



## **EA eDIP240-7 compiler manual**

Oktober 2008  
© ELECTRONIC ASSEMBLY GmbH

# Table of Contents

<b>Part I eDIP240compiler</b>	<b>2</b>
<b>Part II Syntax rules</b>	<b>3</b>
<b>Part III Compiler Options</b>	<b>5</b>
1 General .....	5
2 Transfer .....	6
3 Font .....	7
4 Picture .....	9
5 SystemMacros .....	10
6 Macro .....	11
7 TouchMacro .....	12
8 MenuMacro .....	13
<b>Part IV EA eDIP240-7 commands</b>	<b>14</b>
1 Terminal .....	14
2 Text .....	15
3 Draw .....	16
4 Flashing .....	17
5 Bitmap .....	18
6 Clipboard .....	19
7 Bargraph .....	20
8 Macros .....	21
9 Touch .....	22
10 Menu .....	24
11 Other commands .....	25

# 1 eDIP240compiler

## General

The EA eDIP240-7 is able to store many pictures, fonts and macros in internal EEPROM memory. The EA KIT Editor is a powerful, free of charge software tool to create those macros and to store the pictures and fonts very easily.

The EA KIT Editor combines 3 functions:

- The editor itself which allows a simple definition of the macros, pictures and fonts like a standard text editor.
- The compiler which translates the text into the uploading code and shows up syntax error.
- The transmitter which search the right connection and uploads the data into the EA eDIP240-7.

## 2 Syntax rules

---

<b>ESC</b>	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.
<b>Comma</b>	The comma is used to separate the parameters of a macro.
<b>Numbers</b>	All numbers are converted to binary values. Decimal, hexadecimal and binary numbers can be written. Example: 163(dez) = \$A3(hex) = %10100011(bin)
<b>Comments</b>	Comments must begin with a semicolon. Example: ; this is a comment
<b>Text</b>	Text (strings) must be enclosed within quotation marks "" or ''. ASCII numbers can also be entered directly. Example (output of "abc-def"): #ZL 0,0, "abc",45,'def'
<b>Commands</b>	Command letters and parameters specified in the EA eDIP240-7 data sheet are valid. Two exceptions facilitate the creation of command lines: <ol style="list-style-type: none"><li>1. The &lt;NUL&gt; is appended automatically by the compiler. This means commands in which a string is output, the &lt;NUL&gt; no longer has to be entered as the end identifier. Example: #ZL 0,0,"Text"</li><li>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"</li></ol>
<b>Constants</b>	Words without quotation marks are interpreted as constants, which have to be defined first. The name of a constant can have be up to 29 characters and must begin with a letter followed by letters, numbers or underscores. Up to 1024 constants can be defined. Please note that Compiler Options like e.g. INFO or MACRO can not be used. Example: CORNER_X=5; the word CORNER_X is replaced with immediate effect by the value 5.
<b>Upper / lower case</b>	No difference is made between upper case and lower case.

---

## Calculating

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

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

following C-style operations are also possible:

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

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

## 3 Compiler Options

### 3.1 General

`eDIP240-7 "title"`

Defines EA eDIP240-7 as target. "title" is a short description for the project. It is shown on the display when uploading the EEPROM memory of the module.

---

`SIMULATION`

Starts the PC simulator after compiling the .kmc file. All contents of the screen can be simulated on the PC.



`DESTINATION "new.eep"`

Specifies a new file name for the EEPROM upload file.

---

`INCLUDE <file>`

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

---

`PATH <path>`

Sets a new path to find the following files.

## 3.2 Transfer

`COMx: baud`

With this statement the COM port and baud rate is defined when the EA eDIP240-7 is connected to the RS-232 (e.g. via EA 9777-1RS232),.

---

`USB: "device"`

If the EA STARTeDIP240 is connected to the USB, "device" is "eDIP Programmer" (only Windows 2000/XP/Vista).

---

`RS485ADR: adr`

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

---

`VERIFY`

Verifies the complete contents of the EEPROM memory after upload.

