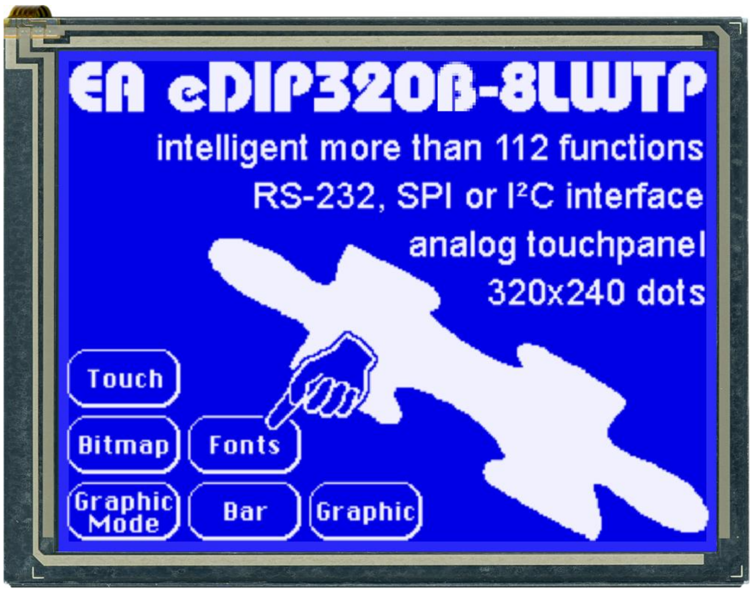


EA eDIP320-8 compiler manual

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

5 EA eDIP320-8 commands 21

5.1 Terminal ... 21
5.2 Text ... 22
5.3 Display ... 23
5.4 Draw ... 24
5.5 Flashing ... 25
5.6 Bitmap ... 26
5.7 Clipboard ... 27
5.8 Bargraph ... 28
5.9 Macros ... 29
5.10 Touch ... 31
5.11 Menu ... 34
5.12 Other commands ... 35

6 Default Fonts 36

6.1 Terminal 8x8 ... 36
6.2 Font 4x6 ... 37
6.3 Font 6x8 ... 38
6.4 Font 7x12 ... 39
6.5 Geneva 10 ... 40
6.6 Chicago 14 ... 41
6.7 Swiss 30 ... 42
6.8 BigZif 57 ... 43

7 Internal Pattern 44

8 Internal Border 45

9 Link Modes 46

10 BH7 format 47

11 How-to-use 49

11.1 Factory Setting 51
11.2 RS485 - Factory Setting 52
11.3 Place Strings - BEGINNER 54
11.4 Text linking - EXPERT 56
11.5 BMP file - BEGINNER 58
11.6 3 simple touch buttons - BEGINNER 59
11.7 Radio group - BEGINNER 61
11.8 Keypad - EXPERT 62
11.9 Free draw area with clipboard - BEGINNER 66
11.10 Free draw area with clipboard - EXPERT 67
11.11 Clipboard - EXPERT 69
11.12 Frame - BEGINNER 71
11.13 GrafikModes - EXPERT 73
11.14 Bargraph by touch - BEGINNER 76
11.15 Bargraph by touch - EXPERT 77
11.16 Menue - BEGINNER 79
11.17 Menue - EXPERT 81
11.18 Languages/Macro Pages - BEGINNER 83
11.19 Automatic Macro - BEGINNER 85
11.20 Automatic Macro - EXPERT 87
11.21 Process Macro - EXPERT 89
11.22 Outputs - BEGINNER 91

1 Overview

General

The EA eDIP320-8 is able to store many pictures, fonts and macros in internal FLASH memory. The EA KIT Editor is a powerful, free of charge software tool to create those macros and to store the pictures and fonts very easily.

The EA KIT Editor combines 3 functions:
- The editor itself which allows a simple definition of the macros, pictures and fonts like a standard text editor.
- The compiler which translates the text into the uploading code and shows up syntax error.
- The transmitter which search the right connection and uploads the data into the EA eDIP320-8.

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 eDIP320-8 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 images following functions are available:

`PICTURE_W`(nr)	**`PICTURE_H`**(nr)	for Images 15
`PICTURE_W`(nr, page)	**`PICTURE_H`**(nr, page)	

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

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

font = font number (eDIP command #ZF 22)
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 22)
height = additional line spacing between two lines 0..15 (eDIP command #ZY 22)
space = spacewidth (eDIP command #ZJ) 22

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

`eDIP320-8 "title"`	Defines EA eDIP320-8 as target. "title" is a short description for the project. It is shown on the display when uploading the FLASH memory of the module.
`DESTINATION <new.df>`	Specifies a new file name for the DATA-FLASH upload file. Optionally you can choose another path for the destination file.

`INCLUDE <file>`
`INCLUDE <file>,number`
	Includes the contents of the file <file> to be used in this actual file. This makes it possible to divide a project up into a number of source files. The file should have the extension *.kmi.
	The optional parameter (number) defines how often the file will be included.

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

`CODETABLE: nr`	A code table is useful adapt different ASCII tables. With that, the ASCII code can be changed for some single character (e.g. "ä", "ß").
	Up to 255 different code tables nr (1..255) can be defined.
	nr = 0 will disable all conversion.
	
	Example:
	`CodeTable: 1 ; use codetable 1 for *.FXT fonts with DOS-Code`
	`'€' = 128`
	`'äöüÄÖÜß' = $84,$94,$81, $8E,$99,$9A, $E1`

4.2 Transfer

`AUTOSCAN: n1`	Scan baudrate for connected eDIP on COM/USB before programming n1=0: autoscan off, use `baud` for connecting and programming n1=1: autoscan on, search baudrate automatically and programm with baudrate `baud`
`COMx: baud`	With this statement the COM port and baud rate is defined.
`USB: baud, "device"`	With this statement the USB device and baud rate is defined. If the EA STARTeDIP320 is connected to the USB, "device" is `"eDIP320-8 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 52)
`VERIFY`	Verifies the complete contents of the FLASH memory after upload.

4.3 Font

`FONT: nr,<file>`	Defines a font file which will be assigned to the number nr (1..31). <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 54)

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

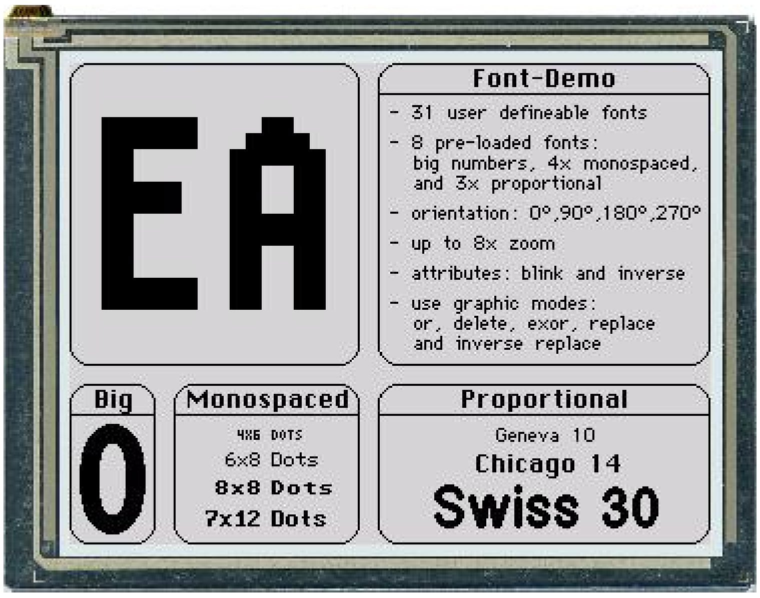
```
; default fonts  (max. 31 fonts number 1..32)
FONT8x8   = 0        ; internal terminal font
FONT4x6   = 1
FONT6x8   = 2
FONT7x12  = 3
GENEVA10  = 4
CHICAGO14 = 5
SWISS30B  = 6
BIGZIF57  = 7
```

```
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 36

Font 4x6 37
Font 6x8 38
Font 7x12 39

Geneva 10 40
Chicago 14 41
Swiss 30 42
BigZif 57 43

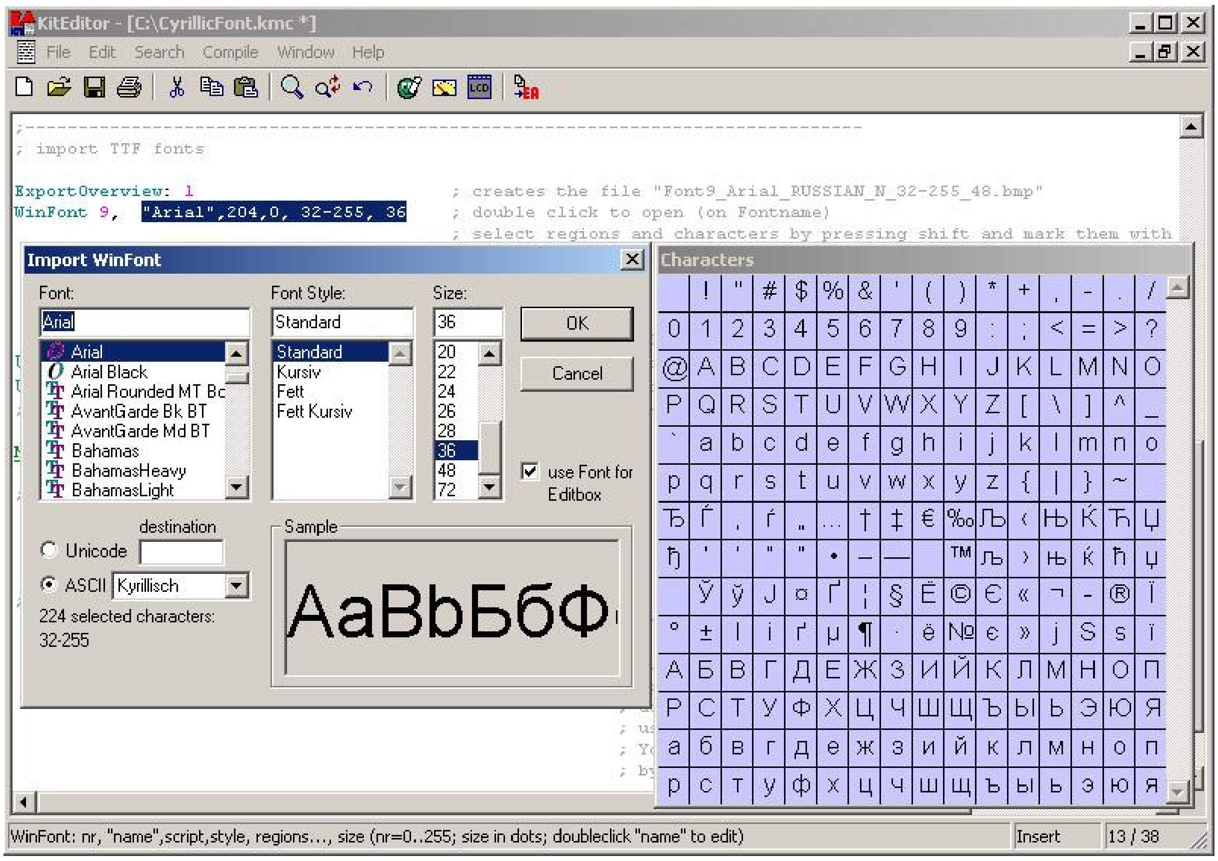


4.4 WinFont

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

Defines a Windows font and assigns to font number nr (1..31).
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 54)


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`

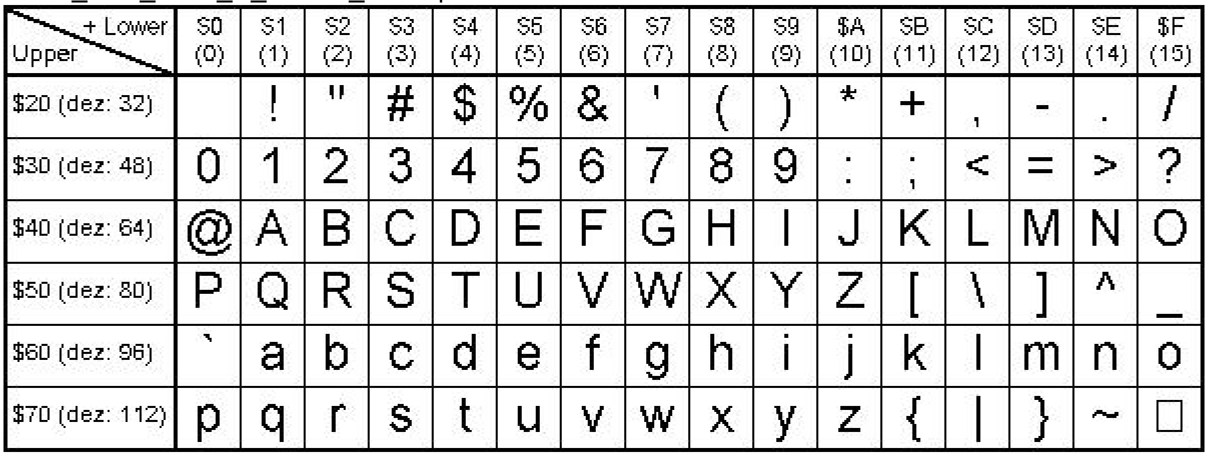
This statement enables the generation of a BMP file for all following WinFonts.
This is good to get an overview which character are available.

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

It is convenient to store all bitmap in FLASH; this will save transfer time via serial interface. The statement PICTURE defines a bitmap <file>with nr (0..255). <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.

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

The pictures can be used with the Bitmap commands. 26
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 58)

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 4,6V or lower.
`WAKEUPPINMACRO:`	Starts up again when the display was in PowerDown mode and PIn13 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, 99,104,114,105,102,116,112,114,111, 98,101,
    0, 27, 90, 70,  3, 27, 90, 76,  4, 30, 70,111,110,116, 55,120, 49, 50, 58, 32,
   83, 99,104,114,105,102,116,  0
};
```

