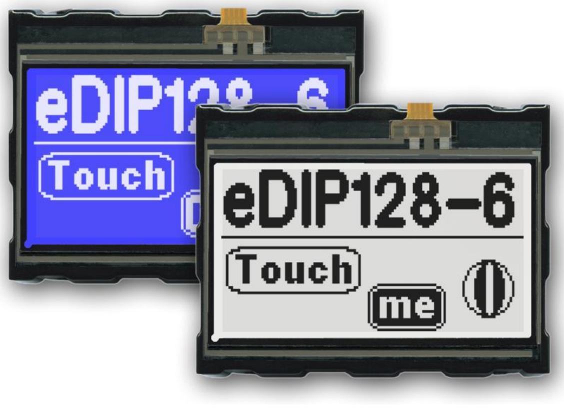


EA eDIP128-6
compiler manual

Mai 2012

Table of Contents

1 Overview 4

2 Syntax rules 5

3 Compiler Functions 6

4 Compiler Options 8

4.1 General 8
4.2 Transfer 9
4.3 Font 10
4.4 WinFont 11
4.5 ExportOverview 12
4.6 ExportWinfont 13
4.7 LogFontWidth 14
4.8 Picture 15
4.9 SystemMacros 16
4.10 ExportMacro 17
4.11 Macro 18
4.12 TouchMacro 19
4.13 MenuMacro 20
4.14 PortMacro 21
4.15 BitMacro 22

5 EA eDIP128-6 commands 23

5.1 Terminal 23
5.2 Text 24
5.3 Display 25
5.4 Draw 26
5.5 Flashing 27
5.6 Bitmap 28
5.7 Clipboard 29
5.8 Bargraph 30
5.9 Macros 31
5.10 Touch 32
5.11 Menu 35
5.12 I/O-Ports 36
5.13 Other commands 37

6 Default Fonts 38

6.1 Terminal 8x8 38
6.2 Font 4x6 39
6.3 Font 6x8 40
6.4 Font 7x12 41
6.5 Geneva 10 42
6.6 Chicago 14 43
6.7 Swiss 30 44
6.8 BigZif 57 45

7 Internal Pattern 46

8 Internal Border 47

9 Link Modes 48

10 BLH format 49

11 How-to-use 50

11.1 Factory Setting 52
11.2 RS485 - Factory Setting 54
11.3 Place Strings - BEGINNER 56
11.4 Text linking - EXPERT 58
11.5 BMP file - BEGINNER 60
11.6 3 simple touch buttons - BEGINNER 61
11.7 Radio group - BEGINNER 63
11.8 Keypad - EXPERT 64
11.9 Free draw area with clipboard - BEGINNER 66
11.10 Free draw area with clipboard - EXPERT 67
11.11 Frame - BEGINNER 69
11.12 GrafikModes - EXPERT 71
11.13 Bargraph by touch - BEGINNER 74
11.14 Bargraph by touch - EXPERT 75
11.15 Menue - BEGINNER 76
11.16 Menue - EXPERT 78
11.17 Bit Macro - BEGINNER 80
11.18 Port Macro - BEGINNER 82
11.19 Automatic Macro - BEGINNER 83
11.20 Automatic Macro - EXPERT 85
11.21 Change display orientation - BEGINNER 87
11.22 Energy save modes - BEGINNER 89
11.23 Energy save modes - EXPERT 90

1 Overview

General

The EA eDIP128-6 is able to store many pictures, fonts and macros in internal EEPROM 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 eDIP128-6.

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 eDIP128-6 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

MIN(value1,value2,...)	returns the minimum value
MAX(value1,value2,...)	returns the maximum value
AVG(value1,value2,...)	returns the average value

RANDOM(min,max)
RANDOM(min,max,delta)
returns a random value from the range min..max
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) **COS**(w, a) **TAN**(w, a)
w = angle in tenth of degree
a = amplitude

to calculate the bounding box of an image 15 following functions are available:
PICTURE_W(nr) returns the width
PICTURE_H(nr) returns the height

to calculate the bounding box of a Stringconstant 5 following functions are available:
STRING_W(!NAME!, par, font) returns the width
STRING_H(!NAME!, par, font) returns the height

font = font number (eDIP command #ZF 24)
par = **STRING_P**(zoomX, zoomY, height, space)
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 24)
height = additional line spacing between two lines 0..15 (eDIP command #ZY 24)
space = spacewidth (eDIP command #ZJ) 24

Example:

```
!TEXT! = "Hello World"

font       = SWISS30B
zoomX      = 1
zoomY      = 1
addheight  = 3
spacewidth = 0

Makro: MnPowerOn
       #ZF font
       #ZZ zoomX,zoomY
       #ZY addheight
       #ZJ spacewidth

par = STRING_P(zoomX,zoomY,addheight,spacewidth)
w = STRING_W(!TEXT!,par,font)
h = STRING_H(!TEXT!,par,font)
x = (XPIXEL-w)/2
y = (YPIXEL-h)/2
       #RS x,y, x+w-1, y+h-1
       #ZV INVERS
       #ZL x,y,!TEXT!
```

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

`eDIP128-6 "title"`	Defines EA eDIP128-6 as target. "title" is a short description for the project. It is shown on the display when uploading the EEPROM memory of the module.
`DESTINATION <new.eep>`	Specifies a new file name for the EEPROM 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 EVALeDIP128 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 54)
`VERIFY`	Verifies the complete contents of the EEPROM memory after upload.

4.3 Font

`FONT: nr,<file>`

Defines a font file which will be assigned to the number nr (1..15).
<file> can be *.FXT format.
Font number 0 is internal 8x8 terminal font and can not be changed

(see How-to-use example Place Strings - BEGINNER 56)

---

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

```
; default fonts  (max. 31 fonts number 1..32)
FONT8x8   = 0        ; internal terminal font
FONT4x6   = 1
FONT6x8   = 2
FONT7x12  = 3
GENEVA10  = 4
CHICAGO14 = 5
SWISS30B  = 6
BIGZIF57  = 7

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: BIGZIF57,  <BIGZIF57.FXT>
```

see Character Table
Terminal 8x8 38

Font 4x6 39
Font 6x8 40
Font 7x12 41

Geneva 10 42
Chicago 14 43
Swiss 30 44

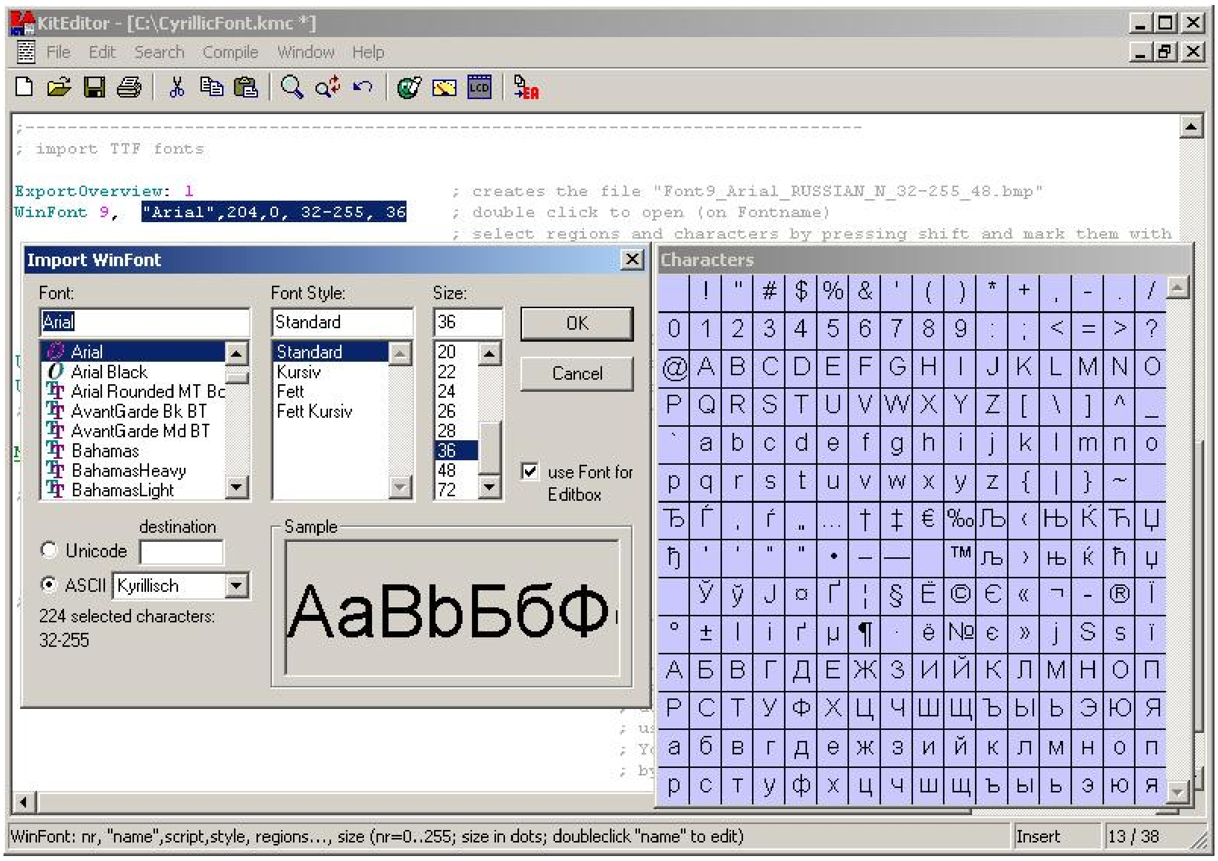
BigZif 57 45

4.4 WinFont

```
WINFONT: nr, "name",script,style, regions.., size
```

Defines a Windows font and assigns to font number nr (1..15).
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.

(see How-to-use example Place Strings - BEGINNER 56)



KitEditor - [C:\CyrillicFont.kmc *]

File Edit Search Compile Window Help

```
;--------------------------------------------------------------------------------
; import TTF fonts

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
```

Import WinFont

Font: Arial — Arial, Arial Black, Arial Rounded MT Bc, AvantGarde Bk BT, AvantGarde Md BT, Bahamas, BahamasHeavy, BahamasLight

Font Style: Standard — Standard, Kursiv, Fett, Fett Kursiv

Size: 36 — 20, 22, 24, 26, 28, 36, 48, 72

OK Cancel

[✓] use Font for Editbox

destination: ( ) Unicode [● ASCII] Kyrillisch

224 selected characters: 32-255

Sample: AaBbБбФ

Characters

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
	Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
°	±	І	і	ґ	µ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

WinFont: nr, "name",script,style, regions..., size (nr=0..255; size in dots; doubleclick "name" to edit) | Insert | 13 / 38

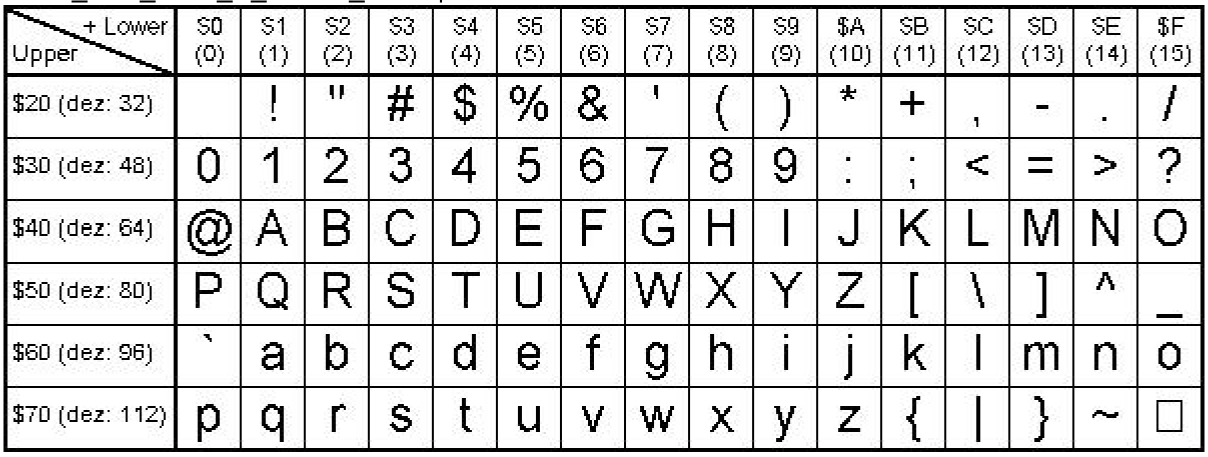
4.5 ExportOverview

`EXPORTOVERVIEW: n1`	This statement enables the generation of a BMP file for all following WinFonts. This is good to get an overview which character are available. n1= 1: an bitmap will be exported n1= 0: no export

Example:

```
ExportOverview: 1
WinFont: 9,  "Arial",0,0, 32-127, 48      ; export "Font9_Arial_ANSI_N_32-
127_48.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	{	|	}	~	□

4.6 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'
..###..
.#...#.
#......
#......
#......
#......
.#...#.
..###..
```