### 3.3 Font

**FONT:** nr,<file>

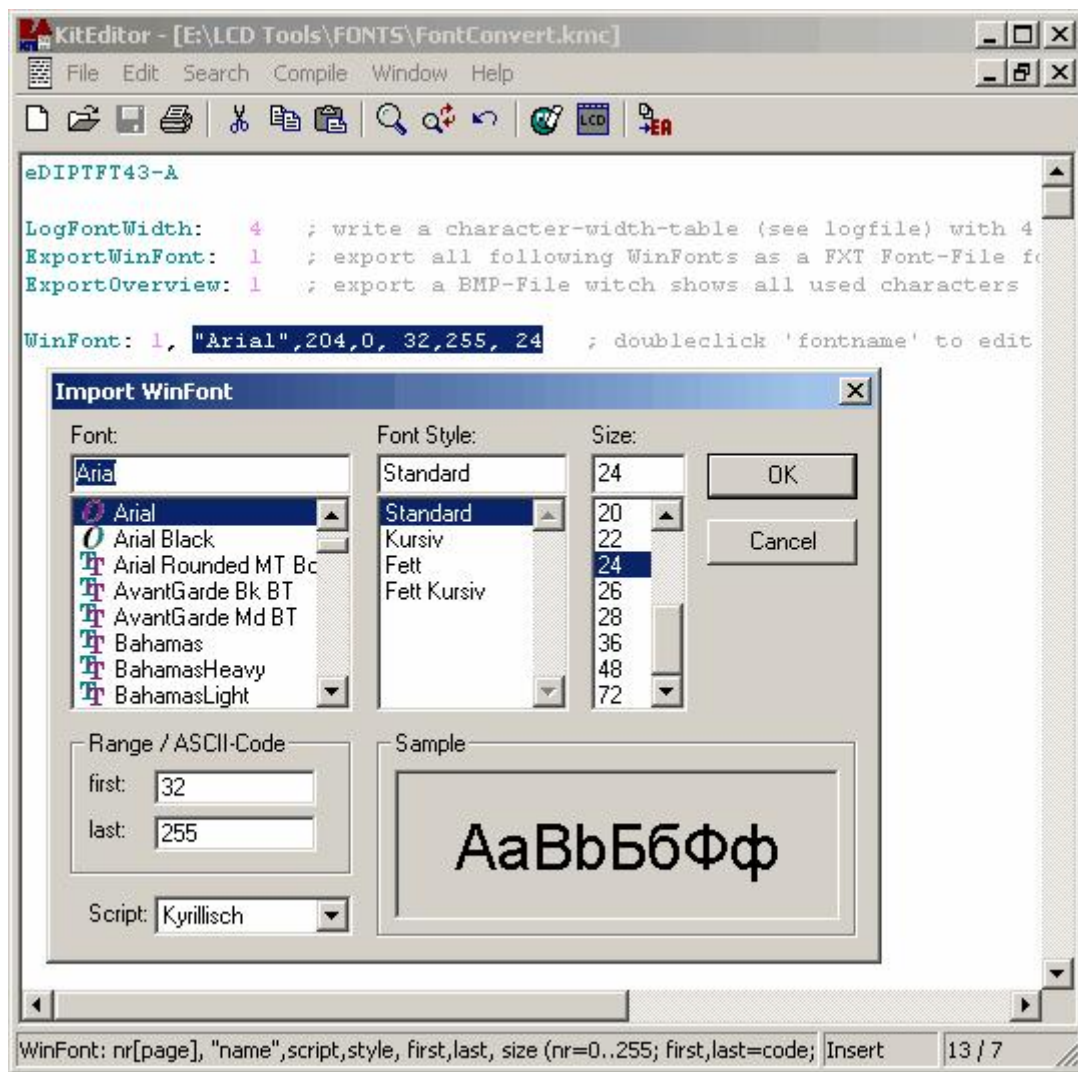
Defines a font file which will be assigned to the number nr (1..15).  
<file> can be \*.FXT format.  
Font number 0 is internal 8x8 terminal font and can not be changed

#### Using Windows fonts:

All Windows fonts can be converted with the eDIPTFT compiler into the FXT-Format.

