EA uniTFT

Multifuncion 5" TFT HMI

FEATURES

- WITH and WITHOUT TOUCHPANEL
- ANALOGUE RESISTIVE TOUCH OR CAPACITIVE - PCAP
- OBJECT BASED SCREEN DESIGN
- CHANGE OBJECTS: SIZE, SHAPE, COLOR, VISABILITY, CONTENT
- VECTOR GRAPHICS, LOSS FREE ROTATION AND ZOOM
- ALPHA BLENDING, MOVING OBJECTS
- VECTORIZED CHARACTER SET AS ASCII AND UNICODE
- SINGLE SUPPLY 3.3 V
- 7 INTERFACES: USB, 2 x I²C, 2 x SPI, 2 x RS232
- 16 DIGITAL I/O ONBOARD, EXPANDABLE UP TO 125
- 4 ANALOGUE INPUTS
- PWM OUTPUT
- BUILT-IN RTC INCL. BATTERY BACKUP
- MICRO SD-CARD USED FOR PICTURES, FONTS, MACROS

ORDERING CODES

DISPLAYS	
MULTI FUNCTION 5" TFT 800x480 DOTS, 24 BIT WITH BACKLIGHT	**EA uniTFT050-A**
INCL. RESISTIVE TOUCHPANEL	**EA uniTFT050-ATP**
INCL. CAPCITIVE TOUCHPANEL - PCAP	**EA uniTFT050-ATC**
ACCESSORIES	
5" uniTFT WITH PCAP PLUS PROGRAMMER BOARD	**EA QUICKuniTFT050C**
5" uniTFT WITH RES: TOUCH PLUS PROGRAMMER BOARD	**EA QUICKuniTFT050P**
MICROMATCH CONNECTOR THT, 26 PIN (2 PCS. REQUIRED)	**EA B2B127M-26T**
MICROMATCH CONNECTOR IDC CRIMP CONNECTION, 26 PIN (2 PCS. REQUIRED)	**EA B2B127M-26Q**

TABLE OF CONTENT

REVISION

Datum	Version / Firmware	Beschreibung
23.08.2016	0.9	First release

GENERAL INFORMATION

METHOD OF WORKING OF THE DISPLAY

The representation on the display is effected based on the commands given by the user. Every item on the display is an independent object and can be manipulated. The appearance, i.e. the colour data, the fonts, the line strengths, etc. is merged to form styles .

OBJECTS

In order to generate various objects, the corresponding commands are available. Every image, every text and every button is a so-called object. Every object needs to be endorsed with an object ID which makes it clearly identifiable. If an object ID is assigned, and if said object ID is re-assigned the previous object is overwritten. Thus, when using objects that occur on multiple screen pages, care is needed so that the latter are not overwritten.

STYLES

There are various styles based on which objects can be displayed, such as colour, line strength or transparency. The corresponding commands and examples can be inferred from the Style Sheets section.
Just as with the objects, styles are saved in corresponding IDs, which can also be overwritten. The various style groups have their own ID range. That means that a button style and a drawing style with the ID 1 can exist alongside one another simultaneously. The maximum number of styles is 255 for every range, which is why using a local definition needs to be taken into consideration when using very many styles.

DRAWING STYLE AND LINE STYLE

Except in the case of images, every object is generated with a border and a filling. The drawing style determines precisely those. In that respect, both the filling and the border (line) can be defined, omitted entirely or designed to be transparent. Besides having a plain colour filling, a colour gradient is, moreover, also possible. In the case of the line, it is not possible to define a colour gradient. In addition, both a dash pattern (dotted) can be defined, and also the ends rounded off. The line style is a component of the drawing style, which is why it is recommended for the overview to always specify both together.

TEXT STYLE

TheText Style needs to be defined if it is intended to work with strings. The latter contain the information on the font used, as well as its formatting. As a string is likewise an object, reference is also made, in the text style, to an existing drawing style, in order to provide the font with a filling and border. For performance reasons, we recommend a drawing style without a border (line).

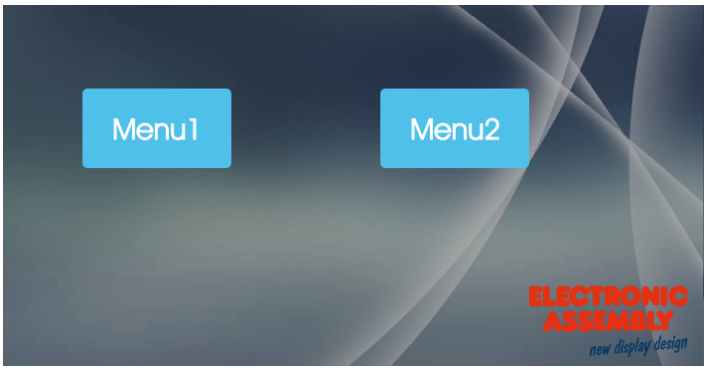
BUTTON STYLE

Text and drawing styles form the basis for the Button Style. In order to design touch control buttons which have a different visual appearance in the pressed state, in comparison to the unpressed state, in certain circumstances multiple drawing and text styles are required. Information, such as touch feedback, e.g. playing short jingles upon activation or enlarging the button is likewise stored in this style.

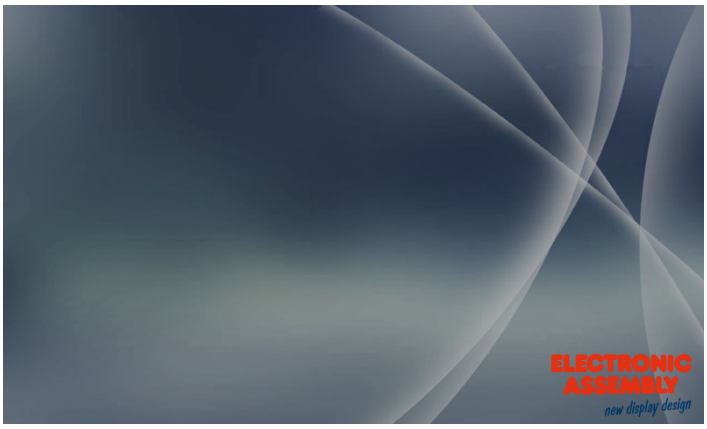
STYLES AND OBJECTS

In order to be able to switch between various different screen pages, all the existing objects first always need to be deleted Precisely in menus, you can often find a fixed structure, so that individual objects only need to be

deleted in a targeted manner, and others can remain in place.
Below is an example, to make it clear how handling objects works.
Once the display has been started, a main menu is opened, with a background (ID=100), Logo(ID=101) and two touch buttons(ID= 1,2) for selecting the sub-menu. A button style(number=1) is, furthermore, defined for the touch control buttons.



The background, logo and button style should also be present in the sub-menus. In order not to overwrite these objects, using the object IDs should be discontinued. A heading with a corresponding style, as well as a line, are supposed to be generated in Menu 1. All objects still existing that are no longer required are to be deleted in advance.In this case that is Objects 1 to 99. The deletion is performed by entering the delete command for objects(→ #ODI 1-99). he result can be seen in the figure below.



Naturally, the objects in Menu 1 and Menu 2 can be used in the same way.

COMMAND SYNTAX

A command always begins with '#'. Subsequently there is a 3-digit sequence of digits - the command code. Depending upon the command code, further parameters are required. In order to separate the parameters, one of the following characters needs to be used:

- Space
- Comma(,)
- Full stop(.)
- A semicolon (;) (only and mandatorily at the end of a string)
- Specified range of multiple object IDs: '-' sign: e.g. 1-5, instead of 1,2,3,4,5.

The command always needs to be concluded by an LF (line feed) Should the line feed not exist, the command is not executed.
Example::

- Individual command to generate Line Style 1 — *#CLS1 $3B7EAE,100,1,0,1(Line Feed)*
- Multiple commands to generate an animation of Object 1 — *#AOA1 501,0 (Line Feed)*
  #AOT1 1,250,100,100 (Line Feed)

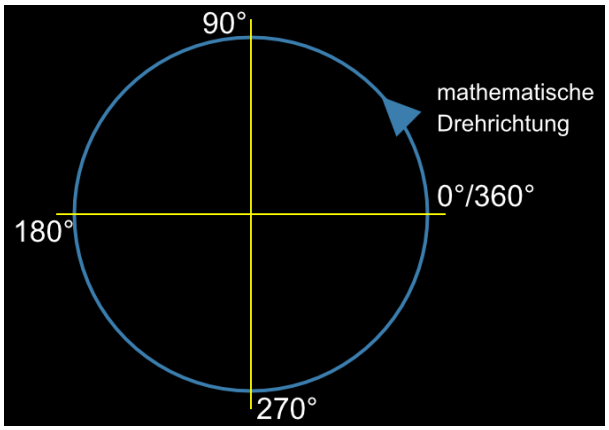
There are commands which relate to files. Those are likewise written in quotation marks or inverted commas. These commands have recorded in them, as their reference point, the default folder, and thus an automated path specification.

COMMAND PARAMETERS

Parameters that are written in **GRAY** n the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** must be transferred.

ANGELS

Angles are specified in the mathematical direction of rotation, i.e. anti-clockwise. It is likewise possible to enter negative angles. The direction of rotation and the angular distribution can be discerned in the figure below.



The co-ordinate system extends in a range from 800(x) * 480(y). The origin (0/0) is in the lower left-hand corner.

COMMENTS

The option exists to add comments in the command input in macro files. Considerations concerning command sequences can be explained thereby, and a better understanding ensured. A comment begins with#-(hash key minus) and applies until the end of the line, i.e. once the comment line is supposed to contain a wrap, the next line likewise has to begin with #- in order to continue the comment.

OBJECT AREA

In the case of commands having the property of influencing one or more objects (marked by: “Object ID”), the object area can be specified by a hyphen, “-”- In the example contained in the section, Styles and Obects, the application is highlighted.

ENTERING FIGURES

Input	Definition
123	Decimal handover as ASCII character/s
$5A	Hexadecimal handover as ASCII character/s
%101001	Binary handover as ASCII character/s
?x	Code of a character (Unicode /ASCII)
R0..R255	Handover of the register value
Q0..Q255	Accept register value indexed = R(R0..255).
(...)	Accept result of the calculation string
G len32 data...	Submit binary data: G len 32-bit - binary, followed by binary data
!string!	Infer values and strings from string file, replacement

ENTERING STRINGS

Should strings be used as parameters, it should be remembered that the latter need to be placed in quotation marks ("") or inverted commas ('), and be ended by a semicolon (;). Should a string be the last parameter in the command code, no semicolon needs to be placed at the end of the string. The maximum length permitted for any string is 255 characters. A line break within a string is implemented with the pipe sign '|' or new line '\n'.

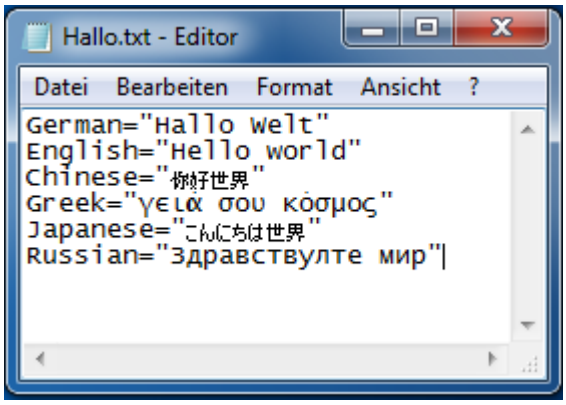
- Beispiel: *"string1" ; 'string2'*

Eingabe	Definition
"Hello"32'World'	Eingabe eines Strings.Da kein ; vorhanden ist werden die Strings zusammengesetzt. Der Code dazwischen wird ebenso eingefügt. Ausgabe: Hello World
"Hello\nWorld" = "Hello World"	Mehrzeilige Eingabe eines Strings
S0..255	Stringregister übernehmen.
T0..255	Stringregister indiziert übernehmen, Stringregisternummer aus Register S(R0..255).
U"Hello"	Zeichen nach dem U als 16-Bit Unicode (bis zum nächsten # oder V auch CR + LF)
V"Hello"	Zeichen nach dem V als 8-Bit ASCII (bis zum nächsten # oder U auch CR + LF)
!string!	Werte und Strings aus Stringdatei entnehmen und einsetzen.

STRING FILE

String files are text files written externally. String files can be used as a sort of database of strings. The strings stored in such a way can be accessed by the name for accessing the string being written between two exclamation marks *!* No quotation marks*""*may be used, as otherwise only the access as a string is shown, however the desired string file is not loaded. To use this function, the text file created only needs to be found in the String folder of the SD card, and be loaded by the command to load a string file(#VFL) . By way of clarification, an example is given below:

The text file *Hallo.txt* created externally and copied onto the SD card looks as follows:



Silt serves the purpose of using multilingualism in regard to the statement “Hello, World”. The text file is first of all loaded through *#VFL "Hallo"*. Should a string now be placed by the commandl *#SSP 1, 1, 400, 240, 5 !German! "|" !English!*, the result looks as follows:

The pipe sign | is necessary for the line break. So that it counts for the string to be displayed, the latter needs to be written in quotation marks *""*Without using the string file, the above command would read *#SSP 1, 1, 400, 240, 5 "Hallo Welt" "|" "Hello World"*.

PATH DETAILS, FILES AND FORMATTING

PATH DETAILS

There are two ways of indicating the path. In absolute and relative terms, respectively.

Designation	Example
Absolute path details	</Ordner/Unterordner>
Relative path details	<.../Unterordner>

The absolute path details should be used to work with files outside the project path set. The project path, which is defined by the command XPS serves to simplify matters. It is not necessary to specify the superordinate directories. By entering “P:” prior to specifying the path, the project paths are automatically added.
If a file is to be accessed in the project folder “picture” set, that can be done absolutely or relatively, as shown in the following example:

Absolute path details: </Ordner/Ordner/Ordner/Projekt/picture/Test.epg
Relative path details: <p:/picture/Test.epg → The project path has been set as "#XPS </Ordner/Ordner/Ordner/Projekt>"

When specifying the path, upper and lower case are to be taken into account.

One more simplification::
There is a series of default folders that are automatically created in the project folder and may NEVER be changed, as, otherwise, command parameters that automatically access these folders no longer work!
This folder structure is defined as follows:

Example:

#PPP1, "Test"; 100, 100

With this command, the “test” image is shown on the display at the position 100,100.

An alternative programming step would be: *#PPP1, <P:/picture/Test.epg>, 100, 100* .

The same occurs with macros, string files (rather than strings), patterns, buttons (images), sounds, instruments and fonts. The default folders may never be changed. Sub-folders in the default folders are allowed, however need to be specified as well in file names.

DATA TYPES

Zur Verwendung von Bild- oder Sounddateien müssen diese umgewandelt beziehungsweise konvertiert werden. Das geschieht automatisch durch die Verwendung der Software *uniSKETCH*, indem dort die Dateien eingebunden werden.
Das uniTFT kann nur mit folgenden Dateitypen arbeiten:

Dateityp	Beschreibung
.evg	Format für Vektorgrafiken
.epa	Format für animierte Bilder
.epg	Format für Pixelgrafiken
.esd	Format für Sounddateien
.evf	Format für Vektorfonts
.epf	Format für Pixelfonts
.epi	Format für Pixelinstrumente
.emc	Format für Macros (Das Makro *start.emc* startet automatisch nach PowerOn oder Reset.)
.txt	Format für Stringfiles

Andere Dateiformate können zwar im internen Speicher abgelegt, jedoch nicht vom uniTFT genutzt werden.

FORMATTED STRINGS

Formatted strings rely on the “printf” output function of C. It is thereby possible to display a string that, for example, contains numerical values of a calculation. The maximum length permitted for any formatted string is 63 characters.

Designation	Character	Description
Integer	**%d**	Displays number as a string.
Double	**%f**	Displays numbers with a decimal place as a string.
Hexadezimal	**%x, %X**	Displays hexadecimal numbers as a string.

A formatted string could look as follows:

#SFP1, 3, 400, 240, 5, "Integer: %d, Float: %.5f, Hexadezimal: %X"; (1+1+1), (3.14159265359), (9+6)

With corresponding text and drawing styles, the result appears as follows:

DATE AND TIME

The file format is a special form of the formatted string and describes the date, including the time. A further explanation on the date format can be found here (#WDF).
In the table below, the input commands for the file format are listed.

Eingabe	Definition
%[]h	Hour
%[]m	Minute
%[]s	Second
%[]D	Day
%[]M	Month
%[]Y	Year
%{}W	Name of weekday as a string
%{}N	Name of month as a string
[] (optional)	**0** = Two digits with a preceding 0 (Default) **1** = At least one digit without a preceding character **2** = Two digits with preceding blank spaces **4** = Four digits (default for years)
{} (optional)	**0...9** = Display the first X characters from the string.

Example:

#SDP

Date and Time

Placing a formatted date

```
#SDP  1, 1, 50, 50, 17, "%W the %D. %3N %Y, %1h:%m"
```

Thursday the 21. Apr 2016, 11:37

IMAGES

The module internally uses a special image format (*.epg). The conversion needs to be done externally. The Windows program EA uniSketch offers the most comfortable option to have most image formats converted. Some commands make it possible to save the screen content on the SD card or transfer image data directly via the serial interface. Various image formats are available here.

IMAGE FORMATS

The following image formats apply to screenshots/hard copies of the display or the video input.

Input	Definition	Storage space requirement/execution time
1	BMP 24Bit → True Color	Very high
2	BMP 16Bit → High Color	High
3	BMP 8Bit grey → grey-scale image	Low
11	epg 32Bit → True Color with alpha channel	Very high
12	epg 16Bit → High Color	High
13	epg 8Bit grey → grey-scale image	Low
21	epg 32Bit RLE compressed → see above	High
22	epg 16Bit RLE compressed → see above	Medium
23	epg 8Bit grey RLE compressed → see above	Very low

POLYLINE AND POLYGONS

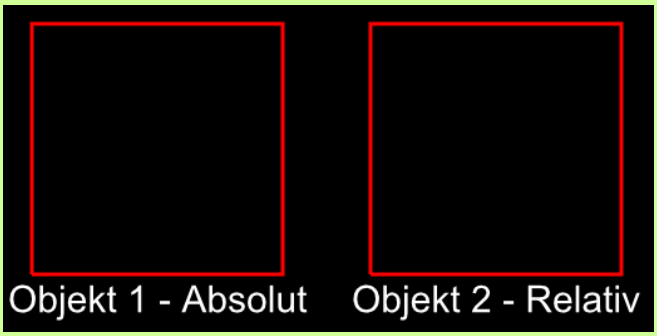
With the polyline (#GPL) and polygon (#GPF) commands, virtually all desired forms can be generated. Each section is designated a segment.

SEGMENT

A segment is generally a section or a part of something whole.
In the present command application, the segments fulfil two primary aspects. Firstly, graphics, and, secondly, operational paths, can be created, segment by segment. As a result, freely selectable forms find their way into the design of the layout.

INDICATORS

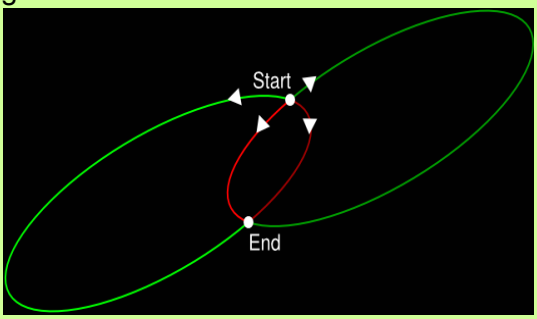
Each segment contains an indicator in the form of a character whereby the type of segment is recognised. The list of indicators can be found here in the command overview.
A segment entered always begins with "?", and the indicator, as well as the corresponding parameters, are added subsequently. In that respect, as many segments as desired can be linked with one another. There is furthermore a difference, in the case of the indicators, between upper and lower case, which represents the difference between absolute and relative co-ordinates.
Here is an example:

Generating a square with absolute and relative segment details	
#GPP1, 1, 100, 100, ?V, 300, ?H, 300, ?V, 100, ?H, 100 #GPP2, 1, 370, 100, ?v, 200, ?h, 200, ?v, -200, ?h, -200	

In this example, two identical squares are generated by linking horizontal and vertical lines. Object 1 is generated absolutely, in other words the absolute co-ordinates of the display need to be specified for positioning the line points. By connecting the point co-ordinates, the corresponding lines are generated from the latter. In the case of Object 2, on the other hand, only the starting point is defined absolutely. In regard to this starting point, the lengths of the respective lines are now specified. Thus, this square relates to the starting point relatively.

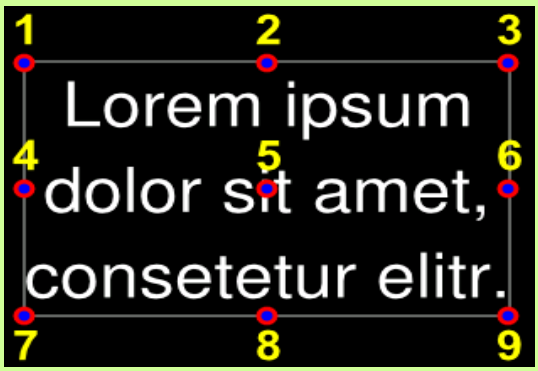
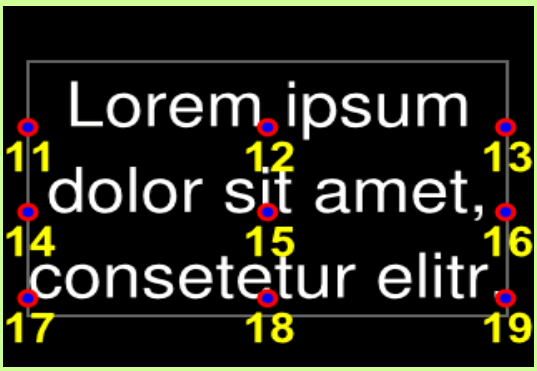
DIRECTIONS OF MOVEMENT AND ARCS

The direction of movement is to be observed with the circular segments. Depending upon the direction of movement, another segment is generated.
In the table below, a segment is defined as a full ellipse, with a starting point and an end point. Due to the radii of the ellipse being given, four options exist for linking the starting point with the end point. The four options differ in regard to the two directions of movement and the two potential arcs. The small arcs are marked red. The two large arcs have been highlighted in green. The clockwise directions of movement are shown darker.
With circles, the differentiation according to large and small arcs does not make a difference. This is due to the fact that a circle only has a radius, as a result of which the large and small arcs are identical.

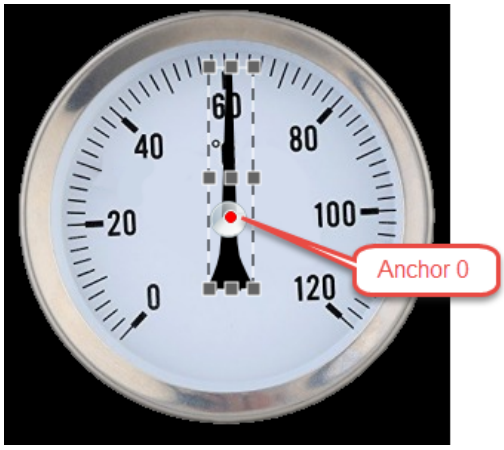
Highlighting the arcs and directions of movement	
#GPP1, 1, 390, 380, ?E0, 200, 70, 25, 340, 260 #GPP2, 2, 390, 380, ?E1, 200, 70, 25, 340, 260 #GPP3, 3, 390, 380, ?E2, 200, 70, 25, 340, 260 #GPP4, 4, 390, 380, ?E3, 200, 70, 25, 340 ,260	0: Small anti-clockwise arc 1: Small clockwise arc 2: Large anti-clockwise arc 3: Large clockwise arc 

ANCHOR

All objects can be created at a position (x, y). By specifying the anchor numbers, it can now be determined whether the object is e.g. with the upper left corner (anchor 1), or e.g. exactly in the middle (anchor 5). In the case of multiline texts, there are further anchors (11-19) which refer to the baseline of the text. In the table below, a string is displayed to illustrate the anchors:

Allgemeine Anker	Zusätzliche Anker für mehrzeilige Texte
	

In addition to the above anchor numbers, there is also anchor 0. Anchor 0 is a special anchor, which can be arbitrarily set by corresponding commands. Thus, for example, an object is able to rotate around a desired point. In the following example, a round thermometer is shown:



The pointer should turn around the center of the instrument. With the general anchors, which are represented as small gray squares, the pointer can also rotate around the center of the thermometer. The standard anchors 1 to 9 are not suitable for this purpose. Therefore, in this case, an individual anchor 0 is determined. This can be determined by pixel accuracy and is shown in red in the image. The pointer is now rotated around this anchor point.

GROUPS

Groups simplify the simultaneous handling of several objects. The desired objects are grouped together in an object group, but they retain their individual functions and definitions. The advantage is that all objects of the group can be affected at the same time, for example a shift, a rotation or a fade-out. In addition to the simple object groups, there are also groups for touch switches. These ensure that only one switch of the group is active (radio buttons), which means that not every switch has to be defined individually with such a function.
Attention: Group IDs and object IDs are in the same number space and thus overwrite each other.

CALCULATION

Each numerical parameter can be replaced by a calculation. The calculation needs to be enclosed in brackets ().