4.7 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
```

4.8 Picture

```
PICTURE: nr,<file>
PICTURE: nr <file1>,<file2>
```

It is convenient to store all bitmap in EEPROM; this will save transfer time via serial interface. The statement PICTURE defines a bitmap <file>with nr (0..255). <file> has to be a monochrome BMP. Optionally 2 different pictures 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.

The pictures can be used with the Bitmap commands. 28
You can use the Compiler Functions 6 `PICTURE_W` and `PICTURE_H` to get the outline in pixels of the picture.

(see How-to-use example BMP file - BEGINNER 60)

4.9 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 3V or lower.
`WAKEUPPINMACRO:`	Starts up again when the display was in PowerDown mode and Pin13 WUP goes to LO.
`WAKEUPTOUCHMACRO:`	Starts up again when the display was in PowerDown mode and the touchpanel is touched.
`WAKEUPI2CMACRO:`	Starts up again when the display was in PowerDown mode and commands are arriving through I2C interface.

4.10 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 thbyte-array (default is "unsigned char")
<filename>: optionally another filename (default is "macroname_macronumber")

Example:

```
ExportMacro: 1, "char flash"

Macro: 5
      #TA

      #ZF FONT4x6
      #ZL 4,10, "Font4x6 0123456789"
      #ZF FONT6x8
      #ZL 4,20, "Font6x8 Schriftprobe"
      #ZF FONT7x12
      #ZL 4,30, "Font7x12: Schrift"
```

Output in Logfile "Macro_5.h":

