cyclical", run from
processmacro no.1 to processmacro no.1 (see line 171)
TouchMacro: Nb2
       #ZT "2"
       #MD Pronumber,2, Pm2,Pm2, DelayTime

TouchMacro: Nb3
       #ZT "3"
       #MD Pronumber,2, Pm3,Pm3, DelayTime

TouchMacro: Nb4
       #ZT "4"
       #MD Pronumber,2, Pm4,Pm4, DelayTime

TouchMacro: Nb5
       #ZT "5"
       #MD Pronumber,2, Pm5,Pm5, DelayTime

TouchMacro: Nb6
       #ZT "6"
       #MD Pronumber,2, Pm6,Pm6, DelayTime

TouchMacro: Nb7
       #ZT "7"
       #MD Pronumber,2, Pm7,Pm7, DelayTime

TouchMacro: Nb8
       #ZT "8"
       #MD Pronumber,2, Pm8,Pm8, DelayTime

TouchMacro: Nb9
       #ZT "9"
       #MD Pronumber,2, Pm9,Pm9, DelayTime

TouchMacro: Nb0
       #ZT "0"
       #MD Pronumber,2, Pm0,Pm0, DelayTime
TouchMacro: FormFeed
       #ZT FF
TouchMacro: Carriage
       #ZT CR LF

;-------------------------------------------------------------------------
;---- 5. Define one touch macro if any touch key 0..9 is released ----
```

```
;-------------------------------------------------------------------

TouchMacro: Stop
        #MZ Pronumber,0                    ; stop autorepeat process (= macro process no.
1) after touchkey release


;-----------------------------------------------------------
;---- 6. Define 10 process macros for repeat function ----
;-----------------------------------------------------------

ProcessMacro: Pm1
        #ZT "1"               ; sends number "1" to terminal window
        #MZ Pronumber,RepeatTime           ; for macro process no.1 change process time to
RepeatTime
                                   ; (was originally DelayTime)

ProcessMacro: Pm2
        #ZT "2"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm3
        #ZT "3"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm4
        #ZT "4"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm5
        #ZT "5"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm6
        #ZT "6"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm7
        #ZT "7"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm8
        #ZT "8"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm9
        #ZT "9"
        #MZ Pronumber,RepeatTime

ProcessMacro: Pm0
        #ZT "0"
        #MZ Pronumber,RepeatTime
```

9.12 Free draw area - BEGINNER

Define a free drawing area. IThere is a an EXPERT example available, too. Please have a look at
EXPERT – free_draw_area.kmc 87

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Draw\

File:
BEGINNER – free_draw_area.kmc

Commands:
#AD

Open file in KitEditor

```
eDIPTFT43-A   "Free drawing area"
...
...
...
;-------------------------------------------------------------------------------
;include Pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,0,1                  ; place Picture no. 1
       #GD 50,100,429,100           ; draw a Line

;--- Place information ---
       #FZ WHITE, BLACK             ; set color for text
       #ZF CHICAGO14                ; set font no.5
       #ZL 10, 115,"Drawing area:"
;---- Place buttons ----
       #FA WHITE, BLUE              ; set color for touchstring
       #AE 10, 0                    ; touch frame and angle 0°
       #FC BLUE, WHITE, BLUE, YELLOW; set color for button
       #AF CHICAGO14                      ; set font no. 5 for Touch area
       #AT 300,160,420,190,1,0, "CClear"   ; place touchbutton 1

;---- Place drawing area ----
       #FG YELLOW, BLACK     ; set color for drawing box
       #GR 10,140,250,270    ; place rectangle around drawing area
       #AD 11,141,249,269,1, GREEN  ; place drawing area, linewith 1 and green drawingline



;-------------------------------------------------------------------------------
TouchMacro: 1
       #RL 11,141,249,269    ; clear drawing area
```

9.13 Free draw area - EXPERT

Define a free drawing area. There is a an BEGINNER example available, too. Please have a look at BEGINNER – free_draw_area.kmc 86

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Draw\

File:
EXPERT – free_draw_area.kmc

Commands:
#AD

Open file in KitEditor

```
eDIPTFT43-A   "Free drawing area"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open

;-------------------------------------------------------------------------------
;define constants for Touchmacro
CLEAR = 1
;-------------------------------------------------------------------------------
Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)

;---- Place drawing area ----
dw_xs = 10
dw_xw = 240
dw_ys = 140
dw_yh = YMAX-dw_ys
       #FG YELLOW, BLACK     ; set color for drawing box
       #GR dw_xs,dw_ys,dw_xs+dw_xw,dw_ys+dw_yh     ; place rectangle around drawing area
       #AD dw_xs+1,dw_ys+1,dw_xs+dw_xw-1,dw_ys+dw_yh-1,1, GREEN ; place drawing area,
linewith 1 and green drawingline

;--- Place information ---
       #FZ WHITE, BLACK             ; set color for text
       #ZF CHICAGO14                ; set font no.5
       #ZL dw_xs, dw_ys-14-5,"Drawing area:"
;---- Place buttons ----
       #FA WHITE, BLUE              ; set color for touchstring
       #AE 10, 0                    ; touch frame and angle 0°
       #FC BLUE, WHITE, BLUE, YELLOW; set color for button
       #AF CHICAGO14                        ; set font no. 5 for Touch area
xs = 30+dw_xs+dw_xw
xw = 100
ys = dw_ys
yh = 20
       #AT xs,ys,xs+xw,ys+yh,CLEAR,0, "CClear"    ; place touchbutton 1 (C=centered)
```

```
;-------------------------------------------------------------------------------
TouchMacro: CLEAR
       #RL dw_xs+1,dw_ys+1,dw_xs+dw_xw-1,dw_ys+dw_yh-1    ; clear drawing area
```

9.14 Frame - BEGINNER

Show the different borders.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Frame\

File:
BEGINNER – frame.kmc

Commands:
#RT, #RR

Open file in KitEditor

```
eDIPTFT43-A "Different Borders"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                              ; Cursor invisible
       #UI 122,0,1                        ; place picture no. 1
       #GD 40,100,439,100                 ; draw a Line

;--- Place 5 Buttons to select different Boarders ---
       ; use default paramters for Border, color and font of Touchbuttons
       #AT 5, 110,80,130,1,0,"Border1"
       #AT 5,140,80,160,2,0,"Border2"
       #AT 5,170,80,190,3,0,"Border3"
       #AT 5,200,80,220,4,0,"Border4"
       #AT 5,230,80,250,5,0,"Border5"

       #MT 2  ; run a TouchMacro to show something on the screen at startup


TouchMacro 1: ; Called by Button Border1
       #FR GREEN,GREEN,TRANSPARENT  ; set color for border
       #RE 2,0               ; set border no. 2
       #MN 1                        ; call Macro 1 (draw frame)

TouchMacro 2: ;Called by Button Border2
       #FR RED,RED,TRANSPARENT      ; set color for border
       #RE 15,0                     ; set border
       #MN 1                        ; call Macro 1 (draw frame)

