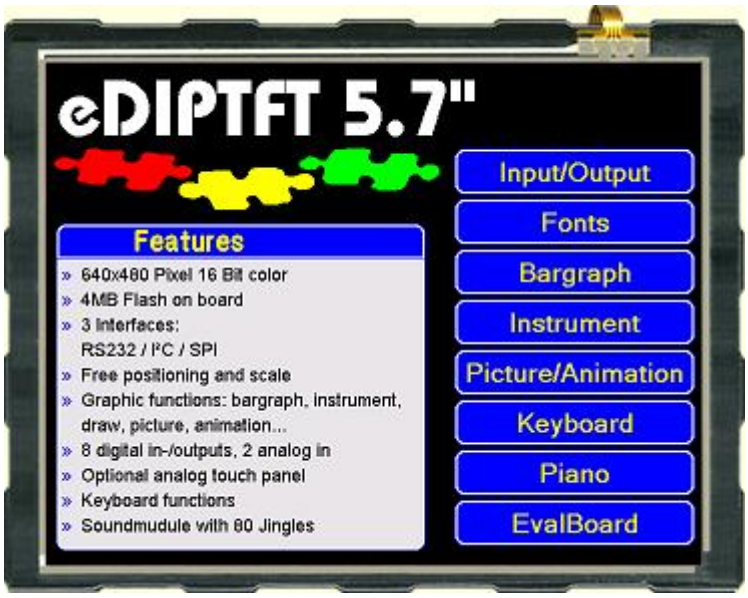
EA eDIPTFT57-A
compiler manual

Juli 2012

Table of Contents

1 Overview



The EA eDIPTFT57-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 eDIPTFT57-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 eDIPTFT57-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 58)

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

`eDIPTFT57-A "title"`	Defines EA eDIPTFT57-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 EVALeDIPTFT57 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 77)

`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 44, #BX 39,#IX 40,#VE 55, #ND 52, #KB,#KL 48, #ES,#EL,#EC,#ER 50)
After the stringcode, defined with #ST n1 58, 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 138)

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

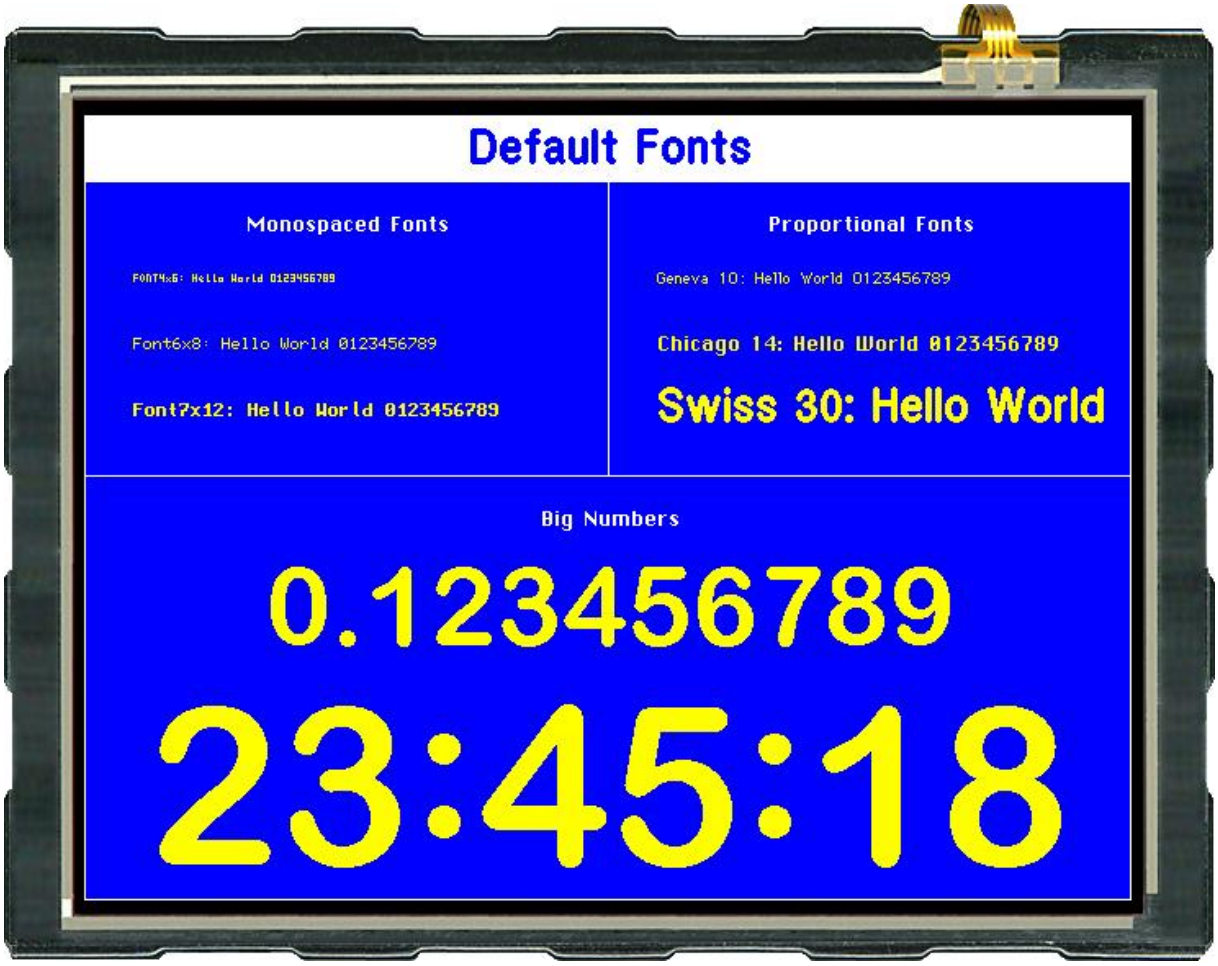
```
FONT: nr,<file>
FONT: nr[page],<file>
```

Defines a font file which will be assigned to the number nr (0..255). <file> can be FXT, G16 [70].
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 [79])

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

```
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 [59]
Terminal 8x16 [60]
Font 4x6 [61]
Font 6x8 [62]
Font 7x12 [63]
Geneva 10 [64]
Chicago 14 [65]
Swiss 30 [66]
BigZif 50 [67]
BigZif 100 [68]

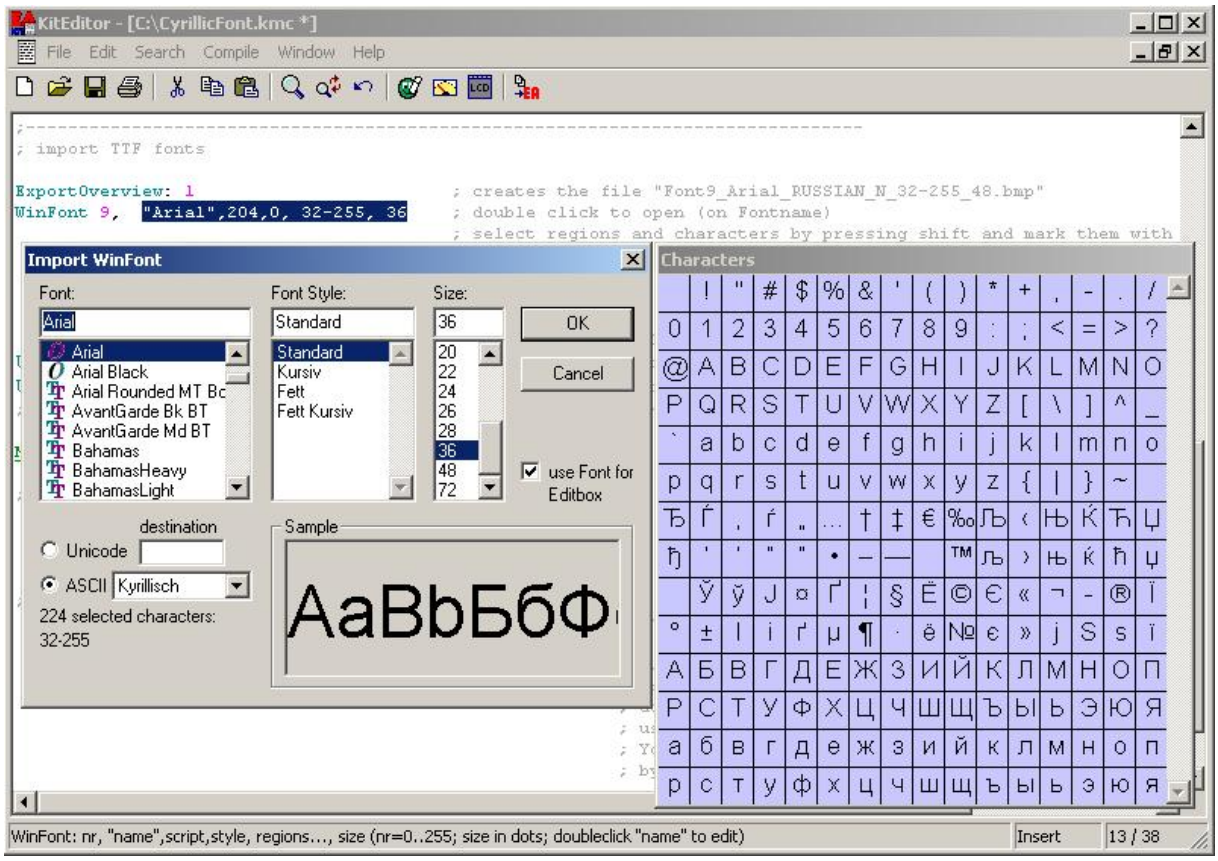


4.4.2 WinFont

```
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 83)



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

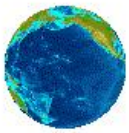
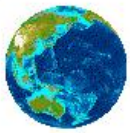
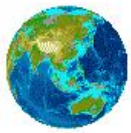
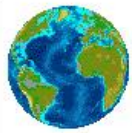
Example:

```
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: 640x480) the size can be reduced automatically to fit to the display.

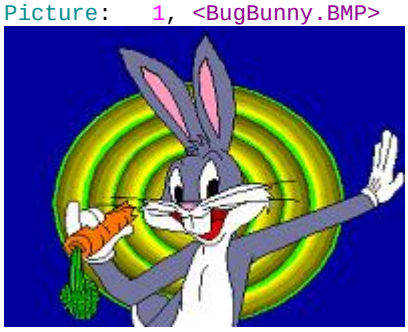
p = 1 reduce proportional
p = 0 reduce non proportional to exact "width" and "height", distortions are possible

---

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

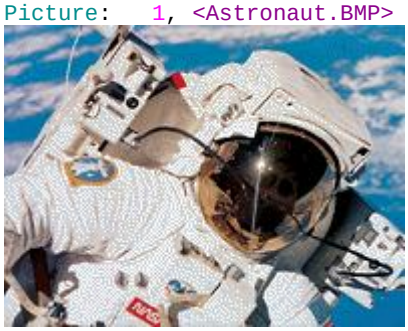
`MAXCOLORDEPTH: bitpixel` Reduces color depth of bitmaps. This saves memory space. Attention: This may effect to the quality of the image.

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 (default)

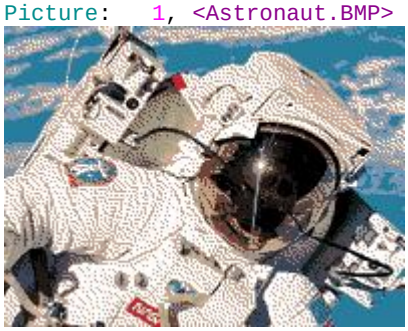
Examples:

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

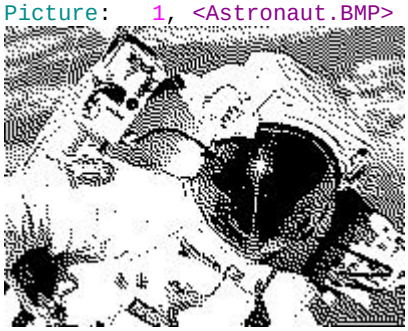
```
MaxColorDepth: 8
Picture:   1, <Astronaut.BMP>
```



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



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



4.5.3 Dithering

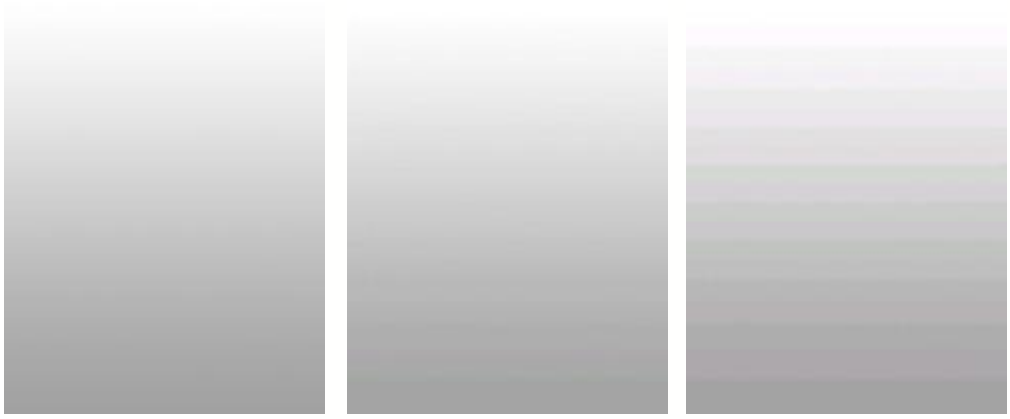
`DITHERING: n1`

The EA eDIPTFT57-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`

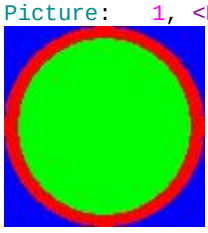
Parts of a picture can be switched to transparent for a nice overlie of a picture on the background. GIF, TGA, PNG and G16 files may already include any transparency information. If not (or a BMP / JPEG format is used) one color can be defined to become transparent. The color will be picked out from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

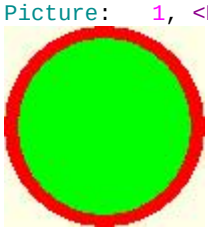
0 = no transparency (default)

Examples:

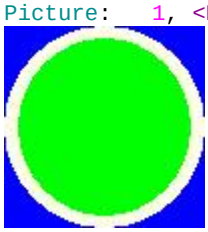
```
MakeTransparent: 0
Picture:   1, <Kreis.BMP>
```



```
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, #CF 41).

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

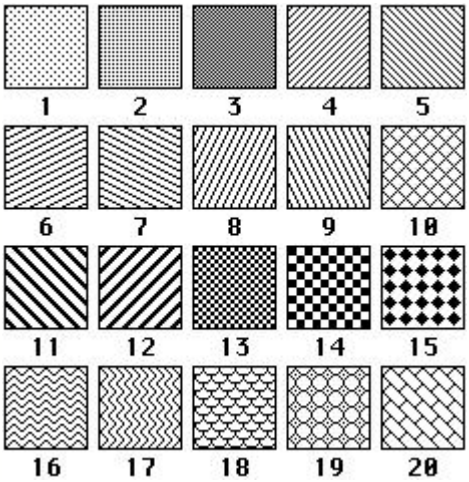
4.5.10 Pattern

```
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>
```

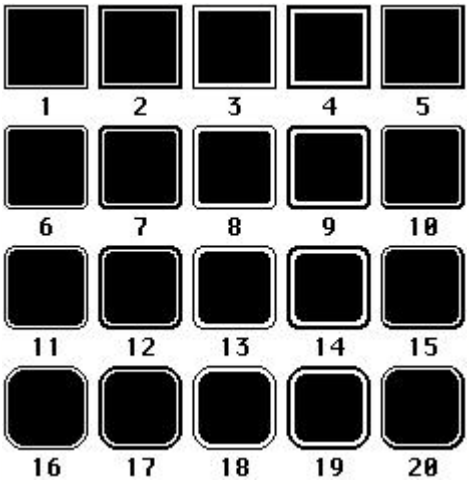
4.5.11 Border

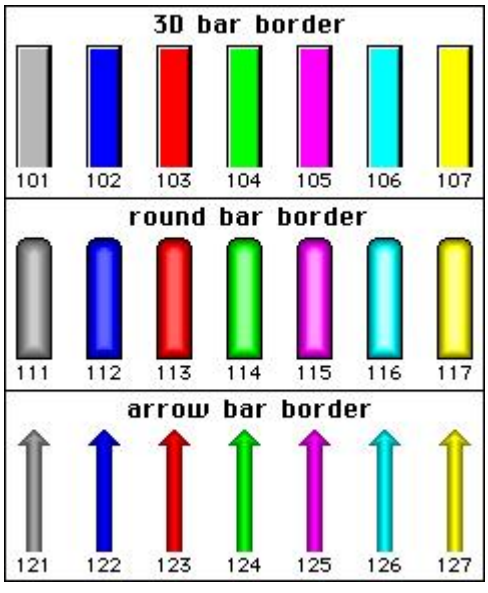
```
BORDER: nr <file>
BORDER: nr[page] <file>
BORDER: nr <file1>,<file2>
BORDER: nr[page] <file1>,<file2>
```

A border is used for rectangle 34, bargraph 39 and touch key/switch 44. 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 118)

---

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 44). 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 91)

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 85)

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).
<file> can be an animated GIF or G16.
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 86)

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 130)