```
/* Macro 5 as include */

#define MACRO_5_LEN  88

char flash MACRO_5[MACRO_5_LEN] =
{
   27, 84, 65, 27, 90, 70,  1, 27, 90, 76,  4, 10, 70,111,110,116, 52,120, 54, 32,
   48, 49, 50, 51, 52, 53, 54, 55, 56, 57,  0, 27, 90, 70,  2, 27, 90, 76,  4, 20,
   70,111,110,116, 54,120, 56, 32, 83, 99,64,114,105,102,116,112,114,111, 98,101,
    0, 27, 90, 70,  3, 27, 90, 76,  4, 30, 70,111,110,116, 55,120, 49, 50, 58, 32,
   83, 99,64,114,105,102,116,  0
};
```

4.11 Macro

`MACRO: nr`	Defines a normal macro with number nr (0..255). This macro will be executed with the command #MN nr 31 A series of macros occurring one after the other can be called cyclically (movie, hourglass, multi-page help text) see command for Automatic (normal-) macros 31. These automatic macros continue to be processed until either a command is received via the interface or a touch macro with a corresponding return code is activated. These macros are also called by macro processes at defined intervals (see command for Macro processes 31). Macro processes are not interrupted when commands are received from the interface or when touch macros are triggered. (see How-to-use example Automatic Macro - BEGINNER 83)

4.12 TouchMacro

`TOUCHMACRO: nr`	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 [31]. (see How-to-use example 3 simple touch buttons - BEGINNER [61])

4.13 MenuMacro

`MENUMACRO: nr`	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 31. (see How-to-use example Menue - BEGINNER 76)

4.14 PortMacro

`PORTMACRO: nr`	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 31. (see How-to-use example Port Macro - BEGINNER 82)

4.15 BitMacro

`BITMACRO: nr`	Defines a bit macro with number nr (0..255). bitmacros will start voltage at a single line IN1..8 (bit) will change or by command #MB nr 31. 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 36. (see How-to-use example Bit Macro - BEGINNER 80)

5 EA eDIP128-6 commands

5.1 Terminal

Terminal commands:

Position cursor	#TP n1,n2	n1=column; n2=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 off	#TA	Terminal display is switched off; outputs are rejected
Terminal on	#TE	Terminal display is switched on

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 eDIP128-6 V1.0 Rev.A"
Output projectname	#TJ	The macrofile-projectname is output in the terminal e.g. "init / delivery state"
Output informationen	#TI	The terminal is initialisized and cleared; the software version, hardware revision, macrofile-projectname and CRC-checksum are output in the terminal

Special ASCII-characters:

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

5.2 Text

Text settings:

Set font	#ZF n1	Set font with the number nr = 0..15 (see compiler option FONT 10 : or WINFONT 11 :)

Font zoom factor	#ZZ n1,n2	n1 = X-zoom factor (1x to 4x)
		n2 = Y-zoom factor (1x to 4x)

Additional line spacing	#ZY n1	Insert n1=0..15 dots between two lines as additional spacing
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°
Text link mode	#ZV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace (see link modes 48)
Text flashing attribute	#ZB n1	n1: 0=no flashing; 1=Text flashing on/off; 2=Text flashes inversely; 3=Text flashing off/on phase shifted

(see How-to-use example Text linking - EXPERT 58)

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). Text between two '~' ($7E) characters flashes on/off. Text between two '@' ($40) characters flashes inversely. Text between two '&' ($26) flashes off/on phase shifted. The character '\' ($5C, backslash) canceles the special function of '|', '~', '@','&' and '\'
Output string centered	#ZC x,y,"text..."	A string (text...) is output centered to x,y. Several lines are separated by the character '|' ($7C, pipe). Text between two '~' ($7E) characters flashes on/off. Text between two '@' ($40) characters flashes inversely. Text between two '&' ($26) flashes off/on phase shifted. The character '\' ($5C, backslash) canceles the special function of '|', '~', '@','&' and '\'
Output string right justified	#ZR x,y,"text..."	A string (text...) is output right justified to x,y. Several lines are separated by the character '|' ($7C, pipe). Text between two '~' ($7E) characters flashes on/off. Text between two '@' ($40) characters flashes inversely. Text between two '&' ($26) flashes off/on phase shifted. The character '\' ($5C, backslash) canceles the special function of '|', '~', '@','&' and '\'
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 56)

5.3 Display

Display commands (effect on the entire display):

Set display orientation	#DO n1	n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270° (0°+180°=128x64; 90°+270°=64x128)
Delete display	#DL	Delete display contents (all pixels off)
Fill display	#DS	Fill display contents (all pixels on)
Invert display	#DI	Invert display contents (invert all pixels)
Switch display off	#DA	Display contents become invisible but are retained, commands are still possible
Switch display on	#DE	Display contents become visible again
Show clip-board	#DC	Show content of clip-board Standard display output is no longer visible
Show current	#DN	Switch back to normal operation Standard display output is visible

(see How-to-use example Change display orientation - BEGINNER 87)

Contrast commands:

Save contrast	#D@	Save actually contrast value in internal EEPROM

Set contrast	#DK n1	n1=0..40: Set the display contrast to value n1 (default = 20)
		n1='+': increase contrast
		n1='-': decrease contrast

5.4 Draw

Draw straight lines and points:

Draw rectangle	#GR x1,y1,x2,y2	Draw four straight lines as a rectangle from x1,y1 to x2,y2
Draw straight line	#GD x1,y1,x2,y2	Draw straight line from x1,y1 to x2,y2
Continue straight line	#GW x1,y1	Draw a straight line from last end point to x1,y1
Draw point	#GP x1,y1	Set a point at coordinates x1,y1
Link mode	#GV n1	Set drawing mode n1: 1=set; 2=delete; 3=inverse; (see link modes 48)
Blink attribute	#GB n1	n1: 0=blink off,solid line; 1=blink on/off; 2=blink inverted; 3=blink off/on phase shifted
Point size/line thickness	#GZ n1,n2	n1=X-point size (1 to 15); n2=Y-point size (1 to 15)

(see How-to-use example GrafikModes - EXPERT 71)

Change/draw rectangular areas:

Delete area	#RL x1,y1,x2,y2	Delete an area from x1,y1 to x2,y2 (all pixels off)
Fill area	#RS x1,y1,x2,y2	Fill an area from x1,y1 to x2,y2 (all pixels on)
Invert area	#RI x1,y1,x2,y2	Invert an area from x1,y1 to x2,y2 (invert all pixels)
Area with fill pattern	#RM x1,y1,x2,y2,no	Draw area from x1,y1 to x2,y2 with pattern no (always set, see internal pattern 46)
Draw box	#RO x1,y1,x2,y2,no	Draw rectangle from x1,y1 to x2,y2 with pattern no (always replace, see internal pattern 46)
Draw frame	#RR x1,y1,x2,y2,no	Draw frame of type no from x1,y1 to x2,y2 (always set, see internal border 47)
Draw frame box	#RT x1,y1,x2,y2,no	Draw frame box of type no from x1,y1 to x2,y2 (always replace, see internal border 47)

(see How-to-use example Frame - BEGINNER 69)

5.5 Flashing

Flashing settings:

Set flashing time	#QZ n1	n1=1..15: Set the flashing time in 1/10sec; n1=0: deactivate flashing

Flashing areas:

Delete flashing attribute	#QL x1,y1,x2,y2	Delete the flashing attribute from x1,y1 to x2,y2 do not use this command for phase shifted areas! (Copies the area from grafiklayer to blinklayer)
Flashing inversely	#QI x1,y1,x2,y2	Define an inverted flashing area from x1,y1 to x2,y2 (Copies the inverted area from grafiklayer to blinklayer)
Flashing area pattern	#QM x1,y1,x2,y2,n1	Define a flashing area (on/off) with pattern n1 from x1,y1 to x2,y2 (Draw the pattern into blinklayer, see internal pattern 46)

Phase shifted areas:

Restore phase shifted area	#QR x1,y1,x2,y2,n1	Delete the phase shifted flashing area from x1,y1 to x2,y2 do not use this command for other flashing attributes! (Copies the area from blinklayer to grafiklayer)
Inverted phase shifted area	#QE x1,y1,x2,y2	Define a phase shifted inverted flashing area from x1,y1 to x2,y2 (Copies the inverted area from blinklayer to grafiklayer)
Phase shifted flashing pattern	#QP x1,y1,x2,y2,n1	Define a phase shifted flashing area (off/on) with pattern n1 from x1,y1 to x2,y2 (Draw the pattern into grafiklayer)

5.6 Bitmap

Bitmap settings:

Image zoom factor	#UZ n1,n2	n1 = X-zoom factor (1x to 4x)
		n2 = Y-zoom factor (1x to 4x)

Image angle	#UW n1	output angle of the image n1=0: 0°; n1=1: 90°
Image link mode	#UV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace; (see link modes 48)
Image flashing attribute	#UB n1	n1: 0=no flashing; 1=image flashing on/off; n1: 2=image flashing inversely; 3=image flashing off/on phase shifted

(see How-to-use example GrafikModes - EXPERT 71)

Output bitmaps:

Image from clipboard	#UC x1,y1	The current contents of the clipboard are loaded to x1,y1 with all the image attributes
Load internal image	#UI x1,y1,nr	Load internal image with the no (0 to 255) from the EEPROM memory to x1,y1 (see compiler option `PICTURE` 15 `:`)
Load image	#UL x1,y1,data...	Load an image to x1,y1; data... = image in BLH-format 49 This command is only for serial interface, do not use this command in a macro !

(see How-to-use example BMP file - BEGINNER 60)

Hardcopy:

Send hardcopy	#UH x1,y1,x2,y2	After this command, the image extract is sent in BLH-format 49

5.7 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 Free draw area with clipboard - BEGINNER 66)

5.8 Bargraph

Define bargraphs:

Define bargraph	#BR #BL #BO #BU no,x1,y1,x2,y2, sv,ev,type,pat	Define bargraph with number no=1..32 to L(eft), R(ight), O(up), U(down) x1,y1,x2,y2 form the rectangle enclosing the bar graph. sv, ev are the values for 0% and 100% type: 0=pattern bar; pat=bar pattern type: 1=pattern bar in rectangle; pat=bar pattern type: 2=line bar; pat=line width type: 3=line bar in rectangle; pat=line width (see internal pattern 46)

(see How-to-use example Bargraph by touch - BEGINNER 74)

Use bargraphs:

Update bargraph	#BA no,value	Set and draw the bargraph no to the new value
Draw bargraph new	#BZ no	Entirely redraw the bargraph with the number no
Send bargraph value	#BS no	Send the current value of bargraph number no
Delete bargraph	#BD no,n2	The definition of the bar graph with the number no becomes invalid. If the bar graph was defined as input with touch, this touch field will also be deleted. n2=0: Bar graph remains visible; n2=1: Bar graph is deleted

5.9 Macros

Run macros:

Run macro	#MN nr	Call the (normal) macro with the number nr (max. 7 levels) (see compiler option MACRO 18 :)
Run touch macros	#MT nr	Call the touch macro with the number nr (max. 7 levels) (see compiler option TOUCHMACRO 19 :)
Run menu macro	#MM nr	Call the menu macro with the number nr (max. 7 levels) (see compiler option MENUMACRO 20 :)
Run port macro	#MP nr	Call the port macro with the number nr (max. 7 levels) (see compiler option PORTMACRO 21 :)
Run bit macro	#MB nr	Call the bit macro with the number nr (max. 7 levels) (see compiler option BITMACRO 22 :)

(see How-to-use example Port Macro - BEGINNER 82)
(see How-to-use example Bit Macro - BEGINNER 80)

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 - BEGINNER 83)

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 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 are stopped with n1=0 and restarted with n1=1 in order, for example, to execute settings and outputs via the interface undisturbed

5.10 Touch

Touch presets:

Touch border form	#AE nr	Set the border n1 for the display of touch keys/switches (see internal border 47)
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 one 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 63)

Label font presets:

Label font	#AF nr	Set font with the number n1 (0 to 15) for touch key label (see compiler option `FONT` 10 `:` or `WINFONT` 11 `:)`
Label zoom factor	#AZ n1,n2	n1=X-zoom factor (1x to 4x); n2=Y-zoom factor (1x to 4x)
Additional line spacing	#AY n1	Insert n1=0..15 dots between two lines as additional spacing
Label angle	#AW n1	Label output angle: n1=0: 0°; n1=1: 90°

Define touch key/switch:

Define touch key	#AT x1,y1,x2,y2, downCode,upCode,"text.." #AU x,y, n1, 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, n1, 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: Image number n1 is loaded to x,y and defined as a key #AJ: Image number n1 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) This is followed by a string "text..." that is placed in the key with the current touch font Multiline texts are separated with the character '|' ($7C, dec: 124)		

(see How-to-use example 3 simple touch buttons - BEGINNER 61)

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 text (C=center,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 76)

Define touch areas:

Define drawing area	#AD x1,y1,x2,y2, n1	A drawing area is defined. You can then draw with a line width of n1 within the corner coordinates x1,y1 and x2,y2.
Define free touch area	#AH x1,y1,x2,y2	A freely usable touch area is defined. Touch actions (down, up and drag) within the corner coordinates x1,y1 and x2,y2 are sent.
Set bar by touch	#AB n1	The bargraph with number n1 is defined for input by touch panel.

(see How-to-use example Free draw area with clipboard - BEGINNER 66)
(see How-to-use example Bargraph by touch - BEGINNER 74)

Global settings:

Touch query on/off	#AA n1	Touch query is
		n1=0: deactivated
		n1=1: activated

Touch key response	#AI n1	Automatic inversion when touch key touched
		n1=0: OFF
		n1=1: ON

Touch key response buzzer	#AS n1	Tone sounds briefly when a touch key is touched
		n1=0: OFF
		n1=1: ON

Send bar value on/off	#AQ n1	Automatic transmission of a new bar graph value by touch input n1=0: deactivated n1=1: is placed in the sendbuffer once at the end of input n1=2: changes are placed continous into sendbuffer during input

Other touch functions:

Invert touch key	#AN code	The touch key with the assigned return code is inverted manually

Set touch switch	#AP code,n1	The status of the switch with the return code is changed to
		n1=0: OFF
		n1=1: ON

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

5.11 Menu

Settings for menu box/touch menu:

Set menu font	#NF n1	Set font with the number n1 (0 to 15) for menu display (see compiler option FONT 10 : or WINFONT 11 :)
Menu font zoom factor	#NZ n1,n2	n1=X-zoom factor (1x to 4x); n2=Y-zoom factor (1x to 4x)
Additional line spacing	#NY n1	Insert n1=0..15 dots between two menu items as additional spacing
Menu angle	#NW n1	Menu display angle: n1=0: 0°; n1=1: 90°
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'

(see How-to-use example Menue - BEGINNER 76)

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

5.12 I/O-Ports

Input-ports:

Read input port	#YR n1	n1=0: Read all input ports as binary value (to sendbuffer) n1=1..8: Read input port <n1> (1=H-level=VDD, 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

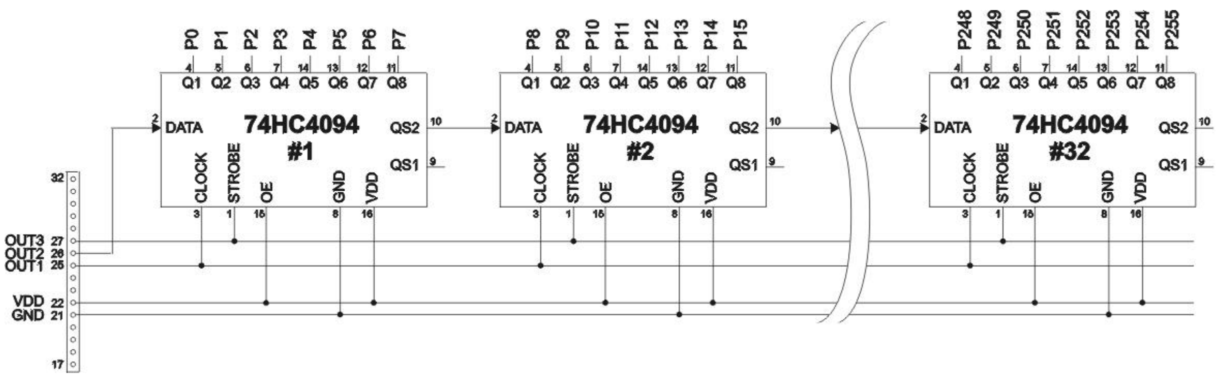
(see How-to-use example Port Macro - BEGINNER 82)
(see How-to-use example Bit Macro - BEGINNER 80)

Output-ports:

Define output port	#YM n1	n1=0: IN1..8 set to input (=default after power-on / reset) n1=1..8: n1 I/O-lines will be set to output (beginning at OUT1 upwards)
Write output port	#YW n1,n2	n1=0: Set all defined output ports in accordance with n2 (=binary value) n1=1..8: Reset output port n1 (n2=0); set (n2=1); invert (n2=2)

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
		(I/O-lines OUT1..OUT3 must be set to output: #YM 3)



5.13 Other commands

Send functions:

Send bytes	#SB data...	bytes are sent to the sendbuffer data... can be numbers or strings e.g #SB "Test",10,13
Send version	#SV	The version is sent as a string to sendbuffer e.g. "EA eDIP128-6 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.

LED backlight:

Illumination on/off	#YL n1	LED illumination n1=0: OFF; n1=1: ON; n1=2 to 255: illumination switched on for n1 tenths of a second.
Illumination brightness	#YH n1	Set brightness of the LED illumination n1=0 to 100%.
Brightness changetime	#YZ n1	Time n1=0..31 in 1/10sec for changing brightness from 0 to 100%
Set backlight PWM	#YF n1	Set the frequency of backlight PWM n1=0: low; n1=1: mid (default), n1=2: high;
Save parameter	#Y@	Save the actual brightness, changetime and PWM-frequency for poweron to EEPROM

Other functions:

Tone on/off	#YS n1	The tone output (pin 16) becomes n1=0:OFF; n1=1:ON; n1=2 to 255:ON for n1/10s
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 54)

Power down	#PD mode,n2	After this command, the display goes into power-down mode.
		mode=0: display invisible, LED disabled
		mode=1: display visible, LED disabled
		mode=2: display visible, LED enabled
		n2=0:Touch WakeUp is disabled
		n2=1:Touch WakeUp is enabled

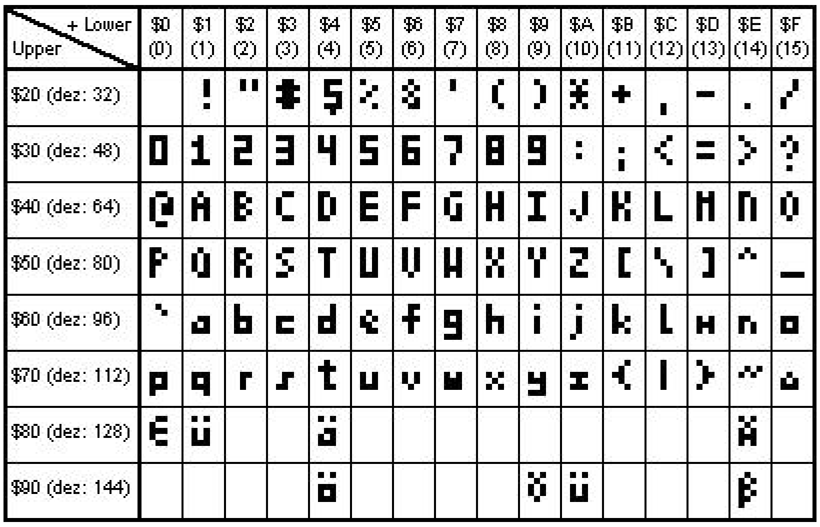
(see How-to-use example Energy save modes - BEGINNER 89)

6 Default Fonts

6.1 Terminal 8x8

+ 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)	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	¢	£	¥	β	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	β	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	³	—

6.2 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)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
$60 (dez: 96)	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
$70 (dez: 112)	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
$80 (dez: 128)	€	ü			ä										Ä	
$90 (dez: 144)					ö					Ö	Ü				ß	

6.3 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)	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	ß	ƒ
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
$F0 (dez: 240)	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	³	¯

6.4 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	¯

6.5 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	{	|	}	~	Δ
$80 (dez: 128)	€	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
$90 (dez: 144)	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü					
$A0 (dez: 160)	á	í	ó	ú	ñ	Ñ	ª	º								
$B0 (dez: 176)																
$C0 (dez: 192)																
$D0 (dez: 208)																
$E0 (dez: 224)		ß														
$F0 (dez: 240)									°							

6.6 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)									°							

6.7 Swiss 30

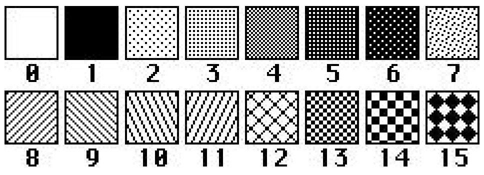
+ Lower Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X	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)		β														
$F0 (dez: 240)									°							

6.8 BigZif 57

+ 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	:					

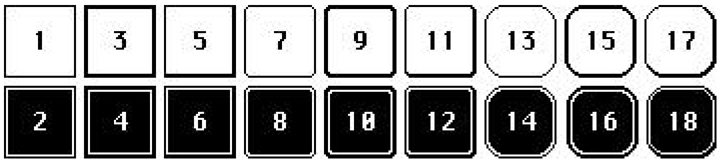
7 Internal Pattern

The internal pattern can be used with any command that uses patterns
e.g #RM,#RO 26, #QM 27, #BR,#BL,#BO,#BU 30



8 Internal Border

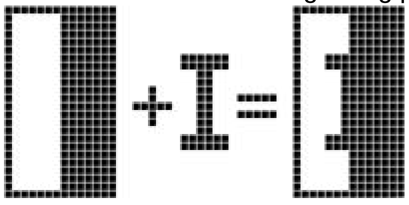
The internal border can be used with any command that uses borders
e.g #RR,#RT 26, #AE 32



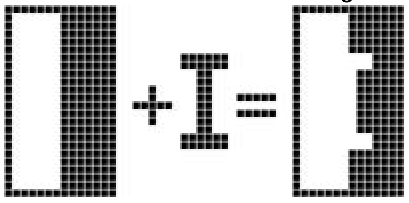
9 Link Modes

The Link Modes can be used with several commands
e.g #ZV 24, #GV 26, #UV 28

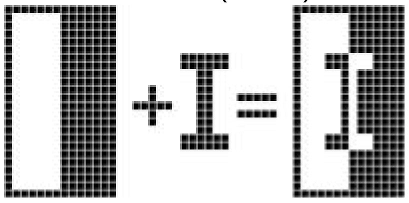
n1=1: set Pixel without regarding previous value (OR)



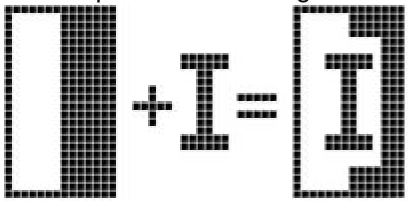
n1=2: delete Pixel without regarding previous value



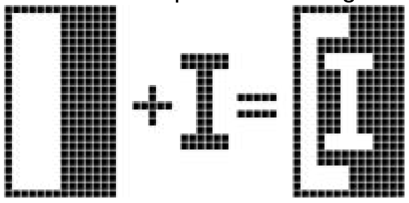
n1=3: invert Pixel (EXOR)



n1=4 replace: clear background and set Pixel



n1=5 invers replace: fill background and delete Pixel



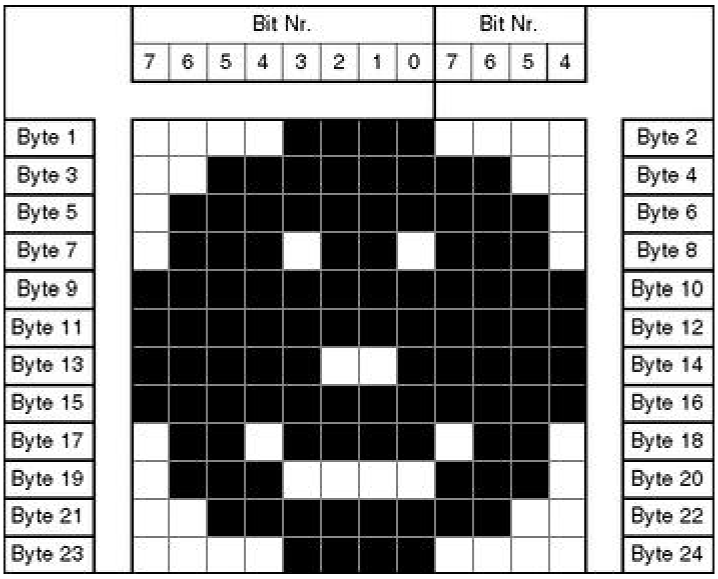
10 BLH format

Use 'BitmapEdit.exe' from the LCD-Tools package to edit/convert images into/from BLH-format.

Structure of an image file in the BLH-format:

description	**number of bytes**
image width	1
image height	1
image data	((width+7) / 8) * height The image is stored from top to down One byte stands for 8 horizontal pixels on the screen (MSB: left, LSB: right; 0=white, 1=black)

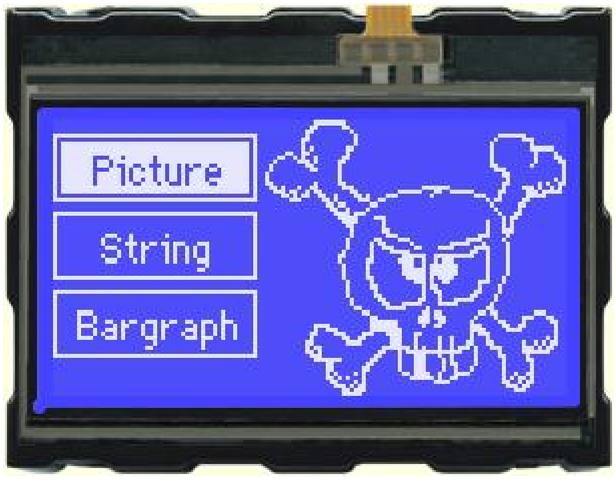
Example:
a small icon with 12x12 dots



The complete BLH-file:
```
$0C $0C
$0F $00 $3F $C0 $7F $E0 $76 $E0 $FF $F0 $FF $F0
$F9 $F0 $FF $F0 $6F $60 $70 $E0 $3F $C0 $0F $00
```

11 How-to-use

To find an easy start, you will find a project under "..\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\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\eDIP128-6\My first project

File:
my_first_project.kmc

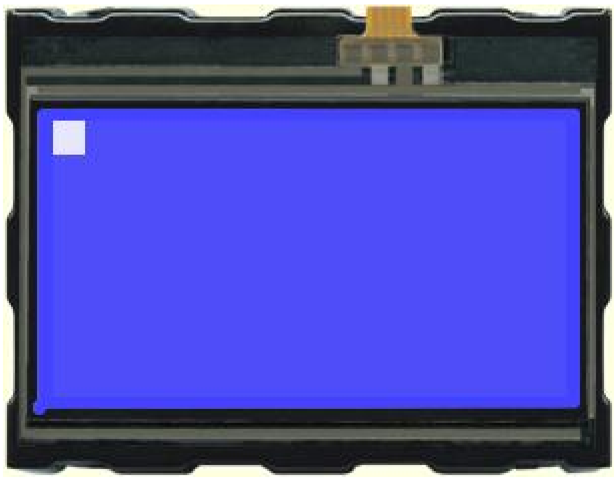
Commands:
#AT, #BR, #ZL, #UI

Open file in KitEditor