TouchMacro 3: ; Called by Button Border3
       #FR BLUE,WHITE,BLUE          ; set color for border
       #RE 18,0                     ; set border no. 18
       #MN 1                        ; call Macro 1 (draw frame)

TouchMacro 4:  ;Called by Button Border4
       #RL 180,110,350,260          ; delete area, to draw button
       #FE BLUE,WHITE,BLUE, YELLOW,WHITE,YELLOW ; set colors for touchbutton
       #AE 18, 0                    ; set touchframe no. 18
       #FA YELLOW,BLUE              ; set colors for fontcolor of touchbutton
```

```
        #AT 180,150,300,190,6,0,"Border-Button" ; define button

TouchMacro 5:  ;Called by Button Border5
        #RL 180,110,350,260
        #BM 0                                   ; set fill pattern for bargraph (none)
        #FB GREEN,BLACK,TRANSPARENT             ; set colour for bargraph pattern, background
and frame
        #BE 18                                  ; set the bargraph frame
type
        #BR 1,180,150,340,180,0,100,5           ; define bargraph no. 1 with size, value and
type
        #AB 1                                   ; define bargraph no. 1 to be adjusted by the
touch
        #BA 1,75                                ; set bargraph no. 1 to value 75
        #ZL 180,200,"Bargraph with border"; place info-text


Macro 1: ; Draw Rectangel with selected Border
        #RL 180,110,350,260                     ; delete area, to draw new frame
        #RR 180,110,350,260                     ; draw new frame

TouchMacro 6: ; called by  Boder-Button
        #ZL 180,200,"Border Button was|pressed"    ; place text, that Border-Button was
touched
                                                  ; '|' means new line
```

9.15 Line recorder - EXPERT

Draw a x-graph in two different ways. Above all interessting for connection with an microcontroller.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT43-A\How to use\Draw\

File:
EXPERT - line recorder.kmc

Commands:
#GS, #GW, #GX

Open file in KitEditor

```
eDIPTFT43-A "Line-recorder"
...
...
...
;-------------------------------------------------------------------------------
;include pictures and buttons
Path <..\..\..\bitmaps\color\>

LOGO = 1 ; define constants makes it easier to use
Picture: LOGO <ea logo making things easy black.bmp>; double click to open

RadioBut = 1
Button: RadioBut  <button\RadioButton95x20_0.bmp>,<button\RadioButton95x20_1.bmp>
;-------------------------------------------------------------------------------
;information: this example shows the difference between drawing stright lines and drawing
a graph directly.
;The difference is hardly remarkable looking at execution time.
;But, if you have to send the values via an interface you will see a great difference.


;constants for grid
pitch=10
xs=10
x=xs
xb=120
xe=XPIXEL-BUTTON_W(RadioBut)-pitch
ys=PICTURE_H(LOGO)+10
yh=(YMAX-ys)
arraypitch=xe/20



Makro: MnPowerOn
;---- Place ELECTRONIC ASSEMBLY Logo ----
#TC 0; Cursor invisible
yoff = 10
#UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO; place Picture no. 1
; draw a centered line directly beneath the picture:
#GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)
#MN GRID

; define constants for radio group
x=xe+pitch
y=ys
#AF CHICAGO14 ;set font for touch
#FA WHITE, YELLOW; set color for touch font
#AC RadioBut, 0, 1, 1 ; use radiobutton
```

```
#AR 1; define first group
#AJ x,y,1,0,"CLine" ;place touch button
y+=BUTTON_H(RadioBut)+pitch
#AJ x,y,2,0,"Graph"
#AR 0; end of first group


Macro: GRID
x=xs
#RL x,ys, xe+pitch-1, YMAX
#FG GREY,BLACK; set line color
; draw vertical lines
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
x+=arraypitch
#GD x,ys,x,ys+yh
; draw horizontal lines
x=xs
y=ys
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
```

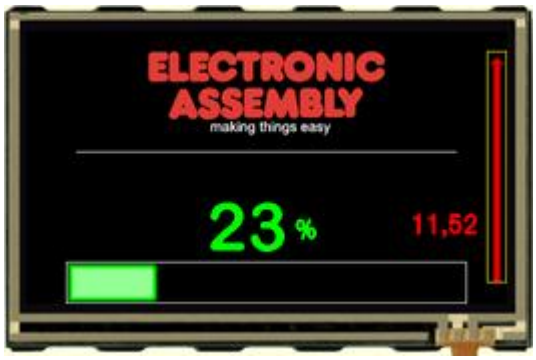
```
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y
y+=arraypitch
#GD x,y,xe,y


;-------------------------------------------------------------------------------
TouchMakro: 1 ;draw graph as lines ( use continue stright lines)
#MN GRID
#FG YELLOW,BLACK ; set color for lines
#GS  xs + 0 ,  247 ; set start point
#GW  xs + 3 ,  237 ;draw line
#GW  xs + 6 ,  95
#GW  xs + 9 ,  155
#GW  xs + 12 , 155
#GW  xs + 15 , 173
#GW  xs + 18 , 176
#GW  xs + 21 , 104
#GW  xs + 24 , 252
#GW  xs + 27 , 213
#GW  xs + 30 , 252
#GW  xs + 33 , 172
#GW  xs + 36 , 147
#GW  xs + 39 , 240
#GW  xs + 42 , 188
#GW  xs + 45 , 210
#GW  xs + 48 , 154
#GW  xs + 51 , 235
#GW  xs + 54 , 164
#GW  xs + 57 , 215
#GW  xs + 60 , 214
#GW  xs + 63 , 177
#GW  xs + 66 , 250
#GW  xs + 69 , 211
#GW  xs + 72 , 160
#GW  xs + 75 , 102
#GW  xs + 78 , 172
#GW  xs + 81 , 111
#GW  xs + 84 , 248
#GW  xs + 87 , 169
#GW  xs + 90 , 206
#GW  xs + 93 , 205
#GW  xs + 96 , 116
#GW  xs + 99 , 192
#GW  xs + 102 ,211
#GW  xs + 105 ,134
#GW  xs + 108 ,154
#GW  xs + 111 ,150
#GW  xs + 114 ,184
#GW  xs + 117 ,198
#GW  xs + 120 ,158
#GW  xs + 123 ,124
#GW  xs + 126 ,210
#GW  xs + 129 ,203
#GW  xs + 132 ,92
#GW  xs + 135 ,195
#GW  xs + 138 ,231
#GW  xs + 141 ,201
#GW  xs + 144 ,104
#GW  xs + 147 ,234
#GW  xs + 150 ,103
#GW  xs + 153 ,147
#GW  xs + 156 ,188
#GW  xs + 159 ,113
#GW  xs + 162 ,231
#GW  xs + 165 ,172
#GW  xs + 168 ,113
#GW  xs + 171 ,108
#GW  xs + 174 ,123
#GW  xs + 177 ,159
#GW  xs + 180 ,156
#GW  xs + 183 ,211
```