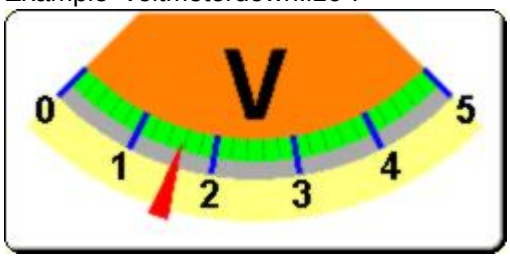
`INSTRUMENT: nr <file??>`

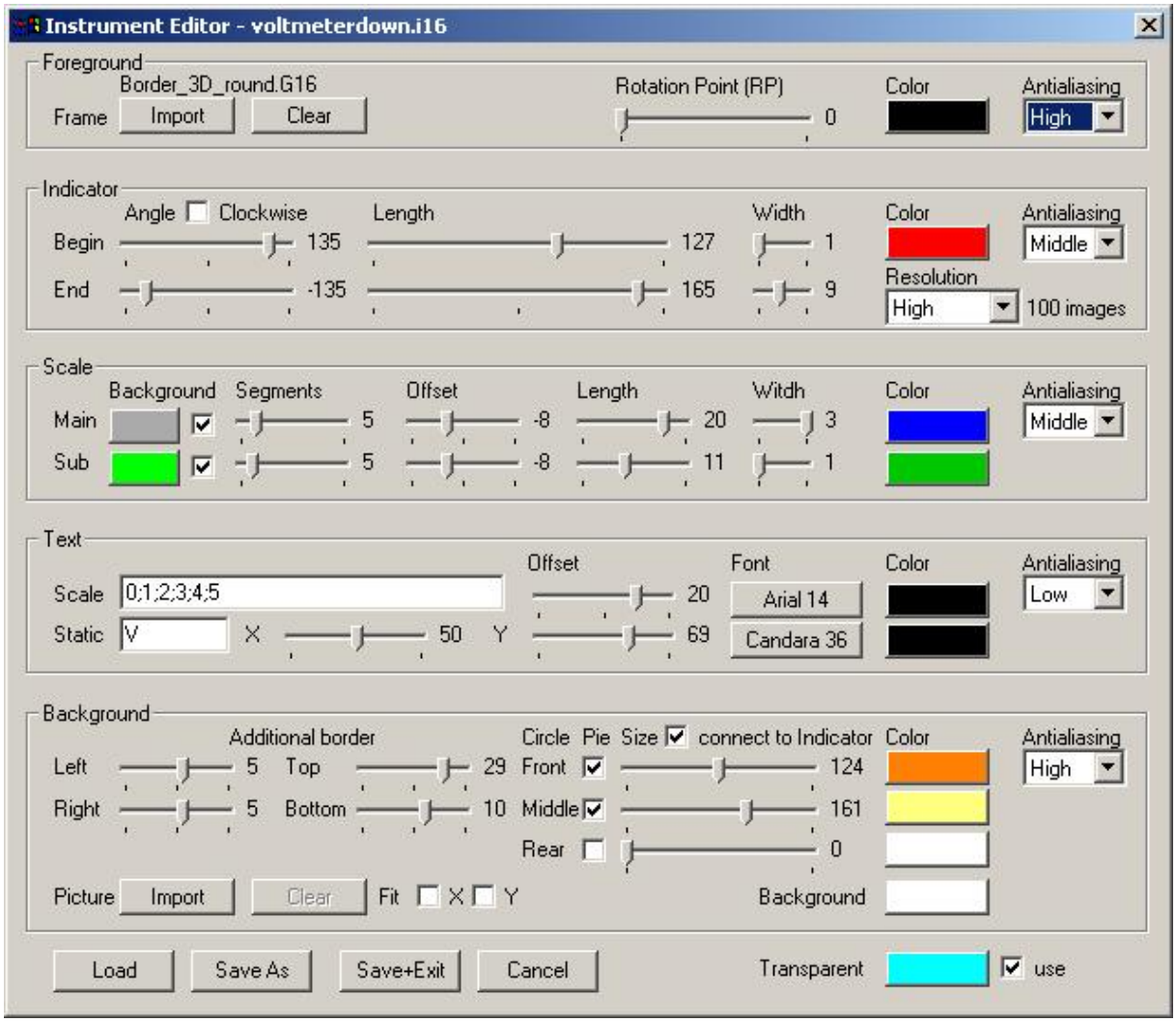
The compiler can create an instrument from single bitmaps, the questionmarks are interpreted as placeholder for numbered images (BMP, GIF, JPG, TGA, PNG or G16).
The image with number 00 is defined as background image only, without the indicator.
Image 1..n are the animations with indicator.
If the backgound 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
```

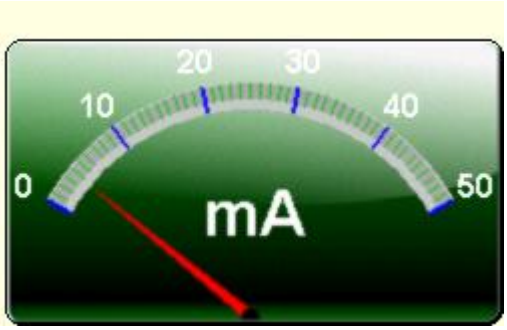
InstrumentParam is used to set the range and position of the indicator for imported animations.
bA, eA = Indicator begin- and end-Angle -180°..+180° (0°=to TOP, Y-axis, see "InstrumentEdit.exe")
rpPos = Rotationpoint position 1..9 (see MAKETRANSPARENT 19) (default=5=middle center of the image)
rpX, rpY = 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 42.
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 42.
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 89)

4.6.5 MenuMacro

`MENUMACRO: nr`
`MENUMACRO: nr[page]`

Defines a menu macro with number nr (0..255). This macro will be executed automatically after choosing an menu entry or by command #MM nr 42.
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.

4.6.6 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 42.
BitMacro 1..8 are good for falling edge at input 1..8.
BitMacro 9..16 are good for rising edge at input 1..8.
It is possible to change the assignment between BitMacro and intput with command #YD n1,n2,n3 54.
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 143)

4.6.7 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 42.
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 - EXPERT 145)

4.6.8 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 42.
Matrix Macro 1..64: start when keypressed.
Matrix Macro 0: start after release of key.
It is possible to change the assignment between keynumber and matrixmacro with command #YX n1,n2,n3 54.
The relating pins for matrix keyboard need to be defined with the command #YM n1,n2,n3 54.
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.

(see How-to-use example Matrix Keyboard - EXPERT 107)

4.6.9 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 42) or by command #MC nr 42.
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 150)

4.6.10 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 42.
It is possible to change the assignment between analogurmacrofunction 0..19 and analogmacronumber with command #VM n1,n2 55.
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 141)

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 eDIPTFT57-A commands

5.1 Terminal

Terminal definition:

Set terminal color	#FT fg,bg	initialisize and clear the terminal with new colors fg=0..32 foreground color; bg=0..32 background color (see default colors 69)
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) C=1..80; 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 eDIPTFT57-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 shown in the terminal

Special ASCII-characters:

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

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 69)
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 69)
Invert display	#DI	Invert display content

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
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	#GS x1,y1	Set the last end point at coordinates x1,y1 for commands #GW, #GX and #GY
Draw X-graph	#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	#GY step,x1,...	Draw lines with fix Y-steps=0..127 (add 128 for neg.steps) and variable count of X-values 1..255

(see How-to-use example Line recorder - EXPERT 120)

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 69)
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 69)
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 69)
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 118)

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 69)
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°

(see How-to-use example Transparent - BEGINNER 87)

Text output:

Output string left justified	#ZL x,y,"text..."	A string (text...) is output left justified to x,y several lines are separated by the character '|' ($7C, pipe) character '\' ($5C, backslash) cancels the special function of '|' and '\'
Output string centered	#ZC x,y,"text..."	A string (text...) is output centered to x,y several lines are separated by the character '|' ($7C, pipe) character '\' ($5C, backslash) cancels the special function of '|' and '\'
Output string right justified	#ZR x,y,"text..."	A string (text...) is output right justified to x,y several lines are separated by the character '|' ($7C, pipe) character '\' ($5C, backslash) cancels the special function of '|' and '\'
Output string in an area	#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 79)
(see How-to-use example Cyrillic font - BEGINNER 83)

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 69)
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 70 format This command is only for serial interface, do not use this command in a macro !

(see How-to-use example BMP file - BEGINNER 85)

Hardcopy:

RLE compression	#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 (see default colors 69) fg=foreground color; bg=background color;
Animation zoom factor	#WZ n1,n2	n1 = X-zoom factor (1x to 8x) n2 = Y-zoom factor (1x to 8x)
Animation angle	#WW n1	output angle of the animation image n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Mirror animation	#WX n1	n1=0: normal display n1=1: the animation image is mirrored horizontally
Transparency for color animation	#WT n1	n1=0: no transparency; show animation with all colors rectangular n1=1: color of the first dot at top left side will be defined as transparent (like a mask) n1=2: if defined - use transparent color from animation-file (.GIF.G16) n1=3: replace transparent color from animation-file with actually background color

Animation bitmap:

Load single image	#WI x1,y1,nr,n2	Load from animation image nr=0..255 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 86)

5.7 Bargraph

Bargraph settings:

Set color for bargraph	#FB fg,bg,fc	Set colors 1..32 for bargraph (see default colors 69) 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..200 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 126)

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 69)
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 for user value 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..20 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 130)

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 69)
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 Clipboard

Clipboard:

Save display contents	#CB	The entire contents of the display are copied to the clipboard as an image area
Save area	#CS x1,y1,x2,y2	The image area from x1,y1 to x2,y2 is copied to the clipboard
Restore area	#CR	The image area on the clipboard is copied back to the display
Copy area	#CK x1,y1	The image area on the clipboard is copied to x1,y1 in the display

(see How-to-use example Transparent - BEGINNER 87)
(see How-to-use example Clipboard - EXPERT 116)
(see How-to-use example Free draw area - BEGINNER 112)

Load bitmap to clipboard:

Load image to clipboard	#CL x1,y1, <file.G16>	Load the image data *.G16 70 and draw it into clipboard to x1,y1 This command is only for serial interface, do not use this command in a macro !
Internal image to clipboard	#CI x1,y1,n1	Draw internal image from DATA-FLASH with number n1=0..255 into clipboard to x1,y1 (see compiler option PICTURE 24 :)
Animation sub image to clipboard	#CF x1,y1,n1,n2	Draw animation sub image from DATA-FLASH with number n1=0..255, n2=framenumber into clipboard to x1,y1 (see compiler option ANIMATION 25 :)
Clipboard image autoupdate	#CU n1	n1=0: autoupdate off n1=1: after loading the image into clipboard (#CL #CI #CF) the content is copied to screen

Extended clipboard layer usage:

Copy to Clipboard layer	#CP x1,y1,x2,y2, x3,y3	The area from x1,y1 to x2,y2 on the screen is copied into the clipboard layer to x3,y3
Copy from Clipboard layer	#CC x1,y1,x2,y2, x3,y3	The area from x1,y1 to x2,y2 on the clipboard layer is copied into the screen to x3,y3
Copy Area on Clipboard layer	#CA x1,y1,x2,y2, x3,y3	The area from x1,y1 to x2,y2 on the clipboard layer is copied to new upper left corner x3,y3
Clipboard Transparent Color	#CT n1	n1=0: no transparency; 1..32: transparent color for clipboard; (only for command #CC, see default colors 69)
Animation through clipborard	#CW n1,n2	n1=1..4: animation nr n2=0: draw direct to screen n2=1: first draw to clipboard layer and then copy to screen

5.10 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 menu macros	#MM nr	Call the menu macro with the number nr (max. 7 levels) (see compiler option MENUMACRO 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','M','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','M','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 136)

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 148)

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 150)

5.11 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 69)

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 69)

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 94)

Label font presets:

Font color	#FA nf,sf	Color 1..32 for touch labeling (see default colors 69) 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 89)
(see How-to-use example Glass button - EXPERT 91)

Define touch menu:

Define touch key with menu function	#AM x1,y1,x2,y2, downCode,upCode,mnuCode, "text..."

The area from x1,y1 to x2,y2 is defined as a menu key.
'down code':(1-255) return/touchmacro when pressed.
'up Code':(1-255) return/touchmacro when menu canceled
'mnu Code':(1-255) return/menumacro+(item number - 1) after selection of a menu item
(down/up code = 0: activation/cancellation is not reported.)
'text':= string with the key text and the menu items.
The first character determines the direction in which the menu opens (R=right, L=left, O=up, U=down).
The second character determines the alignment of the touch key text (C=centered, L=left-, R=right justified).
The menu items are separated by the character '|' ($7C,dec:124) (e.g. "UCkey|item1|item2|item3".
The key text is written with the current touch font and the menu items are written with the current menu font.
The background of the menu is saved automatically.

(see How-to-use example Menue - BEGINNER 96)

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 69)
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	#A+ no	The instrument with no=1..4 is defined for input by touch panel.

(see How-to-use example Free draw area - BEGINNER 112)
(see How-to-use example Bargraph by touch - BEGINNER 126)
(see How-to-use example Instrument by touch - BEGINNER 130)

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

Global settings:

Touch query on/off	#AA n1	Touch query is n1=0: deactivated n1=1: activated
Touch key inversion	#AI n1	Automatic inversion when touch key touched n1=0: OFF n1=1: ON
Touch key Buzzer response	#AS n1	n1=0: beeper output OFF n1=1: beeper output goes ON when a touch key is touched
Touch key Vibration response	#A* n1	n1=0: vibration off n1=1..3: vibration intensity 1=low, 2=mid, 3=high

Touch key
Sound response
	#A- n1,n2,n3	Set Touch sound (plays when touch is touched)
		n1=1..10: touchtype (0:set all types to the same sound)
		n1=1: KEY (#AT #AU)