```
eDIP128-6   "First project"
...
...
...

;--------------------------------------------------------------------------------
;Include picture
Picture: 6 , <..\..\BITMAPS\monochrome\TOTKOPF.bmp> ;store as picture 6 (1-5 are used in
"default_pictures.kmi"

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

Macro: 0 ;define macro 0, called after power on, reset, watchdog reset
       #TA                  ; terminal off
       #AF GENEVA10         ; set touch label font, the font is defined in include file
"default_font.kmi"
       #AT 0, 3,50,18,1,0, "Picture"       ; place 3 touchbuttons at x1,y1 to x2,y2,
Touchmacro 1 is called
       #AT 0,21,50,36,2,0, "String" ; touchmacro 2 is called
       #AT 0,39,50,54,3,0, "Bargraph"      ; 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
       #RL 51,0,127,63      ; delete area on the right (to delete pixels of other
touchmacros)
       #UI 53,0, 6          ;load internal picture 6

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
       #RL 51,0,127,63       ; delete area on the right (to delete pixels of other
touchmacros)
       #ZF CHICAGO14         ;set font for strings (font is defined in "default_font.kmi")
       #ZC 90,10, "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
       #RL 51,0,127,63       ; delete area on the right (to delete pixels of other
touchmacros)
       #AQ 0                 ; deactivate sending barvalues into sendbuffer
       #BO 1,80,63,100,5,0,100,1,3 ; define bar 1 upwards, with statval 0 and endval 100,
pattern 3
       #BA 1,75              ; set bar 1 to new val of 75
       #AB 1                 ; set bar 1 with touch
```

11.1 Factory Setting

This macrofile sets the display back to factroy setting.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\Init\

File:
Init.kmc

Commands:

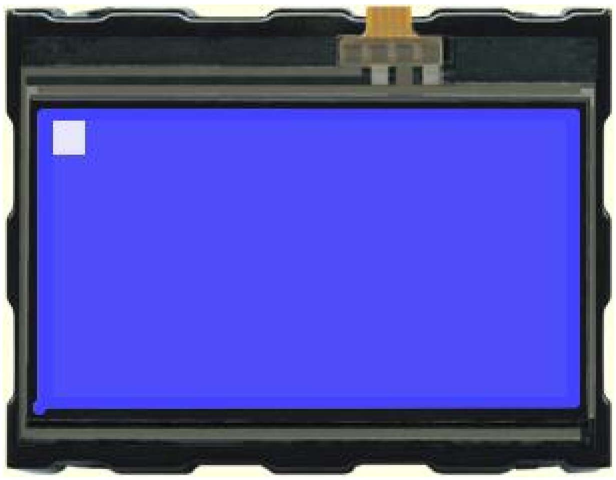
Open file in KitEditor

```
eDIP128-6   "init / delivery state"     ; define eDIP, "Projectname" max. 32 character
;brings the display back to ex-works condition with it's standard-fonts 1..7, standard-
pictures

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

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

;VERIFY                      ; verify after program

;-----------------------------------------------------------------------------
; load defaults (include data)
include <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>     ; the two dots mean an indirect path (one folder back,
like DOS-command)
include <..\default_pictures.kmi>

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

MnAutoStart = 0

PowerOnMacro:          ; runs after power-on
       #MN MnAutoStart

ResetMacro:            ; runs after external reset
       #MN MnAutoStart

WatchdogMacro: ; runs after a crash (>1000ms)
       #MN MnAutoStart

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


WakeupPinMacro:      ; runs after PowerDown and wakeup when pin13=WUP goes LO

WakeupTouchMacro:    ; runs after PowerDown and wakeup when touching the touchpanel

WakeupI2CMacro:      ; runs after PowerDown and wakeup receiving commands from I2C
```

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

Macro: MnAutoStart
```

11.2 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\eDIP128-6\Init\

File:
INIT_with_RS485_address.KMC

Commands:

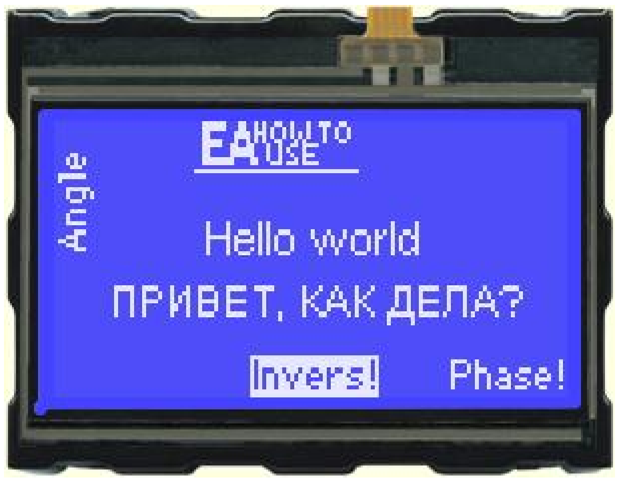
Open file in KitEditor

```
eDIP128-6   "init / delivery state"     ; define eDIP, "Projectname" max. 32 character
;brings the display back to ex-works condition with it's standard-fonts 1..7, standard-
pictures

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

;COM1: 115200                ; program eDIP on COMx with 115200 Baud
USB: 115200, "eDIP Programmer"      ; use EA 9777-USB eDIP Programmer and program eDIP
with 115200 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 data)
include <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>     ; the two dots mean an indirect path (one folder back,
like DOS-command)
include <..\default_pictures.kmi>

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

MnAutoStart = 0

PowerOnMacro:         ; runs after power-on
       #MN MnAutoStart

ResetMacro:           ; runs after external reset
       #MN MnAutoStart

WatchdogMacro: ; runs after a crash (>1000ms)
       #MN MnAutoStart

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

```
WakeupPinMacro:      ; runs after PowerDown and wakeup when pin13=WUP goes LO

WakeupTouchMacro:    ; runs after PowerDown and wakeup when touching the touchpanel

WakeupI2CMacro:      ; runs after PowerDown and wakeup receiving commands from I2C

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

Macro: MnAutoStart
       #KA newadr
```

11.3 Place Strings - BEGINNER

Place different strings with different fonts and orientation. There is a furhter EXPERT example available, containig information about text linking. Please have a look at EXPERT - text_linking.kmc 58.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Font\

File:
BEGINNER – fonts.kmc

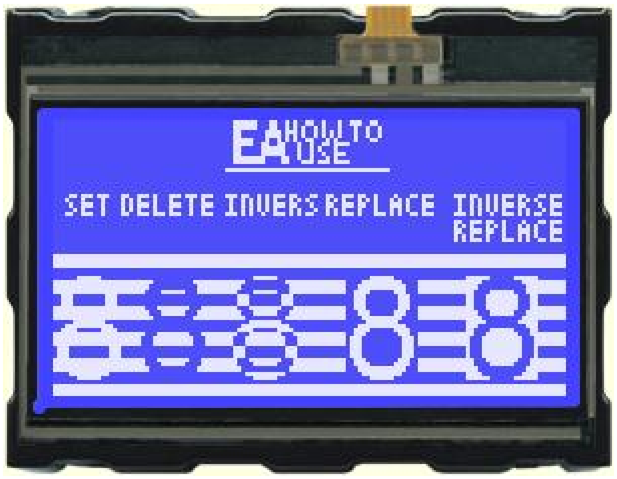
Commands:
WinFont, #ZL, #ZF,#ZW, #ZB

Open file in KitEditor

```
eDIP128-6   "Fonts"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>


; define Winfonts and create overview-file
ExportWinFont: 1      ; export all following WinFonts as *.fxt and save to the project
folder
ExportOverview: 1     ; export all following WinFonts as BMP-File and save to the project
folder
WinFont: 8, "Arial",0,0, 32,255, 14 ; doubleclick 'fontname' to edit
WinFont: 9, "Arial",204,0, 32-255, 12


;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                           ; Terminal off