4.11 Macro

`MACRO: nr`	
`MACRO: nr[page]`	Defines a normal macro with number nr (0..255). This macro will be executed with the command #MN nr 29 A series of macros occurring one after the other can be called cyclically (movie, hourglass, multi-page help text) see command #MA, #MJ 29. 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. Macro processes are not interrupted when commands are received from the interface or when touch macros are triggered see command #MD 29. Optionally different normal macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages. (see How-to-use example Automatic Macro - BEGINNER 85)

4.12 TouchMacro

`TOUCHMACRO: nr`	
`TOUCHMACRO: nr[page]`	Defines a touch macro with number nr (0..255). This macro will be executed if a touch key / switch with the return code nr is defined and the touch key/switch is pressed or by command #MT nr 29. Optionally different touch macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages. (see How-to-use example 3 simple touch buttons - BEGINNER 59)

4.13 MenuMacro

`MENUMACRO: nr`	
`MENUMACRO: nr[page]`	Defines a menu macro with number nr (0..255). This macro will be executed automatically after choosing an menu entry or by command #MM nr 29. Optionally different process macros can be stored for different pages [0..15]. If no page is selected it is set to 0. The 16 pages are helpful to realize e.g. screens in different languages. (see How-to-use example Menue - BEGINNER 79)

5 EA eDIP320-8 commands

5.1 Terminal

Terminal definition:

Define window	#TW C,L,W,H, A	The terminal output is executed only within the window from column C and line L (=upper-left corner) with a width of W and a height of H (specifications in characters) A=angle (0=0°; 1=90°; 2=180°; 3=270°) of the terminal display
Terminal off	#TA	Terminal display is switched off; outputs are rejected
Terminal on	#TE	Terminal display is switched on;

Cursor commands:

Position cursor	#TP C,L	C=column; L=line; origin upper-left corner (1,1)

Cursor on/off	#TC n1	n1=0: Cursor is invisible;
		n1=1: Cursor flashes;

Save cursor position	#TS	The current cursor position is saved
Restore cursor position	#TR	The last saved cursor position is restored

Terminal output:

String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal
Output version	#TV	The version no. is output in the terminal e.g. "EA eDIP320-8 V1.0 Rev.A"

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..31 (see compiler option FONT 10 : or WINFONT 11 :)
Font zoom factor	#ZZ n1,n2	n1 = X-zoom factor (1x to 8x); n2 = Y-zoom factor (1x to 8x)
Add. line spacing	#ZY n1	Insert n1=0..15 dots between two lines as additional line spacing
Text angle	#ZW n1	Text output angle n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Spacewidth (firmware V1.2)	#ZJ n1	n1=0: use spacewidth from font n1=1: same witdh as a number n1>=2: width in dot
Text link mode	#ZV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace (see link modes 46)
Text pattern	#ZM n1	link Text with pattern number n1 (0 to 15) (see internal pattern 44)
Text flashing attribute	#ZB n1	n1: 0=no flashing; 1=Text flashes on/off; 2=Text flashes inversely

(see How-to-use example Text linking - EXPERT 56)

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. 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. 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. 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 54)

5.3 Display

Display commands (effect on the entire display):

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

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 46)

Point size/line thickness	#GZ n1,n2	n1=X-point size (1 to 15)
		n2=Y-point size (1 to 15)

Pattern	#GM n1	set straight line/point pattern number n1 (0 to 15) (see internal pattern 44)

(see How-to-use example GrafikModes - EXPERT 73)

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=0..15 (always set, see internal pattern 44)
Draw box	#RO x1,y1,x2,y2,no	Draw rectangle from x1,y1 to x2,y2 with pattern no=0..15 (always replace, see internal pattern 44)
Draw frame	#RR x1,y1,x2,y2,no	Draw frame of type n0=1..18 from x1,y1 to x2,y2 (always set, see internal border 45)
Draw frame box	#RT x1,y1,x2,y2,no	Draw frame box of type n1n0=1..18 from x1,y1 to x2,y2 (always replace, see internal border 45)

(see How-to-use example Frame - BEGINNER 71)

5.5 Flashing

Flashing areas:

Delete flashing attribute	#QL x1,y1,x2,y2	Delete the flashing attribute from x1,y1 to x2,y2
Flash inversely	#QI x1,y1,x2,y2	Define an inverted flashing area from x1,y1 to x2,y2
Flashing area pattern	#QM x1,y1,x2,y2,n1	Define a flashing area (on/off) with pattern n1=0..15 from x1,y1 to x2,y2 (see internal pattern 44)
Set flashing time	#QZ n1	Set the flashing time n1=1..15 in 1/10sec; 0=deactivate flashing

5.6 Bitmap

Bitmap settings:

Image zoom factor	#UZ n1,n2	n1 = X-zoom factor (1x to 8x)
		n2 = Y-zoom factor (1x to 8x)

Image angle	#UW n1	output angle of the image n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Mirror Image	#UX n1	n1=0: normal display n1=1: the image is mirrored horizontally
Image link mode	#UV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace; (see link modes 46)
Image pattern	#UM n1	link Image with pattern number n1 (0 to 15) (see internal pattern 44)
Image flashing attribute	#UB n1	n1: 0=no flashing; 1=image flashes on/off; n1: 2=image flashes inversely; 3=flashes with flash image

(see How-to-use example GrafikModes - EXPERT 73)

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 data flash memory to x1,y1 (see compiler option `PICTURE` 15 `:`)
Load image	#UL x1,y1,data...	Load an image to x1,y1; data... = image in BH7-format 47 This command is only for serial interface, do not use this command in a macro !

(see How-to-use example BMP file - BEGINNER 58)

Hardcopy:

Send hardcopy	#UH x1,y1,x2,y2	After this command, the image extract is sent in BH7-format 47

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)
(see How-to-use example Clipboard - EXPERT 69)

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=0..255 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=pattern line; pat=line width type: 3=pattern line in rectangle; pat=line width (see internal pattern 44)

(see How-to-use example Bargraph by touch - BEGINNER 76)

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 :)

Macro settings:

Disable macros	#ML type,n1,n2	Macros of the type'N','T' or 'M' (type 'A' = all macro types) are disabled from the number n1 to n2; i.e. no longer run when called
Enable macros	#MU type,n1,n2	Macros of the type 'N','T', or 'M' (type 'A' = all macro types) are enabled from number n1 to n2; i.e. run again when called
Select macro/image page	#MK n1	A page is selected for macros and images n1=0 to 15 if a macro/image is not defined in the current page 1 to 15, this macro/image is taken from page 0 (e.g. to switch languages or for horizontal/vertical installation).
Save macro/image page	#MW	the current macro/image page is saved (when used in process macros)
Restore macro/image page	#MR	the last saved macro/image page is restored

(see How-to-use example Languages/Macro Pages - BEGINNER 83)

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 85)

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

(see How-to-use example Process Macro - EXPERT 89)

5.10 Touch

Touch presets:

Touch border form	#AE nr	Set the border n1 for the display of touch keys/switches (see internal border 45)
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 61)

Label font presets:

Label font	#AF nr	Set font with the number n1 (0 to 31) for touch key label (see compiler option FONT 11 :)
Label zoom factor	#AZ n1,n2	n1=X-zoom factor (1x to 8x); n2=Y-zoom factor (1x to 8x)
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°; n1=2: 180°; n1=3: 270°

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 59)
(see How-to-use example Bargraph by touch - BEGINNER 76)

Define touch menu:

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

(see How-to-use example Menue - BEGINNER 79)

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 76)

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
Rotate touch query	#AO n1	n1=0: normal query; n1=1: Touch query for top view (solder straps changed over)