		n1=2: SWITCH (#AJ #AK)
		n1=3: BAR (#AB)
		n1=4: INSTRUMENT (#A+)
		n1=5: DRAW (#AD)
		n1=6: FREE (#AH)
		n1=7: MAKEMENU (#AM open menue)
		n1=8: MENUBOX (change menue entry)
		n1=9: KEYBOARD (#KS)
		n1=10: EDITBOX (#ET)
		n2=1..254: sound number (0=no sound; 255=no change)
		n3=0..7: sound volume (255=no change)

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

(see How-to-use example Sound - BEGINNER 153)
(to select sounds see Sound selection - EXPERT 152)

5.12 Keyboard

Keyboard frame:

Keyboard frame colors	#FK n1,n2,n3,s1,s2,s3	Set the frame colors 1..32 (0=transparent) for keys (code >= 32) n=normal; s=selected; 1=frame outside; 2=frame inside; 3=filling (see default colors 69)
Special key frame colors	#FS n1,n2,n3,s1,s2,s3	Set frame colors 1..32 (0=transparent) for special keys (code<32) n=normal; s=selected; 1=frame outside; 2=frame inside; 3=filling (see default colors 69)
Keyboard frame	#KE n1,n2	n1=1..255: frame number for normal keys n2=1..255: frame number for special keys n1/n2=0: no frame; (see compiler option BORDER 23:)

Keyboard label:

Keyboard label color	#FF n1,s2	Set the textcolor 1..32 for keys (code >= 32) n1=normal textcolor; s2=textcolor for selection (see default colors 69)
Special key label color	#FY n1,s2	Set the colors 1..32 for special keys (code < 32) n1=normal textcolor; s2=textcolor for selection (see default colors 69)
Keyboard label font	#KF n1,n2,s1,s2	Set keyboard label font n=font for normal keys s=font for special keys 1=0..255 font for single letters 2=0..255 font for strings (see command #KL) (see compiler option FONT 11: or WINFONT 12:)
Key label string	#KL code, "text"	code=keycode; "text"=alternative text for key code=0: Clear alternative text for all keys
Label offset for selection	#KO n1,n2	n1=X-offset; n2=Y-offset n1/n2=0..7 (add +8 for negative direction)

Keyboard layout:

Keyboard position	#KP x1,y1,x2,y2,gap	x1,y1,x2,y2 = rectangle enclosing the keyboard gap = space between the keys
Define keyboard	#KB n1,"codestr"	define keyboard with number n1=1..4 "codestr" = keycodes several lines are separated by '|' use backslash '\' for special keycodes: \1: show keyboard no. 1 \2: show keyboard no. 2 \3: show keyboard no. 3 \4: show keyboard no. 4 \5: SHIFT use keyboard no. 2 for one key autofallback to no. 1 \6: CAPSLOCK toggle between keyboard no. 1 and no. 2 \8: (code 8) BACKSPACE \A: (code 10) CANCEL \C: (code 12) CLEAR \D: (code 13) SEND \N: placeholder for a key witch is not used and drawn

Use keyboard:

Keyboard background	#KH n1,n2	n1=0: background is not saved n1=1: save background into clipboard layer n2=1..32: background color or 0=transparent
Key matrixmacro	#KM n1, n2	n1=keycode; n1=0: Set Martixmakro for all codes n2=Martixmakro number if key is pressed
Show keyboard	#KS no, n2	Show the keyboard with no=1..4 Activate editbox 50 n2=1..15 and send pressed keys to it n2=0: put pressed keys into sendbuffer
Delete keyboard	#KD n1	remove actually shown keyboard from screen n1=0: area remains visible n1=1: delete area or restore from clipboard (see command #KH)

(see How-to-use example Keyboard - BEGINNER 101)

5.13 Editbox

Editbox presets:

Editbox frame colors	#FQ n1,n2,n3	Set the frame colors 1..32 (0=transparent) for Edibox: n1=frame outside; n2=frame inside; n3=filling; (see default colors 69)
Editfield text colors	#FH dfg,dbg, afg,abg	Set the textcolors 1..32 for Editfield (see default colors 69) dfg=deactivated foreground; dbg=deactivated background afg=activated foreground; abg=activated background
Offset for editfield	#EO left,top,right,bottom	Set offset from editbox frame to editfield
Editbox font	#EF n1	Set font number n1=0..255 for editfield (see compiler option FONT 11: or WINFONT 12:)
Editbox frame	#EE n1	n1=1..255: frame number for editbox; n1=0: no frame (see compiler option BORDER 23:)
Editbox background	#EH n1	n1=0: background is not saved n1=1: save editbox background into clipboard layer

Define Editbox:

Define and show Editbox	#EL #EC #ER
	no,n2, x1,y1,x2,y2,
	"default text"
		Define Editbox #EL=left; #EC=center; #ER=right justified
		no=1..15: editbox number
		n2=max.text length; n2=0: no limit
		The area from x1,y1 to x2,y2 is drawn with actual frame (#EE)
		"default text" = userdefined default text

Delete Editbox	#ED no, n2	Delete editbox with no=1..15; no=0:delete all editboxes n2=0: area remains visible; n2=1: restore or delete area (#EH)
Set Editbox by touch	#ET no, n2,n3	The editbox no=1..15 can be activated by touch panel. n2=activate touchmacro; n3=deactivate touchmacro n2/n3=0: touchmacro is not used
Set password character	#EP no, code	Set editbox no=1..15 input mode code=0: all characters are shown code=1..255: use wildcard for each character

Use Editbox:

Activate Editbox	#EA no	Activates editbox with no=1..15 for editing no=0: deactivates actually editbox
Get Editbox string	#EG no	Put string from editbox no=1..15 into sendbuffer no=0: put all editbox strings into sendbuffer
Send character to Editbox	#EB code	send one character to active editbox special codes: 8: BACKSPACE delete last character 10: CANCEL editing, set to defaulttext 12: CLEAR editfield 13: SEND put string into sendbuffer, copy as defaulttext
Send string to Editbox	#ES "text"	Send a string to active editbox

(see How-to-use example Keyboard - BEGINNER 101)

5.14 Menu

Settings for menu box / touch menu:

Set menu colors	#FN fg,bg,fc	Colors 1..32; fg=for characters; bg=for background; fc= for frame (see default colors 69)
Set menu font	#NF n1	Set font with the number n1 (0 to 31) for menu display
Menu font zoom factor	#NZ n1,n2	n1=X-zoom factor (1x to 8x); n2=Y-zoom factor (1x to 8x)
Additional character width/height	#NY n1,n2	n1=0..15: additional width left/right n2=0..15: additional height top/bottom
Menu angle	#NW n1	Menu display angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Touch menu automation	#NT n1	n1=1: Touch menu opens automatically n1=0:Touch menu does not open automatically; instead, the request 'ESC T 0' to open is sent to the host computer, which can then open the touch menu with 'ESC N T 2'

Menu box commands (control with keys not by touch):

Define and display menu	#ND x,y,no,"text.."	A menu is drawn at corner x,y with the current menu font. no=currently inverted entry (e.g.: 1 = first entry). "text.."=string with menu items, the different items are separated by the character '|' ($7C,dec:124) (e.g. "item1|item2|item3"). The background of the menu is saved automatically. If a menu is already defined, it is automatically canceled+deleted
Next item	#NN	The next item is inverted or remains at the end
Previous item	#NP	The previous item is inverted or remains at the beginning
End of menu/send	#NS	The menu is removed and replaced with the original background. The current item is sent as a number (1 to n) (0=no menu displayed)
End of menu/macro	#NM n1	The menu is removed and replaced with the original background. Menu macro n1 is called for item 1, menu macro nr+1 for item 2, and so on...
End of menu/cancel	#NA	The menu is removed and replaced with the original background

(see How-to-use example Menue - BEGINNER 96)

5.15 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	#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 126)

5.16 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	#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 143)
(see How-to-use example Port Macro - EXPERT 145)

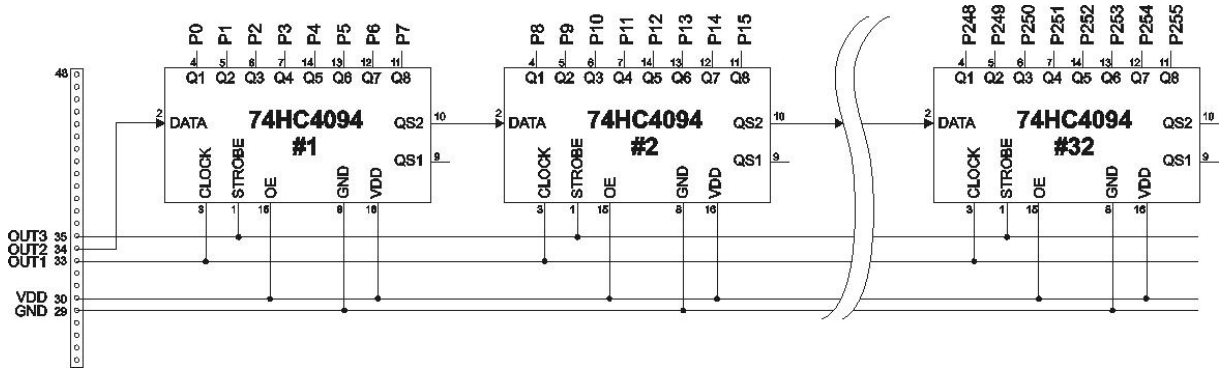
External matrix keyboard:

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	#YX n1,n2	Assign keynumber n1=1..65 with matrixtmacro n2=0..255 After release the key n1=0 run matrixtmacro n2=0..255

(see How-to-use example Matrix Keyboard - EXPERT 107)

Additional Outputs:

Write extended ports (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



5.17 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	#VM n1,n2	Assign analogmacrofunction n1=0..19 with analogmacronumber n2=0..255
Bargraph for AIN1/AIN2	#VB ch,no	Assigns bargraph bn=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 in [mV/20]
Instrument for AIN1/AIN2	#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/instrumment	#VR ch	Redraw all bargraphs and instruments defined for channel ch=1..2

(see How-to-use example Analogue Macro - Beginner 141)
(see How-to-use example Instrument by analogue input - BEGINNER 131)

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 69)
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

5.18 Sound

Sound settings:

Sound settings	#YV n1,n2,n3,n4	n1=0..7: set the volume for a soundqueue and tonscale notes n2=0..127: wait n2 * 10ms between the sounds n3=1..15: repeat counter; n3=0: loop n4=0..2: mode: 0: ignore touchsounds 1: soundlist will be interrupted by touchsounds 2: soundlist stops by touchsounds (note: for n1..n4 you can use the value 255 for NOCHANGE)

Play Sounds:

Play soundqueue	#YQ n1, ...	play soundqueue n1, ...= sound nr 1..80 (max. 90 sounds) #YQ 0 will stop the actually sound
Play tonescale notes	#YT "notestr"	allowed characters in "notestr" (max. 90 notes): 1..8 = time divisor for following notes # = halftone rise for next note CDEFGAH = notes from one-line octave cdefgah = notes from two-line octave P = pause (Duration as long as one note) e.g. #YT "4CDEF2GG4AAAA1G4AAAA1G4FFFF2EE4DDDD1C"
Play touchsound	#Y- n1,n2	play sound number n1=1..80; n2=0..7:volume a playing soundqueue will be interruppted see #YV mode (to assign a sound to a touch-function use #A- 44 command)

(see How-to-use example Sound - BEGINNER 153)
(to select sounds see Sound selection - EXPERT 152)

Buzzer:

Tone buzzer on/off	#YS n1	The tone output (pin 16) becomes n1=0:OFF; n1=1:ON; n1=2 to 255:ON for n1/10s

5.19 Other commands

Use string table:

String table code	#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 138)

Send functions:

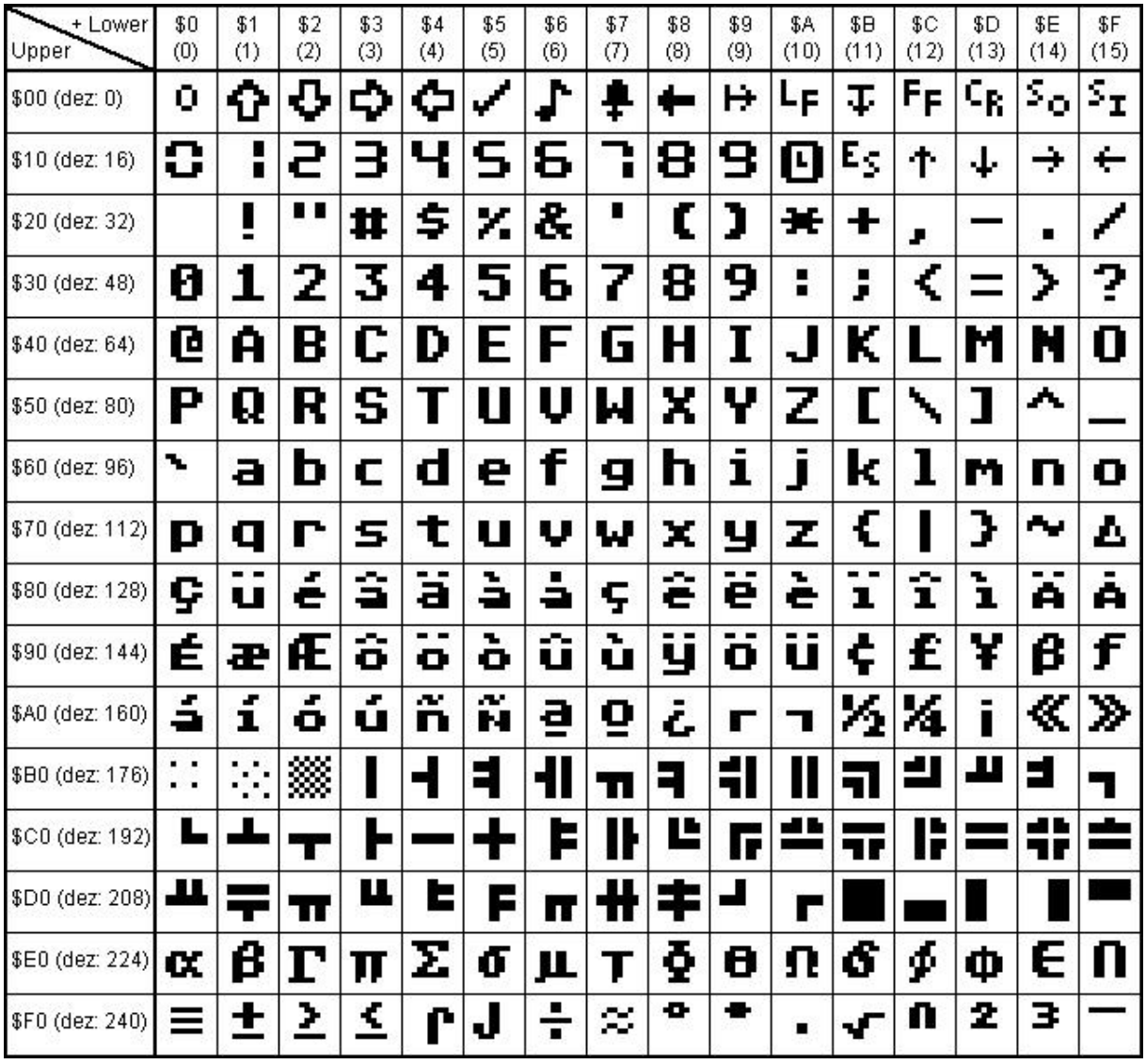
Send bytes	#SB data...	bytes are sent to the sendbuffer data... can be numbers or strings e.g #SB "Test",10,13
Send version	#SV	The version is sent as a string to sendbuffer e.g. "EA eDIPTFT57-A V1.0 Rev.A TP+"
Send projectname	#SJ	The macro-projectname is sent as a string to the sendbuffer e.g. "init / 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 69)
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 77)

6 Default Fonts

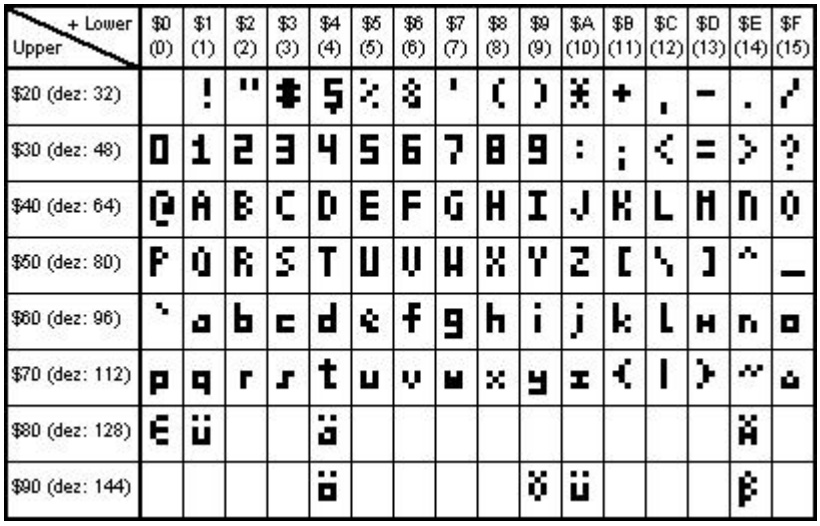
6.1 Terminal 8x8



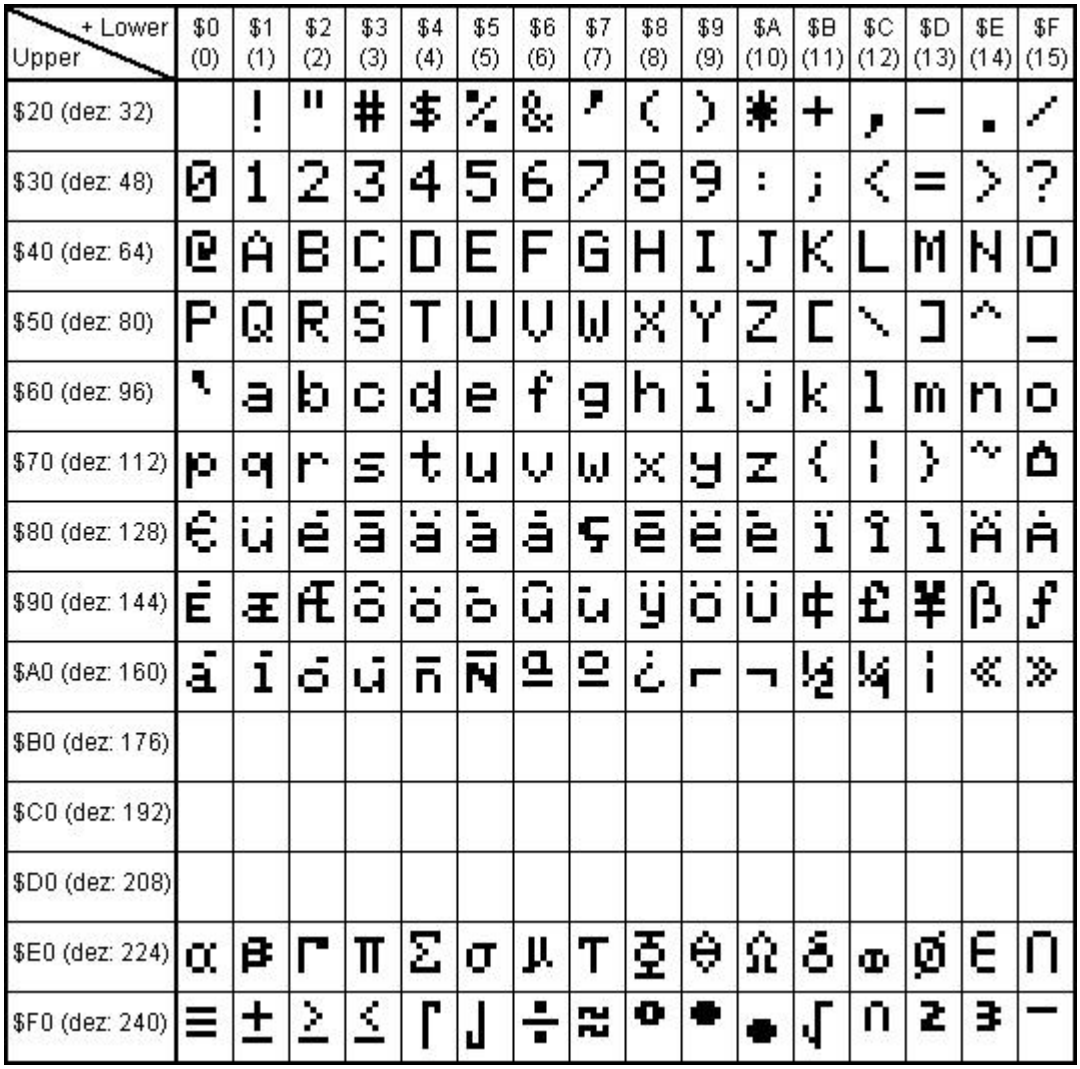
6.2 Terminal 8x16

+ 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

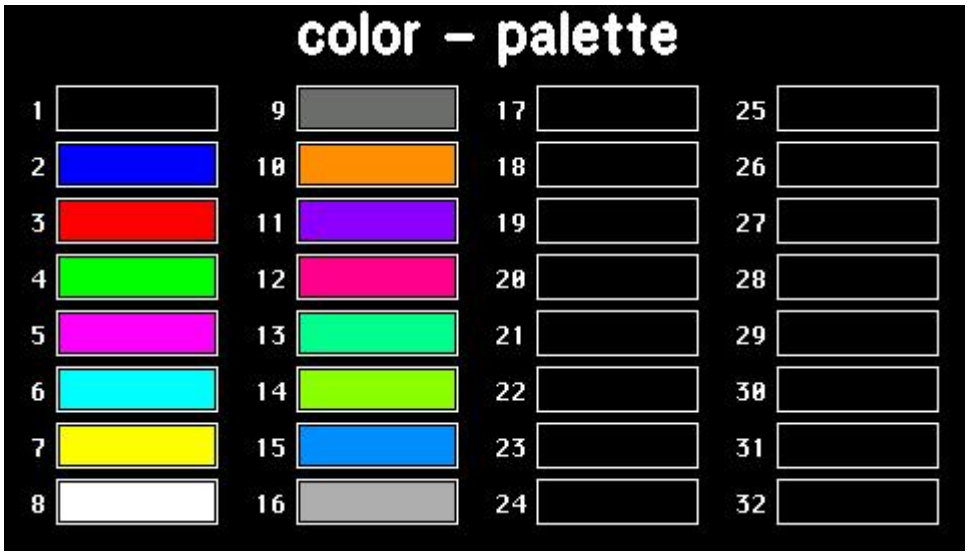
+ Lower / Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)												+		-	.	
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:					

7 Default Colors

EA eDIPTFT57-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 58). 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

BACKGROUND  =200      ; = Clipboard content
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	description	
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

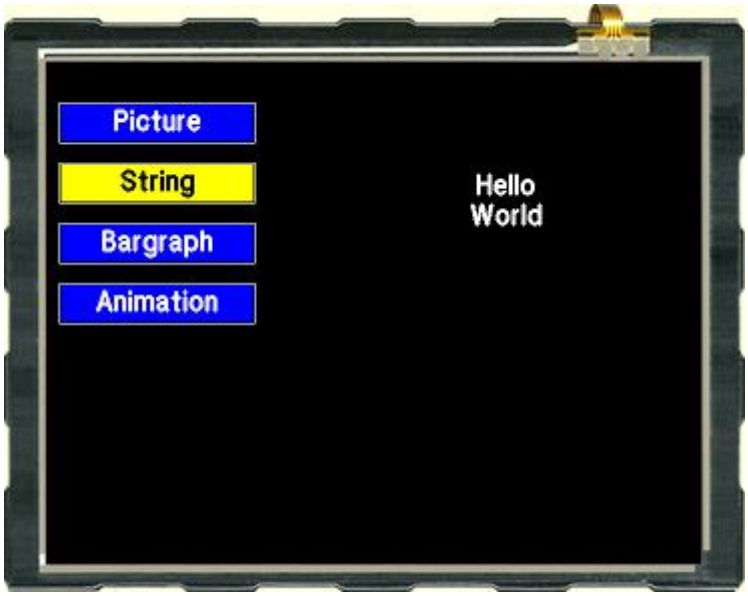
Next, the repeating byte/repeating pixel or the uncompressed bytes/pixels follow.
For pictures with 1 -, 4 and 8-bits per pixel, the data is treated byte by byte.
For High Color 16-Bit picture, the data are treated pixel-wise.

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\eDIPTFT57-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\eDIPTFT57-A\My first project

File:
my_first_project.kmc

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

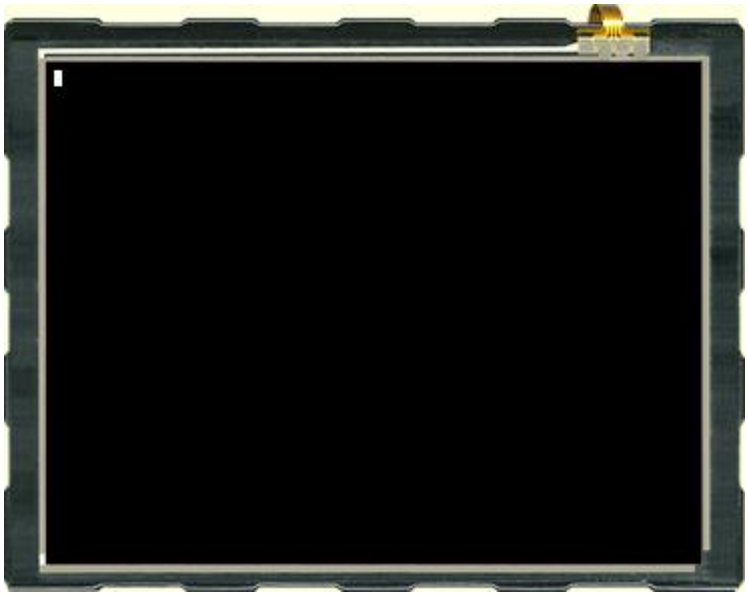
Open file in KitEditor