Name	Command	Description	Integer	Float
Mathematical functions	+, -, * , /, ()	Arithmetical functions	✓	✓
|x| (amount)	abs(x)	Calculation of absolute value	✓	✓
x%y (modulo)	mod(x,y)	Calculation with remainder	✓	
x^y (power)	pow(x,y)	Calculation with power	✓	✓
Root	sqrt(var)	Calculating the root		✓
Logarithm	log(var)	Calculating the logarithm		✓
Natural logarithm	ln(var)	Calculating the natural logarithm		✓
Degrees to radians	rad(deg)	Converting degrees to radians		✓
Radians to degrees	deg(rad)	Converting radians to degrees		✓
Sine	sin(deg)	Calculating the sine		✓
Cosine	cos(deg)	Calculating the cosine		✓
Tangent	tan(deg)	Calculating the tangent		✓
Arc sine	asin(var)	Calculating the arc sine		✓
Arc cosine	acos(var)	Calculating the arc cosine		✓
Arc tangent	atan(var)	Calculating the arc tangent		✓
Arc tangent in the correct quadrant	atan(y,x)	Calculating the arc tangent in the correct quadrant		✓
Random value in the range	rand(sv,ev)	Random value in the range of values	✓	✓
Random value 0 - ev	rand(ev)	Random value greater than 0	✓	✓
Random value 0<= x<=1000	rand()	Random value greater than 0; smaller than 1000	✓	✓
Minima	min(a,b,c...)	Smallest value	✓	✓
Maxima	max(a,b,c...)	Largest value	✓	✓
Average	avg(a,b,c...)	Average	✓	✓
Bit operators	<<, >>, &, |, ^, ~	Bit Operatoren	✓	
Logical operators	<, >, <=, >=, !=, ==, &&, ||, !	Logical operators	✓	✓
Increase, reduce	++,--	Increase, reduce	✓	✓
Port (a=0..15) = 0..255	port(a)	Port value	✓	✓

Name	Command	Description	Integer	Float
Portbit (a=0..127) = 0/1	bit(a)	Bit value	✓	✓
Analogport (a=0..3) = 0..4095	analog(a)	Analogue port value	✓	✓
Current date in seconds	date()	Current date	✓	
Day in seconds	date(D)	Current day	✓	
Day + month + year (1932 - 2067) in seconds	date(D,M,Y)	Any date desired	✓	
Current time in seconds	time()	Current time	✓	
Hour to seconds	time(h)	Current hour	✓	
Hour + mins to seconds	time(h, m)	Current hour and minute	✓	
Hour + min. + sec. in seconds	time(h, m, s)	Current hour, minute and second	✓	
Current time and date	datetime()	Current date with time	✓	
Hour +min.+sec. + Day+month+year to seconds	datetime(h,m,s,D,M,Y)	Any desired date and time specification	✓	
Current year	year()	Current year	✓	
Seconds as from 01/01/2000, 0:00:00 to year	year(a)	Save year	✓	
Current month	month()	Current month	✓	
Seconds as from 01/01/2000, 0:00:00 to month	month(a)	Save month	✓	
Current day	day()	current day	✓	
Seconds as from 01/01/2000, 0:00:00 to days	day(a)	save day	✓	
Current weekday (0=Sunday)	weekday()	Current weekday	✓	
Seconds as from 01/01/2000, 0:00:00 to weekday	weekday(a)	Save weekday	✓	
Current hour	hour()	Current hour	✓	
Seconds as from 01/01/2000, 0:00:00 to hours	hour(a)	Save hour	✓	
Current minute	minute()	Current minute	✓	
Seconds as from 01/01/2000, 0:00:00 to minutes	minute(a)	Save minute	✓	
Current second	second()	Current second	✓	
Seconds as from 01/01/2000, 0:00:00 to seconds	second(a)	Save second	✓	
Global 10 ms timer	timer()	Global timepiece 10ms	✓	
24-bit colour red channel	getR(x)	Determine colour portion red	✓	

Name	Command	Description	Integer	Float
24-bit colour green channel	getG(x)	Determine colour portion green	✓	
24-bit colour bluec hannel	getB(x)	Determine colour portion blue	✓	
24-bit colour RGB	RGB(R, G, B)	Determine colour com-pletely	✓	
Displays RGB of ramp number	rampRGB(nr, offset)	Displays colour of colour gradient	✓	✓
Displays transparency of ramp number	rampO(nr,offset)	Displays transparency of colour gradient	✓	✓
Display width	scrW()	Width of the display	✓	✓
Display height	scrH()	Height of the display	✓	✓
Video width	vidW()	Width of the video	✓	✓
Video height	vidH()	Height of the video	✓	✓
Number of video objects	vidC()	Number of video objects	✓	✓
Object width (without transformation)	objW(id)	Width of the object	✓	✓
Object height (without transformation)	objH(id)	Height of the object	✓	✓
Object position x of the current anchor	objX(id)	X position of the object	✓	✓
Object position x of any desired anchor	objX(id, anchor)	X position of the object		
Object position Y of the current anchor	objY(id)	Y position of the object	✓	✓
Object position Y of any desired anchor	objY(id, anchor)	Y position of the object		
Object scaling, width	objSW(id)	Scaling of the object width	✓	✓
Object scaling, height	objSH(id)	Scaling of the object height	✓	✓
Object shearing X	objSX(id)	Shearing in the X direction of the object	✓	✓
Object shearing Y	objSY(id)	Shearing in the Y direction of the object	✓	✓
Object rotation	objR(id)	Rotation of the object	✓	✓
Object opacity	objO(id)	Transparenz des Objekts	✓	✓
Object level	objL(id)	Ebene des Objekts	✓	✓
Object Style Number	objC(id)	Style Nummer der Objekts	✓	✓
Current object anchor	objA(id)	Current anchor of the object	✓	✓
Instrument bar of current value	objIV(id)	Current instrument value	✓	✓

Name	Command	Description	Integer	Float
Instrument bar of drawn value	objID(id)	Drawn instrument value	✓	✓
Instrument bar of end value	objIE(id)	End value of the instrument	✓	✓
Instrument bar start value	objIS(id)	Start value of the instru-ment	✓	✓
Displays the length of the action path	pathL(id)	Länge des Aktionspfades	✓	✓
Displays relative X co-ordinates from the beginning of the action path	pathX(id,distance)	X position beginning of path	✓	✓
Displays relative Y co-ordinates from the beginning of the actionpath	pathY(id,distance)	Y position beginning of path	✓	✓
Displays the angle of the tangent of the action path	pathR(id,distance)	Tangential angle of the path	✓	✓
Displays touch button/switch status	butS(id)	Status button or switch	✓	✓
Displays the active touch switch of the radio group	butR(groupid)	Active switch in group	✓	✓
Displays last key/switch pressed	butI()	Last switch pressed	✓	✓
Displays code of the last keyboard keys	butC()	Last key pressed on the keyboard	✓	✓
Displays the length of the string	strL(nr)	Length of the string	✓	
Displays ASCII code from string regis-ters	strA(nr, pos)	ASCII code of the stored string	✓	
Displays Unicode from string register	strU(nr, pos)	Unicode of the stored string	✓	
Comparison between two string regis-ters	strC(n1, n2)	Compare string registers	✓	
Convert numerical string into value	strV(nr)	Convert numerical string into value described	✓	✓
Checks whether file exists (<name> in String Register No.)	fileE(nr)	Check existence of the file in the string register	✓	
Gibt Dateigröße aus (<name> in Stringregister nr)	fileS(nr)	Check size of the file in the string register	✓	
Displays file size (<name> in String Register No.)	fileT(nr)	Zeit der Datei im Stringregister prüfen	✓	
Gibt Dateieigenschaft aus(<name> in Stringregister nr)	fileA(nr)	Eigenschaft der Datei im Stringregister prüfen	✓	
Displays file time (<name> in String Register No.)	fileD(nr)	Daum der Datei im Stringregister prüfen	✓	
Seconds in FatTime, 16-bit	fatT(Sekunde)		✓	
Seconds in FatDate, 16-bit	FatD(Sekunde)		✓	
Display integer calculation as a float	int(calculation)	Integer value from float cal-culation		✓

Name	Command	Description	Integer	Float
Display float calculation as an integer	float(calculation)	Float value from integer calculation	✓	

ACTION AND ANIMATION

Actions and animations can be used to "bring life" to objects or graphics on the display. There are significant differences between an action and an animation.

ACTION

Actions are used to change objects. or example, objects can be moved or their transparency can be changed. Accordingly, an apparent or vanishing behaviour can also be defined. If an action affects the parameters of an object, they remain on the newly assigned value.
This means, for example, if an object should disappear, it is deleted at the end of the action. By changing the position in combination with the actions paths , actions also allow you to follow a defined path. Action paths have the advantage that objects can rotate, scale, position, or change their percentages as a percentage. The action sequence can be adapted via the action curves. For example, a linear sequence or a process with delay or acceleration are already available as templates.

ANIMATION

Animations are only valid **for** GIF files as well as for color fills and line patterns. The various animation types can be used to influence the image sequence of GIFs or to animate line patterns and color fillings of graphics. For the time sequence of the animation, the action curves can be used for better adaptation, as already described for the actions.

SAFETY INSTRUCTIONS

AVOID ELECTRIC SHOCK AND FIRE

Do not use a damaged power cord or plug.
Do not use loose sockets.
Do not touch the power cord with wet hands.
Do not unplug the power cord by pulling the cord.
Do not bend or damage the power cord

CARE AND USE

Keep the unit dry.

- Moisture and liquids of all types can damage parts of the device or electronic circuits.

Do not store the unit in dusty or dirty environment.

- Dust can cause malfunction of the unit.

Do not place the unit on inclined surfaces.

- The unit can be damaged by falling down.
- Do not place the unit near magnetic fields, as this can cause malfunction of the unit.

Do not drop or subject the unit to shocks.

- The screen of the unit may be damaged.
- Bending or deforming can damage both the device and parts thereof.

Only use approved original spare parts.

Do not use the unit when the display is cracked or broken.

Do not use the appliance for purposes other than those intended.

Handle memory cards with care.

- Do not remove a memory card from the device while data is being read or stored on the card. Failure to do so may result in data loss or damage to the card or device.
- Do not touch the gold-colored contacts of memory cards with your fingers or metallic objects. If the contacts are dirty, clean them with a soft cloth.

Clean the device without chemicals or solvents and use a clean cloth.

HARDWARE

PIN ASSIGNMENT

An overview on the pin assignment of the display is shown below.

Pin	Symbol	I/O	Beschreibung	Pin	Symbol	I/O	Beschreibung
1	GND		Ground 0 V	27	I/O 1.7 I/O SDA	I/O I/O	Input/Output 1.7 (Bit 15) Port Expander Serial Data
2	VDD		Power Supply 3,3 V	28	I/O 1.6 I/O SCL	I/O I/O	Input/Output 1.6 (Bit 14) Port Expander Serial Clock
3		I	!Reset	29	I/O 1.5 I/O INT	I/O I/O	Input/Output 1.5 (Bit 13) Port Expander Interrupt
4	SPI CS	I	SPI: Chip select	30	I/O 1.4	I/O	Input/Output 1.4 (Bit 12)
5	SPI MOSI	I	SPI: Serial in	31	I/O 1.3	I/O	Input/Output 1.3 (Bit 11)
6	SPI MISO	O	SPI: Serial Out	32	I/O 1.2	I/O	Input/Output 1.2 (Bit 10)
7	SPI CLK	I	SPI: Serial Clock	33	I/O 1.1	I/O	Input/Output 1.1 (Bit 9)
8	RS232 RxD	I	RS232: Receive Data	34	I/O 1.0	I/O	Input/Output 1.0 (Bit 8)
9	RS232 TxD	O	RS232: Transmit Data	35	I/O 0.7	I/O	Input/Output 0.7 (Bit 7)
10	DE (RS485)	O	RS232: Transmit Enable RS485	36	I/O 0.6	I/O	Input/Output 0.6 (Bit 6)
11	I²C SDA	I/O	I²C: Serial Data	37	I/O 0.5	I/O	Input/Output 0.5 (Bit 5)
12	I²C SCL	I	I²C: Clock Data	38	I/O 0.4	I/O	Input/Output 0.4 (Bit 4)
13	A/D 0	I	Analogeingang 0	39	I/O 0.3	I/O	Input/Output 0.3 (Bit 3)
14	USB DPn	I/O	USB Data negative	40	I/O 0.2	I/O	Input/Output 0.3 (Bit 2)
15	A/D 1	I	Analogeingang 1	41	A/D 3	I	Analogeingang 3
16	USB DPp	I/O	USB Data positive	42	A/D 2	I	Analogeingang 2
17	I/O 0.0	I/O	Input/Output 0.0 (Bit 0)	43	I²C SCL	I	User I²C: Clock Data
18	USB VBus		USB Supply Voltage	44	I²C SDA	I/O	User I²C: Serial Data
19	I/O 0.1	I/O	Input/Output 0.1 (Bit 1)	45	RS232 TxD	I	User RS232: Transmit Data
20	Testmode SBUF	I O	PowerOn low: Testmode low: Data available in sendbuffer	46	RS232 RxD	O	User RS232: Receive Data
21	PWM	O	PWM output	47	SPI CLK	O	User SPI: Serial Clock
22	DPROT	I	open (high): Protocoll active low: Protokoll deaktiviert	48	SPI MISO	I	User SPI: Serial In
23	SND+	O	Loudspeaker Output (8 Ohm)	49	SPI MOSI	O	User SPI: Serial Out
24	SND-	O	Loudspeaker Output (8 Ohm)	50	SPI CS	O	User SPI: Chip select
25	VDD		Power Supply 3,3 V	51	VIN		Power Supply 3,3 V
26	GND		Ground 0 V	52	GND		Ground 0 V

RS232

RS232 is a standard for a serial interface. Data is transmitted serially in an asynchronous manner. The data is thus converted into a bit stream and transmitted. No common clock line exists. In other words, every bus subscriber needs to work with the same transmission rate (the so-called “baud rate”). RS232 is a voltage interface. That means that the data is transmitted via a voltage level. In the PC world and industrial control systems, levels of +12V or -12V are defined as the standard. Within PCBs or in micro-controllers, it is usual to work with 0V or VDD (in the case of the EA uniTFT050 3.3 V). To adjust the signal level, there are one or two options in the form of level shifters. RS232 consists of “hearing” and “speaking” lines which are crossed between the two subscribers.

DATA FORMAT

The format for the data transmission is 8-N-1:

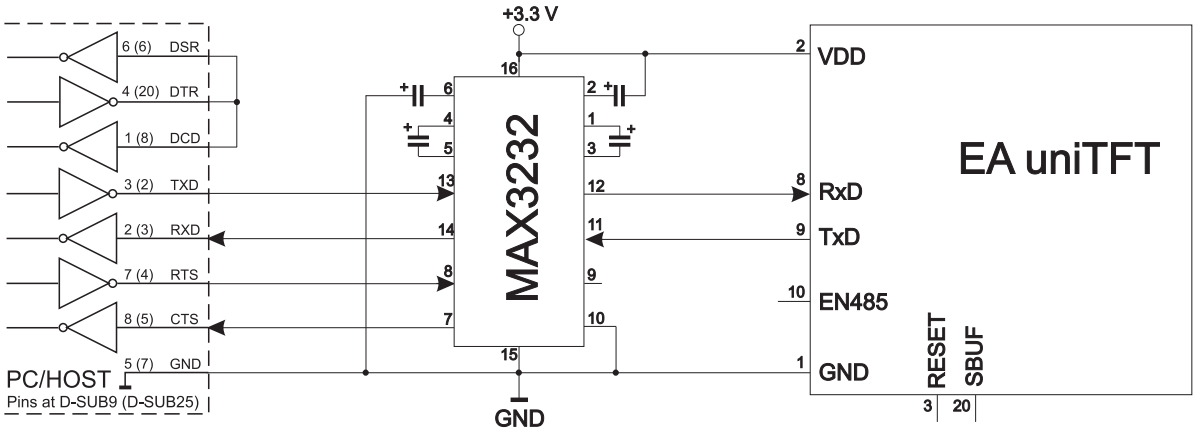
BAUD RATES

The baud rate is preset to 115.200 baud in the delivery state.The command Beispiel siehe S. 97 can change the baud rates. Alternativly you can set this command also in the file <start.emc>.

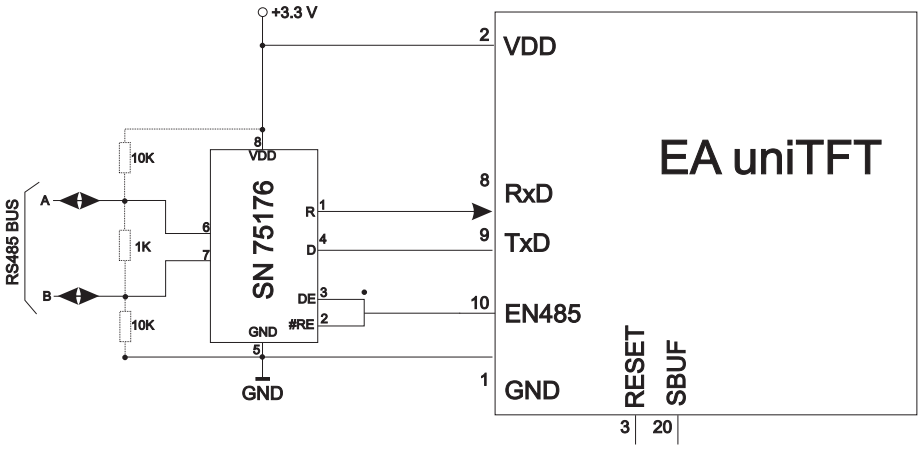
Baudrates	Error rate in %
9600	+0,04
19200	-0,08
38400	+0,16
57600	-0,08
115200	+0,64
230400	-0,80
460800	+2,08
921600	-3,68

APPLICATION EXAMPLES

RS232 V24 - CONNECTION TO A PC



RS485 - BUS SYSTEM



SPI

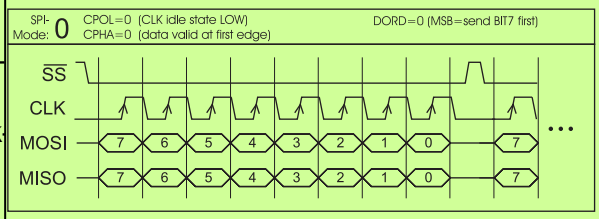
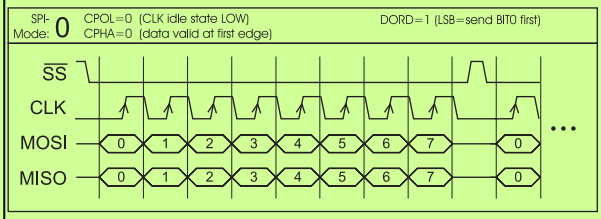
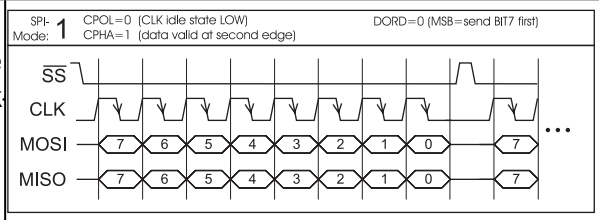
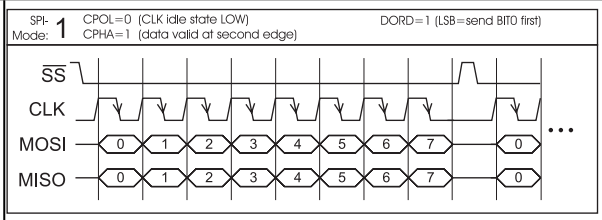
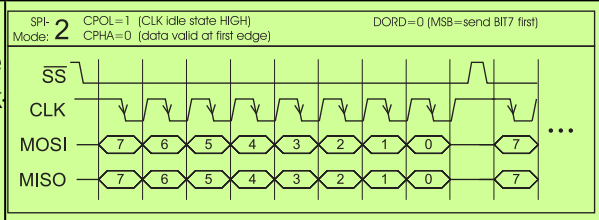
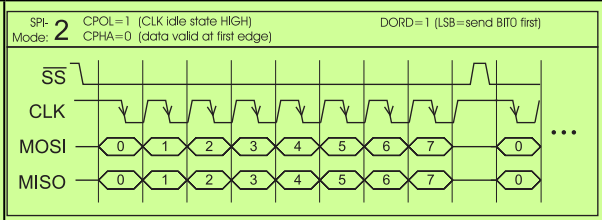
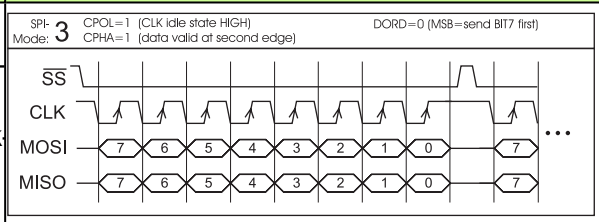
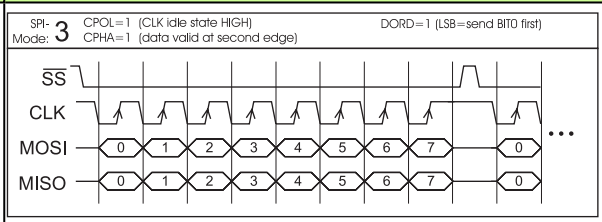
Das **S**erial **P**eripheral **I**nterface is a bus system for a serial synchronous data transmission between various ICs. The bus consists of the following lines:

- MOSI (**M**aster **O**ut → **S**lave **I**n) otherwise known as SDO (Serial Data Out) or DO
- MISO (**M**aster **I**n ← **S**lave **O**ut) also known as SDI (Serial Data In) or DI
- SCK (**S**erial **C**loc**k**) - shift clock
- SS (**S**lave **S**elect → Addressing the partner), also known as CS (Chip Select)

SPI works with a bidirectional transmission principle, i.e. data is exchanged between the partners simultaneously. Every communication is determined by the master with the aid of the SCK line. The display works in the slave mode.

The log for the data transmission is not laid down with SPI, so that there are various configuration options. These are laid down by the parameters of clock polarity, clock phase and data order:

The SPI-Mode 3 with DORD=0 is preset. The command Beispiel siehe S. 97 can change the modes 0..3. Alternativly you can set this command also in the file <start.emc>.

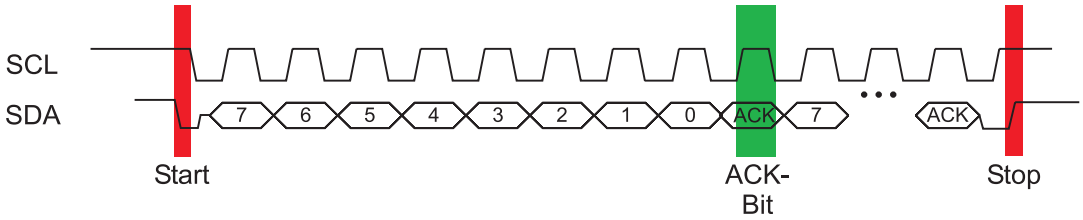
Mode	CPOL (Clock Polarity)	CPHA (Clock Phase)	DORD(0) MSB first	DORD(1) LSB first
0	Low (0)	steigende Flanke (0)		
1	Low (0)	fallende Flanke(1)		
2	High (1)	fallende Flanke(0)		
3	High (1)	steigende Flanke(1)		

I²C

I²C stands **I**nter-**I**ntegrated **C**ircuit and is a serial data bus developed by Phillips. The bus designed as a master-slave bus requires 2 signal lines:

- SCL (**S**erial **C**lock **L**ine)
- SDA (**S**erial **D**ata Line)

The electrical specification provides that both lines are terminated with a pull-up resistor to VDD, since all devices connected to the bus have open-collector outputs. The bus clock is always specified by the master, which determines the entire communication:



After the start condition, the slave address always follows in the transmission protocol. Bit 0 is the so-called R/W bit and determines whether the slave should read (1) or transmit data (0). The data is exchanged until the master executes the stop condition. More detailed information can be found in the I²C specification. The display operates in slave mode.

The default I²C bus address is 0xDE (depending on the notation, 0x6F). The command Beispiel siehe S. 97 can be used to re-define an x-like other address. Alternatively, this can be done via the <start.emc> file.

USB

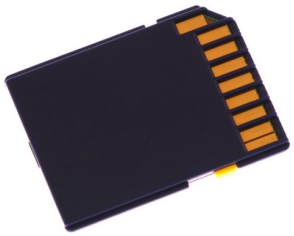
Der **U**niversal **S**erial **B**usis a serial bus system for connecting with a computer or other device. It is based on differential data transmission. The bus topology is a strict master/slave communication (exception: “On the Go” devices). In the case of the EA uniTFT050, it is always the PC/Master which has to lead the communication.
The module has a CDC device class and thus connects to the PC as a virtual serial COM interface:

Description	Value
Device class	2
USB Vendor ID	0x2DA9
USB Product ID	0x2454
Gerätebeschreibung	EA uniTFT

Um Einstellungen am Display vorzunehmen oder für erste Tests empfehlen wir die USB-Schnittstelle. Sie ist einfach anzuschliessen und es müssen keine Schnittstellenparameter angepasst werden.

SD CARD

The module has a removable micro SD card (up to 32GB). The micro SD card must be formatted with FAT32.



The memory is used for all project data, e.g. Different styles, objects, and images. The memory may additionally be used e.g. for data log functions at runtime.

VIDEO INPUT/CAMERA

The module has an analogue video input and supports three colour systems: **PAL** (**P**hase **A**lternating **L**ine), **Secam** (**Sé**quentiel **c**ouleur **à m**émoire) and **NTSC** (**N**ational **T**elevision **S**ystems **C**ommittee). The module automatically selects the right colour system settings.

Most digital cameras and video cameras come with a suitable interface, just as most CMOS cameras do. The resolution is limited to 720x576 (PAL and SECAM), 640x480 (NTSC)pixels. The input is also referred to as a composite video signal input.

he two connections (gold-plated solder pads) are located next to pin 27(see Dimensions). The input levels are 3.3V compatible.

ANALOGUE INPUT

The module has 4 analog inputs with a resolution of 12 bits. The values can be used for further processing. The crossing or subtraction of a boundary region can also trigger an alarm.

Measuring range: 0V to 3.3 V. The supply voltage is used as reference input for the ADC.

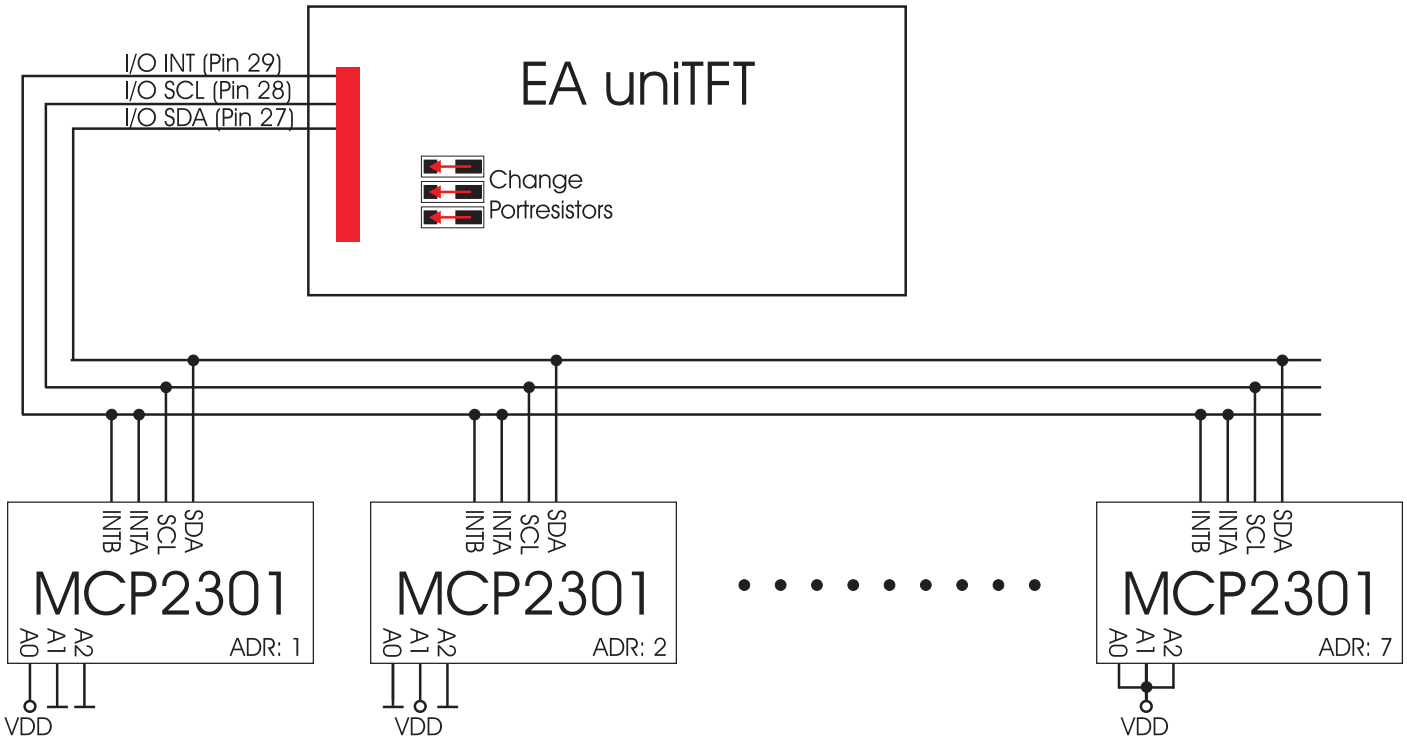
PULSE WIDTH MODULATION (PWM)

The module has the possibility to control external components via a PWM signal (pulse width modulation).
The duty ratio of the rectangular pulse is changed at a constant frequency (adjustable from 2 Hz to 1 MHz with the command Beispiel siehe S. 91). The duty ratio between on / off time can be changed, too. Electromechanical components, e.g. Motors or a quasi-analog voltage can be driven and generated. The variation of the duty cycle then ensures low motor speed / voltage with a short on-time or a high motor speed / voltage with a long on-time.
The output levels are 0V and 3.3V.