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 31) for menu display
Menu font zoom factor	#NZ n1,n2	n1=X-zoom factor (1x to 8x); n2=Y-zoom factor (1x to 8x)
Additional line spacing	#NY n1	Insert n1=0..15 dots between two menu items as additional line spacing
Menu angle	#NW n1	Menu display angle: n1=0: 0°; n1=1: 90°; n1=2: 180°; n1=3: 270°
Touch menu automation	#NT n1	n1=1: Touch menu opens automatically n1=0:Touch menu does not open automatically; instead, the request 'ESC T 0' to open is sent to the host computer, which can then open the touch menu with 'ESC N T 2'

(see How-to-use example Menue - BEGINNER 79)

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 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 eDIP320-8 V1.0 Rev.A TP+"
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%.
		n1=250 save current brightness as starting brightness
		n1=254 switch LED off immediately
		n1=255 switch to 100% immediately

Output port:

Write output port	#YW n1,n2	n1=0: Set all output ports in accordance with n2 (=6/8-bit binary value). n1=1 to 6/8: Reset port n1 (n2=0); set (n2=1); invert (n2=2);
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

(see How-to-use example Outputs - BEGINNER 91)

Other functions:

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 52)

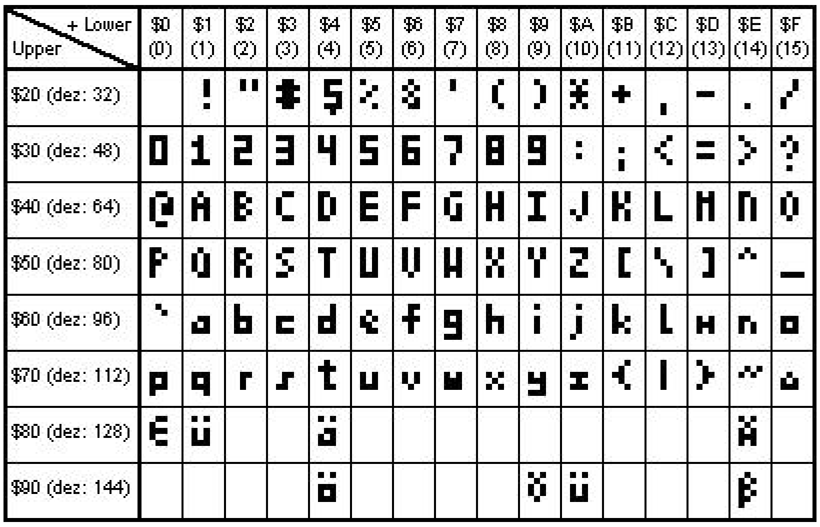
Power down	#PD n1	After this command, the display goes into power-down mode. n1=0: wake up only after reset;
		n1=1: wake up on LO-level at WUP Pin 13
		n1=2: wake up on touch;
		n1=3: wake up on WUP Pin or Touch

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)	€	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	ß	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)	Ç	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

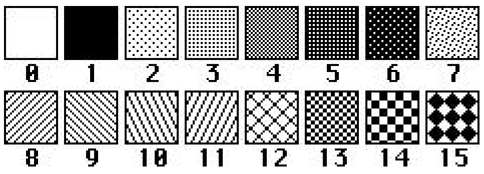
+ Lower Upper	$0 (0)	$1 (1)	$2 (2)	$3 (3)	$4 (4)	$5 (5)	$6 (6)	$7 (7)	$8 (8)	$9 (9)	$A (10)	$B (11)	$C (12)	$D (13)	$E (14)	$F (15)
$20 (dez: 32)		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
$30 (dez: 48)	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
$40 (dez: 64)	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
$50 (dez: 80)	P	Q	R	S	T	U	V	W	X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

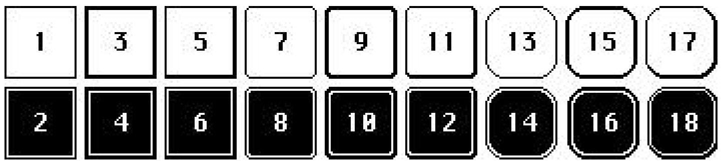
7 Internal Pattern

The internal pattern can be used with any command that uses patterns
e.g #ZM, 22 #GM,#RM,#RO 24, #UM 26, #QM 25, #BR,#BL,#BO,#BU 28



8 Internal Border

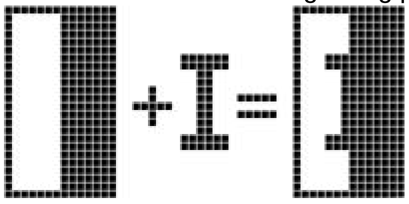
The internal border can be used with any command that uses borders e.g #RR,#RT[24], #AE[31]



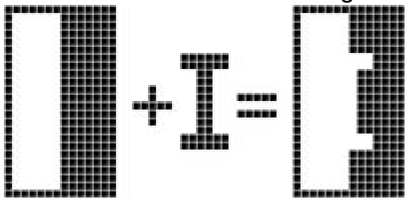
9 Link Modes

The Link Modes can be used with several commands
e.g #ZV 22, #GV 24, #UV 26

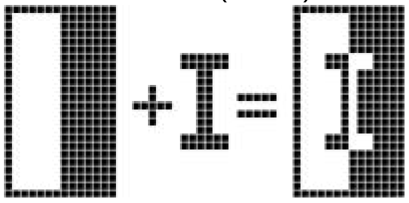
n1=1: set Pixel without regarding previous value (OR)



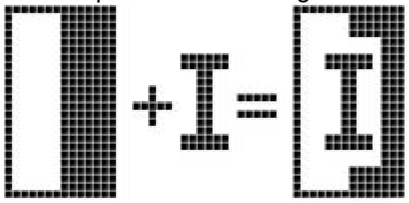
n1=2: delete Pixel without regarding previous value



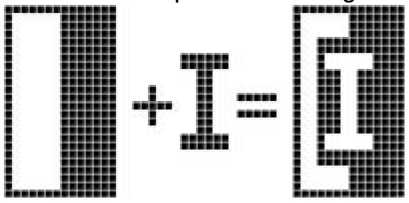
n1=3: invert Pixel (EXOR)



n1=4 replace: clear background and set Pixel



n1=5 invers replace: fill background and delete Pixel



10 BH7 format

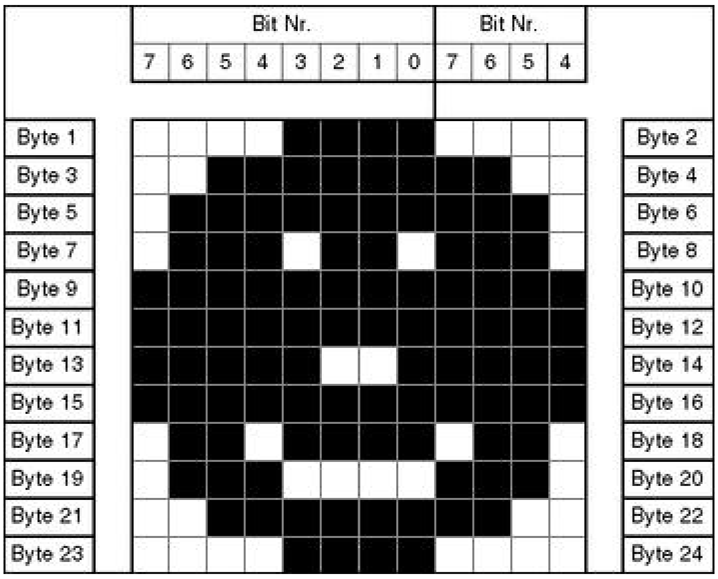
Use 'BitmapEdit.exe' from the LCD-Tools package to edit/convert images into/from BH7-format.

Structure of the picture header:

Byte	value	descripion
1.	$1B	ESC An image file always begins with
2.	$75	' u ' the image load instruction
3.	$6C	' l '
4.	$00	\ LOW byte X-coordinate
5.	$00	/ HIGH byte
6.	$00	\ LOW byte Y-coordinate
7.	$00	/ HIGH byte
8.	$42	'B' \
9.	$48	'H' identification
10.	$37	'7' /
11.	width	\ Width of the picture Low byte
12.		/ High byte
13.	height	\ Height of the picture Low byte
14.		/ High byte
15.	$01	Bit per Pixel 1=monochromes image
16.	$00	reserved

Image data:
After the header the image data follow.
number of 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)

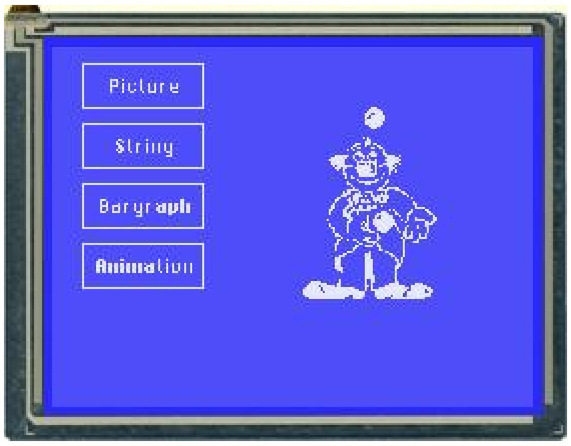
Example:
a small icon with 12x12 dots



The complete BH7-file:
```
$1B $75 $6C $00 $00 $00 $00
$42 $48 $37 $0C $00 $0C $00 $01 $00
$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\eDIP320-8\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\eDIP320-8\My first project

File:
my_first_project.kmc

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

Open file in KitEditor

```
eDIP320-8   "First project"
...
...
...
;-------------------------------------------------------------------------------
;Include picture
Picture: 6, <..\..\BITMAPS\monochrome\TOTKOPF.bmp> ;store as picture 6 (1-5 are used in
"default_pictures.kmi"
; define pictures for animation
Picture: 7, <..\..\BITMAPS\monochrome\Clown\clown1.bmp>
Picture: 8, <..\..\BITMAPS\monochrome\Clown\clown2.bmp>
Picture: 9, <..\..\BITMAPS\monochrome\Clown\clown3.bmp>
Picture: 10, <..\..\BITMAPS\monochrome\Clown\clown4.bmp>
Picture: 11, <..\..\BITMAPS\monochrome\Clown\clown5.bmp>
Picture: 12, <..\..\BITMAPS\monochrome\Clown\clown6.bmp>
Picture: 13, <..\..\BITMAPS\monochrome\Clown\clown7.bmp>
Picture: 14, <..\..\BITMAPS\monochrome\Clown\clown8.bmp>
Picture: 15, <..\..\BITMAPS\monochrome\Clown\clown9.bmp>


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

Macro: 0 ;define macro 0, called after power on, reset, watchdog reset
       #TA                     ; terminal off
       #AF CHICAGO14           ; set touch label font, the font is defined in include file
"default_font.kmi"
       #AT 20, 10,100,40,1,0, "Picture"    ; place 3 touchbuttons at x1,y1 to x2,y2,
Touchmacro 1 is called
       #AT 20,50,100,80,2,0, "String"      ; touchmacro 2 is called
       #AT 20,90,100,120,3,0, "Bargraph"   ; touchmacro 3 is called
       #AT 20,130,100,160,4,0, "Animation" ; touchmacro 4 is called