       #UI 35,0,6                    ; place logo

;--- Place font examples---
       #ZF GENEVA10                  ; use font Geneva 10
                                     ; same as #ZF 4 (see default_font line 10)
       #ZZ 1,1                ; set font zoom factor to 1(x-axses), 1(y-axes) (default)
       #ZB 1                         ; text flashes
       #ZL 0,54, "Blink!"            ; place text left justified
       #ZB 2                         ; text flashes inversly
       #ZC 64,54, "Invers!"          ; place text center aligned
       #ZB 3                         ; text flashes phase shifted
       #ZR 128,54, "Phase!" ; place text right justified
       #ZB 0                         ; blink off
       #ZW 1                         ; change textangle to 90°
       #ZL 0,30, "Angle"             ; place text
       #ZW 0                         ; change textangle back to 0°
       #ZF 8                         ; use Windowsfont arial (see line 48)
       #ZC 64,20, "Hello world"      ; place text center aligned
       #ZF 9                         ; switch to cyrillic font
       ;#ZC 64,35,223," ",227,238,226,238,240,254," ",239,238,"
",240,243,241,241,234,232," !"      ; place cyrillic text
       #ZC 64,35, {CFD0C8C2C5D28220CAC0CA20C4C5CBC03F} ; character table: see file
"Font9_Arial_RUSSIAN_N_32-255_48.bmp"
                                                           ; double click between the
curly brackets to open EditBox for fonts
                                                           ; use mouse to select
characters
                                                           ; You have to select Font no.9
for EditBox to see the characters correctly
                                                           ; by clicking right on the
Fontname and "Select Font for EditBox"
```

11.4 Text linking - EXPERT

Show the differences between the five link modes. There is a furhter BEGINNER example available, containig information about placing strings. Please have a look at BEGINNER - fonts.kmc 56.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Font\

File:
EXPERT – text_linking.kmc

Commands:
#ZV

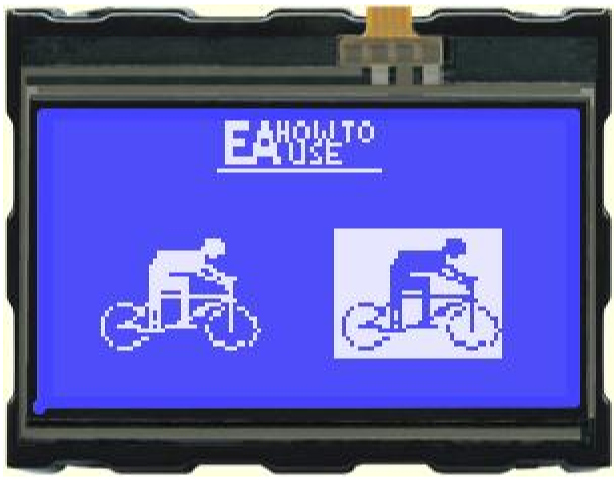
Open file in KitEditor

```
eDIP128-6   "Text linking"
...
...
...
;--------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier
Picture: LOGO <howtouselogo2.bmp>


; define font
ExportWinFont: 0      ; export all following WinFonts as *.fxt and save to the project
folder
ExportOverview: 0     ; export all following WinFonts as BMP-File and save to the project
folder
ARIAL = 8
WinFont: ARIAL, "Arial",0,0, 48,58, 24     ; doubleclick 'fontname' to edit, only numbers
(48-58 --> 0-9)


;--------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                    ; place logo

;--- Place stripes ---
xs=0
xe=XMAX
yh=2
pitch=4
ys=YMAX-6*yh-5*pitch
y=ys
       #RS xs,y,xe,y+yh
y+=yh+pitch
       #RS xs,y,xe,y+yh
y+=yh+pitch
       #RS xs,y,xe,y+yh
y+=yh+pitch
       #RS xs,y,xe,y+yh
y+=yh+pitch
       #RS xs,y,xe,y+yh
```

```
y+=yh+pitch
       #RS xs,y,xe,y+yh



;--- Place information ---
       #ZF FONT4x6                    ; use font Geneva 10
                                      ; same as #ZF 4 (see default_font line 10)
       #ZZ 1,1                 ; set font zoom factor to 1(x-axses), 1(y-axes) (default)
       #ZL 3, 17, "SET"                      ; place text
       #ZL 17, 17, "DELETE"                  ; place text
       #ZL 43,17, "INVERS"                   ; place text
       #ZL 68,17, "REPLACE"                  ; place text
       #ZR 128,17, "INVERSE|REPLACE"         ; place text




;--- Text link modes ---
       #ZF ARIAL               ; set text font (Arial see line 48)

       #ZV 1                          ; text link mode: 1 = set
       #ZL 0, 36, "8"          ; place text in the box
       #ZV 2                          ; text link mode: 2 = delete
       #ZL 21,36,"8"                  ; place text in the box
       #ZV 3                          ; text link mode: 3 = inverse
       #ZL 44,36, "8"          ; place text in the box
       #ZV 4                          ; text link mode: 4 = replace
       #ZL 73,36, "8"          ; place text in the box
       #ZV 5                          ; text link mode: 5 = inverse replace
       #ZR 120,36, "8"                ; place text in the box
```

11.5 BMP file - BEGINNER

Show simple pictures invertered and normal.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Picture\

File:
BEGINNER – bmp_file.kmc

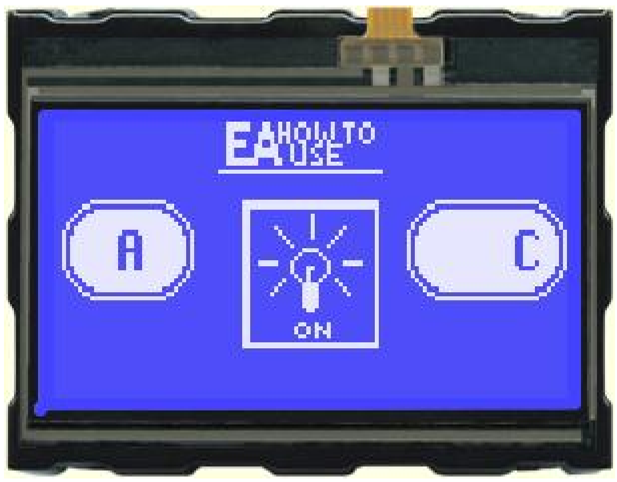
Commands:
#UI, #UV

Open file in KitEditor

```
eDIP128-6   "BMP-File"
...
...
...
;--------------------------------------------------------------------------------
Path <..\..\..\Bitmaps\monochrome>  ; define standard path
Picture: 6 <howtouselogo2.bmp>      ; define picture no. 6
Picture: 7 <cycling3.bmp>           ; define poicture no. 7

;--------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 41,0,6                   ; place logo

;--- Place Picture ---
       #UI 10,25, 7          ; place picture no. 6 at pixelpostion 10|20
       #UV 5                 ; next picture is placed inverse
       #UI 70, 25, 7         ; place same picture but inversed
```

11.6 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 74), Radiogroups (see BEGINNER - radiogroup.kmc 63) and another Example with touch buttons (see EXPERT - keypad.kmc 64)



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Touch\

File:
BEGINNER – 3 simple buttons.kmc

Commands:
#AU, #AT

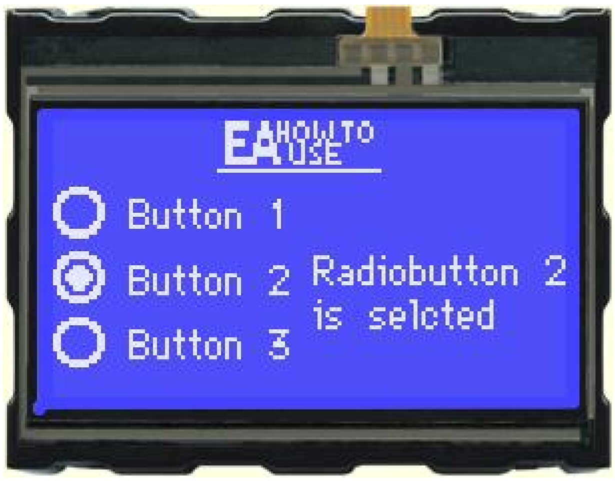
Open file in KitEditor

```
eDIP128-6   "3 simple buttons"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>
Picture: 7 <Button\Lamp34x34_1.bmp>,<Button\Lamp34x34_0.bmp>


;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                            ; Terminal off
       #UI 41,0,6                     ; place logo

;---- Place the left touch ----
       #AE 14                         ; set Frame style no. 14
       #AF 5                          ; set font no. 5 for Touch area
       #AT 2,18,34,40,65,0 "CA"       ;draw Touch area - this will put a $41 (65 dec.) into
send buffer
                                      ; the first "C" means left center aligned

;---- Place the middle touch as a bitmap ----
       #AF 5                          ; set font no. 5 for Touch area
       #AJ 47,18,7,1,2 ""             ; draw Touch area - as bitmap in this example without
text
                                      ; touch area is a switch

;---- Place the right touch as a bitmap ----
       #AF 5                                 ; set font no. 5 for Touch area
       #AT 88,18,128,40,67,68 "RC "          ; draw Touch area - this will run TouchMacro 67
(button down) and
                                             ; afterwards TouchMacro 68 (button up)
                                             ; the first "R" means rigth justify text

;---- Touch Macro for the middle touch (set) ----
TouchMacro: 1
       #YL 0          ;Backlight off
```

```
;---- Reset the middle touch ----
TouchMacro: 2
       #YL 1          ; Backlight on

;---- Touch Macro for the right touch ----
TouchMacro: 67
       #ZF GENEVA10
       #ZC 64,52, "#Macro 67, 'C' pressed"

;---- Release the right touch ----
TouchMacro: 68
       #RL 0,52,128,64      ; delete area
```

11.7 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 74), Buttons (see BEGINNER - 3 simple buttons.kmc 61) and another Example with touch buttons (see EXPERT - keypad.kmc 64)



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Touch\

File:
BEGINNER – radiogroup.kmc

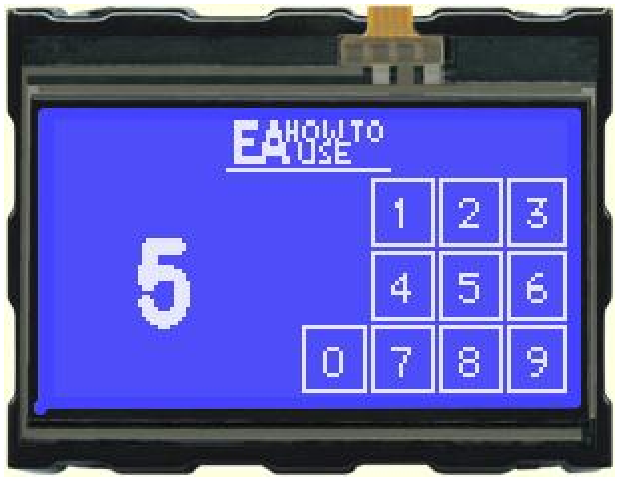
Commands:
#AR, #AJ

Open file in KitEditor

```
eDIP128-6   "Radiogroup"
...
...
...
;------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>
Picture: 7 <button\Radio60x12_0.bmp>,<button\Radio60x12_1.bmp>


;------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 41,0,6                   ; place logo

;---- Place radiobuttons ----
       #AR 1
       #AF GENEVA10                        ; define next buttons as radiogroup 1
       #AJ 0, 15, 7, 1,0,"RButton 1"       ; place radiobutton 1 with picture radiobutton
and downcode 1, 'R'= right aligned
       #AJ 0, 30, 7, 2,0,"RButton 2"       ; place radiobutton 1 with picture radiobutton
and downcode 2, 'R'= right aligned
       #AJ 0, 45, 7, 3,0,"RButton 3"       ; place radiobutton 1 with picture radiobutton
and downcode 3, 'R'= right aligned
       #AR 0                               ; next buttons do not belong to any radiogroup
       #AP 1,1                      ; activate radiobutton 1
       #MT 1                               ; call Touchmacro from radiobutton 1

;------------------------------------------------------------------------------
TouchMacro: 1
       #ZF GENEVA10                 ; use textfont no. 4
       #ZL 65,30,"Radiobutton 1|is selcted"
TouchMacro: 2
       #ZF GENEVA10                 ; use textfont no. 4
       #ZL 65,30,"Radiobutton 2|is selcted"
TouchMacro: 3
       #ZF GENEVA10                 ; use textfont no. 4
       #ZL 65,30,"Radiobutton 3|is selcteded"
```

11.8 Keypad - EXPERT

Place a keypad (0..9). There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 74), Buttons (see BEGINNER - 3 simple buttons.kmc 61) and Radio groups (see BEGINNER - radiogroup.kmc 63).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Touch\

File:
EXPERT – keypad.kmc

Commands:
#AT

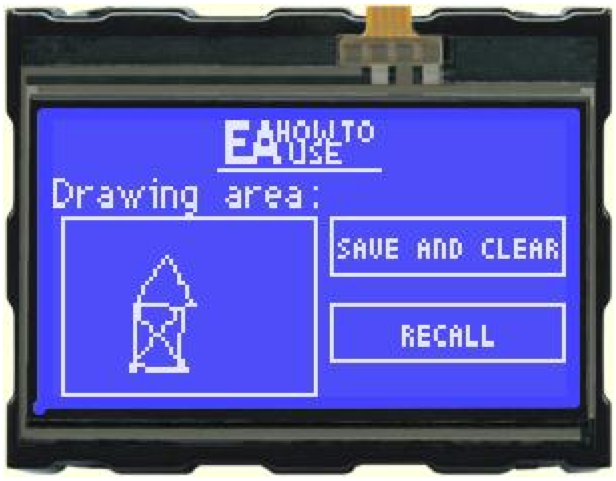
Open file in KitEditor

```
eDIP128-6   "Keypad"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using contsntas makes it easier
Picture: LOGO, <howtouselogo2.bmp>