```
eDIPTFT57-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 SWISS30B         ; set touch label font, the font is defined in include file
"default_font.kmi"
       #AT 5,30,200,70,1,0, "Picture"      ; place 3 touchbuttons at x1,y1 to x2,y2,
Touchmacro 1 is called
       #AT 5,90,200,130,2,0, "String"      ; touchmacro 2 is called
       #AT 5,150,200,190,3,0, "Bargraph"   ; touchmacro 3 is called
       #AT 5,210,200,250,4,0,"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 220,0,639,410    ; delete area on the right (to delete pixels of other
```

```
touchmacros)
       #UI 400,70, 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 220,0,639,410     ; 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 450,100, "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 220,0,639,410     ; 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 1,400,300,440,10,0,100,1 ; 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 220,0,639,410     ; delete area on the right (to delete pixels of other
touchmacros)
       #WD 1, 400,70, 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\eDIPTFT57-A\Init\

File:
Init.kmc

Commands:

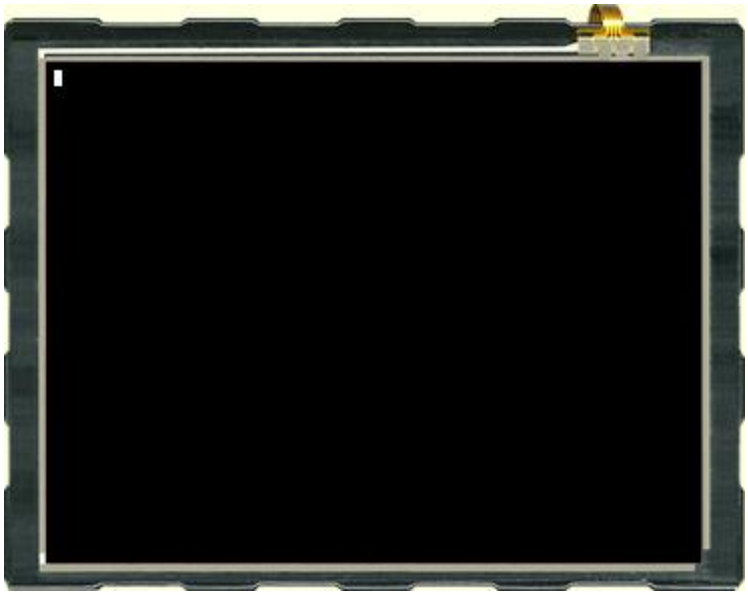
Open file in KitEditor

```
eDIPTFT57-A "init / delivery state" ; 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>
include <..\default_font.kmi>
include <..\default_pattern.kmi>
include <..\default_border.kmi>

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

MnPowerOn = 0

PowerOnMakro:         ; wird nach dem Einschalten ausgeführt
       #MN MnPowerOn

ResetMakro:           ; wird nach einem externen Reset ausgeführt
       #MN MnPowerOn

WatchdogMakro: ; wird nach einem Fehlerfall/Timeout >500ms ausgeführt
       #MN MnPowerOn

BrownOutMakro: ; wird nach einem Spannungseinbruch <3V ausgeführt
       #MN MnPowerOn

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

Makro: MnPowerOn

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\eDIPTFT57-A\Init\

File:
INIT_with_RS485_address.KMC

Commands:

Open file in KitEditor

```
eDIPTFT57-A "init / delivery state" ; 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>

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

MnPowerOn = 0

PowerOnMakro:         ; wird nach dem Einschalten ausgeführt
       #MN MnPowerOn

ResetMakro:           ; wird nach einem externen Reset ausgeführt
       #MN MnPowerOn
```

```
WatchdogMakro: ; wird nach einem Fehlerfall/Timeout >500ms ausgeführt
       #MN MnPowerOn

BrownOutMakro: ; wird nach einem Spannungseinbruch <3V ausgeführt
       #MN MnPowerOn

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

Makro: MnPowerOn
       #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 81). In addition you will find help to include WinFonts under BEGINNER - Cyrillic Font.kmc 83.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Font\

File:
BEGINNER - Place text.kmc

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

Open file in KitEditor

```
eDIPTFT57-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 202,20,1                       ; place picture no. 1
       #GD 200,120,438,120                ; 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 320,230,"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 639,400,"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 540,240,"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 79). In addition you will find help to include WinFonts under BEGINNER - Cyrillic Font.kmc 83.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Font\

File:
EXPERT - Place text.kmc

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

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                           ; Terminal off
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 0,150,!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 XPIXEL/2,250,!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 XMAX,450,!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 650,YPIXEL/2,!TEXT4!            ; place text "vertical centered" in the center
```

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 79, EXPERT - Place text.kmc 81).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Font\

File:
BEGINNER - Cyrillic Font.kmc

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

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                             ; Terminal off
       #UI 202,20,1                    ; place Picture no. 1
       #GD 200,120,438,120             ; 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 320,200, {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
```

```
for EditBox to see the characters correctly
Fontname and "Select Font for EditBox"
; You have to select Font no.9
; by clicking right on the
```

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 86.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Picture\

File:
BEGINNER – show a bmp file.kmc

Commands:
#UI

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; 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 85.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Picture\

File:
BEGINNER - Show an animated giff file.kmc

Commands:
#WD

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; draw a Line

;---- Place Information logo ----
       #WD 1,270,200,5,2,1          ; place Animation no. 5 (process no. 1)
```

9.8 Transparent - BEGINNER

Show the use of clipboard by means of an animation an font with color 200. If you want to use the clipboard as display memory, refer to EXPERT - Clipboard.kmc [116].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Clipboard\

File:
BEGINNER - Transparent.kmc

Commands:
#CB, #FZ

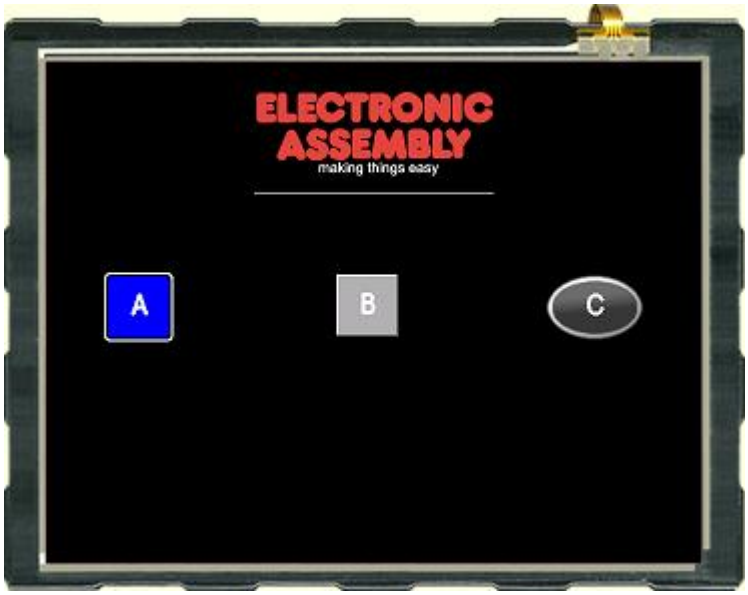
Open file in KitEditor

```
eDIPTFT57-A "Transparent Animation"
...
...
...
;-------------------------------------------------------------------------------
; aditional fonts
Path: <..\..\..\Fonts>
ARIALZIF73 = 10
Font: ARIALZIF73 , <Arial_Color_0-9.G16>

;-------------------------------------------------------------------------------
; include picture
Path <..\..\..\bitmaps\color\>
Picture: 1 <ea logo making things easy black.bmp> ; double click to open
Picture: 2 <.\Snake.G16>
;-------------------------------------------------------------------------------
; include animations
PATH: <..\..\..\BITMAPS\color\Animation>
Animation: 1  <ClownColorTransparent.G16>


Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 0,0, 2                   ; place background
       #UT 1                        ; make logo transparent
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line

       #ZF ARIALZIF73       ; set font for counting characters
       #CB                          ; save display content,neccessary for fonts
(backgroundcolor)
       #FZ NOCHANGE,BACKGROUND      ; color for fonts; it's to restore background from
clipboard
       #MD 1,2, 1,10, 2             ; define process macro
```

```
        #WD 1, 120, 200, 1, CYCLIC,DEFAULTTIME     ; define animation


x = 370
y = 210

ProzessMacro: 1
        #ZL x,y, "1"
ProzessMacro: 2
        #ZL x,y, "2"
ProzessMacro: 3
        #ZL x,y, "3"
ProzessMacro: 4
        #ZL x,y, "4"
ProzessMacro: 5
        #ZL x,y, "5"
ProzessMacro: 6
        #ZL x,y, "6"
ProzessMacro: 7
        #ZL x,y, "7"
ProzessMacro: 8
        #ZL x,y, "8"
ProzessMacro: 9
        #ZL x,y, "9"
ProzessMacro: 0
        #ZL x,y, "0"
```

9.9 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 126), Radiogroups (see BEGINNER - radiogroup.kmc 94).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Touch\

File:
BEGINNER - 3 simple touch areas.kmc

Commands:
#AU, #AT

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; 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 50,200,118,268,65,0 "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 280,200,66,0 "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 490,200,67,68 "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 320,350 "Macro #67"      ; place text

;---- Release the right touch ----
Touch: 68
       #RL 0,300,639,400            ; delete area (text)
```

9.10 Glass button - EXPERT

Example of using pictures as buttons. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 126), and another Example with touch buttons (see BEGINNER - 3 simple buttons.kmc 89).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Touch\

File:
EXPERT - Glass buttons.kmc

Commands:
#AU

Open file in KitEditor

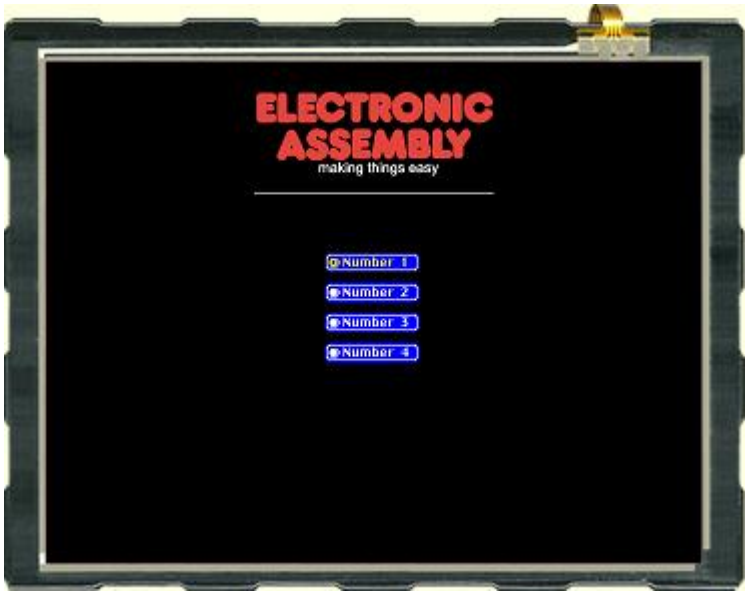
```
eDIPTFT57-A "Glass buttons"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\glas-button\>
LOGO = 1
BACKGROUND = 2
Picture: LOGO, <ea_transparent.g16> ; double click to open
Picture: BACKGROUND <background.g16>

;--- 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 = 9
DAUPHIN = 10
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 ----
       #TA                          ; Terminal off
       #FD BLACK,BLACK              ; set display colors
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)

       #UI (XPIXEL-PICTURE_W(BACKGROUND))/2,
2*yoff+PICTURE_H(LOGO),BACKGROUND                 ; place background

;---- Place buttons ----
xs = (XPIXEL-PICTURE_W(BACKGROUND))/2 + 250
pitch = 15
xw = BUTTON_W(GLASBUT)
ys = 2*yoff+PICTURE_H(LOGO)+200

       #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 (XPIXEL-PICTURE_W(BACKGROUND))/2+10,ys,!EARCD!              ; place earphone,
cd...


x=xs
st_x=x
st_y=2*yoff+PICTURE_H(LOGO)+280
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
```

9.11 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 126), Buttons (see BEGINNER - 3 simple buttons.kmc 89).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Touch\

File:
BEGINNER - touch as radio button.kmc

Commands:
#AR, #AJ

Open file in KitEditor

```
eDIPTFT57-A "Touch as Radio Button"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
Button 1 <button\RadioButton95x20_0.bmp>,<button\RadioButton95x20_1.bmp>
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; draw a Line


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

OriginX = 270                       ; using a constant makes it more easy to move the
group later
OriginY = 180
PitchY = 30

       #AF 5                                                ; set font no. 4 for
Touch area
       #FA WHITE,YELLOW                                     ; 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
        #AR 0                                             ; end of group 1
```

9.12 Menue - BEGINNER

Show a menue and call MenuMacros. There is a EXPERT example available under EXPERT - menue.kmc 98.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Menue\

File:
BEGINNER - menue.kmc

Commands:
#AM

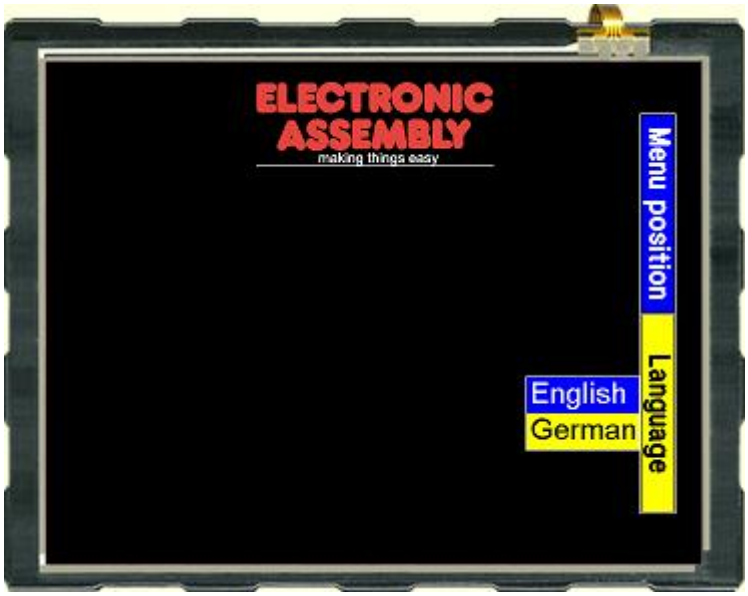
Open file in KitEditor

```
eDIPTFT57-A "Menu"
...
...
...
;-------------------------------------------------------------------------------
Picture 1 <..\..\..\bitmaps\color\ea logo making things easy black.bmp> ; double click to
open Bitmap Editor
;-------------------------------------------------------------------------------

Macro: MnAutoStart
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,90,438,90            ; draw a Line

;--- Place Menue ---
       #NF SWISS30B          ; set Menue font to Geneva10
                             ; same as #NF 4
       #NY 2,2         ; adding 2 additional dots between two menue items
       #NW 0                 ; menu angle
       #NT 1                 ; touch menu opens automatically

       #AM 50,190,200,220,0,0,10, "UCMenu1|Item1|Item2|Item3"          ; place Menu1,
opening down (U), text centered (C)
                                                                       ; MenuMacro
10+Itemnumber is called
                                                                       ; Item1 has no. 0,
Item2 has 1...
       #AM 203,190,353,220,0,0,20, "UCMenu2|Item1|Item2"               ; place Menu2,
opening down (U), text centered (C)
                                                                       ; MenuMacro
20+Itemnumber is called
       #AM 356,190,506,220,0,0,30, "UCMenu3|Item1|Item2|Item3|Item4"   ; place Menu3,
opening down (U), text centered (C)
                                                                       ;MenueMacro
30+Itemnumber is called
```

```
        #ZF SWISS30B
        #FZ WHITE, BLACK

;-------------------------------------------------------------------------------
MenueMacro: 10;Menu 1 Item 1
        #ZL 60,300,"Selected: Menue 1, Item 1" ; place text

MenueMacro: 11;Menu 1 Item 2
        #ZL 60,300,"Selected: Menue 1, Item 2"

MenueMacro: 12;Menu 1 Item 3
        #ZL 60,300,"Selected: Menue 1, Item 3"

;-------------------------------------------------------------------------------
MenueMacro: 20;Menu 2 Item 1
        #ZL 60,300,"Selected: Menue 2, Item 1" ; place text

MenueMacro: 21;Menu 2 Item 2
        #ZL 60,300,"Selected: Menue 2, Item 2"

;-------------------------------------------------------------------------------
MenueMacro: 30;Menu 3 Item 1
        #ZL 60,300,"Selected: Menue 3, Item 1" ; place text

MenueMacro: 31;Menu 3 Item 2
        #ZL 60,300,"Selected: Menue 3, Item 2"

MenueMacro: 32;Menu 3 Item 3
        #ZL 60,300,"Selected: Menue 3, Item 3"

MenueMacro: 33;Menu 3 Item 3
        #ZL 60,300, "Selected: Menue 3, Item 4"
```

9.13 Menue - EXPERT

Show a menue and call MenuMacros. The Menue can be rotated in all directions. There is a BEGINNER example available under BEGINNER - menue.kmc 96.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Menue\

File:
EXPERT - menue.kmc

Commands:
#AM, #NW

Open file in KitEditor

```
eDIPTFT57-A   "Menu"
...
...
...
;---------------------------------------------------------------------------------
LOGO = 1 ; define constants makes it easier to use
Picture: LOGO, <..\..\..\bitmaps\color\ea logo making things easy black.bmp>   ; double
click to open Bitmap Editor

;---------------------------------------------------------------------------------
;define normal macros
;in the NM_MENUE_.. macros 4 postions of the menu are drawn
NM_MENUE_TOP    = 1
NM_MENUE_LEFT   = NM_MENUE_TOP+1
NM_MENUE_RIGHT  = NM_MENUE_LEFT+1
NM_MENUE_BOTTOM = NM_MENUE_RIGHT+1

;define menu macros
;menu macros that are called
MM_POS  = 10
MM_LANG = MM_POS+10

;---------------------------------------------------------------------------------
;define additional fonts
ARIAL28=9

ExportOverview: 0 ; deactivate overview bitmap
Winfont: ARIAL28, "Arial",-32,0, 32-127, 28

;---------------------------------------------------------------------------------
;string constants for menu alignment
!MEN_DOWN!  = "UC"
!MEN_LEFT!  = "LC"
!MEN_RIGHT! = "RC"
!MEN_UP!    = "OC"

;string table
```

```
ENG = 0 ; constants for language (macropages)
GER = 1

MENU_POS    = 1 ; constants for string table
MENU_LANG   = 2
MENBOX_POS  = 3
MENBOX_LANG = 4