Macro: 1 ; define 9 macros to show animation of clown
       #UI 160,40, 7 ;place picture
Macro: 2 ; define 9 macros to show animation of clown
       #UI 160,40, 8
```

```
Macro: 3 ; define 9 macros to show animation of clown
       #UI 160,40, 9
Macro: 4 ; define 9 macros to show animation of clown
       #UI 160,40, 10
Macro: 5 ; define 9 macros to show animation of clown
       #UI 160,40, 11
Macro: 6 ; define 9 macros to show animation of clown
       #UI 160,40, 12
Macro: 7 ; define 9 macros to show animation of clown
       #UI 160,40, 13
Macro: 8 ; define 9 macros to show animation of clown
       #UI 160,40, 14
Macro: 9 ; define 9 macros to show animation of clown
       #UI 160,40, 15


;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
       #MS 0                 ; stop macro process
       #RL 110,0,319,239     ; delete area on the right (to delete pixels of other
touchmacros)
       #UI 160,40, 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
       #MS 0                 ; stop macro process
       #RL 110,0,319,239     ; 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 200,40, "Hello|World" ;write string centered, '|' means next line

TouchMacro: 3 ;Bargraph
       #BD 1, 0              ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                             ; because pixels are deleted with next command
       #MS 0                 ; stop macro process
       #RL 110,0,319,239     ; delete area on the right (to delete pixels of other
touchmacros)
       #AQ 0                 ; deactivate sending barvalues into sendbuffer
       #BO 1,200,200,230,10,0,100,1,3 ; 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

TouchMacro: 4 ;Start of Animation
       #BD 1, 0              ; delete bargraph 1, because of touchmacro 3 ("Bargraph"), it
can stay visible,
                             ; because pixels are deleted with next command
       #MS 0                 ; stop macro process
       #RL 110,0,319,239     ; delete area on the right (to delete pixels of other
touchmacros)
       #MS 1                 ; run all macro processes
       #MD 1,2, 1,9, 1       ; start new macros process and show pictures (call cyclically
every 1/10s normal macros 1-9)
```

11.1 Factory Setting

This macrofile sets the display back to factroy setting.

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\Init\

File:
Init.kmc

Commands:

Open file in KitEditor

```
eDIP320-8   "init/delivery state"     ; define eDIP320-8, "Projectname" max. 30 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 <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>
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


;------------------------------------------------------------------------------
Makro: 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\eDIP320-8\Init\

File:
INIT_with_RS485_address.KMC

Commands:

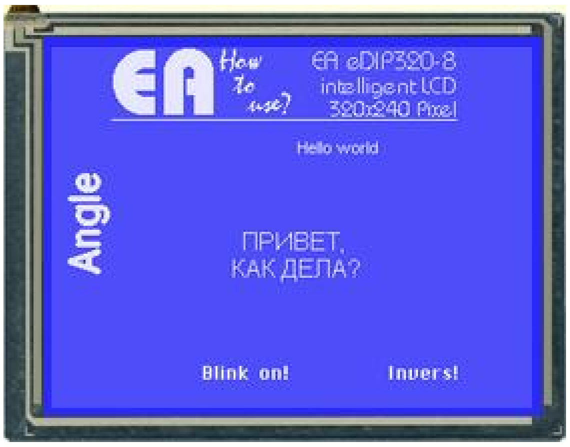
Open file in KitEditor

```
eDIP320-8   "init/delivery state"     ; define eDIP320-8, "Projectname" max. 30 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 <..\default_constant.kmi>   ; double click to open
include <..\default_font.kmi>
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
```

```
;-------------------------------------------------------------------------------
Makro: 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 56.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Font\

File:
BEGINNER – fonts.kmc

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

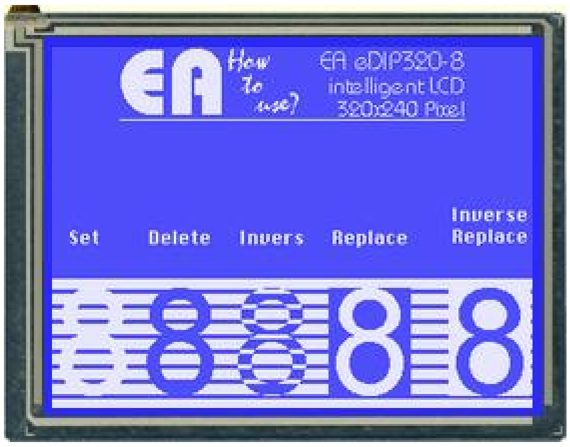
Open file in KitEditor

```
eDIP320-8   "Fonts"
...
...
...
;------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>


PATH: <..\..\..\FONTS\>

GENEVA10  = 4
CHICAGO14 = 5
SWISS30B  = 6

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

; 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, 18


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

       #UI 40,0,6                   ; place logo

;--- Place font examples---
       #ZF CHICAGO14                ; use font 5 (Chicago 14)
                                    ; 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 100,210, "Blink on!"            ; place text left justified
       #ZB 2                        ; text flashes inversly
       #ZR 270,210, "Invers!"       ; place text right aligned
```

```
        #ZB 0                         ; blink off
        #ZF SWISS30B                  ; use font 6 (SWISS 30 B)
        #ZW 1                         ; change textangle to 90°
        #ZL 10,150, "Angle"           ; place text
        #ZW 0                         ; change textangle back to 0°
        #ZF 8                         ; use Windowsfont arial (see line 48)
        #ZC 190,60, "Hello world"     ; place text center aligned
        #ZF 9                         ; switch to cyrillic font
        #ZC 160,120, {CFD0C8C2C5D282},'|',{20CAC0CA20C4C5CBC03F} ; 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 54.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Font\

File:
EXPERT – text_linking.kmc

Commands:
#ZV

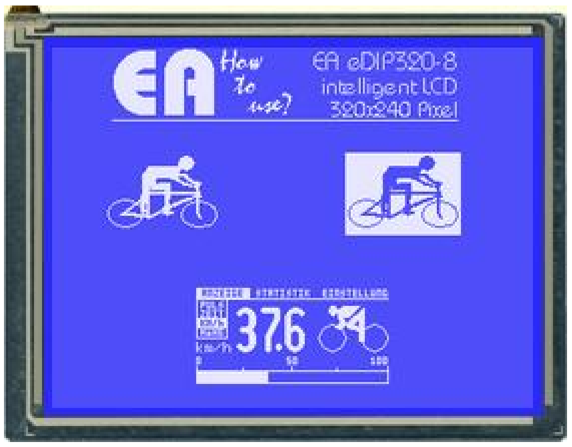
Open file in KitEditor

```
eDIP320-8   "Text link mode"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier to use
Picture: LOGO <howtouselogo_230x50.bmp>

Winfont 8: "Arial",-48,0, 48-57, 72 ; import winfont Arial with 72pt, but only the numbers
                                           ; double click on fontname to open
;-------------------------------------------------------------------------------
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 = 6
pitch = 4
ys=YPIXEL-9*yh-8*pitch
y=ys
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
y+=yh+pitch
       #RS xs,y,xe,y+yh      ; fill area (all pixels on)
```

```
;--- Place information ---
       #ZF CHICAGO14                  ; 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)
       #ZC 20, 120, "Set"                    ; place text
       #ZC 84, 120, "Delete"          ; place text
       #ZC 145,120, "Invers"          ; place text
       #ZC 210,120, "Replace"               ; place text
       #ZC 290,105, "Inverse|Replace"      ; place text



;--- Text link modes ---
       #ZF 8

       #ZV 1                          ; text link mode: 1 = set
       #ZC 20, 160, "8"               ; place text in the box
       #ZV 2                          ; text link mode: 2 = delete
       #ZC 84,160,"8"                 ; place text in the box
       #ZV 3                          ; text link mode: 3 = inverse
       #ZC 145,160, "8"               ; place text in the box
       #ZV 4                          ; text link mode: 4 = replace
       #ZC 210,160, "8"               ; place text in the box
       #ZV 5                          ; text link mode: 5 = inverse replace
       #ZC 290,160, "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\eDIP320-8\How to use\Picture\

File:
BEGINNER – show a bmp file.kmc

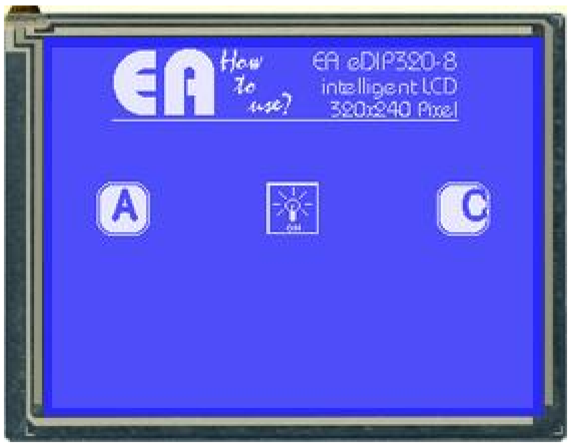
Commands:
#UI, #UV

Open file in KitEditor

```
eDIP320-8   "BMP-File"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\Bitmaps\monochrome>  ; define standard path
Picture: 6 <howtouselogo_230x50.bmp>       ; define picture no. 6
Picture: 7 <cycling.bmp>            ; define poicture no. 7
Picture: 8 <CYCLO.BMP>              ; define picture no. 8
                                    ; double-click on the name to open BitmapEdit



;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 40,0,6                   ; place logo

;--- Place Picture ---
       #UI 35,70, 7          ; place picture no. 6 at pixelpostion 10|20
       #UV 5                 ; next picture is placed inverse
       #UI 195, 70, 7; place same picture but inversed
       #UV 1                 ; next picture is placed normal (set mode)
       #UI 96, 160, 8
```

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 76), Radiogroups (see BEGINNER - radiogroup.kmc 61) and another Example with touch buttons (see EXPERT - keypad.kmc 62)



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Touch\

File:
BEGINNER – 3 simple buttons.kmc

Commands:
#AU, #AT

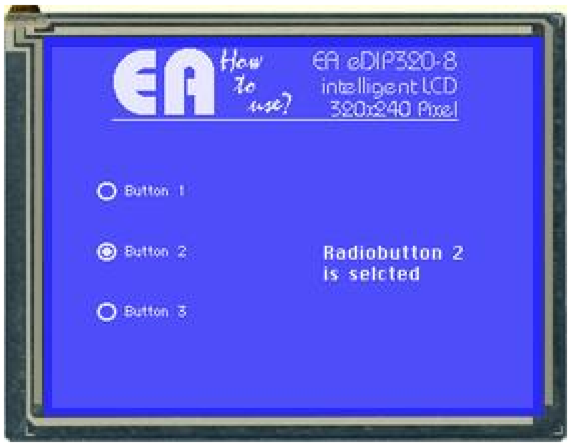
Open file in KitEditor