Just open the file:

'..\ELECTRONIC ASSEMBLY LCD Tools\Fonts\FontConvert.kmc'  
choose your favorite font and then compile (F5) this creates a FXT fontfile and an overview bitmap.



Example:

convert an cyrillic Arial font with 24 dot height:

After compiling the FXT-file 'Font1\_Arial\_RUSSIAN\_N\_32-255\_24.fxt'  
and an overview bitmap is generated.

Font1\_Arial\_RUSSIAN\_N\_32-255\_24.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	{		}	~	□
\$80 (dez: 128)	Ѡ	Ѓ	,	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ѡ	ѡ
\$90 (dez: 144)	ђ	‘	’	“	”	•	–	—	□	™	љ	›	њ	ќ	ћ	ѡ
\$A0 (dez: 160)		Ў	ў	Ј	Ѡ	Ѓ	Ѓ	Ѓ	Ё	©	Є	«	¬	-	®	ї
\$B0 (dez: 176)	°	±	І	і	ѓ	μ	¶	·	ё	№	є	»	ј	ѕ	ѕ	ї
\$C0 (dez: 192)	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
\$D0 (dez: 208)	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
\$E0 (dez: 224)	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
\$F0 (dez: 240)	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

## 3.4 Picture

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

It is convenient to store all bitmap in EEPROM; this will save transfer time via serial interface. The statement PICTURE defines a bitmap <file>with nr (0..255). <file> has to be a monochrome BMP. Optionally 2 different pictures can be defined as <file1> and <file2>. <file1> is for touch key/ switch and <file2> will be used if the touch key/ switch is pressed.

The pictures can be used with the [Bitmap commands](#).<sup>18</sup>

## 3.5 SystemMacros

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

---

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

---

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

---

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

## 3.6 Macro

MACRO: `nr`

Defines a normal macro with number `nr` (0..255). This macro will be executed with the command `#MN nr`<sup>[21]</sup>

A series of macros occurring one after the other can be called cyclically (movie, hourglass, multi-page help text) see command for [Automatic \(normal-\) macros](#)<sup>[21]</sup>.

These automatic macros continue to be processed until either a command is received via the interface or a touch macro with a corresponding return code is activated.

These macros are also called by macro processes at defined intervals (see command for [Macro processes](#)<sup>[21]</sup>). Macro processes are not interrupted when commands are received from the interface or when touch macros are triggered.

### 3.7 TouchMacro

`TOUCHMACRO: nr`

Defines a touch macro with number nr (0..255). This macro will be executed if a touch key / switch with the return code nr is defined and the touch key/switch is pressed or by command `#MT nr`<sup>[21]</sup>.

## 3.8 MenuMacro

`MENUMACRO: nr`

Defines a menu macro with number nr (0..255). This macro will be executed automatically after choosing an menuentry or by command `#MM nr`<sup>[21]</sup>.

## 4 EA eDIP240-7 commands

### 4.1 Terminal

#### Terminal commands:

Position cursor	#TP n1,n2	n1=column; n2=line; origin upper-left corner (1,1)
Cursor on/off	#TC n1	n1=0: Cursor is invisible; n1=1: Cursor flashes;
Save cursor position	#TS	The current cursor position is saved
Restore cursor position	#TR	The last saved cursor position is restored
Terminal off	#TA	Terminal display is switched off; outputs are rejected
Terminal on	#TE	Terminal display is switched on

#### Terminal output:

String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal
Output version	#TV	The version no. is output in the terminal e.g. "EA eDIP240-7 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