```
#GW  xs + 186 ,238
#GW  xs + 189 ,198
#GW  xs + 192 ,137
#GW  xs + 195 ,209
#GW  xs + 198 ,246
#GW  xs + 201 ,198
#GW  xs + 204 ,250
#GW  xs + 207 ,162
#GW  xs + 210 ,100
#GW  xs + 213 ,239
#GW  xs + 216 ,170
#GW  xs + 219 ,113
#GW  xs + 222 ,145
#GW  xs + 225 ,205
#GW  xs + 228 ,102
#GW  xs + 231 ,98
#GW  xs + 234 ,203
#GW  xs + 237 ,252
#GW  xs + 240 ,127
#GW  xs + 243 ,152
#GW  xs + 246 ,93
#GW  xs + 249 ,106
#GW  xs + 252 ,156
#GW  xs + 255 ,150
#GW  xs + 258 ,125
#GW  xs + 261 ,96
#GW  xs + 264 ,205
#GW  xs + 267 ,110
#GW  xs + 270 ,197
#GW  xs + 273 ,103
#GW  xs + 276 ,97
#GW  xs + 279 ,241
#GW  xs + 282 ,221
#GW  xs + 285 ,215
#GW  xs + 288 ,150
#GW  xs + 291 ,211
#GW  xs + 294 ,178
#GW  xs + 297 ,179
```

```
TouchMakro: 2
#MN GRID
#FG GREEN,BLACK ; set color for lines
#GS  xs + 0 ,  247 ; set start point
;draw graph in x-steps of 3. Wee need the command two times, because the parameters exeed
the limit of 255 values
#GX 3, 237, 95, 155, 155, 173, 176, 104, 252, 213, 252, 172, 147, 240, 188, 210, 154, 235,
164, 215, 214, 177, 250, 211, 160, 102, 172, 111, 248, 169, 206, 205, 116, 192, 211, 134,
154, 150, 184, 198, 158, 124, 210, 203, 92, 195, 231, 201, 104, 234, 103, 147, 188, 113,
231, 172, 113, 108, 123, 159, 156, 211, 238, 198, 137, 209, 246, 198, 250, 162, 100, 239,
170, 113, 145, 205, 102, 98, 203, 252, 127, 152, 93, 106, 156, 150, 125, 96, 205, 110,
197, 103, 97, 241, 221, 215, 150, 211, 178, 179
```

9.16 Bargraph by touch - BEGINNER

Place a bargraph, that is adjustable by touch and controls the backlight. There is a an EXPERT example available, too. Please have a look at EXPERT - 2 Bargraphs with backlight dimming.kmc 97. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 76.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Bargraph\

File:
BEGINNER - 2 Bargraphs with backlight dimming.kmc

Commands:
#BR, #YB

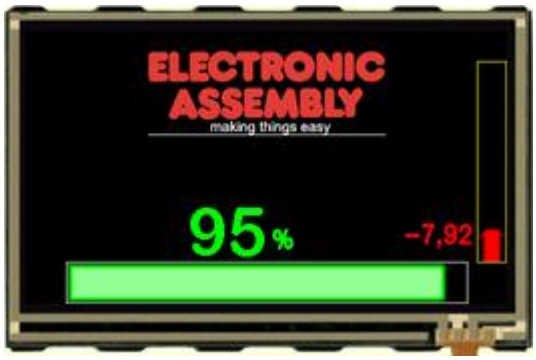
Open file in KitEditor

```
eDIPTFT43-A "2 Bargraphs with backlight dimming"
...
...
...

;-------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
       #UI 122,20,1                   ; place Picture no. 1
       #GD 50,120,429,120             ; draw a Line


;---- Place a bargraph no. 1 ----
       #BM 0                                  ; set fill pattern for bargraph (none)
       #FB GREEN,BLACK,WHITE          ; set colour for bargraph pattern, background and
frame
                                              ; same as "#FB 4,1,8
       #BE 114                        ; set the bargraph frame type
       #BR 1,40,230,439,277,0,100,5 ; define bargraph no. 1 with size, value and type
       #BA 1,57                               ; actualize bargraph no. 1 value
       #AB 1                                  ; define bargraph no. 1 to be adjusted by the
touch

;---- show the value of the bargraph 1 on the display ----
       #FX GREEN,BLACK                        ; set the colour of the value text and the
background
       #BF 7                                  ; set the textfont no.7
       #BZ 1,1                        ; set the zoom for the text size 1 in horizontal and
vertical
       #BX 1,260,170,"0=0;100=100"            ; place the value of bargraph 1 to row 240 and
line 170
                                              ; set the bottom value to 0 and the top to 100

;---- writing "%" as Text to the display ----
       #ZF 6                                  ; set the textfont no.6;
       #FZ GREEN,BLACK                        ; set the colour of the value text and the
background
       #ZZ 1,1                        ; set the zoom for the text size 1 in horizontal and
```

```
vertical
       #ZR 290,185,"%"                    ; place the text "%" to row 290 and line 170

;---- Place another bargraph no. 2 ----
       #FB YELLOW,BLACK,YELLOW            ; set colour for bargraph pattern, background
and frame
       #BE 123                       ; set the bargraph frame type
       #BO 2,460,250,480,20,0,100,5 ; define bargraph no. 2 with size, value and type
       #BA 2,38                           ; actualize bargraph no. 2 value
       #AB 2                              ; define bargraph no. 2 to be adjusted by the
touch

;---- show the value of the bargraph 2 on the display ----
       #FX RED,BLACK                      ; set the colour of the value text and the
background
       #BF 6                              ; set the textfont no.6;
       #BZ 1,1                       ; set the zoom for the text size 1 in horizontal and
vertical
       #BX 2,450,180,"0=-12,00;100=12,00"  ; place the value of bargraph 2 to row 210 and
line 190
                                          ; set the bottom value to -12,0 and the top to
12,0

;---- Brightness adjustment by bargraph 1 ----
       #YB 1                              ; brightness is adjusted by bargraph no. 1
                                          ; the actual brightness value has higher
priority than the bargraph value
                                          ; in this example 100% brightness is adjusted
after the YB command
       #BA 1,87                           ; actualize bargraph no. 1 value; now
brightness is set to 87%

;---- Deactivate transmission of bar ----

       #AQ 0                              ; deactivate transmision of bar-value to
display's sendbuffer
```

9.17 Bargraph by touch - EXPERT

Place a bargraph, that is adjustable by touch and controls the backlight. There is a a BEGINNER example available, too. Please have a look at BEGINNER - 2 Bargraphs with backlight dimming.kmc 95. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 76.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Bargraph\

File:
EXPERT - 2 Bargraphs with backlight dimming.kmc

Commands:
#BR, #YB

Open file in KitEditor

```
eDIPTFT43-A "2 Bargraphs with backlight dimming"
...
...
...