```
eDIP320-8   "3 simple buttons"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>
Picture: 7 <Button\Lamp34x34_1.bmp>,<Button\Lamp34x34_0.bmp>


;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                          ; Terminal off
       #UI 40,0,6                   ; place logo

;---- Place the left touch ----
       #AE 14                       ; set Frame style no. 14
       #AF 6                        ; set font no. 6 for Touch area
       #AT 30,90,64,124,65,0 "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 6                        ; set font no. 6for Touch area
       #AJ 143,90,7,1,2 ""          ; draw Touch area - as bitmap in this example without
text
                                    ; touch area is a switch

;---- Place the right touch as a bitmap ----
       #AF 6                        ; set font no. 6for Touch area
       #AT 256,90,290,124,67,68 "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 160,160, "#Macro 67, Button C pressed"

;---- Release the right touch ----
TouchMacro: 68
       #RL 0,160,320,200     ; 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 76), Buttons (see BEGINNER - 3 simple buttons.kmc 59) and another Example with touch buttons (see EXPERT - keypad.kmc 62)



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Touch\

File:
BEGINNER - touch as radio button.kmc

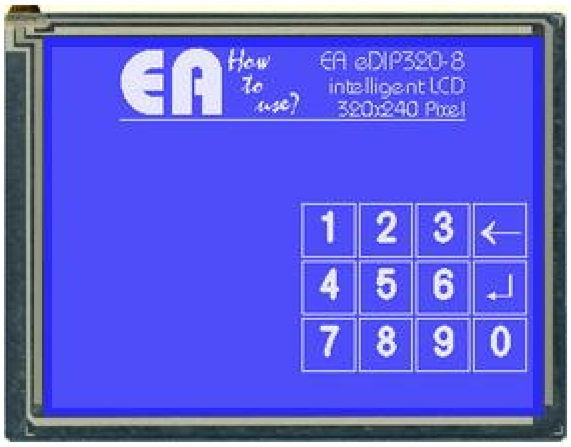
Commands:
#AR, #AJ

Open file in KitEditor

```
eDIP320-8   "Radiogroup"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo1.bmp>
Picture: 7 <button\Radio60x12_0.bmp>,<button\Radio60x12_1.bmp>


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

;---- Place radiobuttons ----
       #AR 1
       #AF GENEVA10                       ; define next buttons as radiogroup 1
       #AJ 5, 30, 7, 1,0,"RButton 1"      ; place radiobutton 1 with picture radiobutton
and downcode 1, 'R'= right aligned
       #AJ 5, 55, 7, 2,0,"RButton 2"      ; place radiobutton 1 with picture radiobutton
and downcode 2, 'R'= right aligned
       #AJ 5, 80, 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 80,50,"Radiobutton 1|is selcted"
TouchMacro: 2
       #ZF GENEVA10                ; use textfont no. 4
       #ZL 80,50,"Radiobutton 2|is selcted"
TouchMacro: 3
       #ZF GENEVA10                ; use textfont no. 4
       #ZL 80,50,"Radiobutton 3|is selcted"
```

11.8 Keypad - EXPERT

Place a keypad (0..9) and send the numbers to the terminal. There are further examples available, containig information about Bargraph (see BEGINNER - bargraph_by_touch.kmc 76), Buttons (see BEGINNER - 3 simple buttons.kmc 59) and Radio groups (see BEGINNER - radiogroup.kmc 61).



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Touch\

File:
EXPERT - numbers to terminal with autorepeat.kmc

Commands:
#AT

Open file in KitEditor

```
eDIP320-8   "Numbers to terminal"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6
Picture: LOGO, <howtouselogo_230x50.bmp>

SYMBOL = 9

Winfont: SYMBOL, "Symbol",-172,1, 172 + 191, 18 ; import Symbol (Return und Backspace)

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

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

Stop = 20

; define constants for (process)macros
Pm1 = 1
```

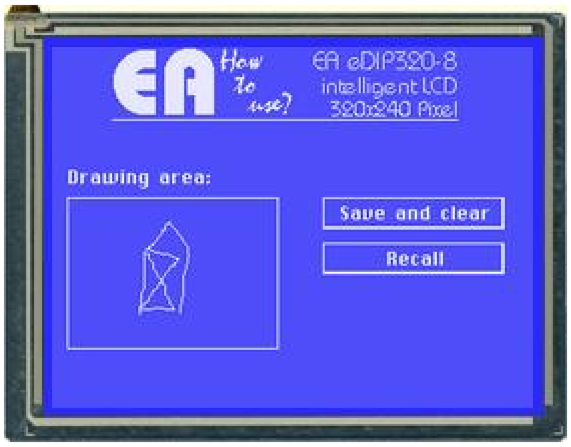
```
Pm2 = Pm1+1
Pm3 = Pm2+1
Pm4 = Pm3+1
Pm5 = Pm4+1
Pm6 = Pm5+1
Pm7 = Pm6+1
Pm8 = Pm7+1
Pm9 = Pm8+1
Pm0 = Pm9+1

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

;--- Define Terminal for outputs ---
t_y = 14
t_x = 5
lines=20
column=15
       #TW t_x,t_y,column,lines, 0        ; define t origin width (20 character) and
height (7 lines)

       #TE                         ; Terminal is visibl
       #TC 1                       ; Terminal on with flashing cursor

;--- Place Keypad ---
       #AF SWISS30B         ; define font for touchbuttons
;define some constants to place buttons easily
;WIDTH = 35
;HIGHT = 35
;PITCH = 5
;XSTART = 320 - 4*WIDTH - PITCH - 5
;YSTART = 80
;define some constants to place buttons easily
xw=35
yh=xw
pitch=3
xs=XPIXEL-4*xw-3*pitch-5
ys=(t_y-1)*8
x=xs
y=ys
       #AT x,y,x+xw,y+yh, Nb1,Stop, "1"
x+=pitch+xw                                    ; while release, touchmacro no. 20 will
be executed
       #AT x,y,x+xw,y+yh, Nb2,Stop, "2"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb3,Stop, "3"
x+=pitch+xw
       #AF SYMBOL           ; define font for return (Symbol)
       #AT x,y,x+xw,y+yh, CLEAR, 0, {AC} ; double click on curly brackets to see string

                  ; be sure that you selected Font 9 for fontbox

                  ; just rightclick on fontname and click "Select Font for FontBox"
       #AF SWISS30B         ; define font for touchbuttons
x=xs
y+=pitch+yh
       #AT x,y,x+xw,y+yh, Nb4,Stop, "4"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb5,Stop, "5"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb6,Stop, "6"
       #AF SYMBOL
x+=pitch+xw          ; define font for return (Symbol)
       #AT x,y,x+xw,y+yh, RETURN, 0, {AD}
       #AF SWISS30B         ; define font for touchbuttons
x=xs
y+=pitch+yh
       #AT x,y,x+xw,y+yh, Nb7,Stop, "7"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb8,Stop, "8"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb9,Stop, "9"
x+=pitch+xw
       #AT x,y,x+xw,y+yh, Nb0,Stop, "0"
```

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

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

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

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

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

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

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

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

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

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

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

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


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

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

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

Macro: Pm3
       #ZT "3"
       #MZ Pronumber,RepeatTime
```

```
Macro: Pm4
       #ZT "4"
       #MZ Pronumber,RepeatTime

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

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

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

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

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

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

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\eDIP320-8\How to use\Draw\

File:
BEGINNER – free_draw_area_clipboard.kmc

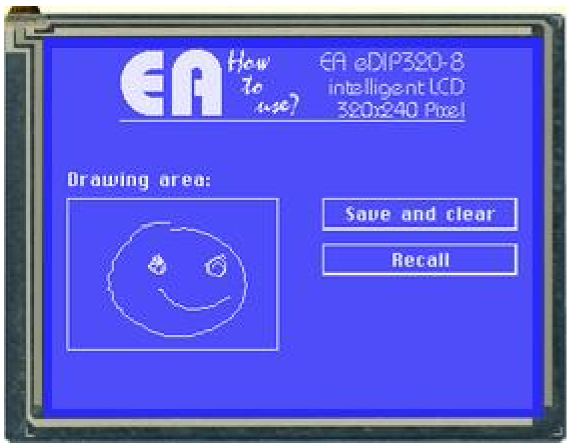
Commands:
#AD

Open file in KitEditor

```
eDIP320-8   "Free drawing area with clipboard"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>
Picture: 7 <Button\Lamp34x34_1.bmp>,<Button\Lamp34x34_0.bmp>


;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                           ; Terminal off
       #UI 40,0,6                    ; place logo

;--- Place information ---
       #ZF CHICAGO14                 ; set font no.5
       #ZL 10, 80,"Drawing area:"
;---- Place buttons ----
       #AF CHICAGO14                        ; set font no. 5 for Touch area
       #AE 5
       #AT 180,100,300,120,1,0, "Save and clear"  ; place touchbutton 1
       #AT 180,130,300,150,2,0, "CRecall"         ; place touchbutton 2 (C means center)

;---- Place drawing area ----
       #GR 10,100,150,200    ; place rectangle around drawing area
       #AD 11,101,149,199,1  ; place drawing area, linewith 1



;-----------------------------------------------------------------------------
TouchMacro: 1
       #CS 11,101,149,199    ; drawing area is copied to the clipboard
       #RL 11,101,149,199    ; 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\eDIP320-8\How to use\Draw\

File:
EXPERT – free_draw_area_clipboard.kmc

Commands:
#AD

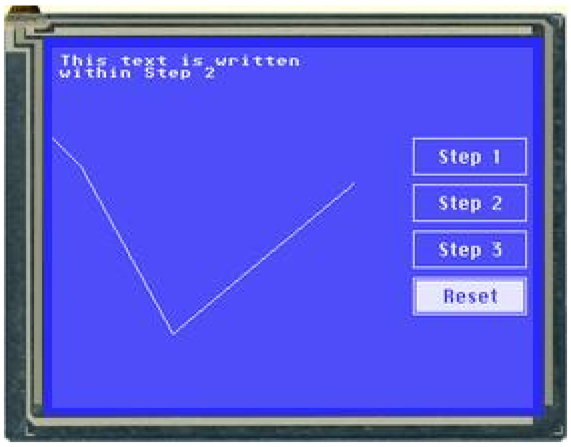
Open file in KitEditor

```
eDIP320-8   "Free drawing area with clipboard"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier
Picture: LOGO, <howtouselogo_230x50.bmp>

;-----------------------------------------------------------------------------
;define constants for touch macros
SAVE = 1
RECALL = 2

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

;--- Store postion of drawing are in constants ---
;it's easier to calculate position, if you align on the main task
xs=10
ys=100
xw=140
yh=100