INPUT/OUTPUT (I/O)

The module has 16 digital I/O's build in.
The input voltage range is between 0..3,3V. All 16 I/O's are configured as 100kOhm Weak Pull-up at start up.

Through the use of one or more external port-expanders MCP23017 (16 I/O for each IC) the number of the entire I/O can be extended up to 128-3. For this purpose, the ports "I/O 1.5 to 1.7" (pin 27 to 29) are switched over and used as serial data lines for the Portexpanders (see application example). This is done by soldering 3 resistors:



The maximum output of the component is 700*mW*. The maximum current load for a single pin is 25*mA*.

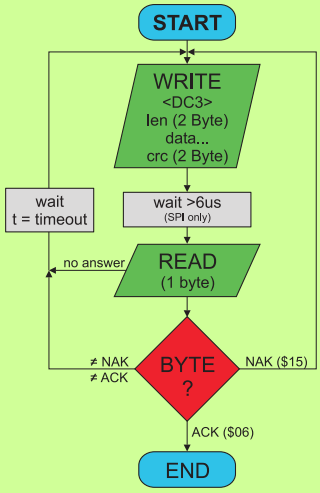
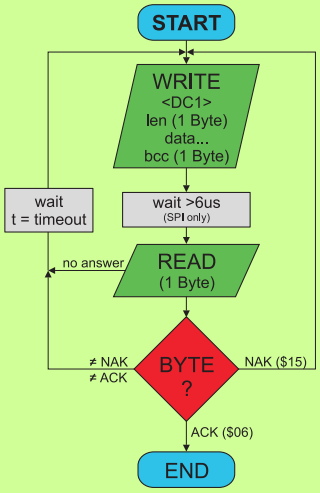
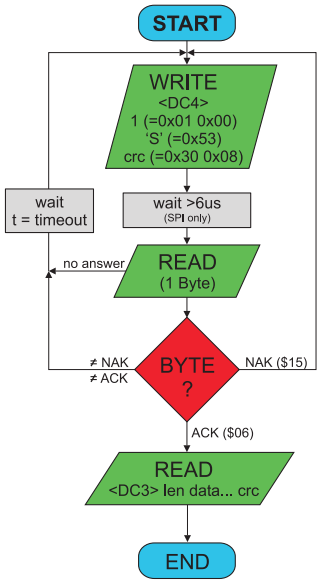
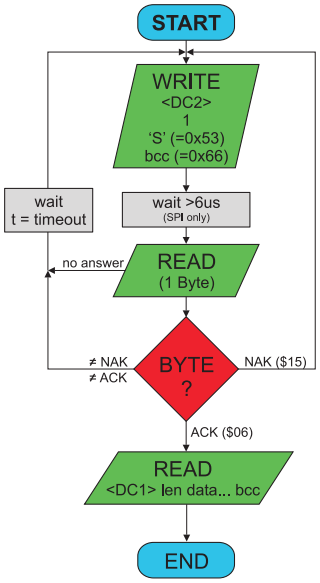
SHORT PROTOCOLL & SMALL PROTOCOLL

In order to ensure secure data transmission, 2 differently complex transmission protocols are integrated. In both protocols, the user data is embedded into a packet by adding a start and a final sequence. We recommend to use the short protocol, but for compatibility with the EA eDIPxxx series you can continue with the small protocol.

For initial tests on the serial interface, the protocol can also be switched off completely (Pin 22 "DPROT" to low).

Both protocols are independent of the interface RS-232, SPI, I²C or USB used. The data transmission is embedded in a fixed frame with checksum. The module acknowledges this package with the character <ACK> (= $ 06) on successful reception or <NAK> (= $ 15) if the checksum is incorrect or receive buffer is overflowed. In either case, after received <NAK> the complete package is discarded and must be sent again.

Note: The <ACK> must be actively read SPI and I²C). If the host does not receive an acknowledgment, at least one byte has been lost. In this case, the timeout time must be waited before the packet is completely repeated. The number (len) of the user data per packet can be max.2042 Byte. Commands larger than 2042 byte (for example, images or file #FWD ...) must be split into several packages. The user data in the individual packets are reassembled after correct reception

Action	Short Protokoll	Small Protokoll
Sending		
Receiption		

SHORT PROTOCOLL

Command	Display	Byte										
		0	1	2	3	4	5	6	7	8	9	10
Send commands / data to display	send	DC3[1]	len[2]		data	...	...	crc16[3]				
	receive	ACK										
Repeat last packet	send	DC4	1	0	'S'	crc16						
	send	ACK										
	receive	DC3	len		data	...	...	crc16				
Request protocoll infromation	send	DC4	1	0	'R'	crc16						
	send	ACK										
	receive	DC3	len		data	...	...	crc16				
Request / Release interface	send	DC4	5	0	'D'	packet size sendbuffer[4]		timeout[5] [10 ms]		crc16		
	receive	ACK										
Hardware Reset / Reboot	send	DC4	1	0	'I'	crc16						
	send	ACK										
	receive	DC4	4	0	Sendbuffer bytes ready		Receivebuffer bytes free		crc16			
Repeat last packet	send	DC4	1	0	'P'	crc16						
	send	ACK										
	receive	DC4	6	0	max. Packet size		akt. Packetsize		timeout [10 ms]		crc16	
Request protocoll infromation	send	DC4	3	0	'A'	**S**elect / **D**eselect	adr	crc16				
	receive	ACK										
RS485 direction enable delay	send	DC4	2	0	'T'	Verzögerung [10 us][6]						
	receive	ACK										
Break-command	send	DC4	2	0	'G'	0=Freigabe 1=Anfordern	crc16					
	send	ACK										
	receive	DC4	1	0	aktiv 7	crc16						
Call the display's send buffer	send	DC4	2	0	'C'	mask[8]	crc16					
	receive	ACK										
Protocoll settings	Display	DC4	2	0	'B'	option[9]	crc16					

[1]DC3 = 19d = 0x13; DC4 = 20d = 0x14

[2]len: number of use data, without checksum, without DC3/4. Consists of 2 bytes, low byte is transmitted first

[3]crc: Checksum over all transmitted bytes, including DC3/4 and length bytes. CRC16-CCITT's start value is 0xFFFF. Low byte is transmitted first

[4]Maximum buffer size is 2042

[5]Delay in [10ms] till deletion of data packet. The default is 200 = 2 seconds.

[6]Delay in [10us] till EA uniTFT RS485-direction is altered. Default is 0 = immediately.

[7]active: 0=ALLE, 1=RS232, 2=SPI, 3=I²C, 4=USB

[8]mask: 1=break wait-command (#XXW), 2=actual macros, 4=Clear send buffer, 8=Clear receive buffer. The individual bits can be OR-combined.

[9]
option: 1=Testmode, 2=Disable PowerOnMacro, 3=Disable defaults 4=Boot menu

SMALL PROTOCOLL

Command	Display	Byte					
		0	1	2	3	4	5
Send commands / data to display	receive	DC1[1]	len[2]	data	...	...	bcc[3]
	send	ACK					
Call the display's send buffer	receive	DC2	1	'S'	bcc		
	send	ACK					
	send	DC1	len	data	...	...	bcc
Repeat last packet	receive	DC2	1	'R'	bcc		
	send	ACK					
	send	DC1	len	data	...	...	bcc
Protocoll settings	receive	DC2	3	'D'	packet size send-buffer[4]	timeout[5] [10 ms]	bcc
	send	ACK					
Request buffer information	receive	DC2	1	'I'	bcc		
	send	ACK					
	send	DC2	2	Sendbuffer bytes ready	Receivebuffer bytes free	bcc	
Request protocoll in-fromation	receive	DC2	1	'P'	bcc		
	send	ACK					
	send	DC2	3	max. Packet size	akt. Packetsize	timeout [10 ms]	bcc
Adressing RS485	receive	DC2	3	'A'	**S**elect / **D**eselect	adr	bcc
	send	ACK					
RS485 direction enable delay	receive	DC4	2	'T'	Verzögerung [10 us][6]		
	send	ACK					
Request / Release interface	receive	DC2	1	'G'	0=Freigabe 1=Anfordern		
	send	ACK					
	send	DC2	1	aktiv[7]	bcc		
Break-command	receive	DC2	1	'C'	mask[8]		
	send	ACK					
Hardware Reset / Reboot	receive	DC2	1	'B'	option[9]	bcc	

[1]DC1 = 17d = 0x11; DC2 = 18d = 0x12

[2]len: number of use data, without checksum, without DC1/2.

[3]bcc: Sum of all bytes including DC1 / 2 and length (Modulo 256)

[4]The maximum buffer size is 255

[5]Delay in [10ms] till deletion of data packet. The default is 200 = 2 seconds.

[6]Delay in [10us] till EA uniTFT RS485-direction is altered. Default is 0 = immediately.

[7]active: 0=ALLE, 1=RS232, 2=SPI, 3=I²C, 4=USB

[8]mask: 1=break wait-command (#XXW), 2=actual macros, 4=Clear send buffer, 8=Clear receive buffer. The individual bits can be OR-combined.

[9]option: 1=Testmode, 2=Disable PowerOnMacro, 3=Disable defaults 4=Boot menu

CHECKSUM CALCULATION

In the following two sample functions, checksum calculation for Short or Small protocol, are shown.

SHORT PROTOCOLL

A cyclic redundancy check (CRC) is used to calculate the checksum. A common and well-known CRC test is the CRC-CCITT. The initial value 0xFFFF is used. The following is a typical C-implementation. The functions must be called externally. The checksum must be pre-assigned with the start value.

```
//-------------------------------------------------------------------------
-
//Function: buffer2crc16()
//input: ptr data, ptr to CRC, len
//output: ---
//Descr: CRC-CCITT of a buffer
//-------------------------------------------------------------------------
-
void buffer2crc16(UBYTE *dat, UINT16 *pCRC, UINT32 anz)
{
   while(anz--)
      crc16(*dat++, pCRC);
}

//-------------------------------------------------------------------------
-
//Function: sp_crc16()
//input: data, ptr to CRC
//output: ---
//Descr: CRC_CCITT (x^16+x^12+x^5+1 = 1 0001 0000 001>0 0001 = 0x1021
//-------------------------------------------------------------------------
-
void crc16 (UBYTE dat, volatile UINT16 * crc)
{
   register UINT16 lcrc = *crc;

   lcrc = (lcrc >> 8) | (lcrc << 8);
   lcrc ^= dat;
   lcrc ^= (lcrc & 0xFF) >> 4;
   lcrc ^= lcrc << 12;
   lcrc ^= (lcrc & 0xFF) << 5;

   *crc = lcrc;
}
```

SMALL PROTOCOLL

For the calculation of the checksum, a simple 8-bit sum check (modulo 256) is performed for this protocol type.

```
//-------------------------------------------------------------------------
-
//Function: buffer2bcc()
//input: ptr data, len
//output: ---
//Descr: Byte bcc of a buffer
//-------------------------------------------------------------------------
-
UBYTE buffer2bcc(UBYTE *dat, UBYTE anz)
{
   UBYTE bcc = 0;
   while(anz--)
      bcc += *dat++;

   return bcc;
}
```

COMMANDS

COMMAND OVERVIEW

The module is controlled by a series of graphic commands. They allow the creation of a screen as a macro as well as via the serial interfaces. The following tables explain each command with the necessary and optional parameters.

TERMINAL

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designition	Command code		Description
Define window and font	**#YDW**	PosX(old Pos), PosY(old Pos), Anchor (7), Columns(scrW()/Font width), Rows(scrH()/Font height), Font-nr(2)	Defines the terminal window through the font **Font No**.(1=8x8;2=8x16), number of columns **Columns**, number of rows **Rows**, position **PosX, PosY** and anchor **Anchor**. After performing the reset, the position in the lower left corner is (0,0), the anchor stands at 7. Should the position not be additionally specified when the window is redefined, the latter is retained. Beispiel siehe S. 121
Set colour	**#YDC**	Text-RGB, Text-Opacity(100), Background-RGB(0x00000000), Background-Opacity(0)	Defines the colour of the terminal window in regard to colour and transparency of text **Text RGB** (e.g. $FF0000 for red **Text Opacity** and background **Background RGB, Background Opacity.** Beispiel siehe S. 121
Terminal level	**#YDL**	Layer	Select the terminal layer, Select the terminal layer, Layer (0= Behind; 1= In front) (0= Behind; 1= In front)
Terminal on/off	**#YDO**	On/Off, Visibility (=On/Off)	The parameter **On/Off**(0=Off; 1=On) determines whether the terminal window is switched on or off. In the switched-on state, the output is saved, however may be hidden through **Visibility**(0= Invisible; 1= Visible), as well as made visible again, if required.
Cursors blink on/off	**#YCB**	Cursor	Determine the blinking of the cursor. **Cursor** 0=off; 1=on
Position cursor	**#YCP**	Column, Row(keine Änderng)	Positions the cursor of the terminal window based on the column **Column**, and row **Row**.
Save cursor position	**#YCS**		**Speichert die aktuelle Cursorposition.**
Restore cursor position	**#YCR**		Restores the cursor position saved.

Designition	Command code	Description
Display string	**#YPA** 'String'	Displays the character string '**String**' in the terminal window. This command may, for example, be used in the output of the Stringregister.
Clear terminal	0x12(FF, Formfeed)	Clears the terminal and puts the cursor in the top left corner.
Show time/date	**#YPD** DateFormat(#WDF), Date32(actual time)	Displays the time in the terminal window. The display can be changed through the date format DateFormat, as well as its content be overwritten by changing the time and the date **Date32**. Information on DateFormat can be found here.
Display string in a formatted manner	**#YPF** Formatstring, Value1...	Displays a formatted character string. The **formatted string** specifies the character string. The values are subsequently specified through (multiple) **Value**(s). nformation on the formatted string can be foundhere.
Display configuration	**#YPI**	Displaying the display parameters, such as the width and height, as well as interface settings in the terminal.
Display version string	**#YPV**	This emits the current version of the display.

IMAGES/VECTOR GRAPHICS

The module internally uses a special image format (*.epg). The conversion needs to be done externally. The Windows program EA uniSketch offers the most comfortable option to have most image formats converted. Some commands make it possible to save the screen content on the SD card or transfer image data directly via the serial interface. Various image formats are available here.

Image size
Should the width **Width** OR the height **Height** , be specified by 0, then this size is adjusted proportionally.
Should the width**Width** AND the height **Height**, be specified by 0, the image is then generated in the original size (in pixels).

ADD IMAGE/VECTOR GRAPHIC.

Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designition	Command code		Description
Place image	**#PPP**	Object-id, 'Picturename', PosX, PosY, Anchor(1), Width(0), Height (0), Angle(0)	Defines as an object **Object-id** , the image file **'Picturename'** and places it independently of the position **PosX, PosY**, the anchor **Anchor**(0...9), the image width **Width**(Pixel), the image height, **Height**(Pixel) and the angle **Angle**(0...360). ttention must be paid to whether the image file is named in upper or lower case letters. Beispiel siehe S. 110
Change image animation	**#PPA**	Object-id, AnimationType (0), Time, Image No	Changes, in the object,**Object ID**, the present animation through the choice of the new animation type **AnimationType**. The expiration time, **Time**, (hrs) and the end image, **Image No**., are only relevant for the animation type Animation Type = 7. This involves the time in which animation is carried out up to the image number set being defined. The prerequisites for animations are described here Beispiel siehe S. 110
Show video input	**#PVP**	Object-id, PosX, PosY,Anchor(1), Width(0), Height(0), Angle(0)	Defines the image of the video input as Object ID, and places it, depending upon the position PosX, PosY, of the anchor **Anchor** (0...9), the image width **Width**(in pixels), the image height **Height**, (in pixels) and the angle **Angle**(0...360). Beispiel siehe S. 110

TYPES OF ANIMATION

Animations relate to objects of the type of origin **.GIF** or animated gradients.

Designation	Parameter	Description
Stop	**0**	Defines an animation end.
Cyclical	**1**	Defines a cyclical animation.
Cyclical, backwards	**2**	Defines a cyclical animation moving in the opposite direction.
Oscillating	**3**	Defines an oscillating animation.
Oscillating, backwards	**4**	Defines an oscillating animation moving in the opposite direction.
One-off	**5**	Defines a one-off animation.
Once, backwards	**6**	Defines a one-off animation moving in the opposite direction.
Once, in a fragmented manner	**7**	Defines an animation section.

STYLE SHEETS

Allgemeine Informationen zu den Styles befinden sich hier.
Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

COLOUR GRADIENTS AND LINE PATTERNS

Designition	Command code	Description
Define colour gradient	**#CCR** Ramp-nr, Offset, RGB, Opacity, reapeate all parameters	Defines a colour gradient **Ramp No**. (1...255) with the **RGB** colours (e.g. $FF0000 for red) and associated transparency **Opacity0** ...100). The offset **Offset** (0...100) determines at what point, as a percentage, the next colour is to be found. A maximum of 10 supporting points can be defined in a colour gradient. Beispiel siehe S. 117
Animate colour gradient	**#CAC** Ramp-nr, AnimationType(1), Time (100)	Animates the colour gradient **Ramp No** . with the animation type AnimationType(1...7) over the time period **Time**(hrs). Animation types
Define dash patterns	**#CDP** Dash-nr, Dashphase, Dash, Space, Dash, Space ...	Defines a dash pattern **Dash No**.(1...10), depending upon the length of the dash **Dash**(in pixels) and length of clearance **Space**(in pixels). The offset **Dash Phase** (0...100) determines, as a percentage, the starting point from which the dash pattern is drawn. A maximum of 10 combinations for line length and clearance may be defined in a dash pattern. Beispiel siehe S. 117
Animate dash patterns	**#CAD** Dash-nr, AnimationType(1), Time(100)	Animates the dash pattern **Dash No**. with the animation type **AnimationType**(1...7) over the time period Time(hrs).. Animation types

DRAWING STYLE

Designition	Command code	Description
No filling	**#CFD** Drawstyle-nr	Defines a **Drawing Style** Number without a filling. The filling is thus transparent.

Designition	Command code	Description
Define filling colour	**#CFC** Drawstyle-nr, RGB, Opacity(100)	Defines a drawing style **Drawing Style No**. with a plain colour filling **RGB**(e.g. $FF0000 for red), as well as its transparency **Opacity**(0...100). Beispiel siehe S. 118
Linear colour gradient	**#CFL** Drawstyle-nr, Ramp-nr, Angle(0)	Defines a drawing style **Drawing Style No**. with linear colour gradient **Ramp No**. and its **Angle**(0...360). In order to be able to use this command, it is necessary to define a colour gradient in advance. Beispiel siehe S. 118
Radial colour gradient	**#CFR** Drawstyle-nr, Ramp-nr, FocusX (5000), FocusY(0)	Defines a drawing style **Drawing Style No**. with a radial colour gradient **Ramp N** o. The focal position **FocusX,Y** (0...100) defines, as a percentage, the starting point of the gradient.Should the focal point **FocusX** be specified as being a number of pixels that is greater than 5,000, then **FocusY** can be set as an anchor. In order to be able to use this command, it is necessary to define a colour gradient in advance. Beispiel siehe S. 118
Conical colour gradient	**#CFK** Drawstyle-nr, Ramp-nr, FocusX (5000), FocusY(0), Direction	Defines a drawing style **Drawing Style No**. with a conical colour gradient Ramp No. The focal position **FocusX,Y**(0...100) defines, as a percentage, the starting point of the gradient. The direction of movement **Direction**(0= Anti-clockwise; 1= Clockwise) determines the orientation. Should the focal point **FocusX** be specified as being a number of pixels that is greater than 5,000, then **FocusY** can be set as an anchor. In order to be able to use this command, it is necessary to define a colour gradient in advance. Beispiel siehe S. 118

Designition	Command code	Description
Filling with a pattern	**#CFP** Drawstyle-nr, 'Patternname', Size(0), Angle(0), Repeat/Reflekt(0), FocusX (5000), FocusY(0), AnchorP(5)	Defines a **Drawing Style No**. with a pattern '**Pattern name**' that depends upon the **Size**(proportional), the **Angle**(0...360) and the representation **Repeat/Reflect**(Repeat= 0; Reflect= 1). The focal position **FocusX,Y**(0...100) defines, as a percentage, the starting point of the gradient. Should the focal point **FocusX** be specified as being a number of pixels that is greater than 5,000, then **FocusY** can be set as an anchor **Anchor** The pattern anchor AnchorP(0...9) sets the reference point of the pattern. Beispiel siehe S. 119
Change angle of the colour gradient	**#CFA** Drawstyle-nr, Angle	Changes the colour gradient angle **Angle** of the drawing style **Drawing Style No**..
Change colour gradient	**#CFG** Drawstyle-nr, Ramp-nr,	Changes the colour gradient **Ramp No**. of the drawing style **Drawing Style No**.. In order to be able to use this command, it is necessary to define a colour gradient in advance.
Change focus of the filling	**#CFF** Drawstyle-nr, FocusX, FocusY, AnchorP(No Change)	Changes the focal position **FocusX,Y** of the drawing style **Drawing Style No**. , as well as the patternanchor **AnchorP** when filled with a pattern.
Change pattern	**#CFN** Drawstyle-nr, 'Patternname'	Changes the pattern '**Pattern name**' of the drawing style **Drawing Style No**. In order to be able to use this command, it is necessary to define a pattern in advance.
Change size of pattern	**#CFS** Drawstyle-nr, Size	Proportionality changes the size **Size** of the pattern of the drawing style **Drawing Style No.**.
Change representation of pattern	**#CFT** Drawstyle-nr, Repeat/Reflect	Changes the **Repeat/Reflect** representation of the filling pattern of the drawing style Drawing **Style No**..

LINE STYLE

Designition	Command code	Description
Refrain from defining a line	**#CLD** Drawstyle-nr	Defines a Drawing **Style No.** without a line style. The line thus becomes invisible.

Designition	Command code	Description
Define line style	**#CLS** Drawstyle-nr, RGB, Opacity(100), Width(1), Joint(0), Dash-nr(0)	Defines a drawing style **Drawing Style No.** with a border, depending upon its **RGB** colour (e.g. $FF0000 for red), transparency **Opacity**(0..100), width **Width** (in pixels), rounding off **Joint**(Angular = 0; Round = 1) and dash pattern **Dash No.**. To use the dash pattern, the latter needs to have been pre-defined. Example
Change line colour	**#CLC** Drawstyle-nr, RGB, Opacity(No Change)	Changes the **RGB** colour (e.g. $FF0000 for red) and transparency **Opacity** (0...100) of the drawing style **Drawing Style No.**. In order to be able to use this command, it is necessary to define a colour in advance.
Change line width	**#CLW** Drawstyle-nr, Width	Changes the **Width** of the line of the drawing style **Drawing Style No.**.
Change line end	**#CLE** Drawstyle-nr, Joint	Changes the line rounding off **Joint** (Angular = 0; Round = 1) of the line of the drawing style **Drawing Style No.**. Beispiel siehe S. 119
Change line pattern	**#CLP** Drawstyle-nr, Dash-nr	Changes the colour gradient **Dash No.** of the drawing style **Drawing Style No.**. In order to be able to use this dash pattern, it is necessary to define it in advance

TEXT STYLE

Designition	Command code	Description
Define text style	**#CTF** Textstyle-nr, 'Fontname', Size(20 *.evf / 0 *.epf), Align(0), Drawstyle-nr(1), Italic(0), SpaceL(0), SpaceC(0)	Defines a text style **Text Style No**. with a font **'Font Name**' which depends upon the latter's size **Size**, orientation **Align**(0=On the left; 1=Centred; 2=On the right), drawing style **Drawing Style No.**, italic angle **Italic**(-45°...45°), line spacing **SpaceL** (in pixels) and character pitch **SpaceC**(in pixels). In order to be able to use this command, it is necessary to define a drawing style style in advance. For performance reasons, it is recommended to refrain from using an outline(#CLD). Beispiel siehe S. 119
Change font	**#CTN** Textstyle-nr, 'Fontname'	Changes the font **'Font Name'** of the text style **Text Style No.**.
Change text size	**#CTS** Textstyle-nr, Size	Changes the size **Size** of the text style **Text Style No.**.

Designition	Command code	Description
Change text orientation	**#CTA** Textstyle-nr, Align	Changes the orientation **Align**(0=On the left; 1=Centred; 2=On the right) of the text style **Text Style No.**.
Change text character style	**#CTC** Textstyle-nr, Drawstyle-nr	Changes the drawing style **Drawing Style No**. of the text style **Text Style No.**.
Change text slant	**#CTI** Textstyle-nr, Italic	Changes the italic angle **Italic** of the text style **Text Style No.**.
Change spacing between characters	**#CTG** Textstyle-nr, SpaceL, SpaceC(No Change)	Changes the line spacing **SpaceL** and character pitch **SpaceC** of the text style **Text Style No.**. Negative spacing and/or pitch is also possible.

TOUCH BUTTON STYLE

Designition	Command code	Description
Image for touch button	**#CBP** Buttonstyle-nr, 'Buttonname'normal, 'Buttonname'down(normal), SizeX(0), SizeY(0)	Defines a button style **Button Style No**. with an image for the unpressed state **'Button Name'(normal)** and pressed state **'Button name'(down),** depending upon the image size **SizeX, SizeY**(in pixels). Beispiel siehe S. 120
Touch button drawing style	**#CBD** Buttonstyle-nr, 'Drawstyle-nr'normal, 'Drawstyle-nr'down(normal), Width, Height, Radius	Defines a button style **Button Style No**. with a drawing style for the unpressed state **Drawing Style No.(normal)**, drawing style for the pressed state **Drawing Style No.(down)**, button width **Width**(in pixels), button height **Height**(in pixels) and corner radius **Radius**(in pixels). In order to be able to use this command, it is necessary to define a drawing style style in advance. Beispiel siehe S. 120
Text style of touch button	**#CBT** Buttonstyle-nr, 'Textstyle-nr'normal, 'Textstyle-nr'down, OffsetX, OffsetY	Defines a button style **Button Style No**. with text style for the unpressed state **Text Style No.(normal)**, text style for the pressed state T**ext Style No.(down)** and a text offset **OffsetX, OffsetY**(in pixels)..
Change the proportion of the touch button when pressed	**#CBO** Buttonstyle-nr, OffestX(0), OffsetY(OffsetX), Size(0), Angle(0)	Defines a button style **Button Style No**. for the pressed state with an offset **OffsetX**, **OffsetY**(in pixels), change in size **Size**(proportional) and change in the angle **Angle**(0...360). Beispiel siehe S. 120

Designition	Command code	Description
Touch button disabled	**#CBG** Buttonstyle-nr, DisableR(-30), DisableG(DisableR), DisableB (DisableR)	Defines a deactivated/greyed out button style **Button Style No**., depending upon a subtraction of the colour portions red **DisableR**(-100...0), green **DisableG**(-100...0)and blue **DisableB**(-100...0).
Sound relating to touch button	**#CBS** Buttonstyle-nr, 'Soundname'(alten löschen)	Defines a sound in the button style **Button Style No**. with the audio file "**Sound Name**". Should no audio file be specified, then the old sound will be deleted. Should a 0 be specified for button style, then the standard sound is selected.