;-------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open

;-------------------------------------------------------------------------------
;define constants for bargraph numbers
BR1 = 1
BR2 = 2

;-------------------------------------------------------------------------------
Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
yoff = 20
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)


;---- Place bargraph no. 1 ----
br1x = 40
br1w = 400
br1y = 230
br1h = YMAX-br1y
       #BM 0                                ; set fill pattern for bargraph (none)
       #FB GREEN,BLACK,WHITE        ; set colour for bargraph pattern, background and
frame
                                            ; same as "#FB 4,1,8
       #BE 114                      ; set the bargraph frame type
       #BR BR1,br1x,br1y,br1x+br1w,br1y+br1h,0,100,5     ; define bargraph no. 1 with
size, value and type
       #AB BR1                      ; define bargraph no. 1 to be adjusted by the touch

;---- show the value of the bargraph 1 on the display ----
       #FX GREEN,BLACK                      ; set the colour of the value text and the
background
```

```
       #BF BIGZIF50                          ; set the textfont no.7
       #BZ 1,1                       ; set the zoom for the text size 1 in horizontal and
vertical
       #BX BR1,br1x+br1w/2,br1y-50-5,"0=0;100=100"       ; place the value of bargraph 1
right above the bar
                                             ; set the bottom value to 0 and the top to 100

;---- writing "%" as Text to the display ----
       #ZF SWISS30B                          ; set the textfont no.6;
       #FZ GREEN,BLACK                       ; set the colour of the value text and the
background
       #ZZ 1,1                       ; set the zoom for the text size 1 in horizontal and
vertical
       #ZL br1x+br1w/2+5,br1y-30-5,"%"     ; place the text "%" next to the value of the
bar

;---- Brightness adjustment by bargraph 1 ----
       #YB BR1                       ; brightness is adjusted by bargraph no. 1
                                             ; the actual brightness value has higher
priority than the bargraph value
                                             ; in this example 100% brightness is adjusted
after the YB command
       #BA BR1,79                            ; actualize bargraph no. 1 value; now
brightness is set to 79%

;---- Place another bargraph no. 2 ----
br2x = XPIXEL-10-br1h/2
br2w = br1h  ;same thickness as bar1
br2y = 230
br2h = 200
       #FB YELLOW,BLACK,YELLOW               ; set colour for bargraph pattern, background
and frame
       #BE 123                       ; set the bargraph frame type
       #BO BR2,br2x,br2y,br2x+br2w,br2y-br2h,0,100,5     ; define bargraph no. 2 with
size, value and type
       #BA BR2,38                            ; actualize bargraph no. 2 value
       #AB BR2                       ; define bargraph no. 2 to be adjusted by the touch

;---- show the value of the bargraph 2 on the display ----
       #FX RED,BLACK                         ; set the colour of the value text and the
background
       #BF SWISS30B                                ; set the textfont no.6;
       #BZ 1,1                       ; set the zoom for the text size 1 in horizontal and
vertical
       #BX BR2,br2x-5,br2y-30-10,"0=-12,00;100=12,00"    ; place the value of bargraph 2
to row 210 and line 190
                                             ; set the bottom value to -12,00 and the top to
12,00

;---- Deactivate transmission of bar ----

       #AQ 0                                 ; deactivate transmision of bar-value to
display's sendbuffer
```

9.18 Instrument by touch - BEGINNER

Place an instrument adjustable by touch. Connect back light with instrument. Instruments can be connected to an analogue input (see BEGINNER - instrument_by_analoginput.kmc[100]). If you want an overview about all instruments, refer to EXPERT - show some instruments.kmc[101].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Instruments\

File:
BEGINNER - compass_by_touch.kmc

Commands:
#IP

Open file in KitEditor

```
eDIPTFT43-A "Compass as instrument"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
path <..\..\..\instruments>
;next line: defines an instrument with kompass.bmp as background, the scale is degree
;double click to open instrument preview and to edit/change instrument settings
Instrument: 1 <compass_small.i16>

;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,0,1                  ; place Picture no. 1
       #GD 50,100,429,100           ; draw a Line

;---- Place Insturment ----
       #IP 1,  169,110, 1,0, 0,100  ; place instument no. 1 with angle and start/end value
       #A+ 1                        ; instrument is controlled by touch
       #Y+ 1                        ; instrument is assigned to brightness
```

9.19 Instrument by analogue input - BEGINNER

Place instrument and connect them with the anlogue input.Instruments can be connected to the backlight (see BEGINNER - compass_by_touch.kmc 99). If you want an overview about all instruments, refer to EXPERT - show some instruments.kmc 101.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Instruments\

File:
BEGINNER - instrument_by_analoginput.kmc

Commands:
#IP, #V+

Open file in KitEditor

```
eDIPTFT43-A "Instruments controlled by analog input"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
path <..\..\..\instruments>
;next line: defines an instrument with kompass.bmp as background, the scale is degree
;double click to open instrument preview and to edit/change instrument settings
Instrument: 1 <voltmeter.i16>
;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,0,1                  ; place Picture no. 1
       #GD 50,100,429,100           ; draw a Line

       #VE 1,"0=0;5000=5.00"

;---- Place Insturment ----
       #IP 1,  130,120, 1,0, 0,250  ; place instument no. 1 with angle and start/end value

       #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)

       #VA 1         ; enable analog inputs

;-------------------------------------------------------------------------------
AnalogeMacro: 0       ; rising analog input 1 (->hysteresis)
       #VR 1          ; redraw insturment 1 with new anlog value
       #VG 1,400,250
```

9.20 Instrument slide show - EXPERT

Place many instrumentsand connect them with the anlogue input.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Instruments\

File:
EXPERT - show some instruments.kmc

Commands:
#IP, #V+

Open file in KitEditor

```
eDIPTFT43-A "Instruments controlled by analog input"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO, <ea logo making things easy black.bmp>    ; double click to open
path <..\..\..\instruments>
;next line: defines an instrument
;double click to open instrument preview and to edit/change instrument settings
Instrument_1=1
Instrument_2=2
Instrument_3=3
Instrument_4=4
Instrument_5=5
Instrument_6=6

Instrument_max=20


Instrument: Instrument_1 <voltmeterdown.i16>
Instrument: Instrument_2 <green amperemeter.i16>
Instrument: Instrument_3 <handwheel4.i16>
Instrument: Instrument_4 <tachometer.i16>
Instrument: Instrument_5 <WattmeterOutside.i16>
Instrument: Instrument_6 <hygrometer.i16>
Instrument: Instrument_max <EA.i16>

Wingdings=9
WinFont: Wingdings,"Wingdings",1,0, 231-232, 18
;-------------------------------------------------------------------------------
BT_Left_xstart=5
BT_Left_ystart=250
BT_width=80
BT_height=YMAX-BT_Left_ystart

BT_Right_xstart=XMAX-BT_width-BT_Left_xstart
BT_Right_ystart=BT_Left_ystart



Macro: MnAutoStart
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
yoff = 10
YoffLOGO=yoff+PICTURE_H(LOGO)
YSPACE=YPIXEL-YoffLOGO
```

```
        #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
        ; draw a centered line directly beneath the picture:
        #GD (XPIXEL-PICTURE_W(LOGO))/2,YoffLOGO,(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),YoffLOGO

        #VA 1          ; enable analog inputs
        #VE "0=0.00;5000=5.00" ; set user string for analog output
        #VF 1, SWISS30B ; set font for analog output