## 4.2 Text

### Text settings:

Set font	#ZF n1	Set font with the number nr = 0..15 (see compiler option <a href="#">FONT</a> [71]:)
Font zoom factor	#ZZ n1,n2	n1 = X-zoom factor (1x to 4x); n2 = Y-zoom factor (1x to 4x)
Additional line spacing	#ZY n1	Insert n1=0..15 dots between two lines as additional spacing
Text angle	#ZW n1	Text output angle n1=0: 0°; n1=1: 90°
Text link mode	#ZV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace
Text flashing attribute	#ZB n1	n1: 0=no flashing; 1=Text flashing on/off; 2=Text flashes inversely

### 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) cancels the special function of ' ', '~', '@' and '\'
Output string centered	#ZC x,y,"text..."	A string (text...) is output centered to x,y. Several lines are separated by the character ' ' (\$7C, pipe). Text between two '~' (\$7E) characters flashes on/off. Text between two '@' (\$40) characters flashes inversely. The character '\' (\$5C, backslash) cancels the special function of ' ', '~', '@' and '\'
Output string right justified	#ZR x,y,"text..."	A string (text...) is output right justified to x,y. Several lines are separated by the character ' ' (\$7C, pipe). Text between two '~' (\$7E) characters flashes on/off. Text between two '@' (\$40) characters flashes inversely. The character '\' (\$5C, backslash) cancels the special function of ' ', '~', '@' and '\'
String for terminal	#ZT "text..."	Command for outputting a string (text...) from a macro to the terminal

## 4.3 Draw

### 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
Show clip-board	#DC	Show content of clip-board Standard display output is no longer visible
Show current	#DN	Switch back to noraml operation Standard display output is visible

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

### Change/draw rectangular areas:

Delete area	#RL x1,y1,x2,y2	Delete an area from x1,y1 to x2,y2 (all pixels off)
Fill area	#RS x1,y1,x2,y2	Fill an area from x1,y1 to x2,y2 (all pixels on)
Invert area	#RI x1,y1,x2,y2	Invert an area from x1,y1 to x2,y2 (invert all pixels)
Area with fill pattern	#RM x1,y1,x2,y2,no	Draw area from x1,y1 to x2,y2 with pattern no (always set)
Draw box	#RO x1,y1,x2,y2,no	Draw rectangle from x1,y1 to x2,y2 with pattern no (always replace)
Draw frame	#RR x1,y1,x2,y2,no	Draw frame of type n1 from x1,y1 to x2,y2 (always set)
Draw frame box	#RT x1,y1,x2,y2,no	Draw frame box of type n1 from x1,y1 to x2,y2 (always replace)

## 4.4 Flashing

### Flashing areas:

Delete flashing attribute	#QL x1,y1,x2,y2	Delete the flashing attribute from x1,y1 to x2,y2
Flashing 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 from x1,y1 to x2,y2
Set flashing time	#QZ n1	Set the flashing time n1=1..15 in 1/10sec; 0=deactivate flashing

## 4.5 Bitmap

### Bitmap settings:

Image zoom factor	#UZ n1,n2	n1 = X-zoom factor (1x to 4x) n2 = Y-zoom factor (1x to 4x)
Image angle	#UW n1	output angle of the image n1=0: 0°; n1=1: 90°
Image link mode	#UV n1	n1: 1=set; 2=delete; 3=inverse; 4=replace; 5=inverse replace;
Image flashing attribute	#UB n1	n1: 0=no flashing; 1=image flashing on/off; n1: 2=image flashing inversely

### Output bitmaps:

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

### Hardcopy:

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

## 4.6 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

## 4.7 Bargraph

### Define bargraphs:

Define bargraph	#BR #BL #BO #BU no,x1,y1,x2,y2, sv,ev,type,pat	Define bargraph with number no to L(ef), 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
-----------------	--	---