DRAWING/GRAPHIC PRIMITIVES

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designition	Command code		Description
Rectangle	**#GRR**	Object-id, Drawstyle-nr, PosX, PosY, Anchor, Width, Height(same as Width), Radius(0), BorderWidth(0), Angle(0)	Defines as an object **Object ID** a rectangle, depending upon the drawing style **Drawing Style No**., the position **PosX, PosY**, the anchor **Anchor**, (0...9), the width **Width** (in pixels), the height **Height** (in pixels), the radius for the corners Radius (in pixels), the border width **BorderWidth**, (0...100) and the angle **Angle** (0...360). In order to be able to use this command, it is necessary to define a drawing style style in advance. Example
Draw polyline	**#GPL**	Object-id, Drawstyle-nr, PosX, PosY, PosX2, PosY2, PosX3, PosY3, ...	Defines as an object, **Object ID** a polyline, depending upon the drawing style, **Drawing Style No**. and the corresponding position points **PosX, PosY.** To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 124
Polygon(filled in)	**#GPF**	Object-id, Drawstyle-nr, PosX, PosY, PosX2, PosY2, PosX3, PosY3, ..., autoclose polygon	Defines as an object, **Object ID** a filled-in polygon, depending upon the drawing style, **Drawing Style No.** and the position points **PosX, PosY**. To use the command, a drawing style needs to have been defined in advance. The polygon is automatically concluded after the last position value. Beispiel siehe S. 124
Extend a polyline/polygon	**#GPA**	Object-id, PosX, PosY, PosX2, PosY2, ...	Adds additional positions **PosX, PosY** to the object, **Object ID,** of #GPF and #GPL . Beispiel siehe S. 124
Polypath segment	**#GPP**	Object-id, Drawstyle-nr, PosX, PosY, Segment1, Segment2, ...	Defines as an object, **Object ID** a filled-in polygon, depending upon the drawing style, **Drawing Style No**., the position points, **PosX, PosY** and the individual segments **Segment**. These also allow for having round shapes. To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 125

Designition	Command code	Description
Polygon(geometric)	**#GGP** Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius, NumberCorners, BorderWidth(0), Angle(0)	Defines as an object **Object ID** a polygon, depending upon the drawing style **Drawing Style No.**, the position points, **PosX, PosY**, the anchor , **Anchor**(0...9), of the radius **Radius** (in pixels), the number of corners **NumberCorners**, the border width **Border Width** (0...100) and the angle **Angle** (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 125
Polygon(star)	**#GGS** Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2, CountPoints, BorderWidth(0), Angle(0)	Defines as an object **Object ID** a star-shaped polygon, depending upon the drawing style **Drawing Style No.**, the position points **PosX, PosY**, the anchor **Anchor** (0...9), the external radius **Radius1** (in pixels), the internal radius **Radius2** (in pixels), the number of points **CountPoints**, the border width **Border Width** (0...100) and the angle **Angle** (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 125
Circle/ellipse	**#GET** Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), BorderWidth(0), Angle(0)	Defines as an object **Object ID** a circular or elliptical polygon, depending upon the drawing style **Drawing Style No.**, the position points, **PosX, PosY**, the anchor **Anchor** (0...9), the external radius **Radius1** (in pixels), the internal radius **Radius2**, the border width **Border Width** (0...100) and the angle **Angle** (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 126
Arc of a circle	**#GEA** Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), StartAngle(0), StopAngle(360), BorderWidth(0), Angle(0)	Defines as an object **Object ID** a curved polygon, depending upon the drawing style **Drawing Style No.**, the position points, **PosX, PosY**, the anchor **Anchor** (0...9), the external radius **Radius1**(in pixel)s, the internal radius **Radius2**, the starting angle **Starting Angle**(0...360), the stop angle **Stop Angle** (0...360), the border width **Border Width** (0...100) and the angle Angle (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 126

Designition	Command code	Description
Straight cut	**#GES** Object-id, Drawstyle-nr, PosX, PosY, Acnhor, Radius 1, Radius2(Radius1), StartAngle(0), StopAngle(360), Angle (0)	Defines as an object **Object ID** a cut circle, depending upon the drawing style **Drawing Style No.**, the position points, **PosX, PosY**, the anchor **Anchor**(0...9),the length radius **Radius1** (in pixels), the height radius **Radius2**, the starting angle **Starting Angle** (0...360), the stop angle **Stop Angle** (0...360), the border width **Border Width** (0...100) and the angle **Angle** (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 126
Section	**#GEP** Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), StartAngle(0), StopAngle(360), Angle (0)	Defines as an object **Object ID** a section of a polygon, depending upon the drawing style **Drawing Style No**., the position points, **PosX, PosY**, the anchor **Anchor**(0...9), the external radius **Radius1** (in pixels), the internal radius **Radius2**, the starting angle **Starting Angle** (0...360), the stop angle **Stop Angle** (0...360), the border width **Border Width**(0...100) and the angle **Angle** (0...360). To use the command, a drawing style needs to have been defined in advance. Beispiel siehe S. 126

SEGMENT TYPES

General information on the segments can be found here.

Designation	Parameter	Description
Horizontal line	**?H**, x	Defines the position **x** for a horizontal line point. Beispiel siehe S. 127
Vertical line	**?V**, y	Defines the position **y** for a vertical line point. Beispiel siehe S. 127
Freehand line	**?L**, x, y	Defines the position **x, y** for the next line point. Beispiel siehe S. 127
Arc of a circle	**?Ct**, r, x, y	Defines the position **x, y** for the next point of the arc, as well as its radius **r**(in pixels). Beispiel siehe S. 128
Elliptical arc	**?Et**, rx, ry, Angle, x, y	Defines the position x, y for the next point of the ellipse, as well as its radius **rx, ry**(in pixels) and **Angle**. Beispiel siehe S. 128
Quadratic Bézier curve	**?Q**, c1x, c1y, x, y	Defines the positions **c1x, c1y**, as well as **x, y** and for the quadratic Bézier curve. Beispiel siehe S. 128
Slight quadratic Bézier curve	**?R**, x, y	Defines the position **x, y** for an addition to the quadratic Bézier curve. Mirrors **c1x** and **c1y** of the previous segment, i.e. this command is not suitable for only one segment. Beispiel siehe S. 128
Cubic Bézier curve	**?S**, c1x, c1y, c2x, c2y, x, y	Defines the positions **c1x, c1y** and **c2x, c2y**, as well as **x, y**, for the cubic Bézier curve. Beispiel siehe S. 129
Slight cubic Bézier curve	**?T**, c2x, c2y, x, y	Defines the positions **c2x, c2y** and **x, y** for an addition to the cubic Bézier curve. Mirrors **c1x** and **c1y** of the previous segment, i.e. this command is not suitable for a single segment. Beispiel siehe S. 129
Close path	**?Z**	Defines the path end, i.e. the next point is the starting point of the first segment. Beispiel siehe S. 129
Jump (new sub-path)	**?M** x, y	Defines a new starting point for the subsequent segment. Beispiel siehe S. 129

STRINGS AND CHARACTER STRING COMMANDS

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designation	Command code		Description
Place character string	**#SSP**	Object-id, Textstyle-nr, PosX, PosY, Anchor, 'String'	Defines as an object **Object ID** a '**String**' and places it, depending upon the text style **Text Style No**., the position **PosX, PosY** and the anchor **Anchor**. Beispiel siehe S. 115
Change character string	**#SSC**	Object-id, String	hanges the string allocated in the object **Object ID** into a new string '**String**'.
Place formatted character string	**#SFP**	Object-id, Textstyle-nr, PosX, PosY, Anchor, 'Formatstring'; Value1...	Defines as an object O**bject ID** a '**Format String**' and places it, depending upon the text style Text Style No., the position **PosX, PosY** and the anchor **Anchor**. The content of the string is, among other things, determined by the successive values **Value**. Beispiel siehe S. 115
Change formatted character string	**#SFC**	Object-id, Value1...	Changes the value **Value** of the formatted string of the object **Object ID**.
Place calculation string (auto update)	**#SAP**	Object-id, Textstyle-nr, PosX, PosY, Anchor, 'Formatstring'; Value1(Calculation), Value2	Defines as an object **Object ID** a 'Format String' and places it, depending upon the text style **Text Style No**., the position **PosX, PosY** and the anchor **Anchor**.The content of the string is, among other things, determined by the successive calculated values **Value**. In the event of the calculation of the first value being amended **Value1**, the display is renewed. Beispiel siehe S. 115
Change auto update of the calculation string.	**#SAC**	Object-id, (ChangeCalculation)	Updates the update calculation **ChangeCalculation** for the automatic display in the object **Object ID**. The calculation of the values of the calculation string is, in that respect, not changed. The changed calculation always has the Integer data type. Beispiel siehe S. 115

Designation	Command code	Description
Place date/time string	**#SDP** Object-id, Textstyle-nr, PosX, PosY, Anchor, DateFormat (Date), Date32 (acutal time)	Defines as an object **Object ID** a string and places it, depending upon the text style **Text Style No**., the postion **PosX, PosY** and the anchor **Anchor**. The content of the string is determined by the choice of date format **DateFormat**(#WDF), the current time and the current date **Date32**. The time is updated automatically. The last optional parameter **Date32** can be specified unless the current time is supposed to be displayed. .
Change date/time string	**#SDC** Object-id, Date32(actual time)	Changes the time or date specification **Date32** of the time string **Object ID**.

EDIT BOX

The edit box is an editable box for strings. The latter is used, for example, in combination with the keyboard. The default string is the first visible display of the edit box, and exists once the corresponding function is accessed.

Designation	Command code	Description
Place edit box for strings	**#SEP** Object-id, Drawstyle-nr, PosX, PoxY, Anchor Width, Height, Radius, Textstyle, OffsetX, OffsetY	Defines as an object **Object ID** an editable string box, and places it, depending upon the drawing style **Text Style No**., the postion **PosX, PosY** and the anchor **Anchor**. The width **Width**, height **Height**, rounding of the corners **Radius**, text style **Text Style** distance from the lateral edge **OffsetX** and offset in the Y direction **OffsetY** likewise need to be specified for the creation. Beispiel siehe S. 115
Send default string to edit box	**#SED** Object-id, Strings...	Defines, as an object **Object ID**, a default string **String**, and transmits it to the edit box.
Send code/string to edit box	**#SEC** Object-id, Codes/Strings...	Defines, as an object **Object ID**, codes or strings **String**, and transmits them to the edit box.
Link keyboard to edit box	**#SEK** Keyboard-id, Object-id...	Links a keyboard **Keyboard ID** with one or more edit boxes **Object ID**. As a result, it is possible to edit multiple edit boxes with one keyboard Beispiel siehe S. 115

Designation	Command code	Description
Enable edit box for key-board	**#SEA** Object-id	Activates an edit box **Object ID** for the linked keyboard. That means that one of the edit boxes allocated to the key-board can be activated for processing. Always only one edit box is active. **Object ID=** 0 disables all edit boxes. Beispiel siehe S. 115

TOUCH OBJECTS / TOUCH FUNCTIOS

Two types of operation are available for touch areas:
Touch button (**B**) = Pushing, they jump into the non-pressed state as soon as the actuation is finished.
Touch switch(**S**) = Switching on, the state is changed once

Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

DEFINITION OF TOUCH OBJECTS

Designition	Command code	Description
Rectangular touch button	**#TBR** **#TSR** Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, Anchor(5), Width(Buttonstyle), Height (Buttonstyle)	The touchbutton **Object-id** with button style **Buttonstyle-nr**, both strings for pressed **'Text'**(normal) and unpressed **Text**(down) state is drawn. The width **Width**(Pixel) and height **Height**(Pixel) out of the button style is used, but you can set it individually, too. You have to define a button style before using this command. Beispiel siehe S. 122
Elliptical touch button	**#TBE** **#TSE** Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, Anchor (5), Width(Buttonstyle), Height (Buttonstyle)	The elliptical touch button **Object-id**with button style **Buttonstyle-nr**, both strings for pressed **'Text'**(normal) and unpressed **Text**(down) state is drawn. The width **Width**(Pixel) and height **Height**(Pixel) out of the button style is used, but you can set it individually, too. You have to define a button style before using this command. Beispiel siehe S. 122
Touch button as an image	**#TBP** **#TSP** Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, Anchor (5), SizeX(Buttonstyle), SizeY (Butonstyle)	An image is used touch object**Object-id**. Properties are set in the button style **Buttonstyle-nr**. The string for unpressed **'Text'** (normal) and pressed **Text**(down) , the posiiton **PosX, PosX** need to be handed over. The anchor **Anchor**, width **SizeX**(Pixel) and height **SizeY**(Pixel) can be set up through the button style.können auch aus dem Buttonstyle übernommen werden. ZYou have to define a button style before using this command (#CBP). Beispiel siehe S. 122

Designition	Command code	Description
Convert Object to a touch switch / button	**#TBO** Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down) **#TSO**	converts any object into a touchable object element **Object-id** . All the additional information required for creating a touch area is taken from the button style **Buttonstyle-nr** . The text for the unpressed **'Text'**(normal) and pressed state **Text**(down) are handeld over, too. You have to define a button style before using this command.

TOUCH FUNCTIONS

Designition	Command code	Description
Change touch status	**#TCS** Touchstate, Object-id ...	Changes the state **Touchstate** (0= pressed; 1= unpressed; 2= deactivated) of one ore more touch objects **Object-id**
Change touch labeling	**#TCL** Object-id, 'Text'(normal), 'Text'(down)	Changes the text **Text**(normal/down) of a touch object **Object-id**.
Disable/Enable Touch	**#TCE** State, Object-id ...	Activates (**State** = 1) or deactivates **State** = 0) one or more touch objects **Object-id**.
Touch responses	**#TCR** Signal, Filter, Object-id ...	Determination of the behavior of one or more touch subjects **Object-id**. **Signal** paramater is used to define which actions can be called 1= unpressed; 2= pressd; 3= pressed and unpressed; 4 = drag; 7 = evrything The filter **Filter**offers the possibility to block the transmission depending on the macro definition: 0= only send if there is no macro defined;1 =send always
Assign a touch response to the object	**#TID** Mask, Object-id ...	Assignes one or more objects **Object-id** a touch action **Mask**(1= internally; 2= move; 4= resize, 8= rotate). **Mask**=1 is used for special touch objects, like bargraphs or indicational instruments. **Mask**= 2 the object is movable on the screen for the user, **Mask**= 4 resizinhg object, **Mask**= 8 rotates the objects around its' anchor
Check status touch switch	**#TQS** Object-id ...	Sends the status of the object **Object-id**. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#TQS)
Create radio-group	**#TRA** Group-id, Object-id...	Creats a new object **Group-id** and inlcudes some touchbuttons **Object-id**, to create a radio group.

Designition	Command code	Description
Check the active switch of the radio group	**#TQR** Group-id …	Sends the status of one or more radio goups **Group-id**. A more precise description of the system response can be found in the chapter Responsens. (#HAR) (#TQR)

BAR GRAPHS AND INSTRUMENTS

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designation	Command code		Description
Bar graph -rectangle	**#IBR**	Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Width, Height, Radius(0), Startvalue(0), Endvalue(100), Course(1), Angle(0)	Defines a rectangular bar graph as an object **Object ID**, depending upon the drawing style of the foreground **Drawing Style No. Front**, the drawing style of the **background Drawing Style** Back, the position **PosX, PosY**, the ancor **Anchor**(0...9), the with o **Width**(Pixel) of the bar, the bar's height**Height**(Pixel), the radius **Radius** for the corners, the **Start Value**, the **End Value**, the direction **Course**(0=right→ left; 1=left→ right)and the angle **Angle**(0...360). Beispiel siehe S. 112
Bar graph - triangle	**#IBT**	Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Width, Height,Tip(0), Startvalue(0), Endvalue (100), Course(1), Angle(0)	Defines a triangular bar graph as an object **Object ID**, depending upon the drawing style of the foreground **Drawstyle Front**, the drawing style of the background **Drawstyle Back**, the position **PosX, PosY**, the anchor **Anchor**(0...9), the width of the bar graphs **Width**(in pixels), the height of the bar graphs **Height**(in pixels), the position of the tip **Tip**(0=Bot; 1=Top; 2=Mid), of the start value **Start Value**, the end value **End Value**, the direction of movement **Course**(0=Large→small; 1=Small→large) and the angle **Angle**(0...360). Beispiel siehe S. 112

Designation	Command code		Description
Bar graph - arc	**#IBA**	Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Radius, Borderwidth, Startangle, Stopangle, Startvalue(0), Endvalue(100), Direction(1)	Defines a curved bar graph as an object **Object ID**, depending upon the drawing style of the foreground **Drawstyle Front**, the drawing style of the background **Drawstyle Back**, the position **PosX, PosY**, the anchor **Anchor**(0...9), the radius to the curvature of the bar graph **Radius**, the height of the bar graph as a **border width** of the defined circle Border Width(in pixels), the starting angle **Start Angle**(0...360), the end angle **Stop Angle**(0...360), the starting value **Start Value**, the end value **End Value** and the direction of movement **Direction**(0=Anti-clockwise; 1=Clockwise Beispiel siehe S. 112
Place instrument picture	**#IPP**	Object-id, "Instrumentname", PosX, PosY, Anchor, Width(0), Height(0), Startvalue(0), Endvalue(100), Angle(0)	Defines as an object **Object ID** a rectangle, depending upon the drawing style, **Drawing Style No**., the position **PosX, PosY**, the anchor **Anchor**(0...9), the width, **Width** (in pixels), the height, **Height** (in pixels), the radius for the corners, **Radius** (in pixels), the border width, **Border Width**, (0...100) and the angle, **Angle** (0...360). Beispiel siehe S. 113
Define rotational instrument	**#IGM**	Group-id, Indicator-id, Startangle, DeltaAngle, Startvalue(0), Endvalue(100)	Defines an object group **Group-ID** as a rotational instrument. For use, the rotating object **Indicator ID**, starting angle **Start Angle** and angle of rotation **Delta Angle** need to be established. Starting value **Start Value** and end value **End Value** determine the range of values. Beispiel siehe S. 112
Define slider	**#IGS**	Group-id, Path-id, Slide-id, Turn(0), Startvalue(0), Endvalue(100)	Defines an object group **Group-ID** as a slider. For use, the path of the slider **Path ID** and sliding object **Slide ID** need to be established. Torsion in the case of the revolution of the path **Turn** is an optional detail. Starting value **Start Value** and end value **End Value** determine the range of value Beispiel siehe S. 113

Designation	Command code		Description
Set value	**#IVS**	Object-id, Current, Time(0), ActionCurve(0)	Setzt das Instrument **Object-id** auf einen Wert **Current** . Die Animation der Werteinstellung wird durch die Dauer **Time**(hs) und die Aktionskurve **ActionCurve**(1...10) bestimmt.
Automatically change value	**#IVA**	Object-id, BarCalculation, Time(0), ActionCurve(0)	Sets the instrument **Object ID** to a value **BarCalculation** . The animation of the value setting is determined by the duration **Time**(hrs) and the action curve **Action Curve**(1...10). Beispiel siehe S. 113
Update calculation	**#IVC**	Object-id, ChangeCalculation(bar calculation)	Initiates in the instrument or bar graphs **Object ID** a recalculation **Change Calculation** . The calculation is always of the “Integer” data type.

OBJECTS

Allgemeine Informationen zu den Objekten befinden sich hier.
Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

DEFINITION OF OBJECTS

Designation	Command code	Description
Objects visible	**#OVV** Object-id, Object-id...	Visibility of objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Object invisible	**#OVI** Object-id, Object-id...	Invisibility of objects **Object-id**.The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Set free anchor (0)	**#OAS** Anchor0X, Anchor0Y, Object-id, Object-id...	The free anchor **Anchor** = 0 of one ore more objects **Object-id** is set through the position **Anchor0X, Anchor0Y**.
Set free anchor (0) relatively	**#OAO** Anchor0X, Anchor0Y, Object-id, Object-id...	The free anchor **Anchor** = 0 of one or more objects **Object-id** is set through the position **Anchor0X, Anchor0Y** relativly to the objects. You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Set active anchor	**#OAA** Anchor, Object-id, Object-id...	Sets the anchor **Anchor**(0-9) of the objects. You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Group objects	**#OGA** Group-id, Object-id, Object-id...	Defines a group **Group-id**, consisting of different **Object-id**. You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs. Informationen zu Gruppen befinden sich hier.

Designation	Command code	Description
Draw object frame	**#OFP** Drawstyle-nr, addL/R, addT/B, Object-id , Object-id...	Draws a frame around the object **Object-id**depending on the style **Drawstyle-nr** and additional dots around the object left/right **addL/R** and top/bottom **addT/B**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Objekt delete	**#ODI** Object-id, Object-id...	Deletes one or more objects **Object-id**. Styles won't be deleted as they aren't objects..The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.

CHANGE OF OBJECTS

Designation	Command code	Description
Change positon absolut	**#OPA** PosX, PosY, Object-id, Object-id...	Change position **PosX, PosY** of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change positon relativ	**#OPR** PosX, PosY, Object-id, Object-id...	Change relatively the position **PosX, PosY** of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change size absolut	**#OSA** Width, Height, Object-id, Object-id...	Changes width **Width** and height **Height** of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.

Designation	Command code	Description
Change size relativ	**#OSR** Width, Height, Object-id, Object-id...	Changes width **Width** and height **Height** with additionor subtraction of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range : ("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change shear absolut	**#OHA** ShearX, ShearY, Object-id, Object-id...	Changes shearing **ShearX, ShearY** (degrees) vof one ore more objects**Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change shear relativ	**#OHR** ShearX, ShearY, Object-id, Object-id...	Changes shearing relativ**ShearX, ShearY** (degrees) of one ore more objects**Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change rotation absolut	**#ORA** Angle, Object-id, Object-id...	Changes the angle **Angle** of one ore more objects**Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range : ("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change rotation relative	**#ORR** Angle, Object-id, Object-id...	Changes the angle realtively **Angle** of one ore more objects**Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range : ("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change transparency absolut	**#OOA** Opacity, Object-id, Object-id...	Changes the transparency **Opacity** of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range : ("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.

Designation	Command code	Description
Change transparency relative	**#OOR** Opacity, Object-id, Object-id...	Changes the transparency relatively**Opacity** of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change Draw-/Text-/Buttonstyle	**#OCS** Style-nr, Object-id, Object-id...	Changes the used style **Style-nr** (Drawstyle-nr, Textstyle-nr oder Buttonstyle-nr) of one ore more objects **Object-id**. The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change color	**#OCC** AddR, AddG, AddB, Object-id, Object-id...	Change color **RGB**(e.g: $FF0000 red) of one ore more objects **Object-id** through adding color parts **AddR**(-100...100), green **AddG**(-100...100) and blue **AddB**(-100...100). The **Object-id** = 0, means that the command is used for all defined objects You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.
Change layer absolut	**#OLA** Nr, Object-id, Object-id...	Changes the object layer **Nr** of one ore more objects **Object-id**. The specified objects are assigned one after the other to the selected object plane, i.e. the last object is located on the desired object level and the other objects under it. If there is no "space" below the selected object level, the other objects are placed over it. Level 1 is the lowest level. You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs. Example: #OLA 1, 1, 2, 3 Object 3 is on layer 1. Object 2 on layer 2 and object 1 on layer 3.
Change layer absolut	**#OLR** Difference, Object-id, Object-id...	Changes the layer raltive to ech object **Object-id**. Positive numbers set the object to a higher, negative numbersto a lower layer. You can also input a range :("ID1 - ID10"). The order of processing depends on the specification of the two object IDs.

VARIABLES AND REGISTERS

Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designition	Command code	Description
Load string file	**#VFL** 'Stringfilename'	Loads the string file **'Stringfilename'**. You can load in maximum 8 String files with 1000 differnet strings in the maximum. Unter 'Stringfile' sind Details über deren Verwendung zu finden
Delete string file	**#VFD** 'Stringfilename'(Alle Stringdateien löschen)	Deletes the selected or all strings of file **'Stringfilename'**.
Count strings of string files	**#VFC**	Sends the amount of strings given through the string files. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#VFC)
Set string register	**#VSS** String-id, 'String', String(id +1), String (id + 2)	Saves a string **'String'** in string register **String-id**. Additonal strings are stored in the following registers. The length is 255 characters for each string register.
Set string register postion	**#VSP** String-id, pos, 'String''	Saves a string **'String'** in string register **String-id** starting at position **pos**.
String register Date/Time	**#VSD** String-id, DateFormat(#WDF), Date32 (datetime())	Uses the date format **DateFormat** to store time and date **Date32**in string register **String-id**.
Stringregister Formatstring	**#VSF** String-id, Formatstring (printf), Value1, Value2...	Saves a formated string **'Formatstring'** in string register **String-id** with the help of **Value**. You can find details about 'formated strings' here..
Copy object sting to string register	**#VSO** String-id, Object-id, Objekt(id+1), Objekt(id+2)	Writes a string of a object **Object-id**to string register **String-id**.
Read string register (ASCII)	**#VSA** String-id, String-id	Sends a string register **String-id** as ASCII. You can define a range of registers to send with '-'. E.g.g S1-S4, so register 1,2,3,4 are send. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#VSA)
Read string register (Unicode)	**#VSU** String-id, String-id	Sends a string register **String-id** as unicode. You can define a range of registers to send with '-'. E.g.g S1-S4, so register 1,2,3,4 are send. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#VSU)

Designition	Command code	Description
Set register (Integer)	**#VRI** Register-id, Integer, Integer(id +1)	Saves an integer value or calculation output**Integer** to register **Register-id**. Further values are stored in the following registers
Set register (Float)	**#VRF** Register-id, Float, Float(id +1)	Saves a float value or calculation output**float** to register **Register-id**. Further values are stored in the following registers
Read register Int/Float	**#VRG** Register-id, Register-id, Register-id	Sends a register **Register-id** value (integer/float). You can define a range of registers to send with '-'. E.g.g S1-S4, so register 1,2,3,4 are send. A more precise description of the system response can be found in the chapter Responsens. (#HAR) (#VRG)

MACROS

General information on macros can be found here.
Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

EXECUTING MACROS

Designation	Command code	Description
Execute macro	**#MRN** "Makroname"	The "**Macro Name**" macro (file name) is executed.
Execute macro with condition	**#MRC** Condition, "Makroname" true;, "Makroname" false(Kein Makro)	The "**Macro Name**" macro (true) is executed if the condition **Condition** is fulfilled. Otherwise, '**Macro Name**'(false) is executed.
Execute port macro	**#MRP** Port	The port macro for the port **Port**(0-15) is executed.
Execute bit macro	**#MRB** Bit, Edge	The bit macro for the bit **Bit**(0-127) with the flank **Edge**(0= Falling; 1= Increasing) is executed.
Execute touch macro	**#MRT** Object-id, TouchState	The touch macro, which is defined by the touch function **Touch State**(1= Not Pressed; 2= Pressed; 4= Drag) in the touch object **Object ID**, is executed.
End macro file prematurely	**#MFE** Condition(wahr)	Ends the macro prematurely if the condition **Condition** is fulfilled.
Skip macro file lines	**#MFS** Condition(wahr), Lines(1)	Skips lines **Lines** in the macro if the condition Condition is fulfilled.
Set macro file marker	**#MFM** Marker-nr(0)	Sets a marking in the macro file **Marker No**.(0-9).
Jump from macro file to marker	**#MFJ** Condition, Marker-nr(0)	Jumps to the previously defined marking **Marker No**.(0-9) if the condition **Condition** is fulfilled.
Delete macro definitions	**#MCD** Mask	Delete all macro definitions. Through **Mask**, it is established which group of macros is deleted (1=Second; 2= Process; 4= Port; 8= Bit;16= Analogue; 32= Touch; 64= Operation). The addition of the individual flag in the Mask results in the combined deletion of the respective macro groups. Thus, when entering Mask=3, second and process macros are deleted.