;--- Place information ---
       #ZF CHICAGO14               ; set font no.5
       #ZL xs, ys-14-5,"Drawing area:" ; set string above the drawing rectangle -14
because of the font height, -5 is the wanted gap
;---- Place buttons ----
       #AF CHICAGO14                      ; set font no. 5 for Touch area
       #AE 5
x=xs+xw+30
pitch=10
height=20
y=ys
       #AT x,y,XMAX-pitch,y+20,SAVE,0, "Save and clear"  ; place touchbutton 1
y+=20+pitch
       #AT x,y,XMAX-pitch,y+20,RECALL,0, "CRecall"             ; place touchbutton 2 (C
means center)
```

```
;---- Place drawing area ----
       #GR xs,ys,xs+xw,ys+yh ; place rectangle around drawing area
       #AD xs+1,ys+1,xs+xw-1,ys+yh-1,1      ; place drawing area, linewith 1



;--------------------------------------------------------------------------------
TouchMacro: SAVE
       #CS xs+1,ys+1,xs+xw-1,ys+yh-1       ; drawing area is copied to the clipboard
       #RL xs+1,ys+1,xs+xw-1,ys+yh-1       ; clear drawing area

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

11.11 Clipboard - EXPERT

Explain the use of clipboard.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Clipboard\

File:
EXPERT - Clipboard.kmc

Commands:
#DO

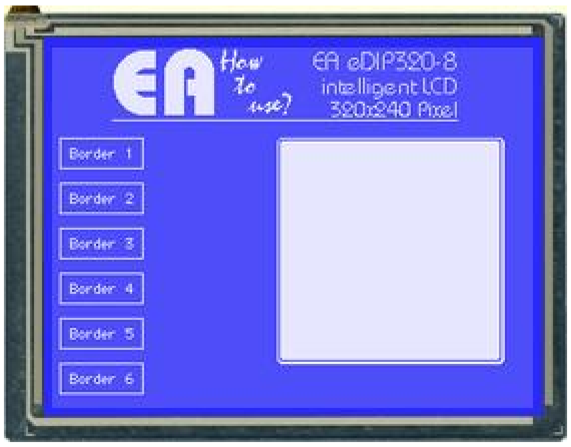
Open file in KitEditor

```
eDIP320-8   "Clipboard"
...
...
...
;-----------------------------------------------------------------------------
;include Pictures
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier to use
Picture: 6 <howtouselogo1.bmp>
;-----------------------------------------------------------------------------
;define constants for touchmacros
ST1 = 1
ST2 = ST1+1
ST3 = ST2+1
RES = 100

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

Macro: 0
       #TA ;Terminal off

       ; define touchbuttons
       #AF CHICAGO14
xs = 240
xb = XPIXEL-xs-5
ys = 60
pitch = 7
yh = (YPIXEL-pitch*3-2*ys)/4
y=ys
       #AT xs,y,xs+xb,y+yh,ST1,0,"Step 1"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,ST2,0,"Step 2"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,ST3,0,"Step 3"
y+=yh+pitch
       #AT xs,y,xs+xb,y+yh,RES,0,"CReset" ;C0center alligned, neccessayry, because R is
used as right justified


TouchMacro: ST1 ; Step one, draw something
       #GD 0,60,20,80
       #GW 80,190
       #GW 200,90
       ;drawing lines finished
       #CB ; save display contents
```

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

TouchMacro: ST3 ; step three, show normal display content
       #CR ;restore are

TouchMacro: RES
       #DL    ; clear display
       #MN 0  ; call start Macro
```

11.12 Frame - BEGINNER

Show the different borders.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Frame\

File:
BEGINNER – frame.kmc

Commands:
#RT, #RR

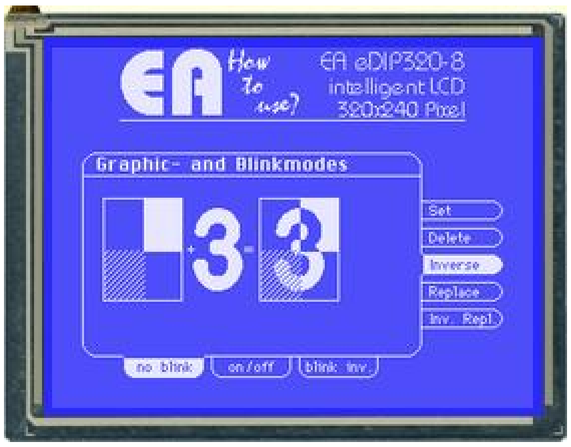
Open file in KitEditor

```
eDIP320-8 "Different Borders"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>
;-------------------------------------------------------------------------------

Macro: MnAutoStart

;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI 40,0,6                  ; 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 5, 60,60, 80,1,0,"Border 1"
       #AT 5, 90,60,110,2,0,"Border 2"
       #AT 5,120,60,140,3,0,"Border 3"
       #AT 5,150,60,170,4,0,"Border 4"
       #AT 5,180,60,200,5,0,"Border 5"
       #AT 5,210,60,230,6,0,"Border 6"


       #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 150,60,300,210,15       ; draw new frame with border no. 15

TouchMacro 2: ; Called by Button Border2
       #MN 1                       ; call Macro 1 (delete area to draw new frames)
       #RR 150,60,300,210,4        ; draw new frame with border no. 4

TouchMacro 3: ; Called by Button Border3
       #MN 1                       ; call Macro 1 (delete area to draw new frames)
       #RR 150,60,300,210,12 ; draw new frame with border no. 10

TouchMacro 4: ; Called by Button Border4
       #MN 1                       ; call Macro 1 (delete area to draw new frames)
       #RR 150,60,300,210,1        ; draw new frame with border no. 1
```

```
TouchMacro 5:  ;Called by Button Border2
       #MN 1
       #AE 18                        ; set touchframe no. 18
       #AT 160,100,280,125,7,0,"Border-Button" ; define button

TouchMacro 6:  ;Called by Button Border3
       #MN 1
       #BR 1,160,100,280,125,0,100,1,5     ; 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 CHICAGO14                              ; switch textfont to CHICAGO 10 (same as
#ZF 5)
       #ZL 170,150,"Bargraph with|fill-pattern"   ; place info-text
                                           ; '|' means new line


Macro 1: ; Draw Rectangel with selected Border
       #RL 150,60,300,210    ; delete area, to draw
       #AV 160,100,0         ; delete old touchareas (Bargraph and border button)


TouchMacro 7: ; called by  Boder-Button
       #ZF CHICAGO14                              ; switch textfont to CHICAGO 10 (same as
#ZF 5)
       #ZL 170,150,"Border Button|was pressed"    ; place text, that Border-Button was
touched
                                                  ; '|' means new line
```

11.13 GrafikModes - EXPERT

Show different link modes and flashing modes.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\GrafikMode\

File:
EXPERT – GrafikMode.kmc

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

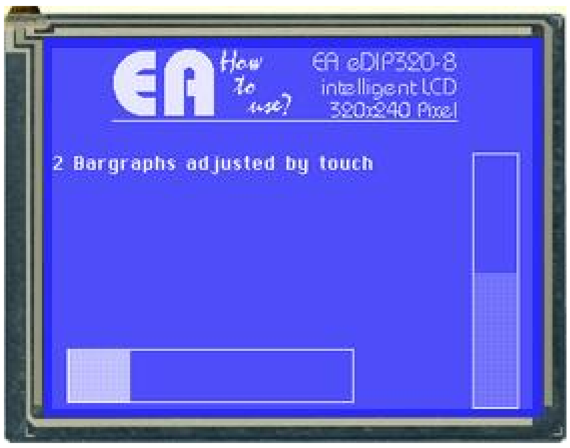
Open file in KitEditor

```
eDIP320-8   "Graphic-Blinkmode"
...
...
...
;-------------------------------------------------------------------------------
;include pictures

LOGO = 6 ; using constants makes it easier to use
ButRegRight_big  = 7
ButRegBottom_big = 8
PATH: <..\..\..\BITMAPS\monochrome\> ;switch to right folder of pictures
Picture: LOGO <howtouselogo_230x50.bmp>
PATH: <..\..\..\BITMAPS\monochrome\button\> ;switch to right folder of pictures
Picture: ButRegRight_big  <RegisterRight55x15_0.bmp>,<RegisterRight55x15_1.bmp>
Picture: ButRegBottom_big <RegisterBottom55x15_0.bmp>,<RegisterBottom55x15_1.bmp>
;-------------------------------------------------------------------------------
;define constants for normal-macros
MnDraw = 1

;define constants for touch-macros
TmNoblink = 1
TmBlinkonoff = 2
TmBlinkinvers = 3



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

;-------------------------------------------------------------------------------
Makro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                         ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO                    ; place logo
x1=20
y1=PICTURE_H(LOGO)+x1
x2=XPIXEL-PICTURE_W(ButRegRight_big)-x1 ;calculate end of big bounding box in the middle
y2=YPIXEL-PICTURE_H(ButRegBottom_big)-x1
       #RR x1,y1,x2,y2, 13
       #ZF CHICAGO14
       #ZV SET
       #ZC (x2-x1)/2,y1+1,"Graphic- and Blinkmodes" ; display title
```

```
        #GR x1,y1+15,x2,y1+15 ;underline title

x=x2 ;start of touchbuttons on the right
pitch=3
y=y1+15+(y2-(y1+15)-5*PICTURE_H(ButRegRight_big)-4*pitch)/2 ; calculate the middle of the
bounding box with 5 touchbuttons

        #AF GENEVA10 ; set font for buttons
        #AR 1 ; start of first radiogroup
        #AJ x,y,ButRegRight_big, TmOr,0,      "L Set"
y+=PICTURE_H(ButRegRight_big)+pitch
        #AJ x,y,ButRegRight_big, TmDelete,0,  "L Delete"
y+=PICTURE_H(ButRegRight_big)+pitch
        #AJ x,y,ButRegRight_big, TmExor,0,    "L Inverse"
y+=PICTURE_H(ButRegRight_big)+pitch
        #AJ x,y,ButRegRight_big, TmReplace,0, "L Replace"
        #AP TmReplace,1
y+=PICTURE_H(ButRegRight_big)+pitch
        #AJ x,y,ButRegRight_big, TmInvRepl,0, "L Inv. Repl."
        #AR 0 ; end of group

x=x1+(x2-x1-3*PICTURE_W(ButRegBottom_big)-2*pitch)/2 ; calculate middle of bounding box
for 3 bottom touch areas
y=y2
        #AR 2
        #AJ x,y,ButRegBottom_big, TmNoblink,0,     "no blink"
        #AP TmNoblink,1
x+=PICTURE_W(ButRegBottom_big)+pitch
        #AJ x,y,ButRegBottom_big, TmBlinkonoff,0,  "on/off"
x+=PICTURE_W(ButRegBottom_big)+pitch
        #AJ x,y,ButRegBottom_big, TmBlinkinvers,0, "blink inv."
        #AR 0

x_offset=x1
x1+=14 ; start of first graphic box
w=52  ; width of graphic box
x2=x1+w
y1+=30 ; start of first graphic box
h=68
y2=y1+h
x3=118+x_offset ; start of second graphic box
xm=x1+(x2-x1)/2 + 1 ; calculate middle of graphic box
ym=y1+(y2-y1)/2 + 1
        #RM x1,ym,xm,y2,8
        #RM xm,y1,x2,ym,1
        #GR x1,y1,x2,y2
        #CS x1,y1,x2,y2 ; save graphic box to clipboard