;-------------------------------------------------------------------------------
;define constants for Touchmaros
KB1 = 1
KB2 = KB1+1
KB3 = KB2+1
KB4 = KB3+1
KB5 = KB4+1
KB6 = KB5+1
KB7 = KB6+1
KB8 = KB7+1
KB9 = KB8+1
KB10= KB9+1


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

Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                    ; place logo

;--- Place Keypad ---
       #AF Geneva10
;define some constants to place buttons easily
xw = 15
yh = xw
pitch = 2
xs = XPIXEL - 3*xw - 2*pitch
ys = PICTURE_H(LOGO)+1
x=xs
y=ys
       #AT x,y,x+xw,y+yh, KB1, 0, "1"
```

```
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB2, 0, "2"
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB3, 0, "3"
x=xs
y+=yh+pitch
       #AT x,y,x+xw,y+yh, KB4, 0, "4"
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB5, 0, "5"
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB6, 0, "6"
x=xs
y+=yh+pitch
       #AT x,y,x+xw,y+yh, KB7, 0, "7"
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB8, 0, "8"
x+=xw+pitch
       #AT x,y,x+xw,y+yh, KB9, 0, "9"
x=xs-xw-pitch
       #AT x,y,x+xw,y+yh, KB10, 0, "0"

       ;define Fonts for Touchmacros (because using the same in all touchmacros
       #ZF SWISS30B         ; define Font


;--------------------------------------------------------------------------------
;Show the numbers which are pressed
x=20
y=25
TouchMacro: KB1
       #ZL x,y, "1"  ; write number
TouchMacro: KB2
       #ZL x,y, "2"
TouchMacro: KB3
       #ZL x,y, "3"
TouchMacro: KB4
       #ZL x,y, "4"
TouchMacro: KB5
       #ZL x,y, "5"
TouchMacro: KB6
       #ZL x,y, "6"
TouchMacro: KB7
       #ZL x,y, "7"
TouchMacro: KB8
       #ZL x,y, "8"
TouchMacro: KB9
       #ZL x,y, "9"
TouchMacro: KB10
       #ZL x,y, "0"
```

11.9 Free draw area with clipboard - 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_clipboard.kmc 67



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Draw\

File:
BEGINNER – free_draw_area_clipboard.kmc

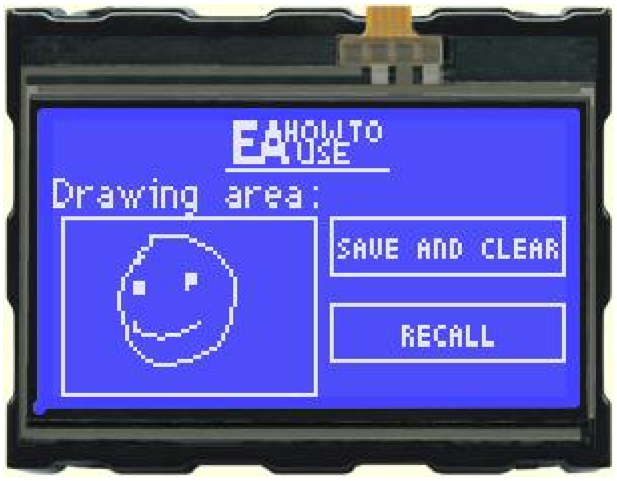
Commands:
#AD

Open file in KitEditor

```
eDIP128-6   "Free drawing area with clipboard"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>        ; define search path
Picture: 6 <howtouselogo2.bmp>                    ; add new picture (logo)

;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 41,0,6                   ; place logo

;--- Place information ---
       #ZF GENEVA10                 ; set font no.4 (see default_font.kmi -> line 13)
       #ZC 0, 12,"Drawing area:"
;---- Place buttons ----
       #AF 1                        ; set font no. 1 for Touch area
       #AE 5                        ; set touchframe
       #AT 69,22,128,35,1,0, "SAVE AND CLEAR"     ; place touchbutton 1
       #AT 69,42,128,55,2,0, "CRECALL"            ; place touchbutton 2

;---- Place drawing area ----
       #GR 2,22,65,64; place rectangle around drawing area
       #AD 3,23,64,63,1      ; place drawing area, linewith 1



;-----------------------------------------------------------------------------
TouchMacro: 1
       #CS 3,23,64,62; drawing area is copied to the clipboard
       #RL 3,23,64,62; clear drawing area

TouchMacro: 2
       #CR                   ; copy clipboard back to the display
```

11.10 Free draw area with clipboard - 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_clipboard.kmc 66



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Draw\

File:
EXPERT – free_draw_area_clipboard.kmc

Commands:
#AD

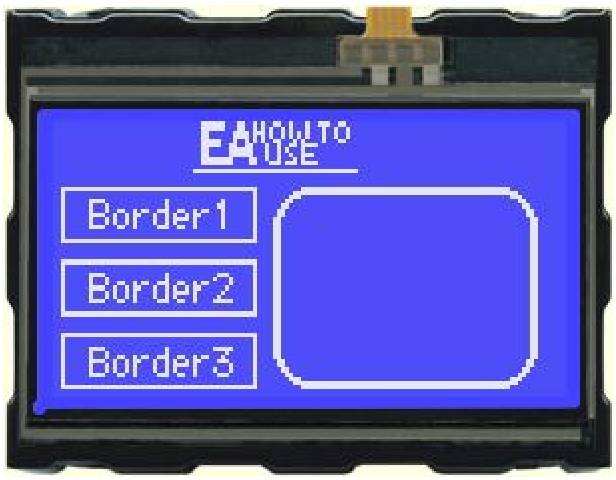
Open file in KitEditor

```
eDIP128-6   "Free drawing area with clipboard"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>        ; define search path
LOGO = 6 ; using constants makes it easier
Picture: LOGO <howtouselogo2.bmp>                ; add new picture (logo)

;-------------------------------------------------------------------------------
;Define constants for Touchmacros
CLEAR = 1
RECALL = 2
;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                     ; place logo

;--- Place information ---
       #ZF GENEVA10                ; set font no.4 (see default_font.kmi -> line 13)
       #ZL 0, PICTURE_H(LOGO),"Drawing area:"
;---- Place buttons ----
xs=69
xw=XMAX-xs
ys=22
yh=13
pitch=7
       #AF FONT4x6                        ; set font no. 1 for Touch area
       #AE 5                       ; set touchframe
       #AT xs,ys,xs+xw,ys+yh,CLEAR,0, "SAVE AND CLEAR"   ; place touchbutton 1
ys+=pitch+yh
       #AT xs,ys,xs+xw,ys+yh,RECALL,0, "CRECALL"         ; place touchbutton 2

;---- Place drawing area ----
da_xs = 2
da_xe = 65
da_ys = 22
da_ye = YMAX
line=1
       #GR da_xs,  da_ys,  da_xe,  da_ye          ; place rectangle around drawing area
```

```
        #AD da_xs+1,da_ys+1,da_xe-1,da_ye-1,line   ; place drawing area, linewith 1



;--------------------------------------------------------------------------------
TouchMacro: CLEAR
        #CS da_xs+1,da_ys+1,da_xe-1,da_ye-1 ; drawing area is copied to the clipboard
        #RL da_xs+1,da_ys+1,da_xe-1,da_ye-1 ; clear drawing area

TouchMacro: RECALL
        #CR                  ; copy clipboard back to the display
```

11.11 Frame - BEGINNER

Show the different borders.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Frame\

File:
BEGINNER – frame.kmc

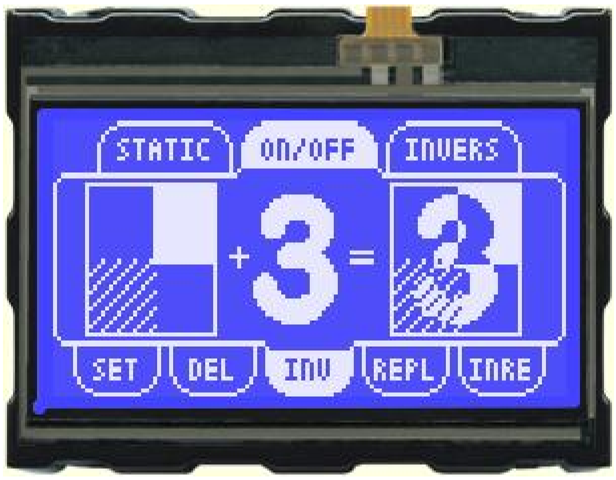
Commands:
#RT, #RR

Open file in KitEditor

```
eDIP128-6 "Different Borders"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>
;-----------------------------------------------------------------------------

Macro: MnAutoStart

;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                            ; Terminal off
       #UI 35,0,6                     ; place logo

;--- Place 3 Buttons to select different Boarders ---
       ; use default paramters for Border, color and font of Touchbuttons
       #AF GENEVA10                          ;use Geneva 10 as Touchbutton font
       #AT 2, 15,50,27,1,0,"Border1"
       #AT 2,32,50,44,2,0,"Border2"
       #AT 2,49,50,61,3,0,"Border3"


       #MT 1  ; run a TouchMacro to show something on the screen at startup



TouchMacro 1: ; Called by Button Border1
       #MN 1                          ; call Macro 1 (delete area to draw new frames)
       #RR 55,15,120,61,15            ; draw new frame with border no. 15


TouchMacro 2:  ;Called by Button Border2
       #MN 1
       #AE 18                         ; set touchframe no. 18
       #AT 55,20,128,35,4,0,"Border-Button" ; define button

TouchMacro 3:  ;Called by Button Border3
       #MN 1
       #BR 1,55,20,125,35,0,100,1,5 ; 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
       #ZF GENEVA10                        ; switch textfont to Geneva 10 (same as #ZF 4)
       #ZL 58,40,"Bargraph with|fill-pattern"     ; place info-text
                                           ; '|' means new line


Macro 1: ; Draw Rectangel with selected Border
       #RL 53,15,128,64      ; delete area, to draw
       #AV 55,20,0           ; delete old touchareas (Bargraph and border button)


TouchMacro 4: ; called by  Boder-Button
       #ZF GENEVA10                               ; switch textfont to Geneva 10 (same as
#ZF 4)
       #ZL 58,40,"Border Button|was pressed"      ; place text, that Border-Button was
touched
                                                  ; '|' means new line
```

11.12 GrafikModes - EXPERT

Show different link modes and flashing modes.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\GrafikMode\

File:
EXPERT – GrafikModeRegister.kmc

Commands:
#ZV, #ZB, #UV, #UB

Open file in KitEditor

```
eDIP128-6   "Grafik-Blinkmode"
...
...
...
;-------------------------------------------------------------------------------
;Include pictures

BMP3 = 10
ButRegBottom_Small    = 12
ButRegTop_Small       = 14

Picture: BMP3 <.\3.bmp>

Path: <..\..\..\BITMAPS\monochrome\button\>
Picture: ButRegBottom_Small <RegisterBottom24x13_0.bmp>,<RegisterBottom24x13_1.bmp>
Picture: ButRegTop_Small    <RegisterTop36x13_0.bmp>,<RegisterTop36x13_1.bmp>


;Include fonts
BIG3 = 8
FONT: BIG3 <.\3.fxt>

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

;define constants for touch-macros
TmNoblink1 = 1
TmBlinkonoff1 = 2
TmBlinkinvers1 = 3



TmOr = 10
TmDelete = TmOr+1
TmExor = TmDelete+1
TmReplace = TmExor+1
TmInvRepl = TmReplace+1

;-------------------------------------------------------------------------------
Makro: MnAutoStart
       #TA            ;terminal off
       #DL            ;clear screen
       #AL 0,0       ;delete all touchareas

; draw the big box in the middle of the screen

yu=YPIXEL-PICTURE_H(ButRegBottom_Small)
       #RT 0,PICTURE_H(ButRegTop_Small)-1,XPIXEL,yu, 7 ; draw rounded box

; draw registers on the top
x=(XPIXEL-PICTURE_W(ButRegTop_Small)*3)/2 ; offset because rounded box in the middle
y=0 ; start at the top
b=PICTURE_W(ButRegTop_Small) ; width of button
       #AF FONT4x6     ; select touch label font
       #AR 1           ; start radio group 1 (top)
       #AJ x,y,ButRegTop_Small, TmNoblink1,0,     "STATIC"
       #AP TmNoblink1,1 ; preset button of radiogroup 1
x+=b
       #AJ x,y,ButRegTop_Small, TmBlinkonoff1,0,  "ON/OFF"
x+=b
       #AJ x,y,ButRegTop_Small, TmBlinkinvers1,0, "INVERS"
x+=b
       #AR 0 ; end of radio group 1

; draw registers at the bottom
x=(XPIXEL-PICTURE_W(ButRegBottom_Small)*5)/2 ; offset because rounded box in the middle
y=yu
b=PICTURE_W(ButRegBottom_Small)
       #AR 2
       #AJ x,y,ButRegBottom_Small, TmOr,0,     "SET"
       #AP TmOr,1
x+=b
       #AJ x,y,ButRegBottom_Small, TmDelete,0,  "DEL"
x+=b
       #AJ x,y,ButRegBottom_Small, TmExor,0, "INV"
x+=b
       #AJ x,y,ButRegBottom_Small, TmReplace,0,  "CREPL"  ;output string c=centered,
because r=right justified
x+=b
       #AJ x,y,ButRegBottom_Small, TmInvRepl,0,  "INRE"
       #AR 0 ; end of radio group 2