        #ZF Chicago14
        #AF Wingdings

        #MT Instrument_1


;-------------------------------------------------------------------------------
TouchMacro: Instrument_1
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        ; place instument no. 1 with angle and start/end value
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_1))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_1))/2, INSTRUMENT_1,0, 0,250
        #ZR XMAX,YoffLOGO+1,"voltmeterdown.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_max,{E7}
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_2,{E8}

        #MV 0 ; run analog macro 0 to update instrument

TouchMacro: Instrument_2
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_2))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_2))/2, INSTRUMENT_2,0, 0,250
        #ZR XMAX,YoffLOGO+1,"green amperemeter.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_1,{E7}
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_3,{E8}

        #MV 0 ; run analog macro 0 to update instrument

TouchMacro: Instrument_3
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_3))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_3))/2, INSTRUMENT_3,0, 0,250
        #ZR XMAX,YoffLOGO+1,"handwheel4.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_2,{E7}
```

```
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_4,{E8}

        #MV 0 ; run analog macro 0 to update instrument


TouchMacro: Instrument_4
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_4))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_4))/2, INSTRUMENT_4,0, 0,250
        #ZR XMAX,YoffLOGO+1,"tachometer.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_3,{E7}
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_5,{E8}

        #MV 0 ; run analog macro 0 to update instrument


TouchMacro: Instrument_5
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_5))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_5))/2, INSTRUMENT_5,0, 0,250
        #ZR XMAX,YoffLOGO+1,"WattmeterOutside.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_4,{E7}
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_6,{E8}

        #MV 0 ; run analog macro 0 to update instrument


TouchMacro: Instrument_6
        #AL 0,0; delete Touchbutton, but remain visible
        #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
        #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
        #IP 1,(XPIXEL-INSTRUMENT_W(INSTRUMENT_6))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(INSTRUMENT_6))/2, INSTRUMENT_6,0, 0,250
        #ZR XMAX,YoffLOGO+1,"Hygrometer.i16"

        #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
        #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
        #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_5,{E7}
        #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_max,{E8}

        #MV 0 ; run analog macro 0 to update instrument

TouchMacro: Instrument_max
        #AL 0,0; delete Touchbutton, but remain visible
```

```
       #RL BT_Left_xstart+BT_width+1,YoffLOGO+1, BT_Right_xstart-1,YMAX ; delete old
instrument including name
       #RL 0,YoffLOGO+1,XMAX,YoffLOGO+1+20
;---- Place Instrument ----
       #IP 1,(XPIXEL-INSTRUMENT_W(Instrument_max))/2,YoffLOGO+(YSPACE-
INSTRUMENT_H(Instrument_max))/2, Instrument_max,0, 0,250
       #ZR XMAX,YoffLOGO+1,"EA.i16"

       #V+ 1,1; connect instrument 1 and analog input 1 (AIN1)
       #VR 1          ; redraw instrument 1 with new anlog value
;---- Place navigation buttons ----
       #AT
BT_Left_xstart,BT_Left_ystart,BT_Left_xstart+BT_width,BT_Left_ystart+BT_height,0,Instrumen
t_6,{E7}
       #AT
BT_Right_xstart,BT_Right_ystart,BT_Right_xstart+BT_width,BT_Right_ystart+BT_height,0,Instr
ument_1,{E8}

       #MV 0 ; run analog macro 0 to update instrument


;-------------------------------------------------------------------------------
;using string constants makes it easier to calculate positions
!VOLTAGE! = "Voltage|AnlogIn 1:"
x=400
y=155
AnalogeMacro: 0       ; rising analog input 1 (->hysteresis)
       #VR 1          ; redraw instrument 1 with new anlog value
       #ZL x,y, !VOLTAGE!
str_par = STRING_P(1,1,0,0,0,0) ; you have to define the string parameters. It's
neccessary for STRIN_W. Please look into help (F1) for further information
       #VG 1, x+STRING_W(!VOLTAGE!, str_par, Chicago14),y+STRING_H(!VOLTAGE!, str_par,
Chicago14)+3 ;diplay analog value
```

9.21 Languages/Macro Pages - BEGINNER

Describe the important function of different languages, with the help of MacroPages. If you want to use string tables, refer to EXPERT - stringtable.kmc 107.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Language\

File:
BEGINNER – multilingual.kmc

Commands:
#MK

Open file in KitEditor

```
eDIPTFT43-A   "Multilingual Support"
...
...
...
;--------------------------------------------------------------------------------
; constants for language support
GERMAN        = 0
ENGLISH       = 1
FRENCH = 2
ITALIAN= 3

;--------------------------------------------------------------------------------
; Bilder einbinden  max. 256 Bilder (0..255)
Path <..\..\..\bitmaps\color\>                    ; specify path
Picture 1 <ea logo making things easy black.bmp>  ; double click to open

PATH: <.\Bitmap\>
Picture: 100[GERMAN] <SausageBeer.jpg>
Picture: 100[ENGLISH]<FishandChips.jpg>
Picture: 100[FRENCH] <Baguette.jpg>
Picture: 100[ITALIAN]<Pizza.jpg>

;--------------------------------------------------------------------------------
Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,0,1                  ; place Picture no. 1
       #GD 50,100,429,100           ; draw a Line
       #GD 240,100,240,270

;---- Place some text ----
       #FZ WHITE, BLACK      ;string color
       #ZF CHICAGO14         ;string font
       #ZL 5,120, "Select Language:"

;--- 3 Touchbuttons (Language selection) ---
       #AE 14,0                                   ; set Frame style no. 14 and rotation
for Touch (1..20)
       #AF 5                                      ; set font no. 5 for Touch area
       #AT 5,150,85,180,1,0 "Deutsch"             ; place button "German" and call
TouchMacro 1
       #AT 5,190,85,220,2,0 "English"             ; place button "English" and call
TouchMacro 2
       #AT 95,150,175,180,3,0 "Fran"135"ais"      ; place button "French" and call
TouchMacro 3
                                                  ; the 135 is the decimal value for
```

```
cdille
       #AT 95,190,175,220,4,0 "Italiano"          ; place button "Italian" and call
TouchMacro 4

;--- Call standard language ---
       #MN 1  ; Call macro 1 with standard language (Page=0) i.e. German

;------------------------------------------------------------------------------
TouchMacro 1:
       #MK GERMAN     ; select page
       #MN 1          ; call macro 1

TouchMacro 2:
       #MK ENGLISH    ; select page
       #MN 1          ; call macro 1

TouchMacro 3:
       #MK FRENCH     ; select page
       #MN 1          ; call macro 1

TouchMacro 4:
       #MK ITALIAN    ; select page
       #MN 1          ; call macro 1


;------------------------------------------------------------------------------
Macro 1[GERMAN]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 250,110,360,140          ; delete area behind text
       #ZL 250,120, "Deutsch"       ; write actual language
       #UI 250,140,100              ; place picture

Macro 1[ENGLISH]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 250,110,360,140          ; delete area behind text
       #ZL 250,120, "English"       ; write actual language
       #UI 250,140,100              ; place picture