; definition of later used strings, devided in header of menu
String: MENU_POS[ENG]  "Menu position"
String: MENU_POS[GER]  "Menü-Position"
String: MENU_LANG[ENG] "Language"
String: MENU_LANG[GER] "Sprache"
; and menu items
String: MENBOX_POS[ENG] "|Top|Bottom|Left|Right"
String: MENBOX_POS[GER] "|Oben|Unten|Links|Rechts"
String: MENBOX_LANG[ENG]"|English|German"
String: MENBOX_LANG[GER]"|Englisch|Deutsch"

; definition of fonts
ft_men = SWISS30B
ft_menboy = ARIAL28

; set parameter for calculation of STRING_W() function
zx=1
zy=zx
w=0
h=w
sp=0
st=$01
st_par = STRING_P(zx,zy,w,h,sp,st)

;-------------------------------------------------------------------------------
;start of macros
Macro: MnAutoStart
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
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)

       #ST st ;define string table code and activate it

       #FN WHITE,BLUE,WHITE ; set colors of menu
       #AF ft_men     ; set font for touchlabels (menu header)
       #NF ft_menboy ; set font for menu (menu box = items)
       #NY 0, 4       ; add additional height in  menubox
       #MN NM_MENUE_TOP ; call normal macro (show menu)



; det the width of the menu strings, every language
w_st1 = STRING_W(MENU_POS, st_par, ft_men, GER)
w_st2 = STRING_W(MENU_POS, st_par, ft_men, ENG)
w_st3 = STRING_W(MENU_LANG, st_par, ft_men, GER)
w_st4 = STRING_W(MENU_LANG, st_par, ft_men, ENG)
; get the maximum stribg witdh and add additional space (width of menue)
xwidth = max(w_st1, w_st2, w_st3, w_st4)+20
xs=(XPIXEL-2*xwidth)/2 ; calculate the start of menu, that it is displayed in the middle
of the screen
ys=180 ; ystart of the menu
yheight=35 ; height of menu-button
offset=20 ;offset for the left, right and bottom side of menu

;place menu to the top
Macro: NM_MENUE_TOP
angle = 0
x=xs
xw=xwidth
y=ys
yh=yheight
       #AL 0,1  ;delete old menu
       #AW angle ; set angle of button
       #NW angle ; set angle of menubox
       ;place first menu, calling menumacro MM_pos. The menu box is openung down
```

```
(=!MEN_DOWN!)
       ;using stringtable (st). Use Stringtable MENU_Pos and MENBOX_POS as menuentries
       #AM x,y,x+xw,y+yh, 0,0, MM_POS, !MEN_DOWN!, st, MENU_POS,MENBOX_POS
x+=xw
       #AM x,y,x+xw,y+yh, 0,0, MM_LANG,!MEN_DOWN!,st, MENU_LANG,MENBOX_LANG

;place menu to the bottom
Macro: NM_MENUE_BOTTOM
angle = 0
x=xs
xw=xwidth
y=YPIXEL-offset-yheight
yh=yheight
       #AL 0,1
       #AW angle
       #NW angle
       #AM x,y,x+xw,y+yh, 0,0, MM_POS, !MEN_UP!, st, MENU_POS,MENBOX_POS ; same as above,
but opening menu box to the top
x+=xw
       #AM x,y,x+xw,y+yh, 0,0, MM_LANG,!MEN_UP!,st, MENU_LANG,MENBOX_LANG


;place menu to the left
Macro: NM_MENUE_LEFT
angle = 1
x=offset
xw=yheight
y=(YPIXEL-2*xwidth)/2
yh=xwidth
y+=yh
       #AL 0,1
       #AW angle
       #NW angle ; spin menu box also with 90 degrees, so it has to open to the bottom =
!MEN_DOWN!
       #AM x,y,x+xw,y+yh, 0,0, MM_POS, !MEN_DOWN!, st, MENU_POS,MENBOX_POS
y-=yh
       #AM x,y,x+xw,y+yh, 0,0, MM_LANG,!MEN_DOWN!,st, MENU_LANG,MENBOX_LANG

;place menu to the right
Macro: NM_MENUE_RIGHT
angle = 3
x=XPIXEL-offset-yheight
xw=yheight
y=(YPIXEL-2*xwidth)/2
yh=xwidth
       #AL 0,1
       #AW angle
       #NW 0 ; don't spin menu box, so it stays correctly readable, but open to the
middle of the screen = !MEN_LEFT!
       #AM x,y,x+xw,y+yh, 0,0, MM_POS, !MEN_LEFT!, st, MENU_POS,MENBOX_POS
y+=yh
       #AM x,y,x+xw,y+yh, 0,0, MM_LANG,!MEN_LEFT!,st, MENU_LANG,MENBOX_LANG


;-------------------------------------------------------------------------------
; menu macros
;change language
MenuMacro: MM_LANG
       #MK ENG ; set macro page
       #MN NM_MENUE_TOP ; call standard macro to show new language
MenuMacro: MM_LANG+1
       #MK GER
       #MN NM_MENUE_TOP

;change position of menue
MenuMacro: MM_POS
       #MN NM_MENUE_TOP
MenuMacro: MM_POS+1
       #MN NM_MENUE_BOTTOM
MenuMacro: MM_POS+2
       #MN NM_MENUE_LEFT
MenuMacro: MM_POS+3
       #MN NM_MENUE_RIGHT
```

9.14 Keyboard - BEGINNER

Show a whole qwertz-keyboard and store inputs in an editbox. This is only a simpel example, there are more features available (see EXPERT - Keyboard.kmc 103, EXPERT - Matrix_Keyboard.kmc 107).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Keyboard\

File:
BEGINNER - Keyboard.kmc

Commands:
#KB, #KE, #KS, #EL, #EA

Open file in KitEditor

```
eDIPTFT57-A "Simple keyboard"
...
...
...
;-------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; draw a Line

;---- Place Editbox ----
       #EF SWISS30B                       ; select font
       #FQ BLUE, WHITE, BLUE        ; set frame colors
       #FH WHITE, BLUE, BLUE, YELLOW      ; set text colors
       #EE 20                             ; define frame
       #EO 10,10,10,10                    ; define offset between frame and text
       #EL 1, 0, 30,140,610,210,""  ; define left justified editbox
       #EA 1                              ; activate editbox

;---- Place Keyboard ----
       #FK BLUE,WHITE,BLUE,YELLOW,BLUE,YELLOW     ; keyboard frame colors
       #FS RED,WHITE,RED,YELLOW,BLUE,YELLOW       ; special keys for keyboard frame colors
       #FF WHITE, BLUE                            ; keyboard text color
       #FY BLACK, BLUE                            ; special key text color
       #KF SWISS30B, CHICAGO14, SWISS30B, CHICAGO14      ; select fonts for keyboard
       #KE 6,6                                    ; frame
       #KL 6, "CAPS"                              ; define labels for special keys
       #KL 8, "<--"
       #KL 12, "CLR"
       #KL 13, "Return"
```

```
#KL 5, "Shift"
#KL ' ',"Space"

; define two keyboards, for capital and small letters '\' means special command
; '|' means next line of keyboard
#KB 1, "^1234567890ß\8\C|\6qwertzuiopü+\D\D|\5asdfghjklöä#\N|<yxcvbnm,.-|    "
#KB 2, "°!""'§$%&/()=?\8\C|\6QWERTZUIOPÜ*\D\D|\5ASDFGHJKLÖÄ\N'"'|<YXCVBNM;:_|    "


#KP 0,240,639,479,3    ; set keyboard position
#KS 1,1         ; show keyboard
```

9.15 Keyboard - EXPERT

Show different keyboards, layout refers to different languages (German, English, Russian). The inputs are stored in different editboxes. This is only one example, there are more available (see BEGINNER - Keyboard.kmc 101, EXPERT - Matrix_Keyboard.kmc 107).

.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Keyboard\

File:
EXPERT - Keyboard.kmc

Commands:
#KB, #KE, #KS, #EL, #EA

Open file in KitEditor

```
eDIPTFT57-A "Keyboard"
...
...
...
;--------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture 1 <ea logo making things easy black.bmp>  ; double click to open

;--------------------------------------------------------------------------------
;set constants for different languages
GER=0
ENG=1
RUS=2

;define constants
border = 7 ;select border for all buttons and frames

;define size of keyboard
kb_x     = 0
kb_xw    = XPIXEL-2*kb_x
kb_y     = 250
kb_yh    = YPIXEL-kb_y
kb_pitch = 2

;define size of all editboxes (the area will be devided by 4 later
eb_x  = kb_x
eb_xw = kb_xw
eb_y  = 120
eb_yh = kb_y-eb_y-25
eb_pitch = 15
eb_offset = 8

;define position and size of menu to select language
```

```
men_xw = 150
men_x  = XMAX-men_xw-kb_x
men_y  = 30
men_yh = 35

;------------------------------------------------------------------------------
;include additonal fonts
f_KEY_EUR   = 9
f_KEY_RUS   = 10
f_SKEY      = 11
f_EDBOX     = 12
f_EDBOX_PW  = 13
f_EDBOX_NUM = 14

WinFont: f_KEY_EUR, "Arial",0,1, 32-255, 28
WinFont: f_KEY_RUS, "Arial",204,1, 32-255, 28
Winfont: f_SKEY , "Wingdings",-33,0, $F0C3 + $F0EF + $F0F1-$F0F2 + $F0F4 + $F0FB, 28
WinFont: f_EDBOX[GER], "Arial",0,1, 32-255, 24
WinFont: f_EDBOX[RUS], "Arial",204,1, 32-255, 24
WinFont: f_EDBOX_PW, "WP IconicSymbolsA",-117,1, 117 + $F029, 28
WinFont: f_EDBOX_NUM, "Arial",0,1, 48-57, 24

; very important to deactivate the codetable, which is necessary for DOS-fonts
CodeTable: 0

st=$01 ; define code for stringtable (#ST), after that code appears, internal strings are
used


; define constants for keyboards
KB_SM =1
KB_CAP=2
KB_ALT=3

;define strings for internal stringtable
;Strings for German keyboard layout, which is used as default
;define keyboard, for capital and small letters '\' means special command
;'|' means next line of keyboard
String: KB_SM [GER], "\N^1234567890ß\8\C\N", "|\6qwertzuiopü+\D\D", "|\5asdfghjklöä#\N",
"|<yxcvbnm,.-", "|\3    \3"
String: KB_CAP[GER], '\N°!"§$%&/()=?\8\C\N', "|\6QWERTZUIOPÜ*\D\D", "|\5ASDFGHJKLÖÄ'\N",
"|<YXCVBNM;:_", "|\3    \3"
String: KB_ALT[GER], "\N^1²³4567890ß\8\C\N", "|\2@w€rtzuiopü~\D\D", "|\1asdfghjklöä#\N",
"|\|yxcvbnm,.-","|\3    \3"
;Strings for English keyboard layout
String: KB_SM [ENG], "'1234567890-=\8\C\N", "|\6qwertyuiop[]\D\D", "|\5asdfghjkl;'\\\N",
"|zxcvbnm,./", "|      "
String: KB_CAP[ENG], '~!@#$%^&*()_+\8\C\N', "|\6QWERTYUIOP{}\D\D", '|\5ASDFGHJKL:"\|\N',
"|ZXCVBNM<>?", "|      "
;Strings for Russian keyboard layout
String: KB_SM [RUS], {B8313233343536373839302D2B}"\8\C\N", "|\6"{E9F6F3EAE5EDE3F8F9FDF5FA}
"\D\D", "|\5"{F4FBE2E0EFF0EEEBE4E6FD5C}"\N", "|"{3CFFF7F1ECE8F2FCE1FE2F}, "|\3    \3"
String: KB_CAP[RUS], {A82122233B3A2C2E2A28295F2B}"\8\C\N", "|\6"{C9D6D3CAC5CDC3D8D9DDD5DA}
"\D\D", "|\5"{D4DBC2C0CFD0CECBC4C6DD}"\|\N", "|"{3EDFD7D1CCC8D2DCC1DE3F}, "|      "


;------------------------------------------------------------------------------
;define macors
KEYBOARD_Def   = 3 ; define keyboard and show

EDIT_BOX       = 10 ; define and show editboxes

;define menu macros
LANGUAGE = 10

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

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA
yoff = 0
YoffLOGO=yoff+PICTURE_H(LOGO)+5
       #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
```

```
;---- Place menu to select language ----
x=men_x
xw=men_xw
y=men_y
yh=Men_yh
       #AF SWISS30B                                ; font for main menu item (= as
touchbutton)
       #FE BLUE,WHITE,BLUE,YELLOW,BLUE,YELLOW      ; define color for main menu item (= as
touchbutton)
       #AE border, 0                               ; define border for main menu item (= as
touuchbutton)
       #FA WHITE,BLUE                        ; set color for main menu item
       #NF SWISS30B                                ; set font for menu box
       #FN WHITE,BLUE,WHITE                        ; set color for menu box
       #AM x,y,x+xw,y+yh,0,0, LANGUAGE, "UCLanguage|German|English|Russian" ; place menu


       #ST st                ; activate internal string table
       #MN EDIT_BOX          ; call default edit boxes
       #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14 ; select fonts for keyboard , have to
be done before keyboard_def, because font is depending on language
       #MN KEYBOARD_DEF      ; call default keyboard


;-------------------- Place Editbox --------------------
Macro: EDIT_BOX

xs=eb_x
ys=eb_y
xw=eb_xw/2-eb_pitch/2
yh=eb_yh/2-eb_pitch/2
pitch=eb_pitch
offset=eb_offset
x=xs
y=ys

       #EF f_EDBOX                           ; select font
       #FQ BLUE, WHITE, BLUE         ; set frame colors
       #FH WHITE, BLUE, BLUE, YELLOW       ; set text colors
       #EE border                            ; define frame
       #EO offset,offset,offset,offset     ; define offset between frame and text
       #EL 1, 0, x,y,x+xw,y+yh,""          ; define left justified editbox
       #EA 1                                 ; activate editbox (default)

x+=xw+pitch
       #ER 2, 15, x,y,x+xw,y+yh,"123ABC"   ; define left justified editbox, with maximum
length
x=xs
y+=yh+pitch
       #EF f_EDBOX_PW               ; select font
       #EP 3, {75}                           ; set wildcard char for editbox
       #EC 3, 8, x,y,x+xw,y+yh,""          ; centered editbox with maximum length and use
of wildcard chars

x+=xw+pitch
       #EF f_EDBOX_NUM                       ; select font (the font includes only numbers)
       #EC 4, 0, x,y,x+xw,y+yh,"123"       ; place centered editbox

       #ET 1, 0,0                            ; set every editbox selectable with touch
       #ET 2, 0,0
       #ET 3, 0,0
       #ET 4, 0,0


;----------------- Place Keyboard -----------------
Macro: KEYBOARD_Def

       #FK BLUE,WHITE,BLUE,YELLOW,BLUE,YELLOW     ; keyboard frame colors
       #FS RED,WHITE,RED,YELLOW,BLUE,YELLOW       ; special keys for keyboard frame colors
       #FF WHITE, BLUE                            ; keyboard text color
       #FY BLACK, BLUE                            ; special key text color
       #KE border,border                                  ; frame
       #KL 6, {24}                                ; define labels for special key
       #KL 8, {22}
       #KL 12,{26}
       #KL 13,{21}
       #KL 5, {23}
```

```
        #KL 3, "Alt"
        #KL 2, {24}
        #KL 1, {23}
        #KL ' ', "Space"
        #KL '|', "\|"

        #KB 1, st, KB_SM      ; define keyboard 1 (small letters)
        #KB 2, st, KB_CAP     ; define keyboard 2 (capital letters)
        #KB 3, st, KB_ALT     ; define keyboard 3 (alternarive letters, only german)

;----------------- Show Keyboard -----------------
x=kb_x
xw=kb_xw
y=kb_y
yh=kb_yh
pitch=kb_pitch
        ; draw a border box behind the keyboard
        #RE border, 0
        #FR GREY,GREY,GREY
        #RR x,y,x+xw,y+yh

        #KP x,y,x+xw,y+yh,pitch      ; set keyboard position
        #KS 1,1               ; show keyboard


;-------------------------------------------------------------------------------
;Menu Macros to select language/keyboard layout
MenuMacro: LANGUAGE+0 ; first menu item
        #ED 0,1 ; delete all editboxes, but remain them visisble
        #KD 0   ; delete keyboard, but remain visisble; because both are overwritten
        #MK GER ; select macropage
        #MN EDIT_BOX ; draw all edit boxes
        #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard ,
have to be done before keyboard_def, because font is depending on language
        #MN KEYBOARD_DEF; call keyboard

MenuMacro: LANGUAGE+1 ; second menu item
        #ED 0,1
        #KD 0
        #MK ENG
        #MN EDIT_BOX
        #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard
        #MN KEYBOARD_DEF

MenuMacro: LANGUAGE+2 ; third menu item
        #ED 0,1
        #KD 0
        #MK RUS
        #MN EDIT_BOX
        #KF f_KEY_RUS, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard
        #MN KEYBOARD_DEF
```

9.16 Matrix Keyboard - EXPERT

Show different keyboards, layout refers to different languages (German, English, Russian). The inputs are stored in different editboxes. In addition, the inputs can be done with the help of an external keypad. The inputs are stored in different editboxes. If you want examples without extarnal input, see BEGINNER - Keyboard.kmc 101, EXPERT - Keyboard.kmc 103).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Keyboard\

File:
EXPERT - Matrix_Keyboard.kmc

Commands:
#YM, #KB, #KE, #KS, #EL, #EA

Open file in KitEditor

```
eDIPTFT57-A "Matrix-Keyboard"
...
...
...
;-------------------------------------------------------------------------------
;Include Pictures
Path <..\..\..\bitmaps\color\>
LOGO = 1 ; using constants makes it easier
Picture 1 <ea logo making things easy black.bmp>  ; double click to open

;-------------------------------------------------------------------------------
;set constants for different languages
GER=0
ENG=1
RUS=2

;define constants
border = 7 ;select border for all buttons and frames

;define size of keyboard
kb_x     = 0
kb_xw    = XPIXEL-2*kb_x
kb_y     = 250
kb_yh    = YPIXEL-kb_y
kb_pitch = 2

;define size of all editboxes (the area will be devided by 4 later
eb_x  = kb_x
eb_xw = kb_xw
eb_y  = 120
eb_yh = kb_y-eb_y-25
eb_pitch = 15
eb_offset = 8

;define position and size of menu to select language
```

```
men_xw = 150
men_x  = XMAX-men_xw-kb_x
men_y  = 30
men_yh = 35

;------------------------------------------------------------------------------
;include additonal fonts
f_KEY_EUR   = 9
f_KEY_RUS   = 10
f_SKEY      = 11
f_EDBOX     = 12
f_EDBOX_PW  = 13
f_EDBOX_NUM = 14