;place mathematical operators
        #ZL x2+3,ym-7,"+"
        #ZC x3-8,ym-7,"="

;place character
        #ZF BIGZIF57
        #ZC x2+(x3-x2)/2-3,ym-29,"3"

; call standard macro and place character in the right graphic box
        #MT TmReplace


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

TouchMakro: TmOr
  #ZV SET
  #MN MnDraw

TouchMakro: TmDelete
  #ZV DELETE
  #MN MnDraw

TouchMakro: TmExor
  #ZV INVERS
  #MN MnDraw

TouchMakro: TmReplace
  #ZV REPLACE
  #MN MnDraw

TouchMakro: TmInvRepl
  #ZV INVREPL
  #MN MnDraw

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

TouchMakro: TmNoblink
  #ZB NOBLINK
  #MN MnDraw

TouchMakro: TmBlinkonoff
  #ZB BLINKONOFF
  #MN MnDraw

TouchMakro: TmBlinkinvers
  #ZB BLINKINVERS
  #MN MnDraw

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

Makro: MnDraw
       #CK x3,y1 ; recal first graphic box from clipboard and position it on the right
       #ZL x3+8,ym-29,"3" ; place character into it
```

11.14 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 [77]. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc [59].


EA How to use?
EA eDIP320-8 intelligent LCD 320x240 Pixel
2 Bargraphs adjusted by touch

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Bargraph\

File:
BEGINNER – bargraph_by_touch.kmc

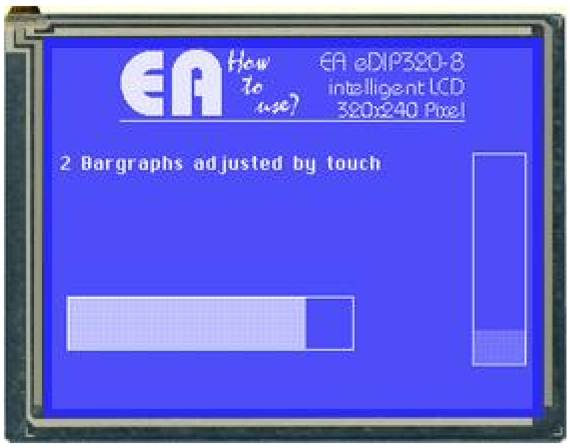
Commands:
#BR

Open file in KitEditor

```
eDIP320-8 "2 Bargraphs"
...
...
...

;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>
;-------------------------------------------------------------------------------

Makro: MnAutoStart
       #TA                       ; Terminal off
       #UI 40,0,6

       #ZF CHICAGO14                    ; set Textfont to Chicago14 (same as #ZF 5)
       #ZL 0,70,"2 Bargraphs adjusted by touch"   ; Place text


;---- Place a bargraph no. 1 ----
       #BR 1,10,200,200,235,0,100,1,5      ; define bargraph no. 1 with size, value and
type
       #BA 1,57                         ; actualize bargraph no. 1 value
       #AB 1                            ; define bargraph no. 1 to be adjusted by the
touch

;---- Place another bargraph no. 2 ----
       #BO 2,280,70,310,240,0,100,1,3      ; define bargraph no. 2 with size, value and
type
       #BA 2,38                         ; actualize bargraph no. 2 value
       #AB 2                            ; define bargraph no. 2 to be adjusted by the
touch

       #BA 1,87                         ; actualize bargraph no. 1 value; now
brightness is set to 87%
```

11.15 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 76. If you need help using touch functions, please refer to BEGINNER - 3 simple buttons.kmc 59.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Bargraph\

File:
EXPERT – bargraph_by_touch.kmc

Commands:
#BR

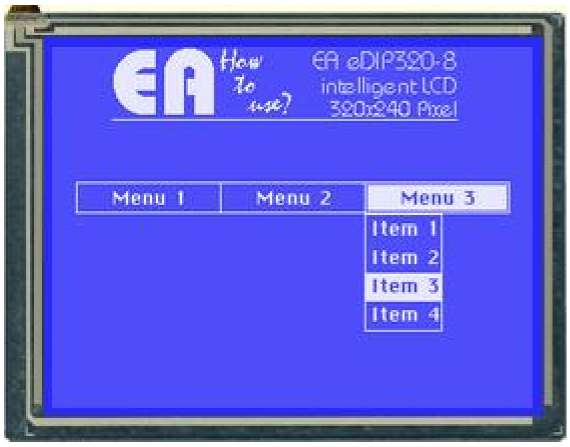
Open file in KitEditor

```
eDIP320-8 "2 Bargraphs"
...
...
...

;--------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ;using constants makes it easier to use
Picture: 6 <howtouselogo_230x50.bmp>
;--------------------------------------------------------------------------------

Makro: MnAutoStart
       #TA                         ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO

       #ZF CHICAGO14                       ; set Textfont to Chicago14 (same as #ZF 5)
       #ZL 5,PICTURE_H(LOGO)+20,"2 Bargraphs adjusted by touch"  ; Place text


;---- Place a bargraph no. 1 ----
BR1=1
BR2=2
sv=0
ev=100
BR1_xs=10
BR1_ys=165
BR1_xw=190
BR1_yh=35
BR1_typ=1
BR1_pat=5
       #BR BR1,BR1_xs,BR1_ys,BR1_xs+BR1_xw,BR1_ys+BR1_yh,sv,ev,BR1_typ,BR1_pat ; define
bargraph no. 1 with size, value and type
       #BA BR1,57                                                    ; actualize
bargraph no. 1 value
       #AB BR1                                               ; define bargraph no. 1
to be adjusted by the touch

;---- Place another bargraph no. 2 ----
BR2_xs=280
BR2_ys=70
BR2_xw=BR1_yh
BR2_yh=140
```

```
BR2_typ=1
BR2_pat=3
       #BO BR2,BR2_xs,BR2_ys,BR2_xs+BR2_xw,BR2_ys+BR2_yh,sv,ev,BR2_typ,BR2_pat ; define
bargraph no. 1 with size, value and typee
       #BA BR2,38                          ; actualize bargraph no. 2 value
       #AB BR2                             ; define bargraph no. 2 to be adjusted by the
touch

       #BA BR2,87                          ; actualize bargraph no. 2 value = 87
```

11.16 Menue - BEGINNER

Show a menu, operable by touch. There is a an EXPERT example available, too. Please have a look at EXPERT - menue.kmc 81.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Menue\

File:
BEGINNER – menue.kmc

Commands:
#AM

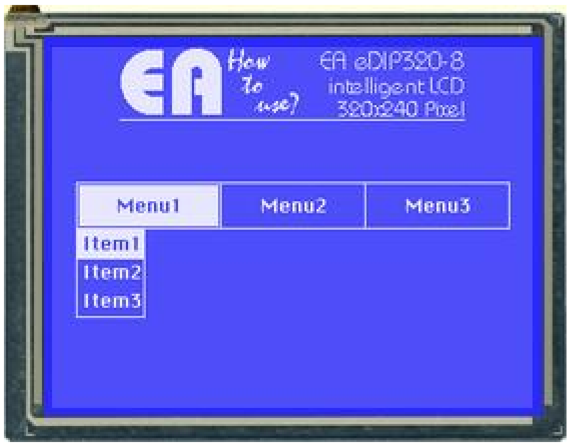
Open file in KitEditor

```
eDIP320-8   "Menue"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>

;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                  ; Terminal off
       #UI 40,0,6           ; place logo
       #ZF CHICAGO14        ; set font for strings

;--- Place Menue ---
       #AF CHICAGO14        ; set font for nenu headline (Chicago 14)
       #NF CHICAGO14        ; set Menue font to CHICAGO14
                            ; same as #NF 5
       #NY 5                ; adding 5 additional dots between two menue items
       #NW 0                ; menu angle
       #NT 1                ; touch menu opens automatically

       #AM 16,90,112,110,0,0,10, "UCMenu 1|Item 1|Item 2|Item 3"       ; place Menu1,
opening down (U), text centered (C)
                                                                               ; MenuMacro
10+Itemnumber is called
                                                                               ; Item1 has
no. 0, Item2 has 1...
       #AM 112,90,208,110,0,0,20, "UCMenu 2|Item 1|Item 2"             ; place
Menu2, opening down (U), text centered (C)
                                                                               ; MenuMacro
20+Itemnumber is called
       #AM 208,90,304,110,0,0,30, "UCMenu 3|Item 1|Item 2|Item 3|Item 4"       ; place
Menu3, opening down (U), text centered (C)
                                                                               ;MenueMacro
30+Itemnumber is called

;-----------------------------------------------------------------------------
MenueMacro: 10;Menu 1 Item 1
       #ZL 10,180,"Selected: Menue 1, Item 1" ; place text

MenueMacro: 11;Menu 1 Item 2
```

```
       #ZL 10,180,"Selected: Menue 1, Item 2"

MenueMacro: 12;Menu 1 Item 3
       #ZL 10,180,"Selected: Menue 1, Item 3"

;------------------------------------------------------------------------------
MenueMacro: 20;Menu 2 Item 1
       #ZL 10,180,"Selected: Menue 2, Item 1" ; place text

MenueMacro: 21;Menu 2 Item 2
       #ZL 10,180,"Selected: Menue 2, Item 2"

;------------------------------------------------------------------------------
MenueMacro: 30;Menu 3 Item 1
       #ZL 10,180,"Selected: Menue 3, Item 1" ; place text

MenueMacro: 31;Menu 3 Item 2
       #ZL 10,180,"Selected: Menue 3, Item 2"

MenueMacro: 32;Menu 3 Item 3
       #ZL 10,180,"Selected: Menue 3, Item 3"

MenueMacro: 33;Menu 3 Item 3
       #ZL 10,180,"Selected: Menue 3, Item 4"
```

11.17 Menue - EXPERT

Show a menu, operable by touch. There is a a BEGINNER example available, too. Please have a look at BEGINNER - menu.kmc 79.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Menue\

File:
EXPERT – menue.kmc

Commands:
#AM

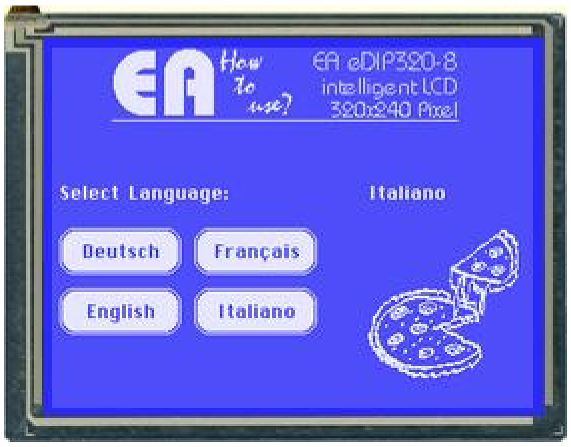
Open file in KitEditor