Macro 1[FRENCH]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 250,110,360,140          ; delete area behind text
       #ZL 250,120,"Fran"135"ais"   ; write actual language
       #UI 250,140,100              ; place picture

Macro 1[ITALIAN]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 250,110,360,140          ; delete area behind text
       #ZL 250,120, "Italiano"      ; write actual language
       #UI 250,140,100              ; place picture
```

9.22 String tables - EXPERT

Different languages, using string table, so you don't need macro pages. If you want to use MacroPages, refer to BEGINNER – multilingual.kmc 105.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Language\

File:
EXPERT - stringtable.kmc

Commands:
#ST, #MK

Open file in KitEditor

```
eDIPTFT43-A   "Stringtable"
...
...
...
;---------------------------------------------------------------------------
; Bilder einbinden  max. 256 Bilder (0..255)
Path <..\..\..\bitmaps\color\>                    ; specify path
LOGO = 1 ; unsing constants makes it easier
Picture: LOGO, <ea logo making things easy black.bmp>    ; double click to open


;---------------------------------------------------------------------------
;define Stringtable (not Stringkonstants)
StringCode = $01 ; define String table code (see command #ST)

ENGLISH = 0
GERMAN  = 1
FRENCH  = 2
ITALIAN = 3

HELLO = 1

STRING: HELLO[ENGLISH] "Hello World!"
STRING: HELLO[GERMAN]  "Hallo Welt!"
;you can use EditBox to write your strings
;please select font (double click on default font.kmi)-> right click on CHICAG14.FXT and
select font for EditBox
;double click on string below and the EditBox will open.
STRING: HELLO[FRENCH]  {426F6E6A6F7572208520746F757321} ;same as "Bonjour à tous!"
STRING: HELLO[ITALIAN] "Ciao a tutti!"

;---------------------------------------------------------------------------
;define constants for normal macros
SHOWSTR = 1

;define constants for touch macros
GER = 1
ENG = 2
FRE = 3
ITA = 4


;---------------------------------------------------------------------------


Macro: MnAutoStart
```

```
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)

;---- Use internal string table ----
       #ST StringCode

;--- 3 Touchbuttons (Language selection) ---
xs = 5
xw = 80
ys = 150
yh = 30
pitch = 10
x = xs
y = ys
       #AE 14,0                                    ; set Frame style no. 14 and rotation
for Touch (1..20)
       #AF CHICAGO14                               ; set font no. 5 for Touch area
       #AT x,y,x+xw,y+yh,GER,0 "Deutsch"           ; place button "German" and call
TouchMacro 1
y+=yh+pitch
       #AT x,y,x+xw,y+yh,ENG,0 "English"           ; place button "English" and call
TouchMacro 2
x+=xw+pitch
y=ys
       #AT x,y,x+xw,y+yh,FRE,0 "Fran"135"ais"      ; place button "French" and call
TouchMacro 3
y+=yh+pitch                                              ; the 135 is the decimal value
for cdille
       #AT x,y,x+xw,y+yh,ITA,0 "Italiano"          ; place button "Italian" and call
TouchMacro 4

;---- Place some text ----
       #FZ WHITE, BLACK      ;string color
       #ZF CHICAGO14         ;string font
       #ZL xs,ys-2*pitch, "Select Language:"

;--- Call standard language ---
       #MN SHOWSTR    ; Call macro 1 with standard language (Page=0) i.e. ENGLISH

;-------------------------------------------------------------------------------
TouchMacro: GER
       #MK GERMAN     ; select page
       #MN SHOWSTR    ; call macro 1

TouchMacro: ENG
       #MK ENGLISH    ; select page
       #MN SHOWSTR    ; call macro 1

TouchMacro: FRE
       #MK FRENCH     ; select page
       #MN SHOWSTR    ; call macro 1

TouchMacro: ITA
       #MK ITALIAN    ; select page
       #MN SHOWSTR    ; call macro 1


;-------------------------------------------------------------------------------
Macro: SHOWSTR
       #FZ WHITE, BLACK              ; string color
       #ZF SWISS30B                  ; string font
       #RL xs+2*xw+pitch+5,ys+5,XMAX,ys+5+35      ; delete area behind text
       #ZC XPIXEL*3/4,ys+5, StringCode, HELLO
```

9.23 Analogue Macro - Beginner

Show the use of analogue inputs and analogue macros. If you want to use I/Os plese refer to BEGINNER – Bit Macro.kmc[111] or EXPERT – Port Macro.kmc[113]. In adition you will find halp using ProcessMacros (see BEGINNER - Prozess Macro.kmc[117]) and AutomaticMacros (see EXPERT – Automatic Macro.kmc[115]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Macro\

File:
BEGINNER - Analog Macro.kmc

Commands:
#VG, #V@, #VE

Open file in KitEditor

```
eDIPTFT43-A "Analog Macro"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
       #UI 122,0,1                    ; place Picture no. 1
       #GD 50,100,429,100             ; draw a Line
       #GR 360,110,460,230            ; draw a rectangle

;---- Write some information ----
       #FZ WHITE, BLACK      ;string color
       #ZF CHICAGO14         ;string font
       #ZC 100,130, "Analog input (AIN1 Pin 23):"
       #ZR 450,130, "Calibration:"

;---- Place a bargraph no. 1 ----
       #BM 0                                  ; set fill pattern for bargraph (none)
       #FB GREEN,BLACK,WHITE          ; set colour for bargraph pattern, background and
frame
                                              ; same as "#FB 4,1,8
       #BE 114                        ; set the bargraph frame type
       #BR 1,10,230,310,260,0,250,5 ; define bargraph no. 1 with size, value and type (250
= Vdd)

;--- Analog input ---
       #VB 1, 1                       ; assign analog input 1 to bargraph 1
       #VR 1                          ; redraw bargraph analog input 1

;---- show the value of the analog input 1 on the display ----
       #FV 1,GREEN,BLACK                      ; set the colour of the value text and the
background
       #VF 1,7                        ; set the textfont no.7
       #VZ 1,1,1                              ; set the zoom for the text size 1 in
horizontal and vertical
       #VE 1,"0=0.000;5000=5.000"             ; set new user format for AIN1
       #VG 1,250,170                          ; place the value of AIN 1
```

```
;---- writing "V" as Text to the display ----
       #ZF 6                                ; set the textfont no.6;
       #FZ GREEN,BLACK                      ; set the colour of the value text and the
background
       #ZZ 1,1                       ; set the zoom for the text size 1 in horizontal and
vertical
       #ZR 275,195,"V"                      ; place the text "%" to row 200 and line 145

;--- 2 Touchbuttons (Calibration) ---
       #AE 14,0                             ; set Frame style no. 14 and rotation for Touch
(1..20)
       #AF 5                                ; set font no. 5 for Touch area
       #AT 370,150,450,180,1,0 "5V"         ;
       #AT 370,190,450,220,2,0 "3.3V"