WinFont: f_KEY_EUR, "Arial",0,1, 32-255, 28
WinFont: f_KEY_RUS, "Arial",204,1, 32-255, 28
Winfont: f_SKEY , "Wingdings",-33,0, $F0C3 + $F0EF + $F0F1-$F0F2 + $F0F4 + $F0FB, 28
WinFont: f_EDBOX[GER], "Arial",0,1, 32-255, 24
WinFont: f_EDBOX[RUS], "Arial",204,1, 32-255, 24
WinFont: f_EDBOX_PW, "WP IconicSymbolsA",-117,1, 117 + $F029, 28
WinFont: f_EDBOX_NUM, "Arial",0,1, 48-57, 24

; very important to deactivate the codetable, which is necessary for DOS-fonts
CodeTable: 0

st=$01 ; define code for stringtable (#ST), after that code appears, internal strings are
used


; define constants for keyboards
KB_SM =1
KB_CAP=2
KB_ALT=3

;define strings for internal stringtable
;Strings for German keyboard layout, which is used as default
;define keyboard, for capital and small letters '\' means special command
;'|' means next line of keyboard
String: KB_SM [GER], "\N^1234567890ß\8\C\N", "|\6qwertzuiopü+\D\D", "|\5asdfghjklöä#\N",
"|<yxcvbnm,.-", "|\3    \3"
String: KB_CAP[GER], '\N°!"§$%&/()=?\8\C\N', "|\6QWERTZUIOPÜ*\D\D", "|\5ASDFGHJKLÖÄ'\N",
"|<YXCVBNM;:_", "|\3    \3"
String: KB_ALT[GER], "\N^1²³4567890ß\8\C\N", "|\2@w€rtzuiopü~\D\D", "|\1asdfghjklöä#\N",
"|\|yxcvbnm,.-","|\3    \3"
;Strings for English keyboard layout
String: KB_SM [ENG], "'1234567890-=\8\C\N", "|\6qwertyuiop[]\D\D", "|\5asdfghjkl;'\\\N",
"|zxcvbnm,./", "|      "
String: KB_CAP[ENG], '~!@#$%^&*()_+\8\C\N', "|\6QWERTYUIOP{}\D\D", '|\5ASDFGHJKL:"\|\N',
"|ZXCVBNM<>?", "|      "
;Strings for Russian keyboard layout
String: KB_SM [RUS], {B8313233343536373839302D2B}"\8\C\N", "|\6"{E9F6F3EAE5EDE3F8F9FDF5FA}
"\D\D", "|\5"{F4FBE2E0EFF0EEEBE4E6FD5C}"\N", "|"{3CFFF7F1ECE8F2FCE1FE2F}, "|\3    \3"
String: KB_CAP[RUS], {A82122233B3A2C2E2A28295F2B}"\8\C\N", "|\6"{C9D6D3CAC5CDC3D8D9DDD5DA}
"\D\D", "|\5"{D4DBC2C0CFD0CECBC4C6DD}"\|\N", "|"{3EDFD7D1CCC8D2DCC1DE3F}, "|      "


;------------------------------------------------------------------------------
;define macors
KEYBOARD_Def   = 3 ; define keyboard and show

EDIT_BOX       = 10 ; define and show editboxes

;define menu macros
LANGUAGE = 10

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

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA
yoff = 0
YoffLOGO=yoff+PICTURE_H(LOGO)+5
       #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
```

```
;---- Place menu to select language ----
x=men_x
xw=men_xw
y=men_y
yh=Men_yh
       #AF SWISS30B                                  ; font for main menu item (= as
touchbutton)
       #FE BLUE,WHITE,BLUE,YELLOW,BLUE,YELLOW      ; define color for main menu item (= as
touchbutton)
       #AE border, 0                                 ; define border for main menu item (= as
touuchbutton)
       #FA WHITE,BLUE                        ; set color for main menu item
       #NF SWISS30B                                  ; set font for menu box
       #FN WHITE,BLUE,WHITE                          ; set color for menu box
       #AM x,y,x+xw,y+yh,0,0, LANGUAGE, "UCLanguage|German|English|Russian" ; place menu

       #YM 4,3,3      ; define Matrix kexboard (4x3 matrix) witch debouncing, please look
at the
                      ; buttom of this file for explanation


       #ST st                ; activate internal string table
       #MN EDIT_BOX          ; call default edit boxes
       #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14 ; select fonts for keyboard , have to
be done before keyboard_def, because font is depending on language
       #MN KEYBOARD_DEF      ; call default keyboard


;--------------------- Place Editbox --------------------
Macro: EDIT_BOX

xs=eb_x
ys=eb_y
xw=eb_xw/2-eb_pitch/2
yh=eb_yh/2-eb_pitch/2
pitch=eb_pitch
offset=eb_offset
x=xs
y=ys

       #EF f_EDBOX                           ; select font
       #FQ BLUE, WHITE, BLUE        ; set frame colors
       #FH WHITE, BLUE, BLUE, YELLOW         ; set text colors
       #EE border                            ; define frame
       #EO offset,offset,offset,offset       ; define offset between frame and text
       #EL 1, 0, x,y,x+xw,y+yh,""            ; define left justified editbox
       #EA 1                                 ; activate editbox (default)

x+=xw+pitch
       #ER 2, 15, x,y,x+xw,y+yh,"123ABC"   ; define left justified editbox, with maximum
length
x=xs
y+=yh+pitch
       #EF f_EDBOX_PW               ; select font
       #EP 3, {75}                           ; set wildcard char for editbox
       #EC 3, 8, x,y,x+xw,y+yh,""            ; centered editbox with maximum length and use
of wildcard chars

x+=xw+pitch
       #EF f_EDBOX_NUM                       ; select font (the font includes only numbers)
       #EC 4, 0, x,y,x+xw,y+yh,"123"         ; place centered editbox

       #ET 1, 0,0                            ; set every editbox selectable with touch
       #ET 2, 0,0
       #ET 3, 0,0
       #ET 4, 0,0


;----------------- Place Keyboard -----------------
Macro: KEYBOARD_Def

       #FK BLUE,WHITE,BLUE,YELLOW,BLUE,YELLOW      ; keyboard frame colors
       #FS RED,WHITE,RED,YELLOW,BLUE,YELLOW        ; special keys for keyboard frame colors
       #FF WHITE, BLUE                             ; keyboard text color
       #FY BLACK, BLUE                             ; special key text color
       #KE border,border                                   ; frame
       #KL 6, {24}                                 ; define labels for special key
```

```
        #KL 8, {22}
        #KL 12,{26}
        #KL 13,{21}
        #KL 5, {23}
        #KL 3, "Alt"
        #KL 2, {24}
        #KL 1, {23}
        #KL ' ', "Space"
        #KL '|', "\|"

        #KB 1, st, KB_SM      ; define keyboard 1 (small letters)
        #KB 2, st, KB_CAP     ; define keyboard 2 (capital letters)
        #KB 3, st, KB_ALT     ; define keyboard 3 (alternarive letters, only german)

;----------------- Show Keyboard -----------------
x=kb_x
xw=kb_xw
y=kb_y
yh=kb_yh
pitch=kb_pitch
        ; draw a border box behind the keyboard
        #RE border, 0
        #FR GREY,GREY,GREY
        #RR x,y,x+xw,y+yh

        #KP x,y,x+xw,y+yh,pitch      ; set keyboard position
        #KS 1,1               ; show keyboard


;-------------------------------------------------------------------------------
;Menu Macros to select language/keyboard layout
MenuMacro: LANGUAGE+0 ; first menu item
        #ED 0,1 ; delete all editboxes, but remain them visisble
        #KD 0   ; delete keyboard, but remain visisble; because both are overwritten
        #MK GER ; select macropage
        #MN EDIT_BOX ; draw all edit boxes
        #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard ,
have to be done before keyboard_def, because font is depending on language
        #MN KEYBOARD_DEF; call keyboard

MenuMacro: LANGUAGE+1 ; second menu item
        #ED 0,1
        #KD 0
        #MK ENG
        #MN EDIT_BOX
        #KF f_KEY_EUR, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard
        #MN KEYBOARD_DEF

MenuMacro: LANGUAGE+2 ; third menu item
        #ED 0,1
        #KD 0
        #MK RUS
        #MN EDIT_BOX
        #KF f_KEY_RUS, CHICAGO14, f_SKEY, CHICAGO14       ; select fonts for keyboard
        #MN KEYBOARD_DEF


;-------------------------------------------------------------------------------
;Matrix Macros
;external matrix keyboard is connected as follows (4x3 Matrix):
;       +---+---+---------MI1 (PIN 44)
;       1 | 2 | 3 |
;         |   |   |
;       +---+---+---------MI2 (PIN 45)
;       4 | 5 | 6 |
;         |   |   |
;       +---+---+---------MI3 (PIN 46)
;       7 | 8 | 9 |
;         |   |   |
;       +---+---+---------MI4 (PIN 47)
;       * | 0 | # |
;         |   |   +--|>|---MO3 (PIN 34)
;         |   +------|>|---MO2 (PIN 35)
;         +----------|>|---MO1 (Pin 36)
; --|>| = diode

MatrixMacro: 1
```

```
        #EB '1'
MatrixMacro: 5
        #EB '2'
MatrixMacro: 9
        #EB '3'
MatrixMacro: 2
        #EB '4'
MatrixMacro: 6
        #EB '5'
MatrixMacro: 10
        #EB '6'
MatrixMacro: 3
        #EB '7'
MatrixMacro: 7
        #EB '8'
MatrixMacro: 11
        #EB '9'
MatrixMacro: 4
        #EB '*'
MatrixMacro: 8
        #EB '0'
MatrixMacro: 12
        #EB '#'
```

9.17 Free draw area - BEGINNER

Define a free drawing area. In addition, the drawing area can be saved and recalled with the help of the clipboard. There is a an EXPERT example available, too. Please have a look at
EXPERT – free_draw_area.kmc

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Draw\

File:
BEGINNER – free_draw_area.kmc

Commands:
#AD, #CS

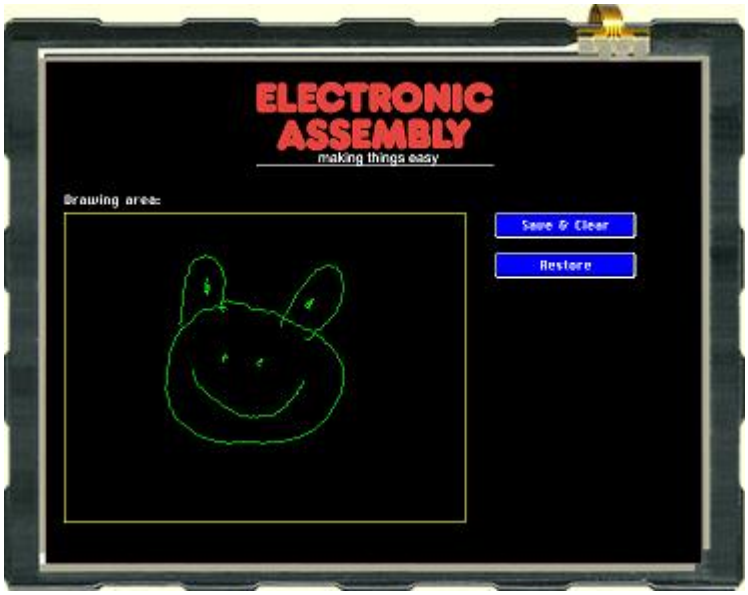
Open file in KitEditor

```
eDIPTFT57-A   "Free drawing area"
...
...
...
;-------------------------------------------------------------------------------
;include Pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line

;--- Place information ---
       #FZ WHITE, BLACK             ; set color for text
       #ZF CHICAGO14                ; set font no.5
       #ZL 10, 130,"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 450,160,580,190,1,0, "Save and Clear" ; place touchbutton 1
       #AT 450,210,580,240,2,0, "CRestore"  ; place touchbutton 2

;---- Place drawing area ----
       #FG YELLOW, BLACK     ; set color for drawing box
       #GR 10,160,400,400    ; place rectangle around drawing area
       #AD 11,161,399,399,1, GREEN  ; place drawing area, linewith 1 and green drawingline
```

```
;-------------------------------------------------------------------------------
TouchMacro: 1
        #CS 11,161,399,399    ; save drawing area
        #RL 11,161,399,399    ; clear drawing area


TouchMacro: 2
        #CR             ; restore clipboard content
```

9.18 Free draw area - EXPERT

Define a free drawing area. In addition, the drawing area can be saved and recalled with the help of the clipboard. There is a an BEGINNER example available, too. Please have a look at BEGINNER – free_draw_area.kmc [112]



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Draw\

File:
EXPERT – free_draw_area.kmc

Commands:
#AD, #CS

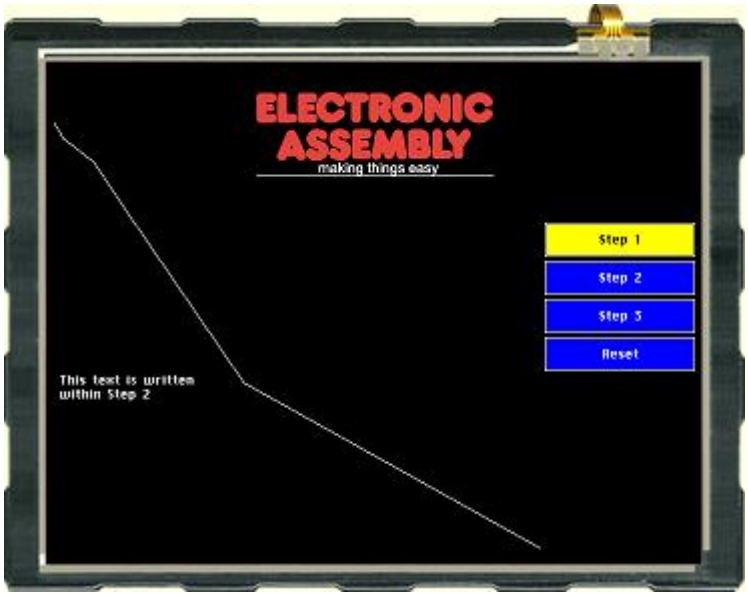
Open file in KitEditor

```
eDIPTFT57-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
RESTORE = CLEAR + 1
;-------------------------------------------------------------------------------
Macro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
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 = 400
dw_ys = 140
dw_yh = YMAX-dw_ys-30
       #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 = 140
ys = dw_ys
yh = 25
pitch = 15
       #AT xs,ys,xs+xw,ys+yh,CLEAR,0, "CSave & Clear"    ; place touchbutton 1
(C=centered)
ys+=yh+pitch
       #AT xs,ys,xs+xw,ys+yh,RESTORE,0, "CRestore"       ; place touchbutton 1
(C=centered)




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

TouchMacro: RESTORE
       #CR
```

9.19 Clipboard - EXPERT

Show the use of clipboard in 3 steps. If you want to place transparent text, refer to BEGINNER - Transparent.kmc 87.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Clipboard\

File:
EXPERT - Clipboard.kmc

Commands:
#CB, #CR

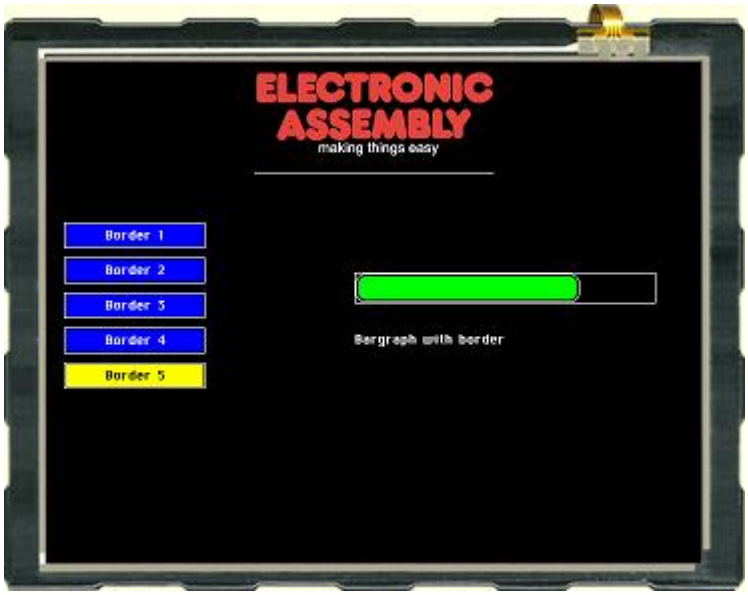
Open file in KitEditor

```
eDIPTFT57-A   "Using Clipboard"
...
...
...
;-------------------------------------------------------------------------------
;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
ST1 = 1
ST2 = ST1+1
ST3 = ST2+1

RES = 100

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

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
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)


       ; define touchbuttons
       #AF CHICAGO14
xs = 490
xb = 150
ys = 150
pitch = 5
yh = (YPIXEL-pitch*3-2*ys-30)/4
```

```
y=ys
       #AT xs,y,xs+xb,y+yh,ST1,0,"Step 1"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,ST2,0,"Step 2"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,ST3,0,"Step 3"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,RES,0,"CReset" ;C0center alligned, neccessayry, because R is
used as right justified


TouchMacro: ST1 ; Step one, draw something
       #GD 0,50,10,67
       #GW 40,90
       #GW 60,120
       #GW 190, 310
       #GW xs-5, YMAX-5
       ;drawing lines finished
       #CB ; save display contents



TouchMacro: ST2 ; step two, write a string on the screen
                ; the string is not seen by user, because #DC of step 1.
                ; Useful to show big pictures
       #ZL 5,300,"This text is written|within Step 2"


TouchMacro: ST3 ; step three, restore display content
       #CR ; restore clipboard content

TouchMacro: RES
       #AL 0,0          ; delete touchbuttons
       #DL              ; clear display
       #MN MnAutoStart ; call start up macro
```

9.20 Frame - BEGINNER

Show the different borders.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Frame\

File:
BEGINNER – frame.kmc

Commands:
#RT, #RR

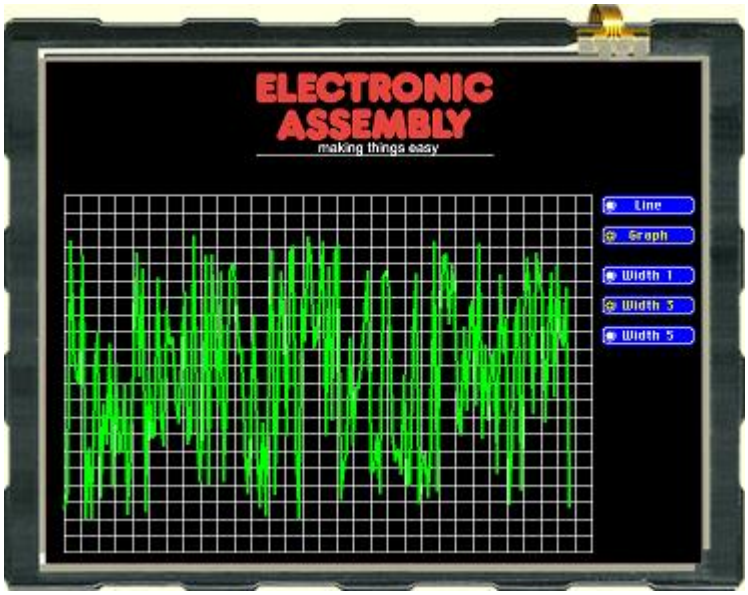
Open file in KitEditor

```
eDIPTFT57-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 202,0,1                          ; place picture no. 1
       #GD 200,100,438,100                  ; draw a Line

;--- Place 5 Buttons to select different Boarders ---
       ; use default paramters for Border, color and font of Touchbuttons
       #AT 10,150,150,175,1,0,"Border 1"
       #AT 10,185,150,210,2,0,"Border 2"
       #AT 10,220,150,245,3,0,"Border 3"
       #AT 10,255,150,280,4,0,"Border 4"
       #AT 10,290,150,315,5,0,"Border 5"

       #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
        #BD 1,0               ; delete bargraph (only touch area)
        #RL 300,150,600,400          ; 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 300,160,500,210,6,0,"Border-Button" ; define button

TouchMacro 5:  ;Called by Button Border5
        #AL 6,0                      ; delete Border Button (only touch area)
        #RL 300,150,600,400
        #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 1,300,200,600,230,0,100,5        ; 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 300,260,"Bargraph with border"; place info-text



Macro 1: ; Draw Rectangel with selected Border
        #AL 6,0                      ; delete Border Button (only touch area)
        #BD 1,0                      ; delete bargraph (only touch area)
        #RL 300,150,600,400                  ; delete area, to draw new frame
        #RR 300,150,600,400                  ; draw new frame

TouchMacro 6: ; called by  Boder-Button
        #ZL 300,250,"Border Button was|pressed"    ; place text, that Border-Button was
touched
                                                   ; '|' means new line
```

9.21 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\ eDIPTFT57-A\How to use\Draw\

File:
EXPERT - line recorder.kmc

Commands:
#GS, #GW, #GX

Open file in KitEditor

```
eDIPTFT57-A   "Free drawing area"
...
...
...
;-------------------------------------------------------------------------------
;include pictures and buttons
Path <..\..\..\bitmaps\color\>

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

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.