DEFINITION OF MACRO PROCESSES AND TOUCH MACROS

Designation	Command code	Description
Macro process	**#MPD** Prozess-nr, Time, "Makroname"; , Startnr(Keine), Endnr(Startnr), Type(1)	Defines a process macro with the process number **Process No**.(1-10). The "**Macro Name**" macro is accessed at time intervals Time(hrs). Multiple macros can, optionally, be executed. For this purpose, the "Macro Name" needs to end with at least one figure. The start **Start Number** and end **End Number** macros are handed over as optional parameters, and equipped with an AnimationTypen **Type**(1-6). Should, for instance, Macro Name= Photo, StartNo.= 1, Time= 100 and End No.= 6 be used, then "Photo1"→"Photo2"→...→Photo6 will be accessed at intervals of a second.
Conditional macro	**#MPC** Prozess-nr, Conditiontime, Condition, "Makroname"; , Startnr(Keine), Endnr (Startnr), Type(1)	Defines a process macro with the process number **Process No**.(1-10). It is queried at the interval **ConditionTime**(hrs) whether the condition Condition is fulfilled. Should the **Condition**=TRUE, then the macro "Macro Name" is executed. The option exists to access multiple macros. For that purpose, the start and end macros are, as it were, give a **Start No**. and **End No**., and equipped with a type of animation **Type**(1-6), see #MPD.
Automatically changing macro	**#MPA** Prozess-nr, Calculationtime, Change, "Makroname";, Startnr(Keine), Endnr (Startnr), Type(1)	Defines a process macro with the process number **Process No**.(1-10). The **"Macro Name"** macro is only accessed if there changes in value in the **Change calculation.** The calculation is always of the “Integer” data type. The time **Calculation Time**(hrs) determines the interval of the query of the calculation result. Should the current result have changed in comparison to the previous result, the "Macro Name" macro is executed. The option exists to access multiple macros. For that purpose, the start and end macros are, as it were, give a **Start No**. and **End No**., and equipped with a type of animation Type(1-6),, see #MPD.
Change process time	**#MPT** Prozess-nr, Time	Changes the time **Time**(hrs) after which a macro is accessed in the process macro **Process No**. In order to be able to use this command, it is necessary to define a process macro in advance.

Designation	Command code	Description
Define port macro	**#MHP** Port, "Makroname"(delete old macro)	Should the status of the I/O pins of the port **Port**(0-15) be altered, the macro is accessed **"Macro Name"**. Should the command be executed without a **"Macro Name"**, the current macro definition is deleted.
Define bit macro	**#MHB** Bit, Edge, "Makroname"(delete old macro)	Defines the macro **"Macro Name"** for the port pin **Pin**(0-127) and the flank Edge (0= Falling; 1= Increasing). Should the command be executed without a **"Macro Name"**, the current macro definition is deleted.
Define analogue macro	**#MHA** Channel, AnalogType, "Makroname" (delete old macro)	Defines in the channel **Channel**(0-3) the **"Macro Name"** macro The Type of the analogue macro AnalogueType needs to be taken into consideration in the process. Should the command be executed without a **"Macro Name"**, the current macro definition is deleted.
Define RTC second macro	**#MDS** , "Makroname" (delete old macro)	Accesses the macro **"Macro Name"** every second. Should the command be executed without a **"Macro Name"**, the current macro definition is deleted.
Define operation end macro	**#MDA** Object-id, "Makroname" (delete old macro)	Defines an end macro in the object **Object ID**. The "Macro Name" macro is only executed once the operation of the object has come to an end Object ID. Should the command be executed without a "Macro Name", the current macro definition is deleted.
Define touch macro	**#MDT** Object-id/Group-id, "Makroname down" (delete old macro); "Makroname up" (Altes Makro löschen), "Makroname drag" (Altes Makro löschen)	Defines a touch function in the object **Object ID** or in the object group **Group ID**. In that respect, **"Macro Name"** macros are specified for the **pressed**(down), **unpressed**(up) and **maintained**(drag) states. The corresponding object **Object ID** needs to be defined for the latter as a touch object.Should the command be executed without a **"Macro Name"**, the current macro definition is deleted.

TYPES OF ANALOGUE MACROS

The hysteresis of the analogue channel set is observed. There is a description here of how to set the limits and the hysteresis..

Designation	Parameter	Description
Change	**0**	The macro is accessed with any changes in value.
Reduction	**1**	The macro is accessed when there are changes in value.
Increase	**2**	The macro is accessed when there are increases in value.
Fail to reach lower limit	**3**	The macro is accessed when failing to reach the lower limit.
Failing to reach lower limit	**4**	The macro is accessed when exceeding the lower limit
Failing to reach upper limit	**5**	The macro is accessed when failing to reach the upper limit.
Exceeding upper limit	**6**	The macro is accessed when exceeding the upper limit
Leaving the border window	**7**	The macro is accessed when leaving the window predefined by the two limits.
Entering the border window	**8**	The macro is accessed when entering the window predetermined by the two limits.

TIME AND DATE

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designation	Command code	Description
Set time and date	**#WTD** Hour, Min(Alte Werte), Sec(Alte Werte), Day(Alte Werte), Month(Alte Werte), Year(Alte Werte)	Sets the time, hour **Hour**, minute **Min** and second **Sec** and date **Day**, month **Month**and year **Year**. The RTC is buffer with a coin battery 364.
Send time in binary format	**#WSB** Date32(Aktuelle Zeit)	Sends the actual timne or the time given thorugh **Date32**. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#WSB)
Set format string for time	**#WDF** DateFormat	Sets the format of time and date. **DateFormat** als formatierter String (Example) angegeben werden. Details about time string format are listed here
Send time ASCII	**#WSA** DateFormat(#WDF), Date32(Aktuelle Zeit)	Sends the actual time or the time given through **Date32**in the date format **DateFormat**as ASCII. Date format commands are shown here. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#WSA)
Send time unicode	**#WSU** DateFormat(#WDF), Date32(Aktuelle Zeit)	Sends the actual time or the time given through **Date32**in the date format **DateFormat**as Unicode. Date format commands are shown here. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#WSU)
Define month names	**#WDM** 'Jan'(January); 'Feb'(February); 'Mar'(March); 'Apr'(April); 'May'(May); 'Jun'(June); 'Jul'(July); 'Aug'(August); 'Sep'(September); 'Okt'(October); 'Nov'(November); 'Dec'(December)	Defines the names for the months **'Jan' - 'Dec'**
Define month names	**#WDW** 'Mon'(Monday), 'Tue'(Tuesday), 'Wed'(Wednesday), 'Thu'(Thursday), 'Fri'(Friday), 'Sat'(Saturday), 'Sun'(Sunday)	Defines the names for the days **'Mon' - 'Sun'**.
Objectgroup as clock	**#WGC** Group-id, IndicatorH-id, IndicatorM-id, IndicatorS-id	Defines an object group **Group-id** as clock.To use the hour- **IndicatorH-id**, minute- **IndicatorM-id** or optionally the scond **IndcatorS-id** indicator, objects of the groups must be assigned to it. Beispiel siehe S. 123

ACTION

Allgemeine Informationen zu Aktionen befinden sich hier.
Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

OPERATIONAL CURVES AND PATHS

Designition	Command code	Description
Define operation curve	**#ACD** ActionCurve-Nr, X1, Y1, X2, Y2	The actrion curve describes the chronological course of an operation, e.g. accelerating and decelerating What is defined is an operational curve Operational **Curve-No**.(1...10) through the positions **X1**, 2 (0...100) and **Y1**, 2(-200...300). For information on default curves and more precise explanations, see Pre-defined operational curves.
Define operation path	**#APD** ActionPath-nr, StartX, StartY, Segment1, Segment2,...	Defines an operational path **action path** (1...10) through the position **Xstart**, **Ystart** and the successive segments. The operational path describes the way in which the operation is supposed to be performed.

DEFINE ACTION

Designition	Command code	Description
Action define	**#ADC** Start/End	**Start/End**(Finish= 0; Start= 1) Commands which chronologically follow **#ADC1** are only executed if the definition of the operation is terminated by **#ADC0**. As a result, multiple objects or operations that are generated or are supposed to begin simultaneously can be defined.

ACTION

Designition	Command code	Description
Define absolute action	**#AOA** Object-id, Action, Action2, ...	Defines an operation **action** for an object **Object ID** absolutely.For an explanation of the operations possible, see definition an action.

Designition	Command code	Description
Define relative action	**#AOR** Object-id, Action, Action2, ...	Defines an operation **action** for an object **Object ID** relatively. For an explanation of the operations possible, see definition an action.
Define type of action	**#AOT** Object-id, ActionType, TotalTime (100), DelayStart(0), DelayEnd(0)	Defines a type of action **Type of Action** for the object **Object ID** assigned, depending upon the total duration **TotalTime**(hs), the start **delay** DelayStart(hrs) and premature termination **DelayEnd**(hrs). An operation for the object Object ID needs to have been determined in advance for this command. For an explanation of the various types of operation, see type of operation
Cancel action	**#AOS** Object-id, Stop(0), Next(0)	Defines the end of an operation **Stop**(0= Now; 1= Start of an operation; 2= End of an operation) and subsequent step **Next**(0= Stopping normally; 1= Jump; 2= Delete object) for the object assigned **Object ID**. An operation for the object Object ID needs to have been determined in advance for this command.

DEFINE ACTIONS

Designation	Parameter	Description
Position curve	**101-110** PosX, PosY	Defines the position **PosX, PosY**. The object is moved in a straight line to the point specified. Beispiel siehe S. 107
Position curve path	**151-160** Actionpath-nr, Offset	The object follows the operational path **Action Path No**., or is placed on the latter. The starting value **Offset** (0...100) determines the starting point on the operational path on a percentage basis. Beispiel siehe S. 107
Scale curve	**201-210** ScaleX, ScaleY	Defines the size **ScaleX, ScaleY** as a percentage of the original size of the object. Beispiel siehe S. 107
Scale curve path	**251-260** Actionpath-nr, Offset	The object is scaled over the operational path **ActionPath No.** The starting value **Offset**(0...100) determines the starting size on the operational path on a percentage basis. Beispiel siehe S. 107
Rotation curve	**301-310** Angle	Defines the rotational angle **Angle**. Negative angle for rotation in a clockwise direction Beispiel siehe S. 108
Rotation curve path	**351-360** Actionpath-nr, Offset	The object is rotated over the operational path **Action Path No**. The starting value **Offset**(0...100) determines the starting angle on the operational path on a percentage basis. Beispiel siehe S. 108
Shear curve	**401-410** ShearX Shear Y	Defines a shearing **ShearX, ShearY** in degrees. Beispiel siehe S. 108
Shear curve path	**451-460** Actionpath-nr, Offset	The object is sheared over the operational path **Action Path No**. The starting value **Offset**(0...100) determines the starting shearing on the operational path on a percentage basis.
Opacity curve	**501-510** Opacity	Defines the transparency **Opacity** (0..100). Beispiel siehe S. 109
Colour offset curve	**601-610** Red, Green, Blue	Defines a colour addition of the colour components red **Red**(-100...100), green **Green**(-100...100) and blue **Blue** (-100...100). Beispiel siehe S. 109

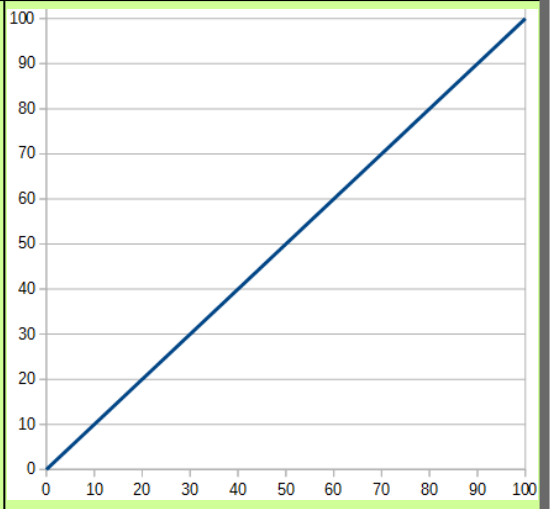
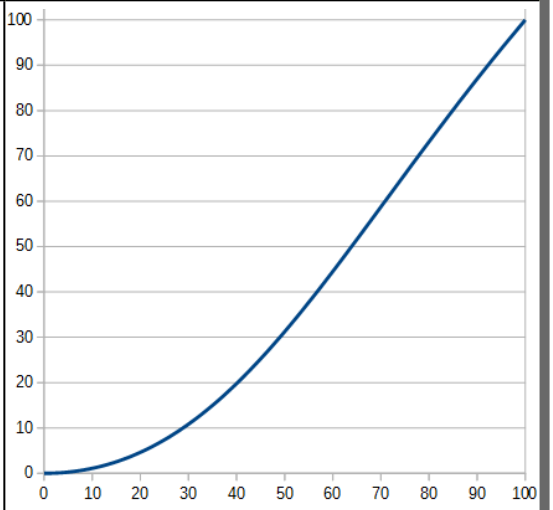
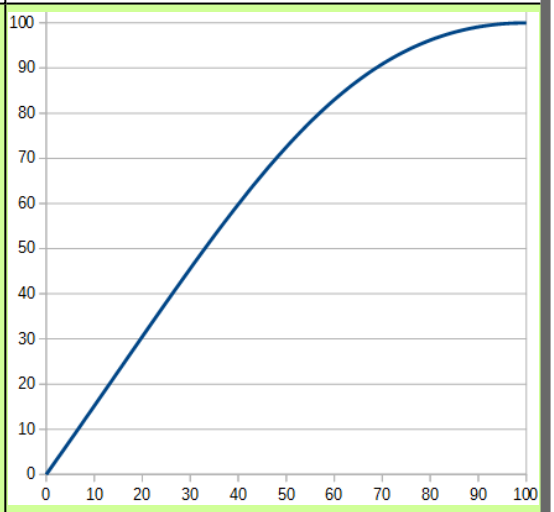
ACTION TYPE

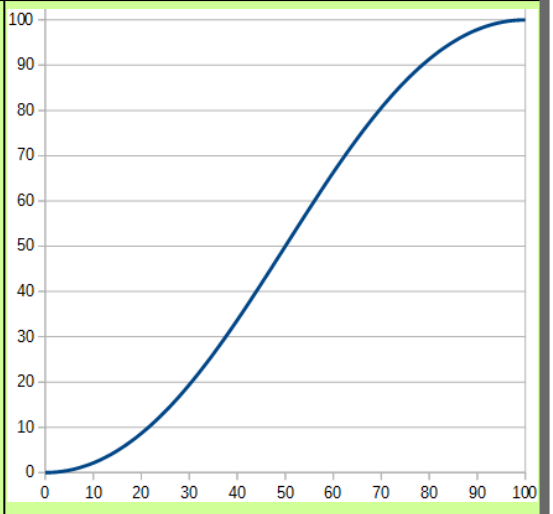
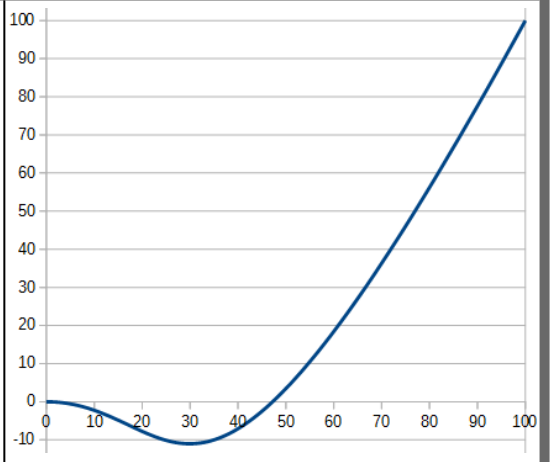
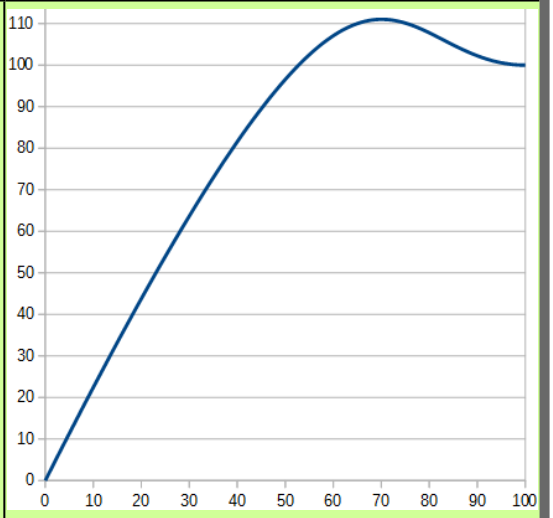
Example on contemplating the types of operation
#AOT id, **X**, 100
The parameter “X” corresponds to the parameters of the types of operation in the following table.

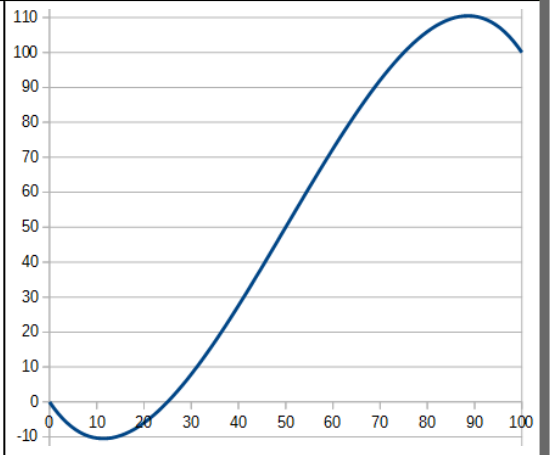
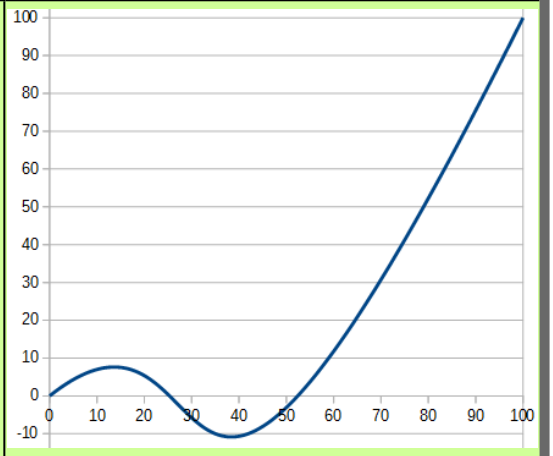
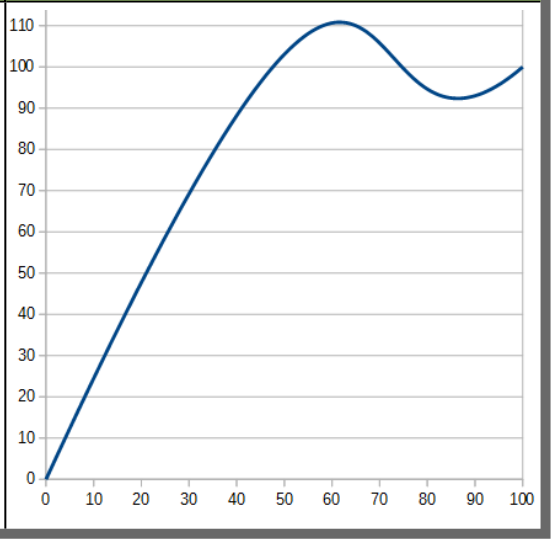
Designition	Parameter	Description
Appearing	**1**	efines an appearing behaviour for the operation selected. The object is generated at its generation point (400,240), depending upon the definition of the operation. That means that it will fly in from the position specified (1000,600), taking into account the size specified (25%), and be scaled to its original size.
Disappearing	**2**	Defines a disappearing behaviour for the operation selected. The course proceeds in the opposite direction to the appearance, in other words the object is moved from the position in which it is generated to the position specified with a corresponding change in its size. The object is subsequently deleted.
Invisible	**3**	Defines a fading out behaviour for the operation selected. The course is the same as that of the disappearing behaviour, however the object is not deleted after the operation, but becomes invisible. That means that it does not need to be re-defined for later uses
Change	**4**	Defines a one-off change in the parameters for the operation selected. The course is the same as that of the disappearing behaviour, however the object is not deleted after the operation.
Cyclical	**5**	Defines a cyclic behaviour for the operation selected. The course corresponds to the changing behaviour, however the latter is constantly repeated
Oscillating (ping-pong)	**6**	Defines an oscillating behaviour for the operation selected. The course resembles that of the cyclic behaviour, however with the difference that the object is not constantly moved from the point of its generation to the position specified, but always oscillates between the latter with the dimensioning specified.

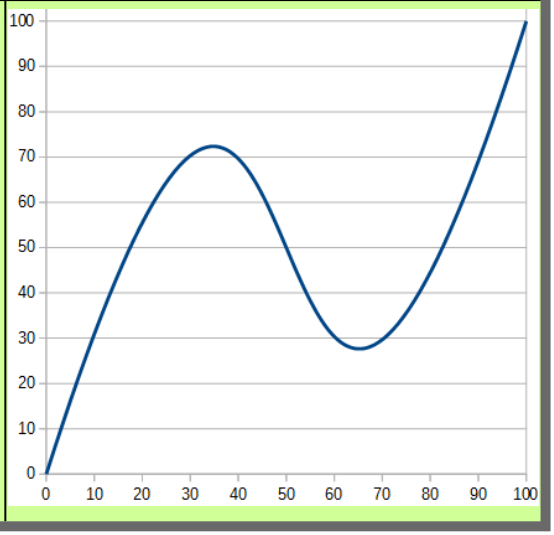
PRE-DEFINED ACTION CURVES

The course of the operation can be defined in the case of the operational curves. The chronological course of 0 to 100% can be found on the X axis. The Y axis specifies the corresponding allocated value. The latter concerns a cubic spline with two supporting points. The EA uniSketch Windows program offers help in generating one's own curve.

Action curve	Curve points	Visualisation
1: Linear	**X1** = 10; **Y1** = 10; **X2** = 90; **Y2** = 90	
2: Accelerating → linear	**X1** = 40; **Y1** = 0; **X2** = 60; **Y2** = 40	
3: Linear → reducing speed	**X1** = 40; **Y1** = 60; **X2** = 60; **Y2** = 100	

Action curve	Curve points	Visualisation
4: Accelerating → linear → reducing speed	**X1** = 40; **Y1** = 0; **X2** = 60; **Y2** = 100	
5: Undershooting → linear	**X1** = 30; **Y1** = 0; **X2** = 30; **Y2** = -60	
6:Linear → overshooting	**X1** =70; **Y1** = 160; **X2** = 70; **Y2** = 100	

Action curve	Curve points	Visualisation
7: Undershooting → linear → overshooting	**X1** = 30; **Y1** = -60; **X2** = 70; **Y2** = 160	
8:Oscillating → linea	**X1** = 40; **Y1** = 40; **X2** = 20; **Y2** = -100	
9: Linear → oscillating	**X1** = 80; **Y1** = 200; **X2** = 60; **Y2** = 60	

Action curve	Curve points	Visualisation
10: Linear → oscillating → linear	**X1** = 60; **Y1** = 200; **X2** = 40; **Y2** = -100	

KEYBOARD

To generate special keys, such as *Shift*, *Backspace* or *CAPSLOCK*, corresponding parameters need to be set in the "Button String". So that the parameters find their application, a *Backslash* '\' needs to be placed in front of them.
The "Button String" command describes the sequence of the keys within a line. In order to insert more lines, the latter need to be separated by entering '|' or ';' at the end of the line.
Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Parameter	Description
\1: (code 1)	Show Keyboard No. 1.
\2: (code 2)	Show Keyboard No. 2.
\3: (code 3)	Show Keyboard No. 3.
\4: (code 4)	Show Keyboard No. 4.
\5: (code 5)	**ShiftKey** switches over to Keyboard No. 2. Once a key has been entered, there is an automatic return to Keyboard No. 1.
\6: (code 6)	Switch over to **CapsLockKey** between Keyboard No. 1 and Keyboard No. 2.
\7: (code 7)	**Delete** deletes the characters to the right of the cursor.
\8: (code 8)	**BACKSPACE** deletes the characters to the left of the cursor.
\A: (code 10)	Move to the beginning of the next line; *reserved for future use*
\B: (code 11)	**Insert / Overwrite** switches between inserting text and overwriting text.
\C: (code 12)	**CLEAR** deletes the entire entry.
\D: (code 13)	**Enter / Return** confirms the entry and sets the entry as a new default string (#SED).
\E: (code 14)	Left arrow key
\F: (code 15)	Rightarrow key
\G: (code 16)	Up arrow key; *reserved for future use*
\H: (code 17)	DownUp arrow key; *reserved for future use*
\I: (code 18)	**Pos1** jumps to the beginning of the entry.
\J (code 19)	**Ende**jumps to the end of the entry.
\L: (code 21)	**Escape** breaks off the entry and returns to the default string(#SED).
\N: (code 23)	Placeholder for an unused and not yet defined key.
\O..\V: (code 24 .. 31)	8 x function keys

Designation	Command code	Description
Define keys	**#KDB** Group-id, Nr, "ButtonStringLine1", "ButtonStringLine2"...	Defines as an object group **Group ID** the key sequence of the keyboard **"Button String Line",** depending upon the keyboard number No. and the sequence of the keys. Further key sequences may be specified. Should the same key be entered several times in a row, then the width of the key generated changes in proportion. The latter shall also apply to spaces. Beispiel siehe S. 114
Define alternative labelling	**#KDL** Group-id, Code," ButtonText"	hanges the labelling "**Button Text**" in the object group **Group ID** with the special **Code** keys Beispiel siehe S. 114
Define styles	**#KDC** Group-id, Gap, ButtonStyle1 normal, ButtonStyle 1 special,Button Style2 normal, ButtonStyle2 special...	efines for the object group **Group ID** the spacing between keys **Gap**, as well as the button style for normal **ButtonStyle normal** and special **ButtonStyle special**.keys A maximum of five button style pairs can be entered. The numbers of the button style specify the key length for which the style is effective. Beispiel siehe S. 114
Place keyboard	**#KPK** Group-id, PosX, PosY, Anchor, Width, Height	Positions the object group **Group ID**, depending upon the position, **PosX**, **PosY**, the anchor **Anchor** and the width **Width**. Entering the height **Height** is optional. Beispiel siehe S. 114

PERIPHERALS - INTERFACE

The module has various interfaces:
- 4 analogue inputs (12-bit precision)
- 1 PWM output (16-bit precision)
- Video input (PAL/SECAM Analogue)
- Audio output (8-ohm loudspeaker, max. 1 W)
- 16 digital I/O, extendable to up to 128 (max. 20 mA)
- 3 serial interfaces RS232, SPI, I²C

Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

ANALOGUE

Designation	Command code	Description
Calibrate analogue converter	**#HAC**	Executes the calibration of the analogue converter.
Analogue input - hysteresis	**#HAH** Channel, Hysteresis (1)	Defines the hysteresis of the analogue input **Channel**(0-3). Not all changes in value are displayed by specifying hysteresis Hysteresis, but only changes in value as from the size specified. If the hysteresis is not specified, all changes in value are displayed, i.e. **Hysteresis** = 1.
Read analogue input	**#HAL** Channel, Limit, Limit2	Defines an upper and lower limit **Limit**, **Limit 2** for the analogue input **Channel**(0-3).
Read analogue input	**#HAR** Channel(0), Number(4)	Scans the analogue input **Channel**(0-3). The number **Number**(1-4) specifies the number of inputs to be read. Example: Channel = 1, Number = 2 leads to the analogue inputs 1 and 2 being scanned, and displayed via the serial interface. A more precise description of the system response can be found in the chapter Responsens.(#HAR)(#HAR)

PULSE WIDTH MODULATION (PWM)

Designation	Command code	Description
PWM output frequency	**#HFO** Frequency32Bit , OnTime(no change), TotalTime(no change)	Defines the frequency **Frequency**(2Hz-1MHz) of the PWM output. Internal splitters lead to not every frequency being precisely adjustable. The ratio of on time to off time is determined by the two parameters **OnTime/TotalValue**. Beispiel siehe S. 111
Duty cycle - PWM output	**#HFD** OnTime, TotalTime(no change)	Changes the ratio of on time to off time **OnTime/TotalValue**.