;-------------------------------------------------------------------------------
AnalogMacro 0:        ; this macro is called by evry change of input voltage AIN1
       #VR 1                ; redraw bargraph analog input 1
       #VG 1,250,170        ; refresh value
       ;#VD 1               ; send analog value



;-------------------------------------------------------------------------------
TouchMacro 1:
       #V@ 1,5000                   ; calibration procedure to set 5.0V
       #VE 1,"0=0.000;5000=5.000"   ; set new user format for AIN1


;-------------------------------------------------------------------------------
TouchMacro 2:
       #V@ 1,3300                   ; calibration procedure to set 3.3V
       #VE 1,"0=0.000;3300=3.300"   ; set new user format for AIN1
```

9.24 Bit Macro - BEGINNER

Get into the use of BitMacros, i.e. get an idea of working with I/Os. There are further examples available, containig information about AutomaticMacro (see EXPERT – Automatic Macro.kmc 115), ProcessMacros (see BEGINNER - Prozess Macro.kmc 117), PortMacros (see BEGINNER - Port Macro.kmc 113) and AnalogueMacros (see BEGINNER - Analog Macro.kmc 109).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Macro\

File:
BEGINNER – Bit Macro.kmc

Commands:
#YW

Open file in KitEditor

```
eDIPTFT43-A "Bit Macro"
...
...
...
;-------------------------------------------------------------------------------
Picture 1 <..\..\..\bitmaps\color\ea logo making things easy black.bmp> ; double click to
open Bitmap Editor
;-------------------------------------------------------------------------------

Macro: MnAutoStart
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,0,1                  ; place Picture no. 1
       #GD 30,90,449,90             ; draw a Line
       #GD 240,90,240,272

;----- OUTPUTS ---
       #FZ WHITE, BLACK      ;String color
       #ZF CHICAGO14         ;String font
       #ZL 5,100, "OUTPUTS"
       #ZL 5,120, "Port 1 (Pin 25):"
       #ZL 5,170, "Port 2 (Pin 26):"
       #ZL 5,220 "All ports (Pin 25-32):"


       #FE BLUE, WHITE, BLUE, YELLOW, WHITE, YELLOW      ; Define button colors
       #FA YELLOW, BLUe                                  ; Define text colors
       #AF CHICAGO14                                ; Touchfont

       #AT 5,140,65,160,1,0,"CToggle"            ; Define Button with TouchMacro 1
                                                 ; "C" means aligment centered
       #AT 5,190,65,210,2,0,"CSet"               ; Define Button with TouchMacro 2(down
code) and 3 (up code)
       #AT 75,190,125,210,3,0,"CReset"           ; Define Button with TouchMacro 2(down
code) and 3 (up code)
       #AT 5,240,65,260,4,0,"CSet"               ; Define Button with TouchMacro 3
       #AT 75,240,125,260,5,0,"CReset"           ; Define Button with TouchMacro 4

;---- INPUTS ---
       #FZ WHITE, BLACK      ;String color
       #ZF CHICAGO14         ;String font
       #ZL 260,100, "INPUTS"
       #ZL 260, 120, "Port 1:"
       #ZL 260, 170, "Port 2:"
```

```
;-------------------------------------------------------------------------------
TouchMacro 1:
       #YW 1, 2              ; Toggle port 1

TouchMacro 2:
       #YW 2, 1              ; Set port 2

TouchMacro 3:
       #YW 2, 0              ; Reset port 2

TouchMacro 4:
       #YW 0, $FF            ; Set all ports

TouchMacro 5:
       #YW 0, 0              ; Reset all ports

;-------------------------------------------------------------------------------
BitMacro 9:                  ;Port 1 rising edge
       #FZ YELLOW, BLACK     ;String color
       #ZL 280, 140, "1"
BitMacro 1:                  ;Port 1 falling edge
       #FZ YELLOW, BLACK     ;String color
       #ZL 280, 140, "0"

BitMacro 10:                 ;Port 2 rising edge
       #FZ YELLOW, BLACK     ;String color
       #ZL 280, 190, "1"
BitMacro 2:                  ;Port 2 falling edge
       #FZ YELLOW, BLACK     ;String color
       #ZL 280, 190, "0"
```

9.25 Port Macro - EXPERT

Get into the use of PortMacros, i.e. get an idea of working with I/Os. There are further examples available, containig information about AutomaticMacro (see EXPERT – Automatic Macro.kmc 115), ProcessMacros (see BEGINNER - Prozess Macro.kmc 117), BitMacros (see BEGINNER – Bit Macro.kmc 111) and AnalogueMacros (see BEGINNER - Analog Macro.kmc 109).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Macro\

File:
EXPERT – Port Macro.kmc

Commands:
#YW

Open file in KitEditor

```
eDIPTFT43-A "Port Macro"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open Bitmap
Editor

;-------------------------------------------------------------------------------
;define constants for bit pattern
PATTERN1= $FE
PATTERN2= $7F

;-------------------------------------------------------------------------------
;define constans for touchmacros
PORT1TOG = 1
PORT2SET = 21
PORT2RES = PORT2SET+1
PORTALLSET = 100
PORTALLRES = PORTALLSET+1
;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)
       #GD XPIXEL/2,yoff+PICTURE_H(LOGO),XPIXEL/2,YMAX

;----- OUTPUTS ---
       #FZ WHITE, BLACK      ;String color
       #ZF CHICAGO14         ;String font
xs=5
y_title = 100
       #ZL xs,y_title, "OUTPUTS"
ys=y_title+20
y=ys
pitch=50
```

```
       #ZL xs,y, "Port 1 (Pin 25):"
y+=pitch
       #ZL xs,y, "Port 2 (Pin 26):"
y+=pitch
       #ZL xs,y "All ports (Pin 25-32):"


       #FE BLUE, WHITE, BLUE, YELLOW, WHITE, YELLOW      ; Define button colors
       #FA YELLOW, BLUE                                  ; Define text colors
       #AF CHICAGO14                                ; Touchfont
x=xs
bt_ypitch=18
y=ys+bt_ypitch
yh=20
xw=60
bt_xpitch = 10
x=xs
       #AT x,y,x+xw,y+yh,PORT1TOG,0,"CToggle"            ; Define Button with TouchMacro
1
                                                    ; "C" means aligment centered
y+=pitch
       #AT x,y,x+xw,y+yh,PORT2SET,0,"CSet"               ; Define Button with TouchMacro
2(down code) and 3 (up code)
x+=bt_xpitch+xw
       #AT x,y,x+xw,y+yh,PORT2RES,0,"CReset"             ; Define Button with TouchMacro
2(down code) and 3 (up code)
y+=pitch
x=xs
       #AT x,y,x+xw,y+yh,PORTALLSET,0,"CSet"                   ; Define Button with
TouchMacro 3
x+=bt_xpitch+xw
       #AT x,y,x+xw,y+yh,PORTALLRES,0,"CReset"           ; Define Button with TouchMacro
4

;---- INPUTS ---
       #FZ WHITE, BLACK      ;String color
       #ZF CHICAGO14         ;String font
       #ZL XPIXEL/2+bt_xpitch,y_title, "INPUTS"