Makro: MnPowerOn
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                              ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO            ; place Picture no. 1
       ; draw a centered line directly beneath the picture:
       #GD (XPIXEL-PICTURE_W(LOGO))/2,PICTURE_H(LOGO),(XPIXEL-
PICTURE_W(LOGO))/2+PICTURE_W(LOGO),PICTURE_H(LOGO)

;constants for grid
pitch=10
xs=10
x=xs
xb=120
xe=XPIXEL-BUTTON_W(RadioBut)-pitch+1
ys=PICTURE_H(LOGO)+40
yh=(YMAX-ys)
arraypitch=xe/30
       #FG WHITE,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
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
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

        #CS xs,ys,xe,ys+yh ; save grid to cliboard (to recall it later fast and easy)


; 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
        #AR 2                 ; start second group
y+=BUTTON_H(RadioBut)+2*pitch
        #AJ x,y,3,0,"Width 1"
y+=BUTTON_H(RadioBut)+pitch
        #AJ x,y,4,0,"Width 3"
y+=BUTTON_H(RadioBut)+pitch
        #AJ x,y,5,0,"Width 5"
        #AR 0

;-------------------------------------------------------------------------------
TouchMakro: 1 ;draw graph as lines ( use continue stright lines)
        #CR ; recall grid
        #FG YELLOW,BLACK ; set color for lines
        #GS  xs + 0 ,  436 ; set start point
        #GW  xs + 3 ,  418 ; draw line
        #GW  xs + 6 ,  168
        #GW  xs + 9 ,  273
        #GW  xs + 12 ,  305
        #GW  xs + 15 ,  311
        #GW  xs + 18 ,  183
        #GW  xs + 21 ,  445
        #GW  xs + 24 ,  375
        #GW  xs + 27 ,  445
        #GW  xs + 30 ,  303
        #GW  xs + 33 ,  259
        #GW  xs + 36 ,  424
        #GW  xs + 39 ,  331
        #GW  xs + 42 ,  371
        #GW  xs + 45 ,  271
```

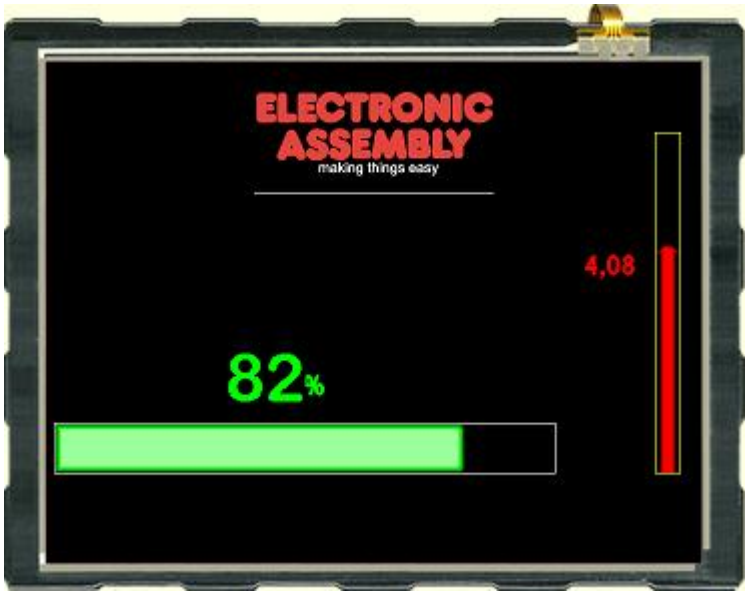
```
#GW  xs + 48 ,  414
#GW  xs + 51 ,  290
#GW  xs + 54 ,  379
#GW  xs + 57 ,  377
#GW  xs + 60 ,  312
#GW  xs + 63 ,  441
#GW  xs + 66 ,  373
#GW  xs + 69 ,  282
#GW  xs + 72 ,  180
#GW  xs + 75 ,  303
#GW  xs + 78 ,  196
#GW  xs + 81 ,  438
#GW  xs + 84 ,  298
#GW  xs + 87 ,  364
#GW  xs + 90 ,  362
#GW  xs + 93 ,  205
#GW  xs + 96 ,  339
#GW  xs + 99 ,  373
#GW  xs + 102 ,  237
#GW  xs + 105 ,  271
#GW  xs + 108 ,  265
#GW  xs + 111 ,  324
#GW  xs + 114 ,  350
#GW  xs + 117 ,  279
#GW  xs + 120 ,  219
#GW  xs + 123 ,  371
#GW  xs + 126 ,  358
#GW  xs + 129 ,  163
#GW  xs + 132 ,  344
#GW  xs + 135 ,  408
#GW  xs + 138 ,  355
#GW  xs + 141 ,  183
#GW  xs + 144 ,  413
#GW  xs + 147 ,  182
#GW  xs + 150 ,  259
#GW  xs + 153 ,  332
#GW  xs + 156 ,  199
#GW  xs + 159 ,  408
#GW  xs + 162 ,  304
#GW  xs + 165 ,  200
#GW  xs + 168 ,  190
#GW  xs + 171 ,  217
#GW  xs + 174 ,  280
#GW  xs + 177 ,  275
#GW  xs + 180 ,  372
#GW  xs + 183 ,  420
#GW  xs + 186 ,  350
#GW  xs + 189 ,  242
#GW  xs + 192 ,  369
#GW  xs + 195 ,  434
#GW  xs + 198 ,  350
#GW  xs + 201 ,  442
#GW  xs + 204 ,  285
#GW  xs + 207 ,  177
#GW  xs + 210 ,  421
#GW  xs + 213 ,  300
#GW  xs + 216 ,  199
#GW  xs + 219 ,  255
#GW  xs + 222 ,  362
#GW  xs + 225 ,  180
#GW  xs + 228 ,  173
#GW  xs + 231 ,  359
#GW  xs + 234 ,  445
#GW  xs + 237 ,  224
#GW  xs + 240 ,  268
#GW  xs + 243 ,  164
#GW  xs + 246 ,  187
#GW  xs + 249 ,  275
#GW  xs + 252 ,  264
#GW  xs + 255 ,  220
#GW  xs + 258 ,  169
#GW  xs + 261 ,  362
#GW  xs + 264 ,  194
#GW  xs + 267 ,  347
#GW  xs + 270 ,  182
#GW  xs + 273 ,  172
#GW  xs + 276 ,  426
```

```
#GW  xs + 279 ,  390
#GW  xs + 282 ,  380
#GW  xs + 285 ,  265
#GW  xs + 288 ,  372
#GW  xs + 291 ,  314
#GW  xs + 294 ,  315
#GW  xs + 297 ,  281
#GW  xs + 300 ,  240
#GW  xs + 303 ,  198
#GW  xs + 306 ,  399
#GW  xs + 309 ,  397
#GW  xs + 312 ,  360
#GW  xs + 315 ,  420
#GW  xs + 318 ,  205
#GW  xs + 321 ,  198
#GW  xs + 324 ,  218
#GW  xs + 327 ,  354
#GW  xs + 330 ,  388
#GW  xs + 333 ,  376
#GW  xs + 336 ,  401
#GW  xs + 339 ,  302
#GW  xs + 342 ,  412
#GW  xs + 345 ,  410
#GW  xs + 348 ,  271
#GW  xs + 351 ,  201
#GW  xs + 354 ,  430
#GW  xs + 357 ,  392
#GW  xs + 360 ,  379
#GW  xs + 363 ,  424
#GW  xs + 366 ,  365
#GW  xs + 369 ,  169
#GW  xs + 372 ,  418
#GW  xs + 375 ,  251
#GW  xs + 378 ,  185
#GW  xs + 381 ,  181
#GW  xs + 384 ,  251
#GW  xs + 387 ,  192
#GW  xs + 390 ,  227
#GW  xs + 393 ,  348
#GW  xs + 396 ,  362
#GW  xs + 399 ,  254
#GW  xs + 402 ,  337
#GW  xs + 405 ,  237
#GW  xs + 408 ,  227
#GW  xs + 411 ,  332
#GW  xs + 414 ,  171
#GW  xs + 417 ,  365
#GW  xs + 420 ,  272
#GW  xs + 423 ,  347
#GW  xs + 426 ,  288
#GW  xs + 429 ,  276
#GW  xs + 432 ,  400
#GW  xs + 435 ,  202
#GW  xs + 438 ,  280
#GW  xs + 441 ,  246
#GW  xs + 444 ,  431
#GW  xs + 447 ,  363
#GW  xs + 450 ,  256
#GW  xs + 453 ,  225
#GW  xs + 456 ,  417
#GW  xs + 459 ,  250
#GW  xs + 462 ,  185
#GW  xs + 465 ,  238
#GW  xs + 468 ,  340
#GW  xs + 471 ,  180
#GW  xs + 474 ,  206
#GW  xs + 477 ,  222
#GW  xs + 480 ,  349
#GW  xs + 483 ,  266
#GW  xs + 486 ,  200
#GW  xs + 489 ,  323
#GW  xs + 492 ,  193
#GW  xs + 495 ,  262
#GW  xs + 498 ,  274
#GW  xs + 501 ,  215
#GW  xs + 504 ,  434
```

```
TouchMakro: 2
       #CR ; recall grid
       #FG GREEN,BLACK ; set color for lines
       #GS  xs + 0 ,  436 ; 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, 418, 168, 273, 305, 311, 183, 445, 375, 445, 303, 259, 424, 331, 371, 271,
414, 290, 379, 377, 312, 441, 373, 282, 180, 303, 196, 438, 298, 364, 362, 205, 339, 373,
237, 271, 265, 324, 350, 279, 219, 371, 358, 163, 344, 408, 355, 183, 413, 182, 259, 332,
199, 408, 304, 200, 190, 217, 280, 275, 372, 420, 350, 242, 369, 434, 350, 442, 285, 177,
421, 300, 199, 255, 362, 180, 173, 359, 445, 224, 268, 164, 187, 275, 264, 220, 169, 362,
194, 347, 182, 172, 426, 390, 380, 265, 372, 314, 315, 281, 240, 198, 399, 397, 360, 420,
205, 198, 218, 354, 388, 376, 401, 302, 412, 410, 271, 201, 430, 392, 379, 424, 365, 169,
418, 251, 185, 181, 251, 192, 227, 348, 362, 254
       #GX 3, 337, 237, 227, 332, 171, 365, 272, 347, 288, 276, 400, 202, 280, 246, 431,
363, 256, 225, 417, 250, 185, 238, 340, 180, 206, 222, 349, 266, 200, 323, 193, 262, 274,
215, 434


; set diffrent line-widths
TouchMakro: 3
       #GZ 1,1
TouchMakro: 4
       #GZ 3,3
TouchMakro: 5
       #GZ 5,5
```

9.22 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 128. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 89.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Bargraph\

File:
BEGINNER - 2 Bargraphs with backlight dimming.kmc

Commands:
#BR, #YB

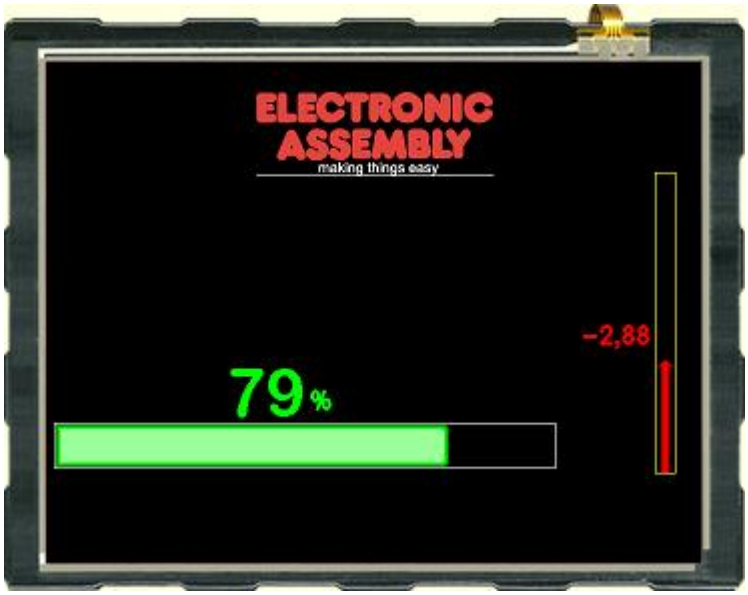
Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; 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 1,0,350,500,400,0,100,5          ; 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 1,250,280,"0=0;100=100"          ; place the value of bargraph 1
                                            ; 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 270,300,"%"                      ; place the text "%" to

;---- 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 2,600,400,625,60,0,100,5 ; 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 2,580,180,"0=-12,00;100=12,00"  ; place the value of bargraph 2
                                            ; 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
```

9.23 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 126.
If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 89.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Bargraph\

File:
EXPERT - 2 Bargraphs with backlight dimming.kmc

Commands:
#BR, #YB

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
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 = 0
br1w = 500
br1y = 350
br1h = 45
       #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 = 600
br2w = 20  ;same thickness as bar1
br2y = 400
br2h = 300
       #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-br2h/2,"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
```

9.24 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[131]). If you want an overview about all instruments, refer to EXPERT - show some instruments.kmc[132].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Instruments\

File:
BEGINNER - compass_by_touch.kmc

Commands:
#IP

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line

;---- Place Insturment ----
       #IP 1, 250,200, 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
```

9.25 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[130]). If you want an overview about all instruments, refer to EXPERT - show some instruments.kmc[132].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Instruments\

File:
BEGINNER - instrument_by_analoginput.kmc

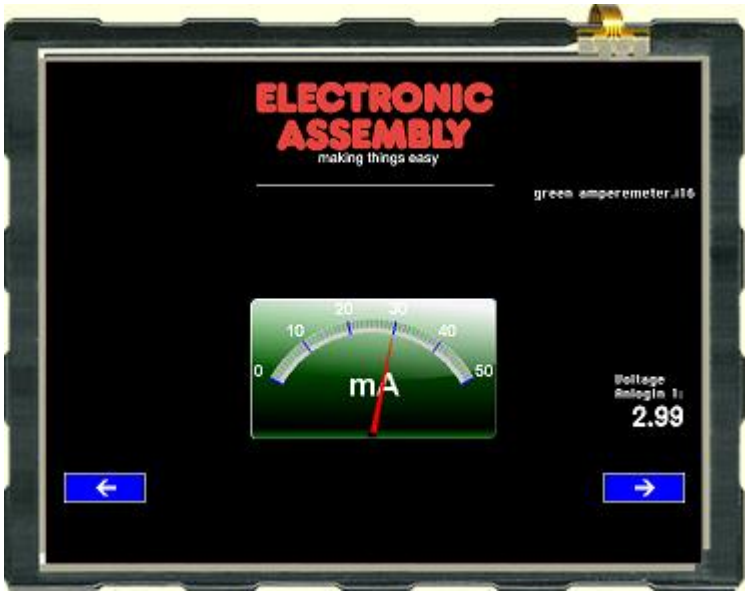
Commands:
#IP, #V+

Open file in KitEditor

```
eDIPTFT57-A "Instrument 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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line

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

;---- Place Insturment ----
       #IP 1,  216,200, 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 1,330,320
```

9.26 Instrument slide show - EXPERT

Place many instrumentsand connect them with the anlogue input.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Instruments\

File:
EXPERT - show some instruments.kmc

Commands:
#IP, #V+

Open file in KitEditor

```
eDIPTFT57-A "Instrument slide show 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=10
BT_Left_ystart=400
BT_width=80
BT_height=YMAX-BT_Left_ystart-50

BT_Right_xstart=XMAX-BT_width-BT_Left_xstart
BT_Right_ystart=BT_Left_ystart
```

```
Macro: MnAutoStart
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
yoff = 10
YoffLOGO=yoff+PICTURE_H(LOGO)+20
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=540
y=300
AnalogeMacro: 0       ; rising analog input 1 (->hysteresis)
        #VR 1          ; redraw instrument 1 with new anlog value
        #ZL x,y, !VOLTAGE!
str_p = STRING_P(1,1,0,0,0,0) ; You have to set STRING_P before using STRING_W. Please
have a look at the help files (F1) if you need further information
        #VG 1, x+STRING_W(!VOLTAGE!, str_p, Chicago14),y+STRING_H(!VOLTAGE!, str_p,
Chicago14)+3 ;diplay analog value
```

9.27 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 138.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Language\

File:
BEGINNER – multilingual.kmc

Commands:
#MK

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line
       #GD 320,140,320,350

;---- Place some text ----
       #FZ WHITE, BLACK      ;string color
       #ZF CHICAGO14         ;string font
       #ZL 65,150, "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 70,170,160,210,1,0 "Deutsch"            ; place button "German" and call
TouchMacro 1
       #AT 70,230,160,270,2,0 "English"            ; place button "English" and call
TouchMacro 2
       #AT 180,170,270,210,3,0 "Fran"135"ais"      ; place button "French" and call
TouchMacro 3
                                                   ; the 135 is the decimal value for
cdille
       #AT 180,230,270,270,4,0 "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 370,150,580,180          ; delete area behind text
       #ZL 370,150, "Deutsch"       ; write actual language
       #UI 370,190,100              ; place picture