       #ZV REPLACE ; set the text modes like the preset values
       #ZB NOBLINK
       #UV REPLACE
       #UB NOBLINK

; draw the inner two boxes and the 3
x1=8     ; start of first graphic box
w=32     ; witdh of graphic box
x2=x1+w  ; end of first graphic box
y1=PICTURE_H(ButRegTop_Small)+1 ; start of first graphic box, +2 because of distance to
register-buttons
h=35 ; height of graphic box
y2=y1+h ; end of first graphic box
x3=84    ; start of second picture
xm=x1+w/2 + 1 ; calculate middle of graphic box,
ym=y1+h/2 + 1 ; for drawing of patterns
       #RM x1,ym,xm,y2,8
       #RM xm,y1,x2,ym,1
       #GR x1,y1,x2,y2

       #CS x1,y1,x2,y2
; place two pictures
       #UI x2+9,y1+2, BMP3
; place the mathematical operators
       #ZF CHICAGO14
       #ZL x2+3,y1+11,"+"
       #ZR x3-4,y1+11,"="
; place the 3 as font
       #ZF BIG3
       #MT TmReplace

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

Makro: MnDraw ; drawing of the both "3" (as picture and font)
       #CK x3,y1 ; place the clipboard back to overwrite old content
       #ZL x3+5,y1+2, "3" ; place picture with flahing-attributes of picture
```

```
;-------------------------------------------------------------------------------------------
--
;Touchmacros of set-modes for pictures and fonts
TouchMakro: TmOr
  #ZV SET
  #UV SET
  #MN MnDraw

TouchMakro: TmDelete
  #ZV DELETE
  #UV DELETE
  #MN MnDraw

TouchMakro: TmExor
  #ZV INVERS
  #UV INVERS
  #MN MnDraw

TouchMakro: TmReplace
  #ZV REPLACE
  #UV REPLACE
  #MN MnDraw

TouchMakro: TmInvRepl
  #ZV INVREPL
  #UV INVREPL
  #MN MnDraw

;-------------------------------------------------------------------------------------------
--
;Touchmacros of blinc-modes for fonts
TouchMakro: TmNoblink1
  #ZB NOBLINK
  #MN MnDraw

TouchMakro: TmBlinkonoff1
  #ZB BLINKONOFF
  #MN MnDraw

TouchMakro: TmBlinkinvers1
  #ZB BLINKINVERS
  #MN MnDraw
```

11.13 Bargraph by touch - BEGINNER

Place a bargraph, that is adjustable by touch. There is a an EXPERT example available, too. Please have a look at EXPERT - Bargraph by touch 75. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 61.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Bargraph\

File:
BEGINNER – bargraph_by_touch.kmc

Commands:
#BR

Open file in KitEditor

```
eDIP128-6   "Bargraph adjusted by touch"
...
...
...

;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>

;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI 43,0,6                  ; place logo

;---- Place text ----
       #ZF GENEVA10                       ; select font
       #ZC 64,15, "Bargraph adjusted|by touch"    ; place text



;---- Place bargraph ----
       #BR 1,5,40,123,60,0,100,1,6        ; place bargraph no. 1 with pattern 6
       #AB 1                              ; set bargraph by touch
       #BA 1, 73                          ; update bar no.1 with new value
       #AQ 0                              ; do not send bar values (send buffer overrun
occurs if 1)
       ;#BS 1                             ; send barvalue of bargraph no. 1, be sure,
                                          ; that the sendbuffer is called regularly
```

11.14 Bargraph by touch - EXPERT

Place a bargraph, that is adjustable by touch. There is a a BEGINNER example available, too. Please have a look at BEGINNER - bargraph_by_touch.kmc 74. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 61.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Bargraph\

File:
EXPERT – bargraph_by_touch.kmc

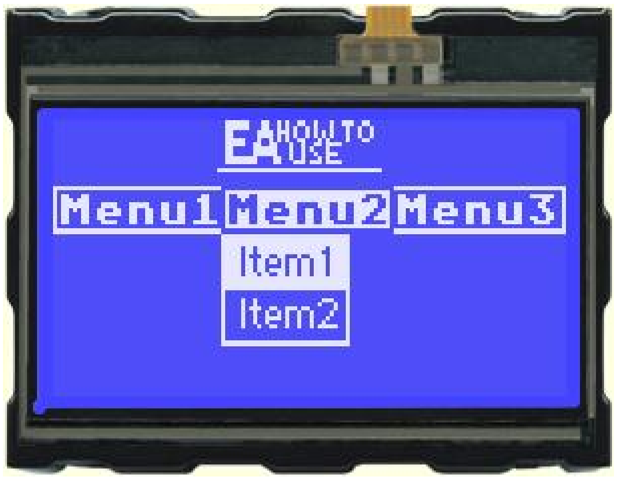
Commands:
#BR

Open file in KitEditor

```
eDIP128-6   "Bargraph adjusted by touch"
...
...
...

;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ;use constants
Picture: LOGO <howtouselogo2.bmp>

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                    ; place logo, using
compiler-functions ans constants

;---- Place text ----
       #ZF GENEVA10                       ; select font
       #ZC XPIXEL/2,PICTURE_H(LOGO)+3, "Bargraph adjusted|by touch"    ; place text



;---- Place bargraph ----
;Bargraph conbstants:
xs=5
xw=XPIXEL-2*xs
ys=45
yh=15
sv=0
ev=100
typ=1
pat=6
       #BR 1,xs,ys,xs+xw,ys+yh,sv,ev,typ,pat             ; place bargraph no. 1 with
pattern 6
       #AB 1                              ; set bargraph by touch
       #BA 1, 73                          ; update bar no.1 with new value
       #AQ 0                              ; do not send bar values (send buffer overrun
occurs if 1)
       ;#BS 1                             ; send barvalue of bargraph no. 1, be sure,
                                          ; that the sendbuffer is called regularly
```

11.15 Menue - BEGINNER

Show a menu, operable by touch. There is a an EXPERT example available, too. Please have a look at EXPERT - menue.kmc 78.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Menue\

File:
BEGINNER – menue.kmc

Commands:
#AM

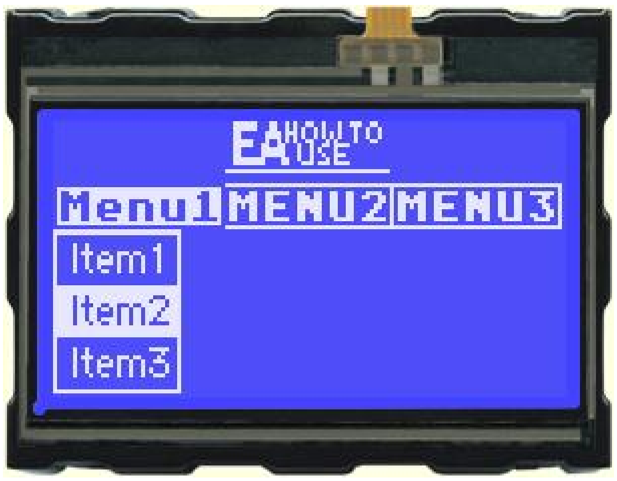
Open file in KitEditor

```
eDIP128-6   "Menue"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                        ; Terminal off
       #UI 41,0,6                 ; place logo
       #ZF GENEVA10

;--- Place Menue ---
       #NF GENEVA10               ; set Menue font to Geneva10
                                  ; same as #NF 4
       #NY 2                      ; adding 2 additional dots between two menue items
       #NW 0                      ; menu angle
       #NT 1                      ; touch menu opens automatically

       #AM 0,15,42,25,0,0,10, "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 42,15,84,25,0,0,20, "UCMenu2|Item1|Item2"                  ; place Menu2,
opening down (U), text centered (C)
                                                                       ; MenuMacro
20+Itemnumber is called
       #AM 84,15,128,25,0,0,30, "UCMenu3|Item1|Item2|Item3|Item4"     ; place Menu3,
opening down (U), text centered (C)
                                                                       ;MenueMacro
30+Itemnumber is called

;-------------------------------------------------------------------------------
MenueMacro: 10 ;Menu 1 Item 1
       #ZL 2,35,"Selected:|Menue 1, Item 1" ; place text

MenueMacro: 11 ;Menu 1 Item 2
```

```
        #ZL 2,35,"Selected:|Menue 1, Item 2"

MenueMacro: 12;Menu 1 Item 3
        #ZL 2,35,"Selected:|Menue 1, Item 3"

;-------------------------------------------------------------------------------
MenueMacro: 20;Menu 2 Item 1
        #ZL 2,35,"Selected:|Menue 2, Item 1" ; place text

MenueMacro: 21;Menu 2 Item 2
        #ZL 2,35,"Selected:|Menue 2, Item 2"

;-------------------------------------------------------------------------------
MenueMacro: 30;Menu 3 Item 1
        #ZL 2,35,"Selected:|Menue 3, Item 1" ; place text

MenueMacro: 31;Menu 3 Item 2
        #ZL 2,35,"Selected:|Menue 3, Item 2"

MenueMacro: 32;Menu 3 Item 3
        #ZL 2,35,"Selected:|Menue 3, Item 3"

MenueMacro: 33;Menu 3 Item 3
        #ZL 2,35,"Selected:|Menue 3, Item 4"
```

11.16 Menue - EXPERT

Show a menu, operable by touch. There is a a BEGINNER example available, too. Please have a look at BEGINNER - menu.kmc 76.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Menue\

File:
EXPERT – menue.kmc

Commands:
#AM

Open file in KitEditor

```
eDIP128-6   "Menue"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ;using constants makes it easier
Picture: LOGO <howtouselogo2.bmp>

;Define string constants for menue
!MENU1! = "UCMenu1|Item1|Item2|Item3"
!MENU2! = "UCMenu2|Item1|Item2"
!MENU3! = "UCMenu3|Item1|Item2|Item3|Item4"
;Define string constants for info text
!SELECT! = "Selected:|Menue "

;-------------------------------------------------------------------------------
;define MenueMacros
MEN1 = 10
MEN2 = 20
MEN3 = 30

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

Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                  ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO             ; place logo
       #ZF GENEVA10

;--- Place Menue ---
       #NF GENEVA10         ; set Menue font to Geneva10
                            ; same as #NF 4
       #NY 2                ; adding 2 additional dots between two menue items
       #NW 0                ; menu angle
       #NT 1                ; touch menu opens automatically

;Constants for menu
xs=0
xb=42
ys=PICTURE_H(LOGO)+2
yh=10
```

```
x=xs
pitch=0
       #AM x,ys,x+xb,ys+yh,0,0,MEN1, !MENU1!            ; place Menu1, opening down (U),
text centered (C)
x+=xb+pitch                                             ; MenuMacro 10+Itemnumber is
called
                                                        ; Item1 has no. 0, Item2 has
1...
       #AM  x,ys,x+xb,ys+yh,0,0,MEN2, "MENU2"           ; place Menu2, opening down (U),
text centered (C)
x+=xb+pitch                                             ; MenuMacro 20+Itemnumber is
called
       #AM x,ys,x+xb,ys+yh,0,0,MEN3, "MENU3"            ; place Menu3, opening down (U),
text centered (C)
                                                        ;MenueMacro 30+Itemnumber is
called

;-------------------------------------------------------------------------------
x=2
y=35
MenueMacro: MEN1+0    ;Menu 1 Item 1
       #ZL x,y,!SELECT! "1, Item 1" ; place text

MenueMacro: MEN1+1    ;Menu 1 Item 2
       #ZL 2,35,!SELECT! "1, Item 2"

MenueMacro: MEN1+2    ;Menu 1 Item 3
       #ZL x,y,!SELECT! "1, Item 3"

;-------------------------------------------------------------------------------
MenueMacro: MEN2      ;Menu 2 Item 1
       #ZL x,y,!SELECT! "2, Item 1" ; place text

MenueMacro: MEN2+1    ;Menu 2 Item 2
       #ZL x,y,!SELECT! "2, Item 2"

;-------------------------------------------------------------------------------
MenueMacro: MEN3+0    ;Menu 3 Item 1
       #ZL x,y,!SELECT! "3, Item 1" ; place text

MenueMacro: MEN3+1    ;Menu 3 Item 2
       #ZL x,y,!SELECT! "3, Item 2"

MenueMacro: MEN3+2    ;Menu 3 Item 3
       #ZL x,y,!SELECT! "3, Item 3"