### 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

## 4.8 Macros

### Run macros:

Run macro	#MN nr	Call the (normal) macro with the number nr (max. 7 levels) (see compiler option <a href="#">MACRO</a> $\uparrow_{11}$ : )
Run touch macros	#MT nr	Call the touch macro with the number nr (max. 7 levels) (see compiler option <a href="#">TOUCHMACRO</a> $\uparrow_{12}$ : )
Run menu macro	#MM nr	Call the menu macro with the number nr (max. 7 levels) (see compiler option <a href="#">MENUMACRO</a> $\uparrow_{13}$ : )

### Automatic (normal-) macros:

Macro with delay	#MG n1,n2	Call the (normal) macro with the number n1 in n2/10s Execution is stopped by commands (e.g. receipt or touch macros).
Autom. macros once only	#ME n1,n2,n3	Automatically run macros n1 to n2 once only; n3=pause in 1/10s Execution is stopped by commands (e.g. receipt or touch macros).
Autom. macros cyclical	#MA n1,n2,n3	Automatically run macros n1 to n2 cyclically; n3=pause in 1/10s Execution is stopped by commands (e.g. receipt or touch macros).
Autom. macros ping pong	#MJ n1,n2,n3	Automatically run macros n1 to n2 to n1 (ping pong); n3=pause in 1/10s Execution is stopped, for example, by receipt or touch macros

### Macro processes:

Define macro process	#MD no,type,n3,n4,zs	A macro process with the number no (1 to 4) is defined (1=highest priority). The 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

## 4.9 Touch

### Touch presets:

Touch border form	#AE nr	Set the border n1 for the display of touch keys/switches
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.

### Label font presets:

Label font	#AF nr	Set font with the number n1 (0 to 15) for touch key label (see compiler option <code>FONT</code> (7) : )
Label zoom factor	#AZ n1,n2	n1=X-zoom factor (1x to 4x); n2=Y-zoom factor (1x to 4x)
Additional line spacing	#AY n1	Insert n1=0..15 dots between two lines as additional spacing
Label angle	#AW n1	Label output angle: n1=0: 0°; n1=1: 90°

### Define touch key/switch:

Define touch key	#AT x1,y1,x2,y2, downCode,upCode,"text.." #AU x,y, n1, downCode,upCode, "text.."	key remains depressed as long as there is contact
Define touch switch	#AK x1,y1,x2,y2, downCode,upCode,"text.." #AJ x,y, n1, downCode,upCode, "text.."	status of the switch toggles after each contact

#AT: The area from x1,y1 to x2,y2 is drawn with actual border and defined as a key  
#AK: The area from x1,y1 to x2,y2 is drawn with actual border and defined as a switch  
#AU: Image number n1 is loaded to x,y and defined as a key  
#AJ: Image number n1 is loaded to x,y and defined as a switch  
'downCode':(1-255) return/touchmacro when key pressed  
'upCode': (1-255) return/touchmacro when key released  
(downCode/upCode = 0 press/release not reported).  
"text...": this is a string that is placed in the key with the current touch font.  
The first character determines the alignment of the text (C=centered, L=left justified, R=right justified)  
This is followed by a string "text..." that is placed in the key with the current touch font  
Multiline texts are separated with the character '|' (\$7C, dec: 124)

### Define touch menu:

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

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

**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.

**Global settings:**

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

**Other touch functions:**

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

## 4.10 Menu

### Settings for menu box/touch menu:

Set menu font	#NF n1	Set font with the number n1 (0 to 15) for menu display
Menu font zoom factor	#NZ n1,n2	n1=X-zoom factor (1x to 4x); n2=Y-zoom factor (1x to 4x)
Additional line spacing	#NY n1	Insert n1=0..15 dots between two menu items as additional spacing
Menu angle	#NW n1	Menu display angle: n1=0: 0°; n1=1: 90°
Touch menu automation	#NT n1	n1=1: Touch menu opens automatically n1=0: Touch menu does not open automatically; instead, the request 'ESC T 0' to open is sent to the host computer, which can then open the touch menu with 'ESC NT 2'

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

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

## 4.11 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 eDIP240-7 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=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 (=5/7-bit binary value). n1=1 to 5/7: 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

### 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).