Macro 1[ENGLISH]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 370,150,580,180          ; delete area behind text
       #ZL 370,150, "English"       ; write actual language
       #UI 370,190,100              ; place picture

Macro 1[FRENCH]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 370,150,580,180                ; delete area behind text
       #ZL 370,150,"Fran"135"ais"   ; write actual language
       #UI 370,190,100              ; place picture

Macro 1[ITALIAN]:
       #FZ WHITE, BLACK             ; string color
       #ZF CHICAGO14                ; string font
       #RL 370,150,580,180          ; delete area behind text
       #ZL 370,150, "Italiano"      ; write actual language
       #UI 370,190,100              ; place picture
```

9.28 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 136.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Language\

File:
EXPERT - stringtable.kmc

Commands:
#ST, #MK

Open file in KitEditor

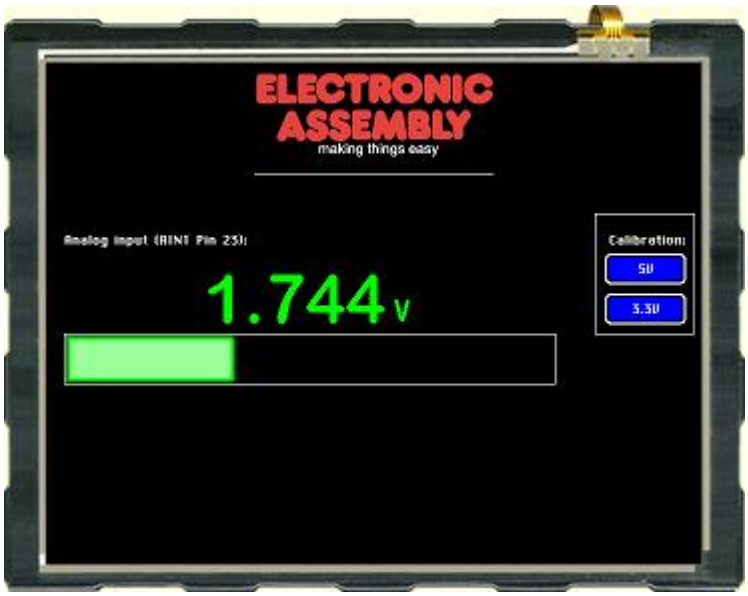
```
eDIPTFT57-A   "Stringtable"
...
...
...
;---------------------------------------------------------------------------
; include pictures
Path <..\..\..\bitmaps\color\>                    ; specify path
LOGO = 1 ; unsing constants makes it easier
Picture: LOGO, <ea logo making things easy black.bmp>     ; 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 ----
       #TA                          ; Terminal off
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 = 200
xw = 100
ys = 190
yh = 40
pitch = 20
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
x=xs+2*xw+pitch+20
```

```
y=ys+yh
        #RL x,y,XMAX,y+35     ; delete area behind text
        #ZL x,y, StringCode, HELLO
```

9.29 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[145] or EXPERT – Port Macro.kmc[145]. In adition you will find halp using ProcessMacros (see BEGINNER - Prozess Macro.kmc[150]) and AutomaticMacros (see EXPERT – Automatic Macro.kmc[148]).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Macro\

File:
BEGINNER - Analog Macro.kmc

Commands:
#VG, #V@, #VE

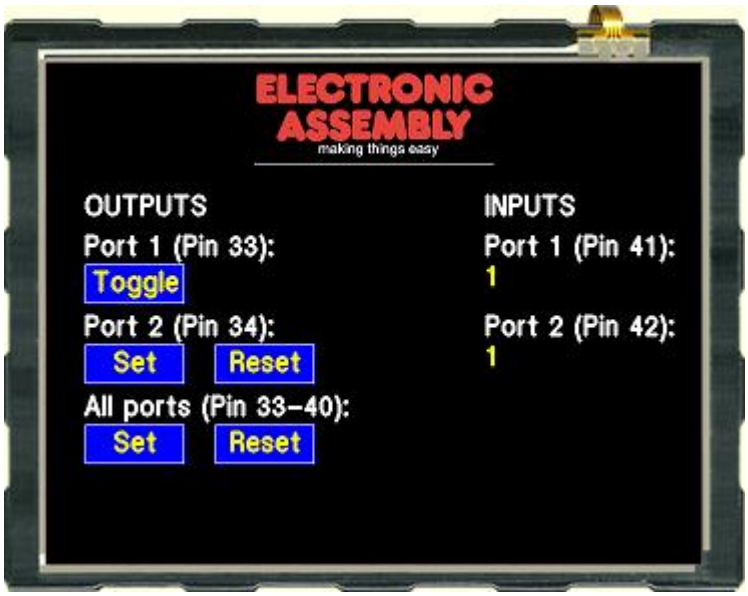
Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,100,438,100          ; draw a Line
       #GR 540,140,639,260          ; draw a rectangle

;---- Write some information ----
       #FZ WHITE, BLACK      ;string color
       #ZF CHICAGO14         ;string font
       #ZC 100,160, "Analog input (AIN1 Pin 23):"
       #ZR 630,160, "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 1,10,260,500,310,0,250,5 ; 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 1,330,200                        ; place the value of AIN 1 to row 190 and line
130

;---- 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 355,225,"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 550,180,630,210,1,0 "5V"         ;
       #AT 550,220,630,250,2,0 "3.3V"



;-------------------------------------------------------------------------------
AnalogMacro 0:        ; this macro is called by evry change of input voltage AIN1
       #VR 1                ; redraw bargraph analog input 1
       #VG 1,330,200        ; 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
```

9.30 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 148), ProcessMacros (see BEGINNER - Prozess Macro.kmc 150), PortMacros (see BEGINNER - Port Macro.kmc 145) and AnalogueMacros (see BEGINNER - Analog Macro.kmc 141).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Macro\

File:
BEGINNER – Bit Macro.kmc

Commands:
#YW

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,90,438,90            ; draw a Line


;----- OUTPUTS ---
       #FZ WHITE, BLACK      ;String color
       #ZF SWISS30B          ;String font
       #ZL 30,120, "OUTPUTS"
       #ZL 30,160, "Port 1 (Pin 33):"
       #ZL 30,240, "Port 2 (Pin 34):"
       #ZL 30,320, "All ports (Pin 33-40):"


       #FE BLUE, WHITE, BLUE, YELLOW, WHITE, YELLOW      ; Define button colors
       #FA YELLOW, BLUe                                  ; Define text colors
       #AF SWISS30B                                      ; Touchfont

       #AT 30,190,130,230,1,0,"CToggle"           ; Define Button with TouchMacro 1
                                                  ; "C" means aligment centered
       #AT 30,270,130,310,2,0,"CSet"                    ; Define Button with TouchMacro
2(down code) and 3 (up code)
       #AT 160,270,260,310,3,0,"CReset"           ; Define Button with TouchMacro 2(down
code) and 3 (up code)
```

```
        #AT 30,350,130,390,4,0,"CSet"                    ; Define Button with TouchMacro
3
        #AT 160,350,260,390,5,0,"CReset"           ; Define Button with TouchMacro 4

;---- INPUTS ---
        #FZ WHITE, BLACK      ;String color
        #ZF SWISS30B          ;String font
        #ZL 430, 120, "INPUTS"
        #ZL 430, 160, "Port 1 (Pin 41):"
        #ZL 430, 240, "Port 2 (Pin 42):"



;-------------------------------------------------------------------------------
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 430, 190, "1"
BitMacro 1:                   ;Port 1 falling edge
        #FZ YELLOW, BLACK     ;String color
        #ZL 430, 190, "0"

BitMacro 10:                  ;Port 2 rising edge
        #FZ YELLOW, BLACK     ;String color
        #ZL 430, 270, "1"
BitMacro 2:                   ;Port 2 falling edge
        #FZ YELLOW, BLACK     ;String color
        #ZL 430, 270, "0"
```

9.31 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[148]), ProcessMacros (see BEGINNER - Prozess Macro.kmc[150]), BitMacros (see BEGINNER – Bit Macro.kmc[143]) and AnalogueMacros (see BEGINNER - Analog Macro.kmc[141]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Macro\

File:
EXPERT – Port Macro.kmc

Commands:
#YW

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
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)

;----- OUTPUTS ---
```

```
       #FZ WHITE, BLACK      ;String color
       #ZF SWISS30B          ;String font
xs=30
y_title = 130
       #ZL xs,y_title, "OUTPUTS"
ys=y_title+40
y=ys
pitch=100
       #ZL xs,y, "Port 1 (Pin 33):"
y+=pitch
       #ZL xs,y, "Port 2 (Pin 34):"
y+=pitch
       #ZL xs,y "All ports (Pin 33-40):"


       #FE BLUE, WHITE, BLUE, YELLOW, WHITE, YELLOW      ; Define button colors
       #FA YELLOW, BLUE                                  ; Define text colors
       #AF SWISS30B                                ; Touchfont
x=xs
bt_ypitch=35
y=ys+bt_ypitch
yh=40
xw=100
bt_xpitch = 30
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 SWISS30B          ;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)!
```

9.32 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[141], BEGINNER – Bit Macro.kmc[143], EXPERT – Port Macro.kmc[145]) and a ProcessMacro example (see BEGINNER - Prozess Macro.kmc[150]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Macro\

File:
EXPERT – Automatic Macro.kmc

Commands:
#MJ

Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
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 = 190

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"
```

9.33 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[141], BEGINNER – Bit Macro.kmc[143], EXPERT – Port Macro.kmc[145]) and an AutomaticMacro example (see EXPERT – Automatic Macro.kmc[148]).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Macro\

File:
BEGINNER - Prozess Macro.kmc

Commands:
#MD

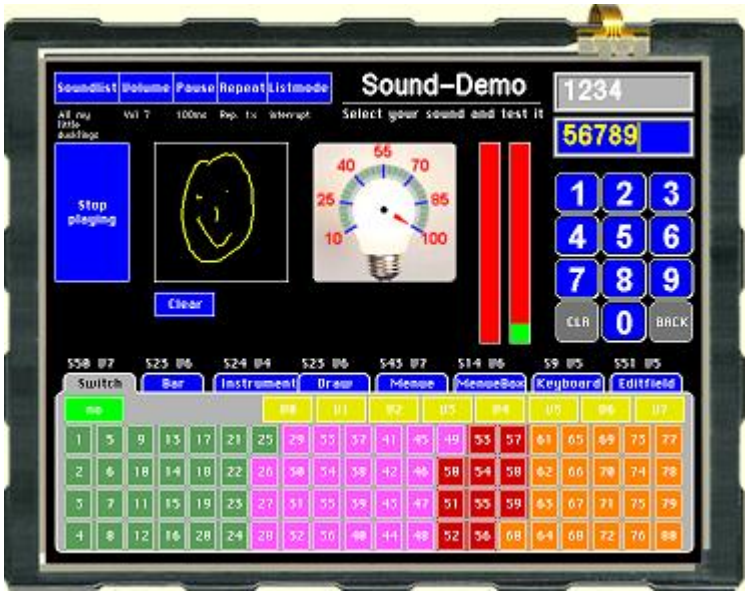
Open file in KitEditor

```
eDIPTFT57-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 ----
       #TA                          ; Terminal off
       #UI 202,0,1                  ; place Picture no. 1
       #GD 200,90,438,90            ; 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 = 320                      ; defining a constant makes it more easy to move the whole
group later
Y = 180

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"
```

9.34 Sound selection - EXPERT

Useful demo to select sounds. All available sounds are hearable and selectable.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Sounds\

File:
EXPERT - sound_selection.kmc

Commands:
#YT, #AU, #A-

Open file in KitEditor

It's not possible to show sample code in this helpfile. Please open the files with KitEditor.

9.35 Sound - BEGINNER

Play a song. It's stored as a different tonescales (All my little ducklings). To select the default sounds, please use demo EXPERT - sound_selection.kmc[152]. Another application is EXPERT - Piano.kmc[154].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Sounds\

File:
BEGINNER - Sound.kmc

Commands:
#YT

Open file in KitEditor

```
eDIPTFT57-A "simple sound demo"
...
...
...
;-------------------------------------------------------------------------------
;include pictures
Path <..\..\..\bitmaps\color\>
Picture 1 <ea logo making things easy black.bmp>  ; double click to open
;-------------------------------------------------------------------------------

Makro: MnAutoStart

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 202,20,1                 ; place Picture no. 1
       #GD 200,120,438,120          ; draw a Line

;---- set sound ----
       #AS 0                ; turn buzzer function off, if touch is inverted
       #A* 0                ; turn vibrator off, if touch is inverted
       #A- 1,50,7           ; set touch key sound to sound nr. 50 ant volume 7


;---- Place 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 270,200,370,250,0,1 "Push" ; draw Touch area - call touchmacro, after release
of touchbutton


TouchMacro: 1 ; play a soundqueue
       #YT "4CDEF2GG4AAAA1G4AAAA1G4FFFF2EE4DDDD1C" ;All my little ducklings
```

9.36 Piano - EXPERT

Show a piano and make it possible to play. To select the default sounds, please use demo EXPERT - sound_selection.kmc[152]. Another application is BEGINNER - Sound.kmc[153].

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\ eDIPTFT57-A\How to use\Sounds\

File:
EXPERT - Piano.kmc

Commands:
#YT, #AU

Open file in KitEditor

```
eDIPTFT57-A "play piano"
...
...
...
;-------------------------------------------------------------------------------
; include pictures
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

; specify pictures for piano
Path<.\Piano\>
PianoLeft   = 10
PianoMiddle = PianoLeft + 1
PianoRight  = PianoLeft + 2
PianoBlack  = PianoLeft + 3
Button: PianoLeft   <PianoWhiteLeft.G16>
Button: PianoMiddle <PianoWhiteMiddle.G16>
Button: PianoRight  <PianoWhiteRight.G16>
Button: PianoBlack  <PianoBlack.G16>

;-------------------------------------------------------------------------------
;constants for touchmacros
STOP_PLAY = 200
PLAY      = 100

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

Makro: MnPowerOn
;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA           ; Terminal off
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 piano ----
       #AS 0                ; turn buzzer off, if touch is pressed     ;
       #A- SNDKEY, 0,0      ; no sound play if touch is pressed

!SPACELINES! = "||||||"     ; this constant is important, to place text not in the middle
of a toucharea (piano is labled at the buttom of each touch field)
w = BUTTON_W(PianoLeft)
h = BUTTON_H(PianoLeft)
x = (XPIXEL - w *14) / 2
y =  YPIXEL - h-40
t = PLAY ; constant for touch macro, it's incrementen every time you place button, please
refer to the lines where a button is placed
bt_down_offset=12
       #AF SWISS30B ; touch label font
       #FA GREY,BLACK ; touch label color
       #AO 0,bt_down_offset ; offset for touch strings at button down event
       #AC PianoLeft,0,1,1                      ; use left piano button as button
picture
       #AU x+w* 0,y, t++,STOP_PLAY, !SPACELINES!,"CC"    ; place button, touchmacro is
set with constant t and incremented for next touchbutton
       #AC PianoMiddle,0,1,1
       #AU x+w* 1,y, t++,STOP_PLAY, !SPACELINES!,"D"
       #AC PianoRight,0,1,1
       #AU x+w* 2,y, t++,STOP_PLAY, !SPACELINES!,"E"
       #AC PianoLeft,0,1,1
       #AU x+w* 3,y, t++,STOP_PLAY, !SPACELINES!,"F"
       #AC PianoMiddle,0,1,1
       #AU x+w* 4,y, t++,STOP_PLAY, !SPACELINES!,"G"
       #AU x+w* 5,y, t++,STOP_PLAY, !SPACELINES!,"A"
       #AC PianoRight,0,1,1
       #AU x+w* 6,y, t++,STOP_PLAY, !SPACELINES!,"H"

       #AC PianoLeft,0,1,1
       #AU x+w* 7,y, t++,STOP_PLAY, !SPACELINES!,"c"
       #AC PianoMiddle,0,1,1
       #AU x+w* 8,y, t++,STOP_PLAY, !SPACELINES!,"d"
       #AC PianoRight,0,1,1
       #AU x+w* 9,y, t++,STOP_PLAY, !SPACELINES!,"e"
       #AC PianoLeft,0,1,1
       #AU x+w*10,y, t++,STOP_PLAY, !SPACELINES!,"f"
       #AC PianoMiddle,0,1,1
       #AU x+w*11,y, t++,STOP_PLAY, !SPACELINES!,"g"
       #AU x+w*12,y, t++,STOP_PLAY, !SPACELINES!,"a"
       #AC PianoRight,0,1,1
       #AU x+w*13,y, t++,STOP_PLAY, !SPACELINES!,"h"

!SPACELINES! = "|||||"
x = x + w - BUTTON_W(PianoBlack) / 2
       #AF CHICAGO14
       #FA GREY,WHITE
       #AO 0,2
       #AC PianoBlack,0,1,1
       #AU x+w*0,y, t++,STOP_PLAY, !SPACELINES!,"CCIS"
       #AU x+w*1,y, t++,STOP_PLAY, !SPACELINES!,"DIS"
       #AU x+w*3,y, t++,STOP_PLAY, !SPACELINES!,"FIS"
       #AU x+w*4,y, t++,STOP_PLAY, !SPACELINES!,"GIS"
       #AU x+w*5,y, t++,STOP_PLAY, !SPACELINES!,"AIS"

       #AU x+w* 7,y, t++,STOP_PLAY, !SPACELINES!,"cis"
       #AU x+w* 8,y, t++,STOP_PLAY, !SPACELINES!,"dis"
       #AU x+w*10,y, t++,STOP_PLAY, !SPACELINES!,"fis"
       #AU x+w*11,y, t++,STOP_PLAY, !SPACELINES!,"gis"
       #AU x+w*12,y, t++,STOP_PLAY, !SPACELINES!,"ais"

;-------------------------------------------------------------------------------
t=PLAY
TouchMacro: t++
       #YT "C" ; play note
TouchMacro: t++
       #YT "D"
TouchMacro: t++
       #YT "E"
TouchMacro: t++
       #YT "F"
TouchMacro: t++
       #YT "G"
TouchMacro: t++
```

```
        #YT "A"
TouchMacro: t++
        #YT "H"

TouchMacro: t++
        #YT "c"
TouchMacro: t++
        #YT "d"
TouchMacro: t++
        #YT "e"
TouchMacro: t++
        #YT "f"
TouchMacro: t++
        #YT "g"
TouchMacro: t++
        #YT "a"
TouchMacro: t++
        #YT "h"

TouchMacro: t++
        #YT "#C"
TouchMacro: t++
        #YT "#D"
TouchMacro: t++
        #YT "#F"
TouchMacro: t++
        #YT "#G"
TouchMacro: t++
        #YT "#A"

TouchMacro: t++
        #YT "#c"
TouchMacro: t++
        #YT "#d"
TouchMacro: t++
        #YT "#f"
TouchMacro: t++
        #YT "#g"
TouchMacro: t++
        #YT "#a"

TouchMacro: STOP_PLAY
        #YT "" ; stop playing
```