```
eDIP320-8   "Menue"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ; using constants makes it easier
Picture: LOGO <howtouselogo_230x50.bmp>

;-------------------------------------------------------------------------------
;define string constants
!MENU1! = "UCMenu1|Item1|Item2|Item3"
!MENU2! = "UCMenu2|Item1|Item2"
!MENU3! = "UCMenu3|Item1|Item2|Item3"

!SELECT! = "Selected: Menue "

;-------------------------------------------------------------------------------
;define constants for menumacros
MEN1 = 10
MEN2 = MEN1+10
MEN3 = MEN2+10

;-------------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                  ; Terminal off
       #UI (XPIXEL-PICTURE_W(LOGO))/2,0,LOGO           ; place logo
       #ZF CHICAGO14        ; set font for strings

;--- Place Menue ---
       #AF CHICAGO14        ; set font for nenu headline (Chicago 14)
       #NF CHICAGO14        ; set Menue font to CHICAGO14
                            ; same as #NF 5
       #NY 5                ; adding 5 additional dots between two menue items
       #NW 0                ; menu angle
       #NT 1                ; touch menu opens automatically

xs=16
xw=(XPIXEL-2*xs)/3
ys=90
yh=30
pitch=0
```

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

;--------------------------------------------------------------------------------
x=10
y=180
MenueMacro: MEN1+0    ;Menu 1 Item 1
       #ZL x,y,!SELECT! "1, Item 1" ; place text (cobine string constant and "normal"
strings)

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

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

;--------------------------------------------------------------------------------
MenueMacro: MEN2+0    ;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"
```

11.18 Languages/Macro Pages - BEGINNER

Describe the important function of different languages, with the help of MacroPages.



Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Language\

File:
BEGINNER – multilingual.kmc

Commands:
#MK

Open file in KitEditor

```
eDIP320-8   "Multilingual Support"
...
...
...
;-------------------------------------------------------------------------------

; constants for language support
GERMAN        = 0
ENGLISH       = 1
FRENCH = 2
ITALIAN= 3

;------------------------------------------------------------------------------
; Include Pictures  max. 256 Bilder (0..255)
Path <..\..\..\bitmaps\monochrome\>                ; specify path
Picture 6 <howtouselogo_230x50.bmp>

PATH: <.\Bitmap\>
Picture: 100[GERMAN] <SausageBeer.bmp>                    ; store picture to no. 100 and
german page
Picture: 100[ENGLISH]<FishandChips.bmp>                   ; store picture to no. 100 and
english page
Picture: 100[FRENCH] <Baguette.bmp>                ; store picture to no. 100 and french
page
Picture: 100[ITALIAN]<Pizza.bmp>                          ; store picture to no. 100 and
italian page

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

;---- Place ELECTRONIC ASSEMBLY Logo ----
       #TA                          ; Terminal off
       #UI 40,0,6

;---- Place some text ----
       #ZF CHICAGO14         ;string font
       #ZL 5,90, "Select Language:"

;--- 3 Touchbuttons (Language selection) ---
       #AE 14                                      ; set Frame style no. 14
       #AF 5                                       ; set font no. 5 for Touch area
       #AT 5,120,85,150,1,0 "Deutsch"              ; place button "German" and call
TouchMacro 1
       #AT 5,160,85,190,2,0 "English"              ; place button "English" and call
TouchMacro 2
```

```
       #AT 95,120,175,150,3,0 "Fran"135"ais"      ; place button "French" and call
TouchMacro 3
                                                  ; the 135 is the decimal value for
cdille
       #AT 95,160,175,190,4,0 "Italiano"          ; place button "Italian" and call
TouchMacro 4

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

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

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

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

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


;--------------------------------------------------------------------------------
Macro 1[GERMAN]:
       #ZF CHICAGO14              ; string font
       #RL 210,90,320,120         ; delete area behind text
       #ZL 210,90, "Deutsch" ; write actual language
       #UI 210,120,100            ; place picture

Macro 1[ENGLISH]:
       #ZF CHICAGO14              ; string font
       #RL 210,90,320,120         ; delete area behind text
       #ZL 210,90, "English" ; write actual language
       #UI 210,120,100            ; place picture

Macro 1[FRENCH]:
       #ZF CHICAGO14              ; string font
       #RL 210,90,320,120         ; delete area behind text
       #ZL 210,90,"Fran"135"ais"  ; write actual language
       #UI 210,120,100            ; place picture

Macro 1[ITALIAN]:
       #ZF CHICAGO14              ; string font
       #RL 210,90,320,120         ; delete area behind text
       #ZL 210,90, "Italiano"     ; write actual language
       #UI 210,120,100            ; place picture
```

11.19 Automatic Macro - BEGINNER

A little animation with the help of automatic macros. There are further examples available, containig information about Outputs (see BEGINNER - Outputs 91), a second AutomaticMacro example (see EXPERT - Automatic_Macro.kmc 87) and another ProcessMacro (see EXPERT – Prozess Macro.kmc 89).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Macro\

File:
BEGINNER – AutomaticMacro_as_animation.kmc

Commands:
#MJ, #UI

Open file in KitEditor

```
eDIP320-8   "Automatic Macro as animation"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>
Picture: 7 <Clown\clown1.bmp>
Picture: 8 <Clown\clown2.bmp>
Picture: 9 <Clown\clown3.bmp>
Picture: 10 <Clown\clown4.bmp>
Picture: 11 <Clown\clown5.bmp>
Picture: 12 <Clown\clown6.bmp>
Picture: 13 <Clown\clown7.bmp>
Picture: 14 <Clown\clown8.bmp>
Picture: 15 <Clown\clown9.bmp>



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

       #UI 40,0,6                     ; place logo

;---- Start animation ----
       #UV 4                          ; replace old pictures
       #MA 1,9,3                      ; run macros 1..6 automatically
                                      ; MA = Cyclical Mode, Pause is 3/10s

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

Macro: 1
       #UI 104, 100, 7

Macro: 2
       #UI 104, 100, 8

Macro: 3
       #UI 104, 100, 9
```

```
Macro: 4
       #UI 104, 100, 10

Macro: 5
       #UI 104, 100, 11

Macro: 6
       #UI 104, 100, 12

Macro: 7
       #UI 104, 100, 13

Macro: 8
       #UI 104, 100, 14

Macro: 9
       #UI 104, 100, 15
```

11.20 Automatic Macro - EXPERT

A little animation with the help of automatic macros. There are further examples available, containig information about Outputs (see BEGINNER - Outputs 91), an AutomaticMacro example (see BEGINNER - AutomaticMacro_as_animation.kmc 85) and another ProcessMacro (see EXPERT – Prozess Macro.kmc 89).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Macro\

File:
EXPERT – Automatic_Macro.kmc

Commands:
#YW, #YM

Open file in KitEditor

```
eDIP320-8   "Automatic Macro"
...
...
...
;-------------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ;using constants makes it easier
Picture: LOGO <howtouselogo_230x50.bmp>

;-------------------------------------------------------------------------------
;define constants for macors
Mn1 = 100
Mn2 = Mn1+1
Mn3 = Mn2+1
Mn4 = Mn3+1
Mn5 = Mn4+1
Mn6 = Mn5+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 BIGZIF57                            ; set font to no. 7 (BIGZIF57)
time=5
       #MJ Mn1,Mn6,time                 ; run macros 1..6 automatically
                                        ; MJ = Ping Pong Mode

;---- Place characters in different Macros -----
x=XPIXEL/2
y=80
Macro: Mn1
       #ZC x,y, "1"

Macro: Mn2
       #ZC x,y, "2"

Macro: Mn3
```

```
        #ZC x,y, "3"

Macro: Mn4
        #ZC x,y, "4"

Macro: Mn5
        #ZC x,y, "5"

Macro: Mn6
        #ZC x,y, "6"
```

11.21 Process Macro - EXPERT

A little animation with the help of process macros. There are further examples available, containig information about Outputs (see BEGINNER - Outputs 91) and another AutomaticMacro example (see BEGINNER - AutomaticMacro_as_animation.kmc 85)

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Macro\

File:
EXPERT – Prozess Macro.kmc

Commands:
#MD

Open file in KitEditor

```
eDIP320-8 "Prozess Macro"
...
...
...
;-------------------------------------------------------------------------------
Path <..\..\..\bitmaps\color\>
Path: <..\..\..\Bitmaps\monochrome>
LOGO = 6 ;using constants makes it easier
Picture: LOGO <howtouselogo_230x50.bmp>

;-------------------------------------------------------------------------------
;define constants for processmacros
Mn1 = 100
Mn2 = Mn1+1
Mn3 = Mn2+1
Mn4 = Mn3+1
Mn5 = Mn4+1
Mn6 = Mn5+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 BIGZIF57                        ; set font to no. 7 (BIGZIF57)
       #MD 1,3, Mn1,Mn6, 5          ; define macro process 1, pingpong mode, call
automatic macro 1 to 6 delay 5/10s

;---- Place Digit -----
;---- Place characters in different Macros -----
x=XPIXEL/2
y=80
Macro: Mn1
       #ZC x,y, "1"

Macro: Mn2
       #ZC x,y, "2"
```

```
Macro: Mn3
       #ZC x,y, "3"

Macro: Mn4
       #ZC x,y, "4"

Macro: Mn5
       #ZC x,y, "5"

Macro: Mn6
       #ZC x,y, "6"
```

11.22 Outputs - 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 85), a second AutomaticMacro example (see EXPERT - Automatic_Macro.kmc 87) and another ProcessMacro (see EXPERT – Prozess Macro.kmc 89).

Folder:
\ELECTRONIC_ASSEMBLY_LCD-Tools-Portable\Data\eDIP - intelligent graphic displays\eDIP320-8\How to use\Macro\

File:
BEGINNER – Outputs.kmc

Commands:
#YW

Open file in KitEditor

```
eDIP320-8   "Output"
...
...
...
;-----------------------------------------------------------------------------
Path: <..\..\..\Bitmaps\monochrome>
Picture: 6 <howtouselogo_230x50.bmp>

;-----------------------------------------------------------------------------
Macro: MnAutoStart
;--- Place ELECTRONIC ASSEMBLY logo ---
       #TA                                 ; Terminal off
       #UI 40,0,6                          ; place logo

;--- Place 1 button ---
       #AF 4                               ; touch font
       #AE 4                               ; touch frame
       #AT 50,100,140,120,1,0,"Port 1 toggle"     ; place touchbutton with touchmacro no.
1



;-----------------------------------------------------------------------------
TouchMacro: 1
       #YW 1, 2       ; toggle output 1 (Pin 6 if interface is RS232)
```