PORT

Designation	Command code	Description
Port setting	**#HPC** Port, I/O, ...	Setting for the entire port(0-15). In this context, **I/O**(0-0xFF) defines the respective port bit as an input (1) or output (0). Further specifying **I/O** defines the next higher value port. Beispiel siehe S. 111
Set outputs	**#HPW** Port, PortState, ...	Bring output of the component **Port** into the **PortState** state (0-255). Further details of PortState describe the next port component. Beispiel siehe S. 111
Read inputs	**#HPR** Port(0), Number(2)	Scans the port component **Port**(0-15). Multiple port components can be read with the number **Number**(1-16). As the standard, two port components exist. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#HPR)
Information on the port module	**#HPI**	Return of the connected port components. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#HPI)
Port pin setting	**#HBC** Pin, Output/Input, ...	Determines whether the port pin **Pin**(0-127) is defined as an input or output **Output/Input**(0= Output; 1= Input). Further specifying Output/Input defines the next higher value **Pin**.
Set port pin output	**#HBW** Pin, BitState, ...	Describes the **Bit Statestate** (0,1, 2= invert) of the output pin **Pin**(0-127). Further specifying Bit State defines the state of the next higher value **Pin**.

Designation	Command code	Description
Read port pin input	**#HBR** Pin(0), Number(16)	Scans the port pin **Pin** (0-127). Multiple port components can be scanned with the number **Number**(1-128). A more precise description of the system response can be found in the chapter Responsens. (#HAR) (#HBR)

VIDEO AND AUDIO

Designation	Command code	Description
Play sound	**#HTP** "Soundname"	Plays back the sound file with the name **Sound Name**. Only *.esd files can be played back. The easiest way to create these files is using the EA uniSketch WindEA uniSketch.
Stop sound	**#HTS**	Stops the playback of the sound file.
Video input window	**#HVB** Left(0), Top(0), Width(720), Height (576)	Window setting of the video input with left **Left** and top **Top** offset, as well as width **Width** and height **Height**.

RS232 MASTER INTERFACE

Designation	Command code	Description
Send ASCII	**#HRA** "string"	Sends a text **'string'** via the RS232 Master Interface as ASCII code.
Send Unicode	**#HRU** "string"	Sends a text **'string'** via the RS232 Master Interface as Unicode.
Send binary code	**#HRS** len32Bit, Data	Sends binary data **data** having the lenghth **len** via the RS232 Master Interface. The length needs to be under 1 kB.
Send file	**#HRF** <Filename>	Sends a file **<Filename>** via the RS232 Master Interface.
Data received	**#HRR** len32Bit (1024)	Scans the length **len** of the data received via RS232. The maximum length permitted is 1 kB. If the length is greater than the available data, only the latter are transmitted. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#HRR)

Designation	Command code	Description
RS232 parameter	**#HRP** Baudrate32Bit	Baud rate setting **Baud rate** of the master RS232 interface with 8-N-1. Baud rates: **Baudrates** / **Error rate in %** 9600 / +0,04 19200 / -0,08 38400 / +0,16 57600 / -0,08 115200 / +0,64 230400 / -0,80 460800 / +2,08 921600 / -3,68

SPI MASTER INTERFACE

Designation	Command code	Description
Send ASCII	**#HSA** "string"	Sends a text **'string'** via the SPIMaster Interface as ASCII code.
Send Unicode	**#HSU** "string"	Sends a text **'string'** via the SPI Master Interface as Unicode.
Send binary code	**#HSS** len, Data	Sends binary data **data** having the lenghth **len** via the SPI Master Interface. The length needs to be under 1 kB.
Send file	**#HSF** <Filename>	Sends a file **<Filename>** via the SPI Master Interface.
Data received	**#HSR** len32Bit	Scans the length **len** of the data received via RS232. The maximum length permitted is 1 kB. If the length is greater than the available data, only the latter are transmitted. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#HRR)
CS pin setting	**#HSC** State	Setup of the CS line: **State**(0=low (manually); 1=high (manually); 2=automatic)).

Designation	Command code	Description
SPI parameter	**#HSP** Frequency32Bit (in Hz), SPI Mode, DataOrder	Defines the parameters of frequency **Frequency**(15.6kHz-1MHz), mode **SPI Mode** (0-3) and data sequence **Dataorder**(0= MSB (Most Significant Bit)) of the SPI. The clock frequency is set to the Frequency, in regard to which internal splitters lead to the actual frequency always being the same or slower.Information on the SPI specification can be found here.

I²C

Designation	Command code	Description
Send ASCII	**#HIA** "string"	Sends a text **'string'** via the I²C Master Interface as ASCII code.
Send Unicode	**#HIU** "string"	Sends a text **'string'** via the I²C Master Interface as Unicode.
Send binary code	**#HIS** len32Bit, Data	Sends binary data **data** having the lenghth **len** via the I²C Master Interface. The length needs to be under 1 kB.
Send file	**#HIF** <Filename>	Sends a file **<Filename>** via the SPI Master Interface.
Data received	**#HIR** len32Bit	Scans the length **len** of the data received via RS232. The maximum length permitted is 1 kB. If the length is greater than the available data, only the latter are transmitted. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#HRR)
I²C Parameter	**#HIP** Adress8Bit, Frequency32Bit	Defines the parameters of address Address and frequency Frequency(3.9kHz-1MHz) for the I²C transmission. IInformation on the I²C specification can be found here.

SYSTEM COMMANDS

Parameters that are written in GREY in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Define project path	**#XPS** </Absolute path>	Defines the project path. Information on the path details can be found here.
Read project path	**#XPG**	Writes the current project path into the transmission buffer. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#XPG).
Send version	**#XIV**	Writes the version string into the transmission buffer. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#XIV).
Display info	**#XID**	Writes the display parameters, such as the width and height, into the transmission buffer. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#XID)
SD Card info	**#XIS**	Writes information on the SD card into the transmission buffer. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#XIS)
RAM info	**##XIR**	Writes information on theinternal RAM into the transmission buffer.. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#XIR)
Display orientation	**#XCO** Angle	Changes the **angle** Angle(0, 90, 180, 270) for aligning the display.
Define virtual display resolution	**#XCV** Width(Displaywidth), Height(Displayheight), Touchwidth(Width), Touchheight(Height)	Defines the virtual display resolution through the width **Width**, height **Height**, touch width **Touchwidth** and touch height **Touchheight**. It is therefore possible for the size of programs to be adjusted to small and large displays automatically.
Define display brightness	**#XCB** Brightness, Time, Flash	Defines the brightness **Brightness**(0-100) of the display, making reference to the modification time **Time**(hrs). The value can be saved permanently. **Flash** 0= Do not save; 1= Save.

Bezeichnung	Befehlscode		Beschreibung
RS232 slave parameter	**#XCR**	Baudrate, RS485 Adress(no change), Flash(0)	Defines the parameters for the RS232 slave: **Baud rate** (default = 115200) and RS485 Address (default = 7). The value can be saved in the flash memory **Flash** (0= Do not save; 1= Save). The format for the data transmission is 8-N-1. Dhe available baud rates are listed here .
SPI slave parameter	**#XCS**	SPI-Mode, DataOrder(no change), Flash(0)	Setting the **SPI modes**(0-3) (default = 3) and bit sequence **DataOrder** (0= MSB First; 1= LSB first) (default= 0). he value can be saved permanently. **Flash** 0= Do not save; 1= Save.Informatione on the SPI-Mode can be found here.
I²C slave adress	**#XCI**	Adress8Bit, Flash(0)	Set I²C slave address 8-bit **address** (default 0xDE). The value can be saved permanently. **Flash** 0= Do not save; 1= Save..
ASCII string to sending buffer	**#XSA**	'String'	Transmits the character string '**String**' to the transmission buffer as an ASCII string. This command is intended for displaying the Stringregister.
Unicode string to sending buffer	**#XSU**	'String'	Transmits the character string '**String**' to the transmission buffer as an Unicode string. This command is intended for displaying the Stringregister.
Hard copy to file	**#XHF**	<File>, PictureFormat(1), x(0), y(0), Anchor (7), width (display width), height (display height)	ave screenshot in the PictureFormat format, and the dimensions x, y, anchor, width, height under the File path and nameGeneral information on the picture format can be found here.
Hard copy to serial interface	**#XHS**	PictureFormat(1), x1(0), y1(0),Anchor (7), width (display width - 1), height (display height -1)	Save screenshot in the **PictureFormat** format, and place the dimensions **x, y, width, height** in the transmission buffer. General information on the picture format can be found here.
Hardcopy of video image to file	**#XVF**	<File>, , PictureFormat	Save still image of the video window in the **PictureFormat** format under the **File path** and name.Save file. General information on the picture format can be found here.
Hardcopy of video image to serial interface	**#XVS**	PictureFormat	Place still image of the video window in the PictureFormat format into the sending buffer.IGeneral information on the picture format can be found here.

Bezeichnung	Befehlscode	Beschreibung
Interrupt program sequence	**#XXW** Wait	Stops the program sequence for the duration of the waiting period Wait(hrs). This command is suitable for debugging purposes. The module works does not process any further commands during this period.
Touch adjustment (only resistive)	**#XXT**	Starts the adjustment of the resistive touch. The user needs to touch two predefined points on the screen.
Firmware reboot (HW-Reset)	**#XFB** Option	The command executes a complete restart of the module. The parameter **Option** selects the boot mode:: 1 Testmode — Display parameters and information of the module on the display. 2 Disable PowerOnMacro — Deactivates the start of the PowerOn macro (start emc). 3 Disable Default — Deactivates all standard macros. No default settings are loaded. 4 Bootmenu — All projects to be found on the SD card are listed, and can be started.

BOOT MENU AND TOUCH ADJUSTMENT THROUGH GESTURES

Both the resistive touch adjustment **#XXT**) and the boot menu **#XFB** can be accessed via two different gestures at the time of the restart.
For that purpose, directly at the start of the display (max. 1 second later), the latter needs to be pressed with a finger until the confirmation is obtained. After a further second, a message is displayed about whether a touch adjustment is supposed to take place or the boot menu is supposed to be accessed.
Should the touch adjustment take place, the finger needs to be briefly lifted, and then pressed on the display again, until the request for adjustment comes. This is the pressing of two points on the display.
Should, on the other hand, the boot menu start, the finger needs to be briefly lifted and subsequently pressed on the display twice at brief intervals.
Should neither of the two things occur, the finger may either remain on the display or be removed from the display.

The boot menu can be seen in the figure below.

The boot menu is divided into three areas.
On the left are the various projects that can be started by dialling.
On the lower right-hand side all macros can be deactivated, the test mode started or the accessing of the boot menu interrupted.
On the right-hand side at the top, the option of a one-off or permanent (saved) accessing of the corresponding selection is selected by pressing.

FILE ACCESS

Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

SD-CARD

The options for specifying the path are listed here.
General information on the SD card can be found here.

Designation	Command	Description
Set working path	**#FDS** <Path>	Defines the working path **Path**. When relative paths are specified, this folder is taken as the basis.
Reading back working path	**#FDG**	Read working path. A more precise description of the system response can be found in the chapter Responsens. (#HAR) (#FDG)
List directory completely	**#FDR** <Path> (Arbeitspfad)	Listing the files and folders that exist in the path **Path**. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#FDR)
List directory completely in the ASCII format	**#FDA** <Path> (Arbeitspfad)	isting the files and folders that exist in the path **Path**. They are displayed in ASCII format. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#FDA)
List directory folders	**#FDL** <Path> (Arbeitspfad)	Listing the folders that exist in the **Path**. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#FDL)
Create directory	**#FDC** <Path>	Create new directory in the working directory **Path**.
Delete directory	**#FDD** <Path>, Content(0)	Delete content **Content**(0= Content only; 1= Folder and content) in the working directory **Path**.
Open/create file for writing	**#FWO** <Path + Filename>, Pos (0)	Open file with the name **File Name** for writing, optionally specifying the path **Path** at the position **Pos**. If the file does not exist, it is created.
Write content into file	**#FWD** Num, Data	Describes the opened and writeable file with the number **Num** of binary values **Data**.
Write (ASCII) content into file	**#FWA** "String"	The ascii string **String** is written into the open file.

Designation	Command	Description
Write (Unicode) content into file	**#FWU** "String"	The Unicode string **String** is written into the open file..
Close writeable file	**#FWC**	Closes the file that is presently open and writeable.
Open file for reading	**#FRO** <Path + Filename>, Pos (0)	Open file with the name **File Name** for reading, optionally specifying the path **Path** at the position **Pos**.
Read content of file	**#FRD** Anz (gesamtes File)	Reads the content of the open and readable file. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#FRD)
Close file for reading	**#FRC**	Closes the file that is presently open and readable.
Delete file	**#FFD** <Path + Filename>	Delete file
Info on file/directory	**#FFI** <Path (Arbeitsverzeichnis)>	ransmits information on the current direction/file **Path**. A more precise description of the system response can be found in the chapter Responsens. (#HAR)(#FFI)
Change file/directory name	**#FFR** <Path>, <NewName>, Replace(0)	Changes the name of the file or directory at the location **Path** with the new designation **NewName**. Should another file exist with the same name, the behaviour is defined by **Replace**(0= Only change if the file does not already exist; 1= Always change → Other file is deleted).
Copy file/directory	**#FFC** <PathOld>, <PathNew>, Replace(0)	Copies the file or directory from the original location **PathOld** to the new location **PathNew**. If a file with the same name exists, the behaviour is defined by **Replace** (0= Copy if file does not already exist; 1= Overwrite).
Move file/directory	**#FFM** <PathOld>, <PathNew>, Replace(0)	Shifts the file or directory from the original location **PathOld** to the new location **PathNew**. If a file with the same name exists, the behaviour is defined by **Replace** (0= Shift if file does not already exist; 1= Overwrite).
File/directory Time stamp	**#FFT** <Path>, time, date	The time stamp is comprised of the **time** and **date** together. .For an explanation on file time,e see p. 102.

Designation	Command	Description
File/directory Change properties	**#FFA** <Path>, Fileattribute	Sets a new attribute for the **Path** file specified. The **File Attribute** parameter is designed as follows: Write-protected 0x01 Hidden 0x02 System 0x04 Archive 0x20 The flag attributes can be combined through simple bit ORing.
Saving a uniTFT file	**#FSO** <Path>, File-Data	Saves an object file of the uniTFT's (e.g. *.epg) in the specified path **Path**.
Load file names into string register	**#FNF** <Path> (Arbeitsverzeichnis)	Saves the number in Register R1 and the name of the files in S1, S2, ... from the destination folder **Path** S. number Should, for example, the two directories "Project1" and "Project2" be present in the destination folder, as well as the file "start.emc", the number “1” is saved in Register R1, and the “start” string in the String Register S1.
Load directory names into string register	**#FND** <Path> (Arbeitsverzeichnis)	Saves the number in Register R1 and the names of the folders in S1, S2, ... from the destination folder **Path** S. number Should, for example, the two directories "Project1" and "Project2" be present in the destination folder, as well as the file "start.emc", the number “2” is saved in the register R1, the "Project1" string in the String Register S1 and the “Project2” string in the String Register S2.

FILE TIME

The file time is comprised of the date and time together. In the FAT file system, the time specification has a resolution of 2 s, calculated as from 00:00:00 hours on 01/01/1980.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Date	JYear, counting from 1980 (0..127)							Month(1..12)				Day(1..31)				
Time	Hour(0..23)					Minute (0..59)						Second/2 (0..29)				

Brief example for converting the time:
Date = ((Year-1980)<<9) + (Month<<5) + Day;
Time =(Hour<<11) + (Minute<<5) + (Second>>1);

RESPONSES

General information on macros can be found here..
Parameters that are written in **GREY** in the command tables are considered optional details, and partially have default values. The default values are given in brackets behind the corresponding parameters. Parameters written in **BLACK** on the other hand, must be assigned values. Commands that may contribute towards changing parameters can only be used if the corresponding parameters have already previously been defined. Some commands allow for entering several object IDs. Should these objects be in numerical sequence, the range may be specified with the "-" sign, e.g. 1-5, instead of 1,2,3,4,5.

Designation	Command code		Description
Send version	#XIV	String, 0	Returns the current version as a **string**. (←XIVEA uniTFT V1.0)
Display info	#XID	Width 16Bit, Height 16Bit, Colordepth 8Bit, Touch 8Bit, Videowidth 16Bit, Videoheight 16Bit	Returns the width **Width**, height **Height**, colour depth **Colour depth**(16-bit/24-bit), touch control function, **Touch**(0 = No touch control; 3 = resistive; 7 = capacitive; 15 = PC mouse) and the dimensions of the video image through the latter's width, **Video Width** and height, **Video Height**. Should the value 1 (top/bottom) or 2 (left/right) be returned when using Touch, then the other respective directional value is missing.
RAM info - byte	#XIR	Gesamt RAM 32Bit, Frei RAM 32Bit	Return of the overall and available memory on the SD card in KB.
SD Card info in KB	#XIS	Gesamt SD 32Bit, Frei SD 32Bit	Rückgabe des Gesamt- und freien Speichers auf der SD-Karte in KB.
Time in binary code	#WSB	Hour 16Bit, Min 16Bit, Sec 16Bit, Day 16Bit, Month 16Bit, Year 16Bit, Weekday 16Bit	Returns the time and date in hours **Hours**, minutes, **mins**, and seconds, **Secs**, day **Day**, month **Month**, year **Year** and weekday **Weekday**(0=Sunday).
Time - ASCII	#WSA	String, 0=Ende	Returns the time and date as a **string** in ASCII code. (e.g. "#WSA08:30:36\n Tuesday, 3 May 2016")
Time in Unicode	#WSU	UnicodeString, 0=Ende	Returns the time and date as a **string** in Unicode code. (e.g. "#WSU08:30:36\n Tuesday, 3 May 2016")
Retain project path	#XPG	String, 0	Returns the project path as a **string**.
Hard copy to serial interface	#XHS	Bitmap Header, Data	Sends a screenshot to the serial interface.
Retain working directory	#FDG	String,0, 0=Ende	Displays the working directory as a **string**.

Designation	Command code	Description
Read directory	**#FDR** NameString , 0, Filesize 32Bit, File-attribute8Bit, Modified Time 16Bit, Modified Date16Bit, ..., ..., 0=ENDE	Reads the content of the directory with name **Name String**, size **File Size** and properties, **File Attributes**. The file attributes can be combined through bit ordering. Write-protected 0x01 Hidden 0x02 System 0x04 Archive 0x20
Read directory (ASCII)	**#FDA** SizeString, Datestring, Timestring, Namestring, 0x0D, 0x0A, ..., ..., 0=ENDE	Reads the content of the directory as ASCII code with size **Size String**, date D**ate String**, time Time String and name **Name String.**
List directories	**#FDL** Namestring, 0, Namestring, ..., 0=ENDE	Lists the directories with the name **Name String**.
Info on directories	**#FFI** NameString , 0, Filesize 32Bit, File-attribute8Bit, Modified Time 16Bit, Modified Date16Bit, ..., ..., 0=ENDE	Displays information on the directory with name, **Name String**, size, **File Size** and properties, **File Attributes**. The file attributes can be combined through bit ORing. Write-protected 0x01 Hidden 0x02 System 0x04 Archive 0x20
Read file content	**#FRD** Len 32Bit, Data	Transmits the content of the file opened for reading, **Data**, complete with the length, **Len**.
Number of port modules	**#HPI** Subscriber16Bit	Information about which port module is connected. **Subscriber** is bit-coded. Bit0 = Address 0 .. Bit7 = Address 7...
Read Port(Byte)	**#HPR** Portnr 8Bit, Number 8Bit, Value 8Bit	The port status **Value** is transmitted. **Portnr**/port number specifies the port as from which data is being read, **number** the number of ports to be read.
Read Port(Bit)	**#HBR** Bitnr 8Bit, Number 8Bit, Value 8Bit	The pin status **Value** is transmitted. **Bit Number** specifies the port pin as from which data is read, **number** the number of pins to be read.
Read analogue input	**#HAR** Channel 8Bit, Number 8Bit, Value 16Bit	The analogue value, **Value**, is transmitted. **Channel** specifies the analogue channel as from which data is read, **number** the number of channels to be read.
Master RS232 data	**#HRR** Len 32Bit, Data	Return of the data received via the Master-RS232, **Data**, with the length **Len**.

Designation	Command code	Description
Master SPI data	**#HSR** Len 32Bit, Data	Return of the data received via the Master-SPI , **Data**, with the length **Len**.
Master I²C data	**#HIR** Len 32Bit, Data	Return of the data received via the Master-I²C, **Data**, with the length **Len**.
String file - number	**#VFC** Number 16Bit	Displays the number, **Number**, of the strings contained in the string file.
String register - ASCII	**#VSA** Number 16Bit, StrLen 16Bit, String	Displays the string, **String** of the string register, with the number, **Number** and the string length, **StrLen**, as an ASCII string..
String register in Unicode	**#VSU** Number 16Bit, StrLen 16Bit, Uni-codeString	Displays the string, **String** of the string register, with the number, **Number** and the string length, **StrLen**, as an Unicode-string...
Read register	**#VRG** Number 16Bit, Type 16Bit, Value 32Bit	Returns the content, **Value** , of the register **number**. In this context, **Type** determines whether the value is an “I” = integer or “F”=Float.
Request touch control button status	**#TQS** Object-id 16Bit, Status 16Bit	Displays the **Status**(1=Up; 2=Down) of the touch control switch, **Object ID.**
Request the status of the active switch of the radio group	**#TQR** Group-id 16Bit, Object-id 16Bit	Displays the **Object ID** of the active touch control switch of the radio group, **Group ID.**.
Display touch code for key-board	**#TGK** Object-id16Bit, Code16Bit	Displays the **Code** of the touch control button of the keyboard, **Object ID** .
Display touch value - instrument	**#TQI** Object-id16Bit, Value16Bit	Displays the **Value**, of the instrument **Object ID.**

EXAMPLES OF COMMANDS

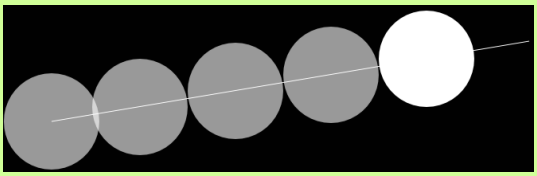
In the following, most of the graphical commands are illustrated by brief examples.

OPERATION AND ANIMATION

Position curve, scale curve

Operational path with change of position

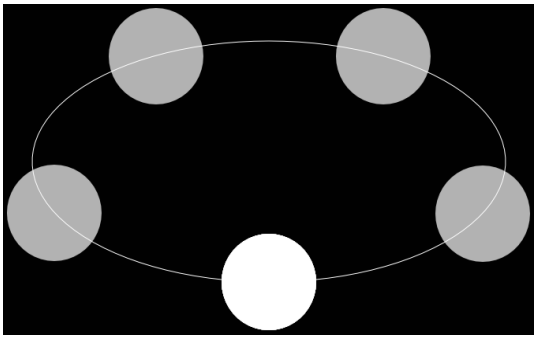
Draw a straight line in order to visualise the operational path
#GPL 1, 2, 100,100, 700,200
Insert a circle with a filling
#GET 2, 1, 100, 100, 5, 60
Move operatoin via Path 1
#AOA 2, 101, 700, 200
Define type of operation and time of operation
#AOT 2, 5, 500



For the complete set of commands of the sample image see p. 131.

Elliptical operational path with change of position

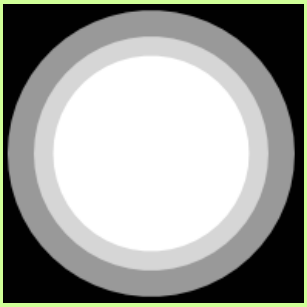
Insert an ellipse polypath in order to visualise the operational path
#GPP 1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
Insert a circle with a filling
#GET 2, 1, 400, 80, 5, 60
Define ellipse and line operational path. The path corresponds to Object 1
#APD 1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
Move operation via Path 1
#AOA 2, 151, 1, 0
Define type of operation and time of operation
#AOT 2, 1, 400



For the complete set of commands of the sample image see p. 133.

Changing the size

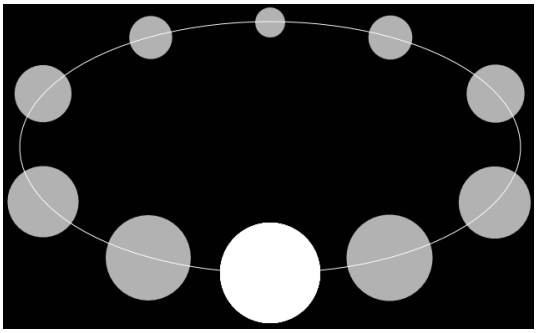
Insert a circle with a filling
#GET 1, 1, 200, 200, 5, 50
Double the size of the operation
#AOA 1, 201, 200, 200
Define type of operation and time of operation
#AOT 1, 5, 500



For the complete set of commands of the sample image see p. 131.

Changing the size and position

Insert a polypath ellipse in order to visualise the operational path
#GPP 1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
Insert a circle with a filling
#GET 2, 1, 400, 80, 5, 60
Define ellipse operational path. The path corresponds to Object 1
#APD 1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
Change in size of operational path
#APD 2, 100, 100 ?L, 30, 30 ?L, 100, 100
Operational movement and change in size
#AOA 2, 151, 1, 0, 251, 2, 0
Define type of operation and time of operation
#AOT 2, 1, 400



For the complete set of commands of the sample image see p. 132.

Rotation Curve, Shear Curve

Rotation

Define rectangle as being horizontal

#GRR 1, 1, 50,100,4, 300,40,5

Operational rotation through 90° anti-clockwise

#AOA 1, 301, 90

Define type of operation and time of operation

#AOT 1, 4, 500

For the complete set of commands of the sample image see p. 132.