MenueMacro: MEN3+3    ;Menu 3 Item 3
       #ZL x,y,!SELECT! "3, Item 4"
```

11.17 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 BEGINNER - AutomaticMacro_as_animation.kmc 83), PortMacros (see BEGINNER - Port Macro.kmc 82) and another AutomaticMacro example (see EXPERT - Automatic_Macro.kmc 85)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Macro\

File:
BEGINNER – BitMacro.kmc

Commands:
#YW, YM

Open file in KitEditor

```
eDIP128-6  "Bit Macro"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI 41,0,6                  ; place logo

;--- Place 1 button ---
       #AF 4                       ; touch font
       #AE 4                       ; touch frame
       #AT 20,15,108,30,1,0,"Port 1 toggle"      ; place touchbutton with touchmacro no.
1

;--- Port settings ---
       #YM 1   ; Set I/O Line 1 as output
       #YA 1   ; activate automatic scan of the input ports


;-------------------------------------------------------------------------------
TouchMacro: 1
       #YW 1, 2      ; toggle output 1

;-------------------------------------------------------------------------------
BitMacro: 10   ; I/O line 2, rising edge
       #ZL 2,40,"Input 2 is high"

BitMacro: 2    ; I/O line 2, falling edge
       #ZL 2,40,"Input 2 is low  "

BitMacro: 11   ; I/O line 3, rising edge
       #ZL 2,40,"Input 3 is high"
```

```
BitMacro: 3    ; I/O line 3, falling edge
       #ZL 2,40,"Input 3 is low  "

PortMacro: 0
       #DL
```

11.18 Port Macro - BEGINNER

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 BEGINNER - AutomaticMacro_as_animation.kmc) 83, BitMacros (see BEGINNER - BitMacro.kmc 80) and another AutomaticMacro example (see EXPERT - Automatic_Macro.kmc 85)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Macro\

File:
BEGINNER – Port Macro.kmc

Commands:
#YW, YM

Open file in KitEditor

```
eDIP128-6  "Port Macro"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                              ; Terminal off
       #UI 41,0,6                       ; place logo

;--- Place 1 button ---
       #AF 4                            ; touch font
       #AE 5                            ; touch frame
       #AT 20,15,108,30,1,2,"Out 1 and 2"  ; place touchbutton with touchmacro no. 1 (set
output) and 2 (reset output)

;--- Port settings ---
       #YM 2   ; Set I/O Line 1 and 2 as output
       #YA 1   ; activate automatic scan of the input ports


;-------------------------------------------------------------------------------
TouchMacro: 1
       #YW 0, $03     ; set output-ports 1 and 2 (=0000 0011b)

TouchMacro: 2
       #YW 0, $00     ; reset output-ports 1 and 2

;-------------------------------------------------------------------------------
PortMacro: $27 ; I/O line 1,2 and 5, rising edge (=0001 0011b)
       #ZL 1,40,"Bit-pattern 0x27"
PortMacro: $3F ; I/O line 1,2 and 5, rising edge (=0001 0011b)
       #RL 0,40,XMAX,YMAX
```

11.19 Automatic Macro - BEGINNER

A little animation with the help of automatic macros. There are further examples available, containig information about BitMacros (see BEGINNER – BitMacro.kmc 80), PortMacros (see BEGINNER - Port Macro.kmc 82) and another AutomaticMacro example (see EXPERT - Automatic_Macro.kmc 85)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Macro\

File:
BEGINNER – AutomaticMacro_as_animation.kmc

Commands:
#MJ, #UI

Open file in KitEditor

```
eDIP128-6   "Automatic Macro as animation"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>
Picture: 7 <Kopf\kopf1.bmp>         ; include pictures for animation
Picture: 8 <Kopf\kopf2.bmp>
Picture: 9 <Kopf\kopf3.bmp>
Picture: 10 <Kopf\kopf4.bmp>
Picture: 11 <Kopf\kopf5.bmp>



;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off

       #UI 41,0,6                   ; place logo

;---- Start animation ----
       #UV 5                        ; invert pictures
       #MJ 1,5,3                    ; run macros 1..6 automatically
                                    ; MJ = Ping Pong Mode, Pause is 3/10s

;---- Place Digit in different Macros -----

Macro: 1
       #UI 41, 19, 7  ; place picture

Macro: 2
       #UI 41, 19, 8

Macro: 3
       #UI 41, 19, 9

Macro: 4
       #UI 41, 19, 10
```

```
Macro: 5
       #UI 41, 19, 11
```

11.20 Automatic Macro - EXPERT

A little animation with the help of automatic macros. There are further examples available, containig information about BitMacros (see BEGINNER – BitMacro.kmc 80), PortMacros (see BEGINNER - Port Macro.kmc 82) and another AutomaticMacro example (see BEGINNER - AutomaticMacro_as_animation.kmc 83)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Macro\

File:
EXPERT – Automatic_Macro.kmc

Commands:
#YW, #YM

Open file in KitEditor

```
eDIP128-6   "Automatic Macro"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier
Picture: LOGO <howtouselogo2.bmp>   ; include logo

BIGZIF50 = 8
Path:  <..\..\..\FONTS>             ; default search path
Font: BIGZIF50, <BIGZIF50.fxt>             ; include font

;-------------------------------------------------------------------------------
;Define Macros
MACRO1 = 1
MACRO2 = MACRO1+1
MACRO3 = MACRO2+1
MACRO4 = MACRO3+1
MACRO5 = MACRO4+1
MACRO6 = MACRO5+1

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

Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off

       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                    ; place logo

;---- Count up and down ----
       #ZF BIGZIF50                        ; set font to no. 6 (SWISS30B)
time=5
       #MJ MACRO1,MACRO6,time                     ; run macros 1..6 automatically
                                    ; MJ = Ping Pong Mode

;---- Place characters in different Macros -----
x=64
y=13
```

```
Macro: MACRO1
       #ZC x,y, "1"

Macro: MACRO2
       #ZC x,y, "2"

Macro: MACRO3
       #ZC x,y, "3"

Macro: MACRO4
       #ZC x,y, "4"

Macro: MACRO5
       #ZC x,y, "5"

Macro: MACRO6
       #ZC x,y, "6"
```

11.21 Change display orientation - BEGINNER

Change the display orientation in every possible direction.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Display orientation\

File:
BEGINNER – change displayorinetation.kmc

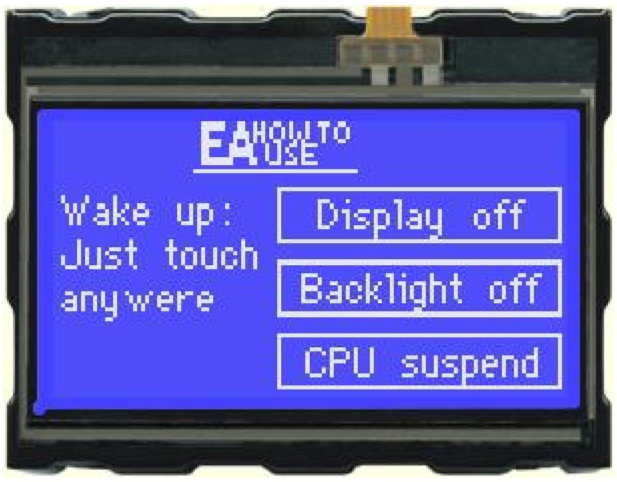
Commands:
#DO

Open file in KitEditor

```
eDIP128-6   "Displayorientation"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo2.bmp>

;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 43,0,6                   ; place logo

;---- Place Button ----
       #AF GENEVA10                                    ; define next buttons as
radiogroup 1
       #AE 2                                           ; slect touchframe
       #AT 14, 25, 114, 55, 1,0,"CChange orientation"    ; place button to change
orientation, 'C'=center aligned



;-----------------------------------------------------------------------------
TouchMacro: 1
       #AL 0,0; delete touchbuttons
       #DL            ; clear display
       #DO 1          ; set displayorientation to 90°
       #UI 11,5,6     ; place logo
       #AT 2, 35, 62, 65, 2,0,"CChange|orientation"      ; place button to change
orientation, 'C'=center aligned

TouchMacro: 2
       #AL 0,0; delete touchbuttons
       #DL            ; clear display
       #DO 2          ; set displayorientation to 180°
       #UI 35,0,6     ; place logo
       #AT 14, 25, 114, 55, 3,0,"CChange orientation"    ; place button to change
orientation, 'C'=center aligned

TouchMacro: 3
       #AL 0,0; delete touchbuttons
       #DL            ; clear display
       #DO 3          ; set displayorientation to 270°
       #UI 11,5,6     ; place logo
       #AT 2, 35, 102, 65, 4,0,"CChange|orientation"     ; place button to change
orientation, 'C'=center aligned

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

11.22 Energy save modes - BEGINNER

Show different energy modes. Wake-up by touch. There is a an EXPERT example available, too. Please have a look at EXPERT – energy_saving.kmc 90



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Energy saving\

File:
BEGINNER – energy_saving.kmc

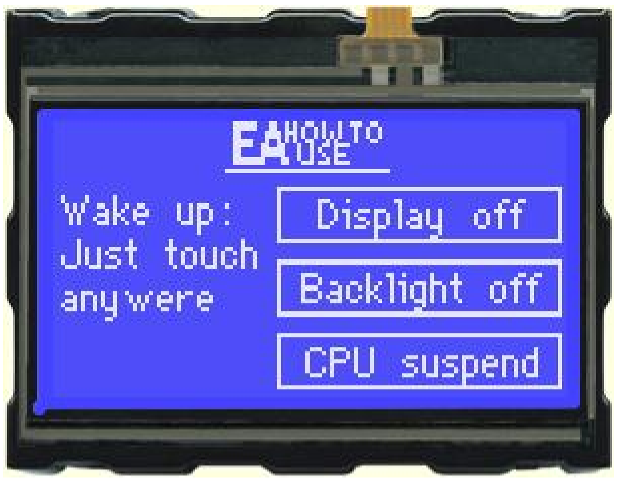
Commands:
#PD

Open file in KitEditor

```
eDIP128-6   "Energy saving modes"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome> ; define searchpath
Picture: 6 <howtouselogo2.bmp>             ; add picture

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 35,0,6                   ; place logo

;---- Place the left touch ----
       #AE 1                        ; set Frame style no. 14
       #AF GENEVA10                 ; set Touchfont (same as #AF 4, see default_font.kmi)
       #AT 56,15,126,27,0,1 "Display off"          ; place touch
       #AT 56,32,126,44,2,0 "Backlight off"        ; place touch
       #AT 56,49,126,61,3,0 "CCPU suspend" ; place touch

;--- Place information ---
       #ZF GENEVA10
       #ZL 2,16,"Wake up:|Just touch|anywere"


;---- Touch Macro for the middle touch (set) ----
TouchMacro: 1
       #PD 0,1        ; Power down mode, LED disabled, display disabled, 1 wakeup by touch
TouchMacro: 2
       #PD 1,1; Power down mode, LED disabled, display enabled, 1 wakeup by touch
TouchMacro: 3
       #PD 2,1; Power down mode, LED endabled, display enabled, 1 wakeup by touch
```

11.23 Energy save modes - EXPERT

Show different energy modes. Wake-up by touch. There is a a BEGINNER example available, too. Please have a look at BEGINNER – energy_saving.kmc 89.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP128-6\How to use\Energy saving\

File:
EXPERT – energy_saving.kmc

Commands:
#PD

Open file in KitEditor

```
eDIP128-6   "Energy saving modes"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome> ; define searchpath
LOGO = 6 ;using constnats makes it easier
Picture: LOGO <howtouselogo2.bmp>          ; add picture

;-----------------------------------------------------------------------------
;Constants for TouchMacros
DISP_OFF = 1
BACK_OFF = 2
CPU_OFF  = 3

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

Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                     ; place logo

;---- Place touch ----
;define constants
xs=56
xw=70
ys=15
yh=12
pitch=5
y=ys
       #AE 1                        ; set Frame style no. 1
       #AF GENEVA10                 ; set Touchfont (same as #AF 4, see default_font.kmi)
       #AT xs,y,xs+xw,y+yh,0,DISP_OFF, "Display off"             ; place touch
y+=yh+pitch
       #AT xs,y,xs+xw,y+yh,0,BACK_OFF, "Backlight off"    ; place touch
y+=yh+pitch
       #AT xs,y,xs+xw,y+yh,0,CPU_OFF, "CCPU suspend"      ; place touch

;--- Place information ---
       #ZF GENEVA10
       #ZL 2,ys+1,"Wake up:|Just touch|anywere"
```

```
;---- Touch Macro for the middle touch (set) ----
TouchMacro: DISP_OFF
       #PD 0,1       ; Power down mode, LED disabled, display disabled, 1 wakeup by touch
TouchMacro: BACK_OFF
       #PD 1,1; Power down mode, LED disabled, display enabled, 1 wakeup by touch
TouchMacro: CPU_OFF
       #PD 2,1; Power down mode, LED endabled, display enabled, 1 wakeup by touch
```