       #ZL XPIXEL/2+bt_xpitch, ys, "Port 0x" !HEXSTR(PATTERN1,2)! ':'         ; use
compiler funtion !HEXSTRING(value, digits)! to complete
       #ZL XPIXEL/2+bt_xpitch, ys+pitch, "Port 0x" !HEXSTR(PATTERN2, 2)! ':'   ; the
string, even if Bit-patterns are changed (transform constant as hexstring)



;-------------------------------------------------------------------------------
TouchMacro: PORT1TOG
       #YW 1, 2              ; Toggle port 1

TouchMacro: PORT2SET
       #YW 2, 1              ; Set port 2

TouchMacro: PORT2RES
       #YW 2, 0              ; Reset port 2

TouchMacro: PORTALLSET
       #YW 0, $FF            ; Set all ports

TouchMacro: PORTALLRES
       #YW 0, 0              ; Reset all ports

;-------------------------------------------------------------------------------
PortMacro: PATTERN1
       #FZ YELLOW, BLACK     ;String color
       #ZL XPIXEL/2+bt_xpitch, ys+bt_ypitch, "Bit-pattern:|" !BINSTR(PATTERN1, 8)!

PortMacro: PATTERN2
       #FZ YELLOW, BLACK     ;String color
       #ZL XPIXEL/2+bt_xpitch, ys+bt_ypitch+pitch, "Bit-pattern:|" !BINSTR(PATTERN2, 8)!
```

9.26 Automatic Macro - EXPERT

A little animation with the help of automatic macros. TThere are further examples available, containig information about I/Os (see BEGINNER - Analog Macro.kmc[109], BEGINNER – Bit Macro.kmc[111], EXPERT – Port Macro.kmc[113]) and a ProcessMacro example (see BEGINNER - Prozess Macro.kmc[117]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Macro\

File:
EXPERT – Automatic Macro.kmc

Commands:
#MJ

Open file in KitEditor

```
eDIPTFT43-A "Automatic Macro"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture: LOGO <ea logo making things easy black.bmp>     ; double click to open Bitmap
Editor
;-------------------------------------------------------------------------------
;define constants for normal macros
Mn1 = 1
Mn2 = MN1+1
Mn3 = MN2+1
Mn4 = MN3+1
Mn5 = MN4+1
Mn6 = MN5+1

;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
yoff = 10
       #UI (XPIXEL-PICTURE_W(LOGO))/2,yoff,LOGO          ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,yoff+PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),yoff+PICTURE_H(LOGO)

;---- Count up and down ----
       #FZ YELLOW,BLACK             ; set text color and background color
                                    ; YELLOW is defined as 7 (compare with line 12:
"include <..\..\default_constant.kmi>"
       #ZF BIGZIF100                ; set font to no. 8 (Big Numbers)
       #MJ Mn1,Mn6,5                ; run macros 1..6 automatically
                                    ; MJ = Ping Pong Mode
;---- Place Digit -----
X = XPIXEL/2                        ; defining a constant makes it more easy to move the
whole group later
Y = 150

Macro: Mn1
       #ZC X,Y "1"
```

```
Macro: Mn2
       #ZC X,Y "2"

Macro: Mn3
       #ZC X,Y "3"

Macro: Mn4
       #ZC X,Y "4"

Macro: Mn5
       #ZC X,Y "5"

Macro: Mn6
       #ZC X,Y "6"
```

9.27 Process Macro - BEGINNER

A little animation with the help of automatic macros. There are further examples available, containig information about I/Os (see BEGINNER - Analog Macro.kmc[109], BEGINNER – Bit Macro.kmc[111], EXPERT – Port Macro.kmc[113]) and an AutomaticMacro example (see EXPERT – Automatic Macro.kmc[115]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Macro\

File:
BEGINNER - Prozess Macro.kmc

Commands:
#MD

Open file in KitEditor

```
eDIPTFT43-A "Prozess Macro"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open Bitmap Editor
;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                          ; Cursor invisible
       #UI 122,0,1                    ; place Picture no. 1
       #GD 30,90,449,90               ; draw a Line

;---- Count up and down ----
       #FZ YELLOW,BLACK               ; set text color and background color
                                      ; YELLOW is defined as 7 (compare with line 12:
"include <..\..\default_constant.kmi>"
       #ZF 8                          ; set font to no. 8 (Big Numbers)
       #MD 1,3, 1,6, 5                ; define macro process 1, pingpong mode, call
automatic macro 1 to 6 delay 5/10s

;---- Place Digit -----
X = 240                       ; defining a constant makes it more easy to move the whole
group later
Y = 150

ProcessMacro: 1
       #ZC X,Y "1"

ProcessMacro: 2
       #ZC X,Y "2"

ProcessMacro: 3
       #ZC X,Y "3"

ProcessMacro: 4
       #ZC X,Y "4"

ProcessMacro: 5
       #ZC X,Y "5"
```

```
ProcessMacro: 6
        #ZC X,Y "6"
```

9.28 Change display orientation - BEGINNER

Change the display orientation in every possible direction.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIPTFT43-A\How to use\Display orientation\

File:
BEGINNER – change displayorinetation.kmc

Commands:
#DO

Open file in KitEditor

```
eDIPTFT43-A   "Displayorientation"
...
...
...
;-------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TC 0                        ; Cursor invisible
       #UI 122,20,1                 ; place Picture no. 1
       #GD 50,120,429,120           ; draw a Line

;---- Place Button ----
       #AF CHICAGO14                                ; define next buttons as radiogroup 1
       #AE 2, 0                                     ; slect touchframe
       #AT 170, 150, 310, 175, 1,0,"CChange orientation" ; place button to change
orientation, 'C'=center aligned




;-------------------------------------------------------------------------------
TouchMacro: 1
       #DO 1          ; set displayorientation to 90°
       #AL 0,0; delete touchbuttons
       #DL            ; clear display
       #UI 18,20,1    ; place logo
       #GD 10,120,262,120           ; draw a Line
       #AT 66, 150, 206, 175, 2,0,"CChange orientation"  ; place button to change
orientation, 'C'=center aligned

TouchMacro: 2
       #DO 2          ; set displayorientation to 180°
       #AL 0,0; delete touchbuttons
       #DL            ; clear display
       #UI 122,20,1                 ; place Picture no. 1
```

```
        #GD 50,120,429,120          ; draw a Line
        #AT 170, 150, 310, 175, 3,0,"CChange orientation" ; place button to change
orientation, 'C'=center aligned

TouchMacro: 3
        #DO 3           ; set displayorientation to 270°
        #AL 0,0; delete touchbuttons
        #DL             ; clear display
        #UI 18,20,1     ; place logo
        #GD 10,120,262,120          ; draw a Line
        #AT 66, 150, 206, 175, 4,0,"CChange orientation"  ; place button to change
orientation, 'C'=center aligned

TouchMacro: 4
        #DO 0           ; set displayorientation to 0°
        #AL 0,0; delete touchbuttons
        #DL             ; clear display
        #MN 0           ; call Macro MnAutoStart
```