Changing the rotation and position

Insert an ellipse polypath in order to visualise the operational path

#GPP 1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80

Insert a circle with a filling

#GRR 2, 1, 400, 80, 5, 20,100,2

Define ellipse operational path. The path corresponds to Object 1

#APD 1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80

Change in size of operational path

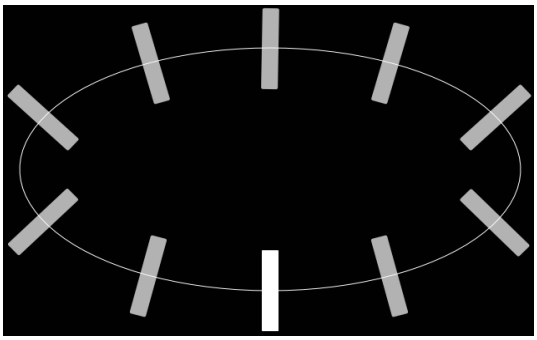
#APD 2, 100, 100 ?L, 30, 30 ?L, 100, 100

Move operation along the path, revolving

#AOA 2, 151, 1, 0, 351,1,0

Define type of operation and time of operation

#AOT 2, 1, 400



For the complete set of commands of the sample image see p. 134.

Distort

Define square

#GRR 1, 1, 100,100,5, 80,80,5

Operation: Shearing of object by 40° on both axes

#AOA 1, 401, 40,40

Define type of operation and time of operation

#AOT 1, 4, 500

For the complete set of commands of the sample image see p. 134.

Opacity curve, colour offset curve

Opacity

Define square
#GRR 1, 1, 100,100,5, 80,80,5
Operation: change the alpha channel
#AOA 1, 501, 20
Define type of operation and time of operation
#AOT 1, 4, 500

For the complete set of commands of the sample image see p. 130.

Colour transformation

Define square
#GRR 1, 1, 100,100,5, 80,80,5
Colour transformation operation Remove red and green portion
#AOA 1, 501, -100,-100
Define type of operation and time of operation
#AOT 1, 4, 500

For the complete set of commands of the sample image see p. 130.

BILD / VEKTORGRAFIKEN

Bilder / Vektorgrafiken

Plazierung eines (Vektor-)Bildes

Tiger platzieren, Größe proportional skalieren

#PPP 1, "tiger"; 400, 240, 5, 150

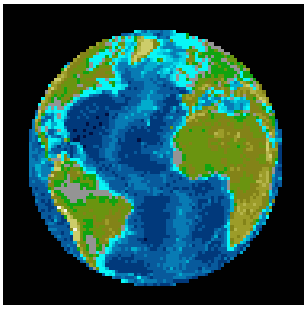
Platzierung einer Animation mit Typänderung

Erde platzieren, Größe proportional skalieren

#PPP 1, "Erde"; 400, 240, 5, 150

Zyklische Animation in eine Ping-Pong-Animation (pendelnd) ändern

#PPA 1,3



Video

Plazierung eines Videobildes

Video platzieren, Größe proportional skalieren

#PVP 1, 400, 240, 5, 300

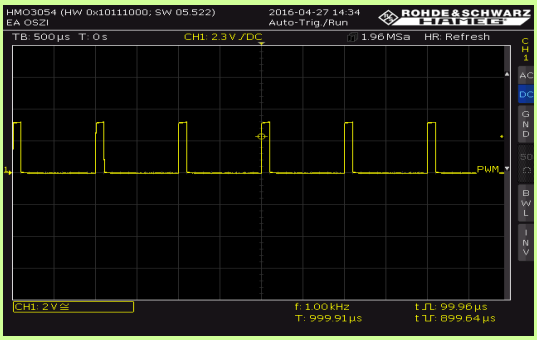
EIN- UND AUSGÄNGE

PWM

PWM Ausgang

PWM Frequenz 1 kHz, 100 µs high, 900 µs low

#HFO 1000,10, 100

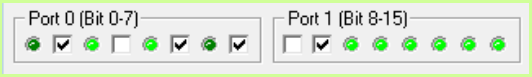


I/O Port

Port Control

Port 0 abwechselnd als Ein und Ausgang, Port 1, 2 Aus, 6 Eingänge

#HPC 0, $AA, $03



Port Ausgänge

Port 0 abwechselnd als Ein und Ausgang,Port 1, 2 Aus, 6 Eingänge

#HPC 0, $AA, $03

Verfügbare Ausgänge Abwechselnd An und Ausschalten

#HPW 0, $44, $A8

INSTRUMENTS

#IBR, IBT, IBA, IGM, IGS, IPP, IVA,

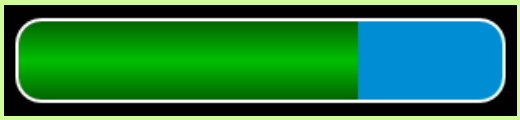
Bar graph with rounded corners

Bar graph with green gradient as a foreground, inclusive of the white frame. Background plain blue.

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15

Set bar graph to the value 70.

#IVS 1, 70



For the complete set of commands of the sample image see p. 136.

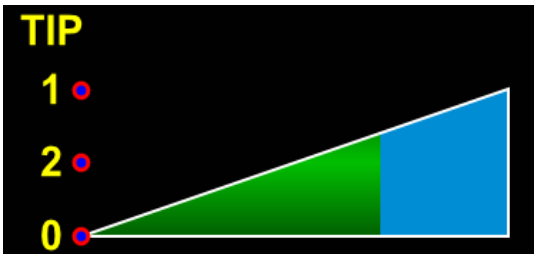
Triangular bar graph

Bar graph with green gradient as a foreground, inclusive of the white frame. Background plain blue.
As an indication, the three possible positions of the point have been high-lighted.

#IBT 1, 1, 2, 150, 70, 4, 300, 100

Set bar graph to the value 70.

#IVS 1, 70



For the complete set of commands of the sample image see p. 135.

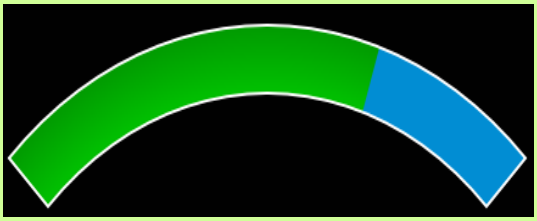
Arc - bar graph

Bar graph with green gradient clockwise as a foreground, inclusive of the white frame. Background plain blue.

#IBA 1, 1, 2, 150, 70, 4, 300, 45, 45,135

Set bar graph to the value 70.

#IVS 1, 70

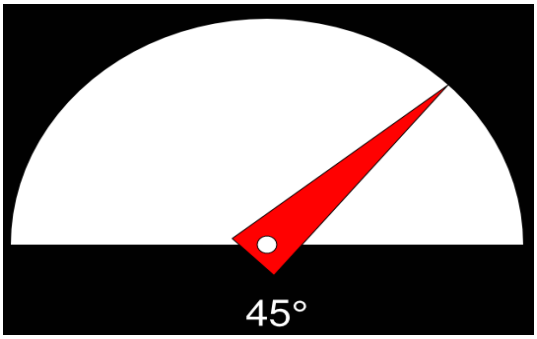


For the complete set of commands of the sample image see p. 137.

Object group as a rotational instrument

Defines Object Group 2 as a rotational instrument.

#IGM 2, 180, -180, -90, 90



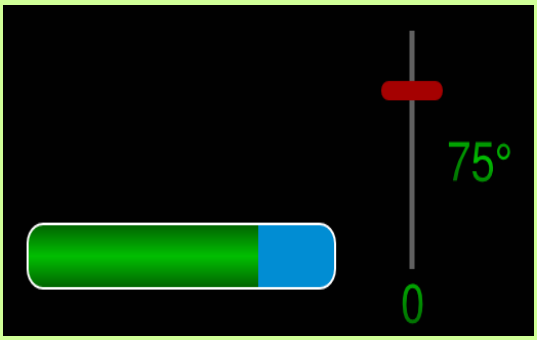
For the complete set of commands of the sample image see p. 137.

#IBR, IBT, IBA, IGM, IGS, IPP, IVA,

Object group as a slider

Defines Object Group 3 as a slider

#IGS 3, 1, 2, 0, 0, 100



For the complete set of commands of the sample image see p. 138.

Wattmeter as an instrument

Place instrument file; original image size, start value =0, end value =500

#IPP 1, "Wattmer"; 400,200,5,0,0,0,500

Set bar graph to 70% (0.7x500)

#IVS 1, 350

For the complete set of commands of the sample image see p. 136.

Bar graph value update

Rectangular bar, with start and end value, adjusted to seconds

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15,0,59

Change bar automatically with the second

#IVA 1,(second())

For the complete set of commands of the sample image see p. 136.

KEYBOARD

#KDB, KDL, KDC, KPK

Object group as a keyboard field

Define Object Group 5 as a keyboard field
Generating the keyboard field for Keyboard1

#KDB 5, 1,"^1234567890ß\8\C"; "\6qwertzuiopü+\D\D"; "\5asdfghjklöä#\N"; "<yxcvbnm,.-"; " "

Generating the keyboard field for Keyboard2

#KDB 5, 2, '°!"§$%&/()=?\8\C'; "\6QWERTZUIOPÜ*\D\D"; "\5ASDFGHJKLÖÄ'\N"; ">YXCVBNM;:_"; " "

For the complete set of commands of the sample image see p. 139.

Change key labelling of the special keys

Change labelling of the special keys in Object Group 5

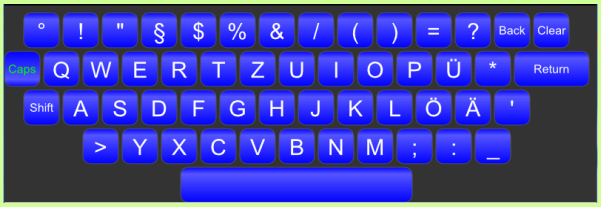
#KDL 5, 5 "Shift"; 6 "Caps"; 8 "Back"; 12 "Clear"; 13 "Return"

For the complete set of commands of the sample image see p. 139.

Define button style for keys

Define button style for normal and special keys of Length 1

#KDC 5, 1, 2



For the complete set of commands of the sample image see p. 139.

Place object group as a keyboard

Places Object Group 5 as a keyboard

#KPK 5, 0, 0, 7, 800

For the complete set of commands of the sample image see p. 139.

STRING ZEICHENKETTENBEFEHLE

#SSP, SFP, SAP

Zeichenkette platzieren

Platzierung einer einfachen Zeichenkette

#SSP 1, 1, 50, 50, 17, "Hello World"

Formatierte Zeichenkette platzieren

Platzierung einer formatierten Zeichenkette

#SFP 1, 1, 50,50,17, "Int: %d, Float: %.3f, Hex: 0x%02X"; 3, 3.14, 13

Formatierte sich selbst erneuernde Zeichenkette plazieren

Sobald sich der Analogeingang 0 ändert wird diese Zeichenkette neu dargestellt

#SAP 1, 1, 50,50,17, "Analog In 0: %.2f V"; (3.3/4095*analog(0))

Update-Kalkulation ändern

Sobald sich der Analogeingang 0 ändert wird diese Zeichenkette neu dargestellt

#SAP 1, 1, 50,50,17, "Analog In 0: %.2f V"; (3.3/4095*analog(0))

Ab jetzt wird die Zeichenkette im Sekundentakt erneuert, wobei weiterhin der Analogwert angezeigt wird

#SAC 1, (second())

Date and Time

Placing a formatted date

#SDP 1, 1, 50, 50, 17, "%W the %D. %3N %Y, %1h:%m"

For the complete set of commands of the sample image see p. 140.

EDITBOX

#SEP, SEK, SEA

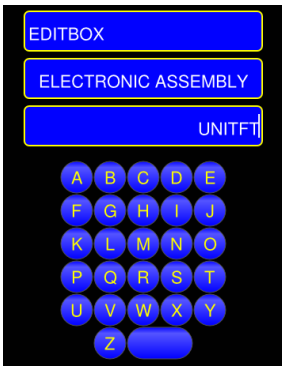
Platzierung Editbox

Platzierung von drei Editboxen

#SEP 6, 20, 400, 280, 8, 300, 50, 6, 21, -5, -3

#SEP 7, 20, 400, 340, 8, 300, 50, 6, 22, 5, -3

#SEP 8, 20, 400, 400, 8, 300, 50, 6, 23, 5, -3

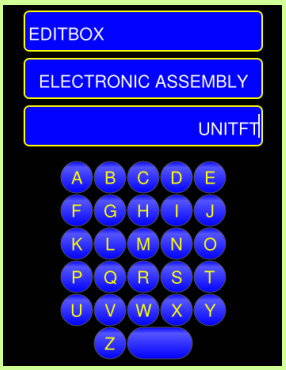


For the complete set of commands of the sample image see p. 140.

Zuweisung von drei Editboxen auf eine Tastatur

Editboxen 6-8 der Tasttatur 9 zuweisen

#SEK 9, 6-8

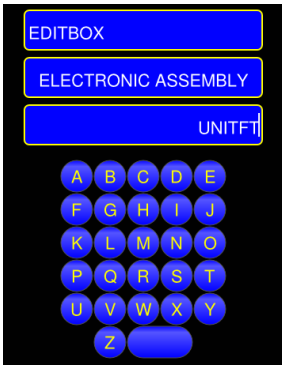


For the complete set of commands of the sample image see p. 140.

Aktivierung Editbox

Aktiviert Editbox 8, die oberste der drei Editboxen

#SEA 8



For the complete set of commands of the sample image see p. 140.

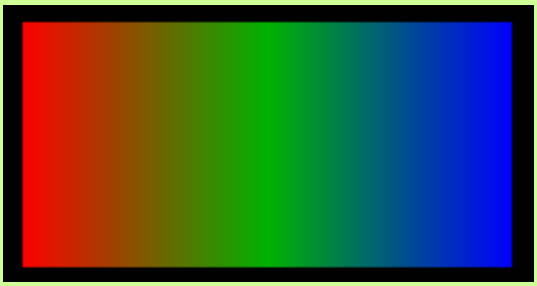
STYLE SHEETS

Farbverlauf und Linienmuster

Farbverlauf

Erstellung eines Gradientenverlaufs mit 3 Stützpunkten RGB,
wobei Grün leicht durch den Alphakanal abgedunkelt wird.

#CCR 1, 0,$FF0000,100, 50,$00FF00,70, 100,$0000FF,100



For the complete set of commands of the sample image see p. 141.

Linienmuster

Erstellung eines Linenmusters (Dash pattern),

#CDP 1, 0, 20,10, 5,10

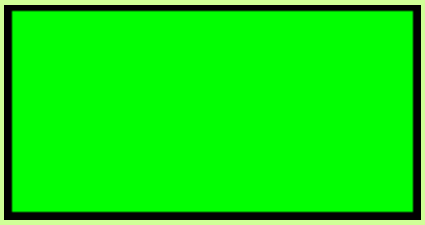
For the complete set of commands of the sample image see p. 141.

Füllungen

Einfarbige Füllung

Füllfarbe definieren, der Alphakanal ist optional,

#CFC 1, $00FF00

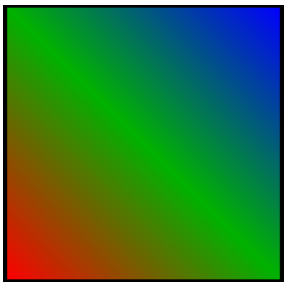


For the complete set of commands of the sample image see p. 142.

Lineare Farbverlaufsfüllung

Gradientenverlauf als linearen Verlauf nutzen - um 45° geneigt

#CFL 1, 1, 45

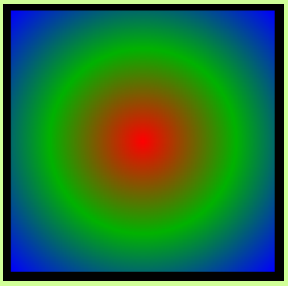


For the complete set of commands of the sample image see p. 142.

Radiale Farbverlaufsfüllung

Gradientenverlauf als radialen Verlauf nutzen.Anker 5 als Fokuspunkt nutzen

#CFR 1, 1, 5000,5



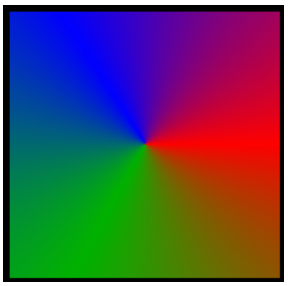
For the complete set of commands of the sample image see p. 142.

Konische Farbverlaufsfüllung

Gradientenverlauf als konischenVerlauf nutzen.

Anker 5 als Fokuspunkt nutzen. Die Drehrichtung is im Uhrzeigersinn

#CFK 1, 1, 5000,5



For the complete set of commands of the sample image see p. 142.

Füllungen

Musterfüllung

Füllung mit sich selbst wiederholendem Muster

#CFP 1, "marmor02"

For the complete set of commands of the sample image see p. 142.

Linienstyle

Linienfarbe und -dicke

Keine Füllung definieren

#CFD 1

grüne Linie mit 5 Pixel dicke

#CLS 1, $00FF00,100, 5

Rechteck zeichnen

#GRR 1,1, 50,50,7,200,100

For the complete set of commands of the sample image see p. 143.

Linienverbindung

Linienverbindung mit abgerundeten Ecken (Round = 1)

#CLE 1, 1

Linienverbindung mit spitzen Ecken (Miter = 0)

#CLE 2, 0

For the complete set of commands of the sample image see p. 143.

Textstyle

Textstyle

Erstellung eines Textstyles mit dem internen Font "Sans"

#CTF 1,<I:Sans.evf>, 60,1,1

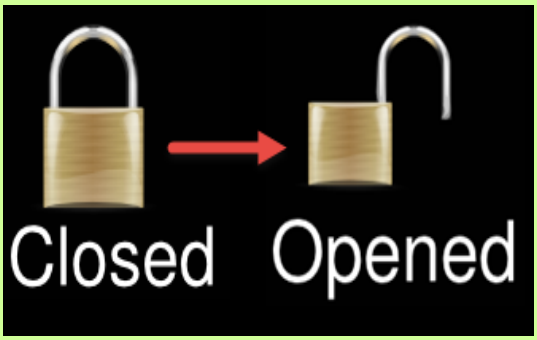
For the complete set of commands of the sample image see p. 143.

Buttonstyle

Touchbutton als Bild

Zwei Bilder als Touchbuttonstyle definieren

#CBP 1, "Padlock_closed"; "Padlock_opend"

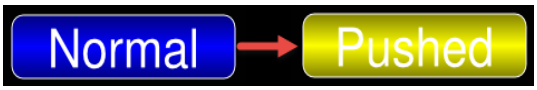


For the complete set of commands of the sample image see p. 144.

Touchbuttonstyle für geometrische Objekte

Touchbuttonstyle mit 200 Breite, 50 Höhe und Eckenradius 50

#CBD 1, 5,6, 200,50,10



For the complete set of commands of the sample image see p. 144.

Touchbutton Proportion bei Drücken ändern

Touchbutton soll seine Größe auf 70% im gedrückten Zustand ändern

#CBO 1, 0,0, 70

For the complete set of commands of the sample image see p. 144.

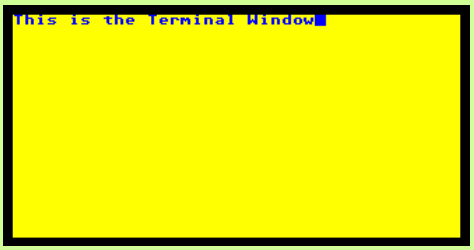
TERMINAL COMMANDS

Terminal window

(Re-)define terminal window

Small terminal window with 40 characters and 20 lines

#YDW 1, 30,30,7, 40,20



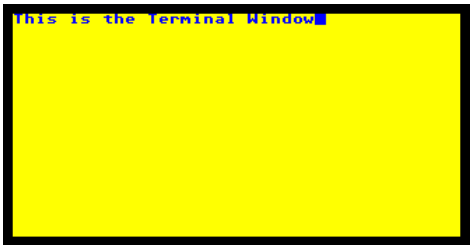
For the complete set of commands of the sample image see p. 145.

Terminal window foreground/background colour

Small terminal window with 40 characters and 20 lines

Font colour blue, window colour yellow, non-transparent

#YDC $0000FF,100, $FFFF00,100



For the complete set of commands of the sample image see p. 145.

TOUCH CONTROL OBJECTS/TOUCH CONTROL FUNCTIONS

Touch control zones

Rectangular touch button

Set touch control switch to position 400,240.
External dimensions are defined by the button style

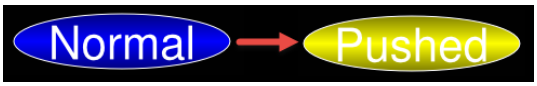
#TSR 1, 1, "Normal"; "Pushed"; 400,240

For the complete set of commands of the sample image see p. 146.

Elliptical touch button

Set touch control switch to position 400,240.
External dimensions are defined by the button style

#TSE 1, 1, "Normal"; "Pushed"; 400,240

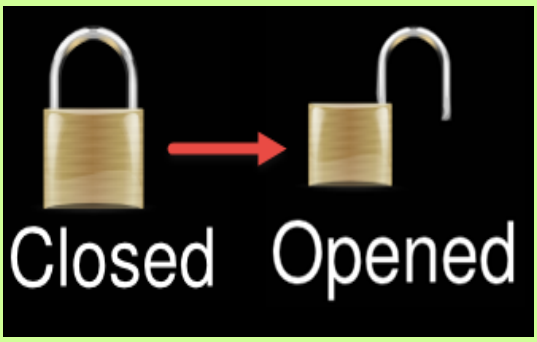


For the complete set of commands of the sample image see p. 146.

Touch button as an image

Use two images as a touch button

#TSP 1, 1, "Closed";"Opened"; 400,240,7



TIME

#WGC

Object group as a clock

Define Object Group 10 as a clock

Objects 6, 7 and 8 form the clock hands

#WGC 10, 6, 7, 8

For the complete set of commands of the sample image see p. 147.

DRAWING/GRAPHIC PRIMITIVES

Rounded rectangle with border

Rounded rectangle with border

Rounded rectangle with border, a type of picture frame

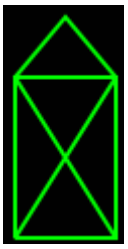
#GRR 1,1, 400,240,5, 200,100,20, 15

For the complete set of commands of the sample image see p. 148.

Polyline - Santa Claus house

Draw a Santa Claus house with the aid of a polypath

#GPL 1,1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,130, 20,100, 70,20

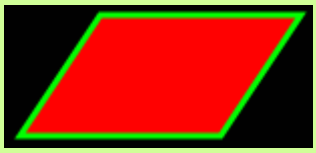


For the complete set of commands of the sample image see p. 148.

Filled-in polyline - trapezium

Draw a filled-in trapezium with the aid of a polypath

#GPF 1,1, 20,20, 60,80, 160,80, 120,20

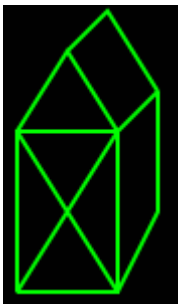


For the complete set of commands of the sample image see p. 148.

Extend a polyline - Santa Claus house

Extend the Santa Claus house by the 3 dimensions

#GPA 1, 1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,130, 20,100, 70,20



For the complete set of commands of the sample image see p. 148.

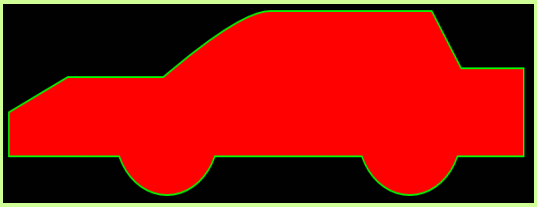
Graphic drawing objects

Polypath - Draw a car

Draw a car with path segments

#GPP 1,1, 50,200, ?v-50, ?h150, ?c0,70,130,0, ?h200, ?c0,70,130,0, ?h90, ?v100, ?h-85, ?l-40,65, ?h-220, ?q-40,0,-145,-75, ?h-130, ?z

Course of the relative polypath:
The line is drawn vertically, 50 pixels downwards from the starting point. A horizontal alignment then follows towards the right. The wheels are drawn between the three horizontal segments using segments of a circle. The wheels are drawn in such a way that their width amounts to 130 pixels. A vertical alignment by 100 pixels upwards follows, as well as a vertical line by 85 pixels to the left. For the rear windscreen, a line segment is used, which sets the end point to 40 pixels to the left and 65 pixels above the last position, and connects it by means of a line. A horizontal line to the left, having a length of 220 pixels, then follows. For the front windscreen, a quadratic Bezier segment is used. A horizontal line to the left with 130 pixels, as well as the end of the path, follows thereafter.

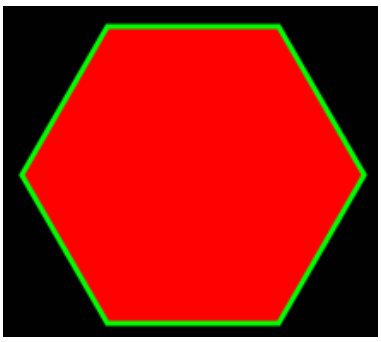


For the complete set of commands of the sample image see p. 148.

Hexagon

A hexagon rotated through 90°

#GGP 1, 1 400,240,5, 100,6,0, 90



For the complete set of commands of the sample image see p. 149.

Five-point star

Draw a filled-in star

#GGS 1, 1 400,240,5, 100,50, 5

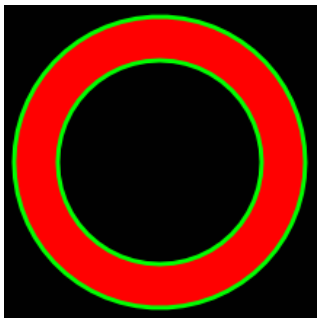
For the complete set of commands of the sample image see p. 149.

Graphic drawing objects

Tyres

A circle with a hole

#GET 1, 1 400,240,5, 100,100, 30



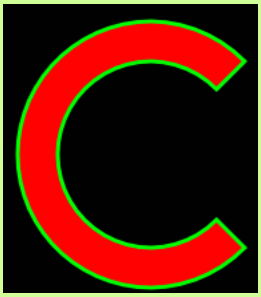
For the complete set of commands of the sample image see p. 149.

Graphic drawing objects

Arc of a circle

Draw a letter “C” as an arc of a circle

#GEA 1, 1 400,240,5, 100,100, 45,-45, 30



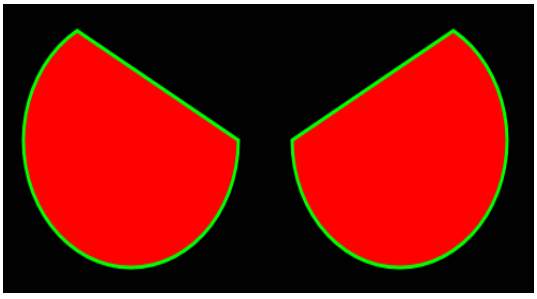
For the complete set of commands of the sample image see p. 149.

Cut segments of a circle

Draw two cut segments of a circle

#GES 1, 1, 400, 200, 5, 100, 100, 90, 0

#GES 2, 1, 450, 200, 5, 100, 100, 180, 60

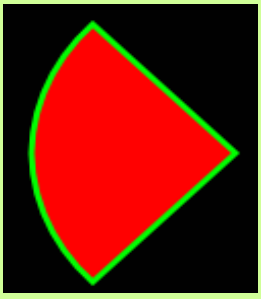


For the complete set of commands of the sample image see p. 150.

Piece of cake

Draw a piece of cake

#GEP 1, 1 400,240,5, 100,100, 135,225



For the complete set of commands of the sample image see p. 150.

SEGEMENTTYPEN

Segmente H(orizontal), V(ertikal), L(inear)

Polypfad Horizontal

Horizontale Linie als Polypfad erzeugen

#GPP 1, 1, 10, 50, ?H 300

For the complete set of commands of the sample image see p. 151.

Polypfad Vertikal

Vertikale Linie als Polypfad erzeugen

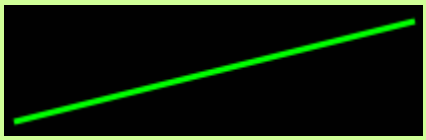
#GPP 1, 1, 50, 50, ?V 150

For the complete set of commands of the sample image see p. 151.

Polypfad Linie

Linie als Polypfad erzeugen

#GPP 1, 1, 50,50, ?L 250,100



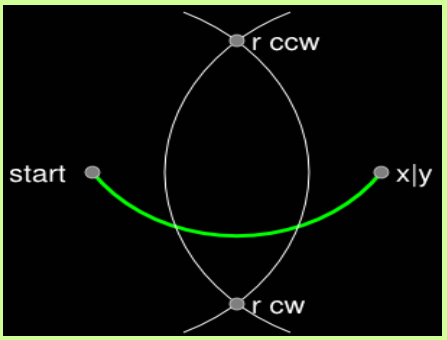
For the complete set of commands of the sample image see p. 151.

Segmente C(ircle), E(llipse)

Polypfad Kreissegment

Kreissegment gegen den Uhrzeigersinn mit dem Radius 120. Der Radius muss mindestens Abstand/2 betragen

#GPP 1, 5, 200,200,?C0,120,400,200

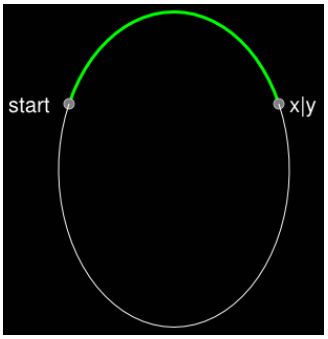


For the complete set of commands of the sample image see p. 152.

Polypfad Ellipse

Ellipse durch die beiden Punkte

#GPP 1, 5, 200,300,?E,110,150,0,400,300



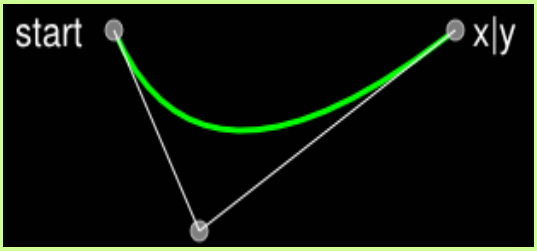
For the complete set of commands of the sample image see p. 152.

Segmente Q(uadratische Bézier Kurve), R

Polypfad quadratische Bézierkurve

Eine quadratische Bézierkurve platzieren. Der Stützpunkt bestimmt die gezeichnete Kurve

#GPP 1, 5, 200,300,?Q 250, 200,400,300

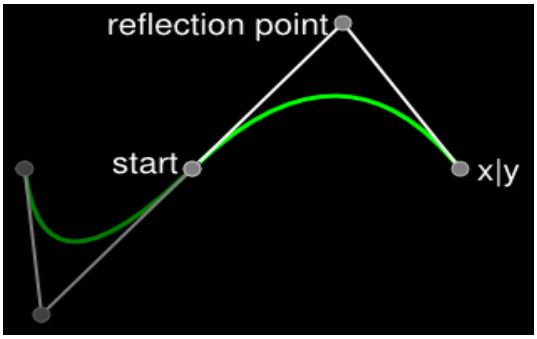


For the complete set of commands of the sample image see p. 153.

Polypfad quadratische Bézierkurve mit Hilfspunkt

Eine quadratische Bézierkurve platzieren. Der Stützpunkt wird durch das vorher gezeichnete Segment vorgegeben (Punktspiegelung) Dadurch werden die einzelnen Segmente immer tangential Zusammengefügt und bilden eine geglätteten Übergang.

#GPP 1, 5, 200,300,?Q 210,200,300,300, **?R 460,300**



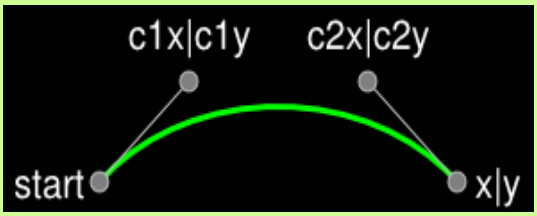
For the complete set of commands of the sample image see p. 154.

Segmente T, S(pline)

Polypfad Spline

Spline zeichnen

#GPP 1, 5, 100,50,?S150,100,250,100,300,50

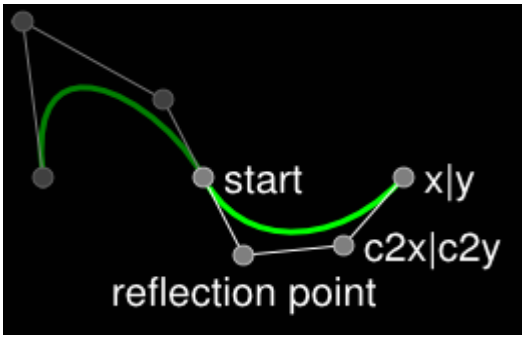


For the complete set of commands of the sample image see p. 155.

Polypfad geglättete kubische Bézierkurve

Die geglättete kubische Bézierkurve übernimmt von dem vorhergegangen Pfad den letzten Stützpunkt und spiegelt diesen am Startpunkt der eigenen Kurve. Es ergibt sich so ein tangentialer also glatter Übergang zwischen beiden Segmenten.

#GPP 1, 5, 200,200, ?s-10,80,60,40,80,0, **?t70,-35,100,0**



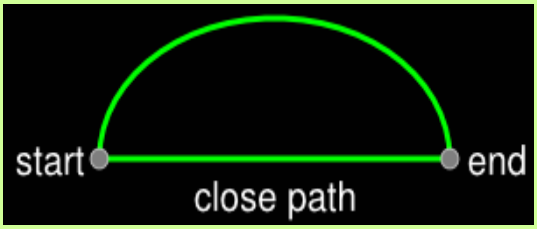
For the complete set of commands of the sample image see p. 156.

Segmente Z, M

Polypfad schließen

Nach Zeichnen der Ellipse wird der Polypfad geschlossen

#GPP 1, 5, 200,200, ?e1,100,70,0,200,0, **?Z**

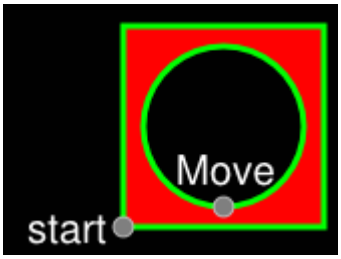


For the complete set of commands of the sample image see p. 157.

Polypfad Sprung

Platzierung eines Kreises in einem Rechteck, als ein Polypfadobjekt. Vor allem mit Füllung interessant

#GPP 1, 5, 200,200, ?h100, ?v100, ?h-100, ?Z, **?M250, 210,** ?c1,40,0,80, ?c1,40,0,-80



For the complete set of commands of the sample image see p. 157.

COMMANDS FOR THE EXAMPLES

The complete commands which are necessary for the creation of the sample images are shown below.

OPERATION AND ANIMATION

Opacity curve, colour offset curve

Opacity

```
        Define square
#GRR    1, 1, 100,100,5, 80,80,5
        Operation: change the alpha channel
#AOA    1, 501, 20
        Define type of operation and time of operation
#AOT    1, 4, 500
        Define white filling
#CFC    4, $FFFFFF, 60
        Create rectangles
#GRR    2, 4, 200,100,5, 80,80,5
#GRR    3, 1, 300,100,5, 80,80,5
```

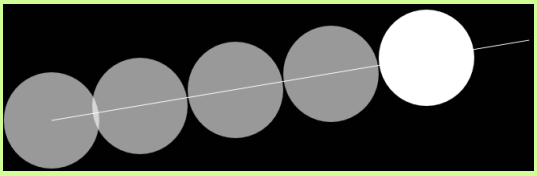
Colour transformation

```
#CFC    4, $7F7FFF, 60
        Define square
#GRR    1, 1, 100,100,5, 80,80,5
        Colour transformation operation Remove red and green portion
#AOA    1, 501, -100,-100
        Define type of operation and time of operation
#AOT    1, 4, 500
        Define remaining squares
#GRR    2, 4, 200,100,5, 80,80,5
#GRR    3, 1, 300,100,5, 80,80,5
```

Position Curve, Scale Curve

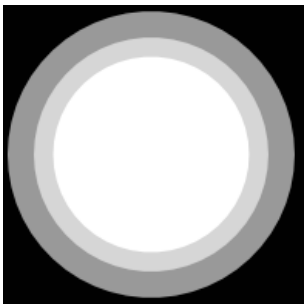
Operation path with change of position

```
        Draw a straight line in order to visualise the operation path
#GPL    1, 2, 100,100, 700,200
        Insert a circle with a filling
#GET    2, 1, 100, 100, 5, 60
        Move operation via Path 1
#AOA    2, 101, 700, 200
        Define type of operation and time of operation
#AOT    2, 5, 500
        Set integer register
#VRI    0, 3, 1,10
        Define white filling
#CFC    4, $FFFFFF, 60
        Define circles with filling, as well as their type of operation and time of
        operation
#GET    R0, 4, 100, 100, 5, 60
#AOA    R0, 101, 700, 200
#AOT    (R0++), 5, 600, 100
#GET    R0, 4, 100, 100, 5, 60
#AOA    R0, 101, 700, 200
#AOT    (R0++), 5, 700, 200
#GET    R0, 4, 100, 100, 5, 60
#AOA    R0, 101, 700, 200
#AOT    (R0++), 5, 800, 300
#GET    R0, 4, 100, 100, 5, 60
#AOA    R0, 101, 700, 200
#AOT    (R0++), 5, 900, 400
```



Changing the size

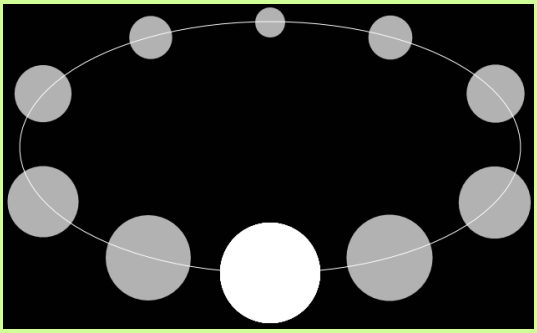
```
        Insert a circle with a filling
#GET    1, 1, 200, 200, 5, 50
        Double the size of the operation
#AOA    1, 201, 200, 200
        Define type of operation and time of operation
#AOT    1, 5, 500,0
        Define white filling
#CFC    4, $FFFFFF, 60
        Set integer register
#VRI    0, 3, 500, 1,300
        Generate circles and define their type of operation, as well as time of
        operation
#GET    R0, 4, 200, 200, 5, 50
#AOA    R0, 201, 200, 200
#AOT    (R0++), 5, (R1+R2*R3), (R2++*R3)
#GET    R0, 4, 200, 200, 5, 50
#AOA    R0, 201, 200, 200
#AOT    (R0++), 5, (R1+R2*R3), (R2++*R3)
#GET    R0, 4, 200, 200, 5, 50
#AOA    R0, 201, 200, 200
#AOT    (R0++), 5, (R1+R2*R3), (R2++*R3)
#GET    R0, 4, 200, 200, 5, 50
#AOA    R0, 201, 200, 200
#AOT    (R0++), 5, (R1+R2*R3), (R2++*R3)
```



Position Curve, Scale Curve

Changing the size and position

```
        Add an ellipse polypath in order to visualise the operation path
#GPP    1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
        Insert a circle with a filling
#GET    2, 1, 400, 80, 5, 60
        Define operational path of ellipse The path corresponds to Object 1
#APD    1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
        Operation path: Change size
#APD    2, 100, 100 ?L, 30, 30 ?L, 100, 100
        Operation: Moving and changing the size
#AOA    2, 151, 1, 0, 251, 2, 0
        Define type of operation and time of operation
#AOT    2, 1, 400
        Set integer register
#VRI    0, 3, 1,10
        Define white filling
#CFC    4, $FFFFFF, 70
        Set macro file mark
#MFM
        Create dots/circles
#GET    R0, 4, 400, 80, 5, 60
        Operation: Moving and changing the size
#AOA    R0, 151,1,(R1*R2), 251, 2, (R1++*R2)
        Define type of operation and time of operation
#AOT    R0, 1,400
        Jump back to the macro file mark
#MFJ    (R0++<20)
```



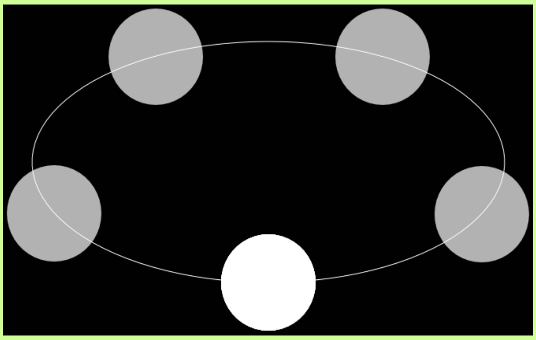
Rotation

```
        Define rectangle as being horizontal
#GRR    1, 1, 50,100,4, 300,40,5
        Operation: Rotate by 90° anti-clockwise.
#AOA    1, 301, 90
        Define type of operation and time of operation
#AOT    1, 4, 500
        Set integer register
#VRI    0, 2, 1,100
        Define white filling
#CFC    4, $FFFFFF, 70
        Set macro file mark
#MFM
        Create rectangles
#GRR    R0, 4, 50,100,4, 300,40,5
        Operation: Rotate by 90° anti-clockwise.
#AOA    R0, 301, 90
        Define type of operation and time of operation
#AOT    R0, 1,400
        Jump back to the macro file mark
#MFJ    (R0++<20)
```

Position Curve, Scale Curve

Elliptical operation path with change of position

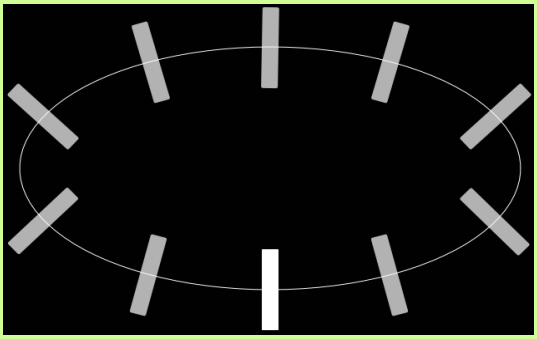
	Insert an ellipse polypath in order to visualise the operation path
#GPP	1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
	Insert a circle with a filling
#GET	2, 1, 400, 80, 5, 60
	Define ellipse and line operation path. The path corresponds to Object 1
#APD	1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
	Operation: Move object via Path 1
#AOA	2, 151, 1, 0
	Define type of operation and time of operation
#AOT	2, 1, 400
	Define white filling
#CFC	4, $FFFFFF, 70
	Set integer register
#VRI	0, 3, 1,20
	Define circles with type of operation and time of operation
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100
#GET	R0, 4, 400, 80, 5, 60
#AOA	R0, 151, 1, (R1++*R2)
#AOT	(R0++), 1, 100



Position Curve, Scale Curve

Changing the rotation and position

```
        Insert an ellipse polypath in order to visualise the operation path
#GPP    1, 2, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
        Insert a circle with a filling
#GRR    2, 1, 400, 80, 5, 20,100,2
        Define operational path of ellipse. The path corresponds to Object 1
#APD    1, 400, 80 ?E0, 300, 150, 0, 400, 380 ?E0, 300, 150, 0, 400, 80
        Move operation along the path, revolving
#AOA    2, 151, 1, 0, 351,1,0
        Define type of operation and time of operation
#AOT    2, 1, 400
        Set integer register
#VRI    0, 3, 1,10
        Define white filling
#CFC    4, $FFFFFF, 70
        Set macro file marker
#MFM
        Create rectangles
#GRR    R0, 4, 400, 80, 5, 20,100,2
        Move operation along the path, revolving
#AOA    R0, 151,1,(R1*R2), 351, 2, (R1++*R2)
        Define type of operation and time of operation
#AOT    R0, 1,400
        Jump back to the macro file marker
#MFJ    (R0++<20)
```



Distort

```
        Define square
#GRR    1, 1, 100,100,5, 80,80,5
        Shearing operation 40° on both axes
#AOA    1, 401, 40,40
        Define type of operation and time of operation
#AOT    1, 4, 500
        Set integer register
#VRI    0, 2, 1,200
        Define white filling
#CFC    4, $FFFFFF, 70
        Set macro file marker
#MFM
        Create rectangles
#GRR    R0, 4, 100,100,5, 80,80,5
        Shearing operaton 40° on both axes
#AOA    R0, 401, 40,40
        Define type of operation and time of operation
#AOT    R0, 4, (500+R1*R2), (R1++*R2)
        Jump back to the macro file marker
#MFJ    (R0++<3)
```

INSTRUMENTS

#IBT, IBR, IVA, IPP, IBA, IGM, IGS

Triangular bar graph

Define green colour gradient

#CCR 1 , 0 , $005f00 , 100 , 50 , $00Bf00 , 100 , 100 , $005f00 , 100

Refrain from defining an outline

#CLD 1

Define monochrome blue filling

#CFC 2, $008dd3, 100

Define line style for outline

#CLS 2 , $ffffff , 100 , 2

Define colour gradient as linear with 90° torsion

#CFL 1, 1, 90,

Bar graph with green gradient as a foreground. Background plain blue. As an indication, the three possible positions of the point have been high-lighted

#IBT 1, 1, 2, 150, 70, 4, 300, 100

Set bar graph to the value 70.

#IVS 1, 70

Define line style for points

#CLS 5,$FF0000,100,3

Define blue filling for points

#CFC 5,$0000FF,100

Define yellow filling for font

#CFC 6,$FFFF00,100

Refrain from defining an outline for font

#CLD 6

Define text style

#CTF 2,</Projekt/Font/Arialbd.evf>,40,1,6

Set integer register

#VRI 0,1

String Register setzen

#VSS 1,"1"

Set string register

#VSS 4,"2"

Set string register

#VSS 7,"0"

Set marker for macro file

#MFM

Create points

#GET (10+R0),5,(objX(1,R0)),(objY(1,R0)),5,5

Generate numbering

#SSP (100+R0),2,(objX(1,R0)-15),(objY(1,R0)),6,T0

Set integer register

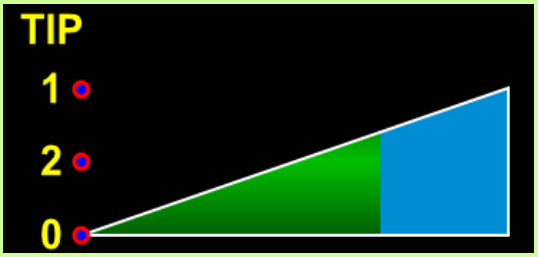
#VRI 0, (R0+3)

Command serves to jump to the marker of the macrofile again

#MFJ (R0<9)

Set numbering

#SSP (100+R0),2,(objX(1,1)),(objY(1,1)+20),9,"TIP"



#IBT, IBR, IVA, IPP, IBA, IGM, IGS

Bar graph with rounded corners

Define green colour gradient

#CCR 1 , 0 , $005f00 , 100 , 50 , $00Bf00 , 100 , 100 , $005f00 , 100

Refrain from defining an outline

#CLD 1

Define monochrome blue filling

#CFC 2, $008dd3, 100

Define line style for outline

#CLS 2 , $ffffff , 100 , 2

Define colour gradient as linear with 90° torsion

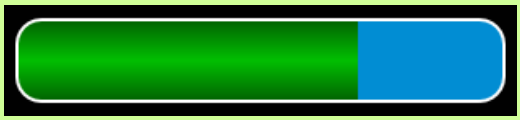
#CFL 1, 1, 90,

Rectangular bar, with start and end value, adjusted to seconds

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15

Change bar automatically with the second

#IVS 1,70



Bar graph value update

Define green colour gradient

#CCR 1 , 0 , $005f00 , 100 , 50 , $00Bf00 , 100 , 100 , $005f00 , 100

Refrain from defining an outline

#CLD 1

Define monochrome blue filling

#CFC 2, $008dd3, 100

Define line style for outline

#CLS 2 , $ffffff , 100 , 2

Define colour gradient as linear with 90° torsion

#CFL 1, 1, 90,

Rectangular bar, with start and end value, adjusted to seconds

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15,0,59

Change bar automatically with the second

#IVA 1,(second())

Define monochrome white filling for label

#CFC 3, $FFFFFF,100

Refrain from defining an outline for the font

#CLD 3

Change text style

#CTC 1,3

Place the calculated string in the centre of the object

#SAP 2,1, (objX(1,5)), (objY(1,5)),5, "%d"; (second())

Wattmeter as an instrument

Specify project path absolutely

#XPS </Projekt>

Place instrument file; original image size, start value =0, end value =500

#IPP 1, "Wattmeter"; 400,200,5,0,0,0,500

Set bar graph to 70% (0.7x500)

#IVS 1, 350

#IBT, IBR, IVA, IPP, IBA, IGM, IGS

Arc - bar graph

Define green colour gradient

#CCR 1 , 60 , $005f00 , 100 , 70 , $00Bf00 , 100 , 100 , $005f00 , 100

Refrain from defining an outline

#CLD 1

Define monochrome blue filling

#CFC 2, $008dd3, 100

Define line style for outline

#CLS 2 , $ffffff , 100 , 2

Define colour gradient as being radial

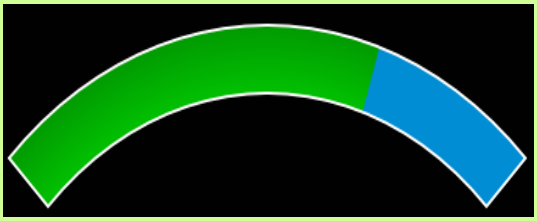
#CFR 1, 1

Bar graph with green gradient clockwise as a foreground. Background plain blue.

#IBA 1, 1, 2, 150, 70, 4, 300, 45, 45,135

Set bar graph to the value 70.

#IVS 1, 70



Object group as a rotational instrument

Define line style and filling of hands

#CLS 2, $000000

#CFC 2, $FF0000

Define line style and filling of point and disc

#CLS 3, $000000

#CFC 3, $FFFFFF

Generate disk and hands

#GES 1, 3, 300, 116, 8, 270, 270, 0, 180

#GPF 2, 2, 0,0, 30,280, 60, 0

Turn hands, set and place free anchor

#ORA 90, 2

#OAO 30, 20, 2

#OPA 300, 116, 2

Generate white point

#GET 3, 3, 270, 116, 5, 10

Generate group and define it as a rotational instrument

#OGA 5, 1, 2, 3, 4

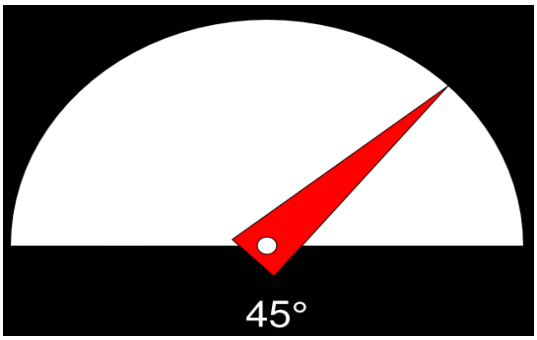
#IGM 5, 2, 180, -180, -90, 90

Assign touch function

#TID 1, 5

Place calculated string

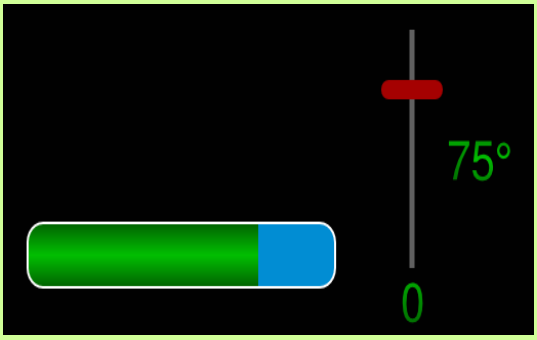
#SAP 4, 1, 310, 10, 9, "%d°"; (objIV(5))



#IBT, IBR, IVA, IPP, IBA, IGM, IGS

Object group as a slider

```
        Define green colour gradient
#CCR    1 , 0 , $005f00 , 100 , 50 , $00Bf00 , 100 , 100 , $005f00 , 100
        Refrain from defining a line style and colour gradient distorted by 90°
#CLD    1
#CFL    1, 1, 90
        Define blue filling and white line style for bar graph
#CFC    2, $008dd3, 100
#CLS    2, $ffffff , 100 , 2
        Define line style for slider path
#CLS    3, $606060, 100, 5
        Refrain from defining a line style and red filling for slider
#CLD    4
#CFC    4, $A50000, 100
        Place slider with path line
#GPL    1, 3, 425, 60, 425, 245
#GRR    2, 4, 425, 60, 5, 60, 15, 15, 0
        Generate group and define it as a slider.
#OGA    3, 1, 2
#IGS    3, 1, 2, 0, 0, 100
        Place bar graph
#IBR    4, 1, 2, 50, 70, 4, 300, 50, 15
        Assign touch function
#TID    1, 3
        Place calculated and fixed string
#SAP    5, 1, 490, 145, 5, "%d°"; (objIV(3))
#SSP    6, 1, 425, 35, 5, "0"
        Automatic change in value in the bar graph
#IVA    4, (objIV(3))
```



KEYBOARD

#KDB, KDL, KDC, KPK

Place object group as a keyboard

Define line styles, colour gradient and fillings for keys

```
#CLD 11
#CFC 11, $FFFFFF, 100
#CLD 12
#CFC 12, $00FF00, 100
#CCR 1, 0, $0000FF, 100, 80, $5555FF, 100, 100, $0000FF, 100
#CFL 15, 1, 90
#CLS 15, $555555, 100, 1
#CFL 16, 1, 270
#CLS 16, $555555, 100, 1
```

Define text styles

```
#CTF 11, <I:Sans.evf>;34, 1, 11
#CTF 12, <I:Sans.evf>;34, 1, 12
#CTF 13, <I:Sans.evf>;18, 1, 11
#CTF 14, <I:Sans.evf>;18, 1 12
```

Define text styles for touch buttons

```
#CBT 1, 11, 12, 0, -5
#CBT 2, 13, 14, 0, -2
```

Define drawing style and change in proportion of the touch button

```
#CBD 1, 15, 16, 100, 50, 10
#CBD 2, 15, 16, 100, 50, 10
#CBO 1, 0, 0, 140
#CBO 2, 0, 0, 100
```

Define keyboard style

```
#KDC 5, 5 , 1, 2
```

Define keyboard fields

```
#KDB 5, 1,"^1234567890ß\8\C"; "\6qwertzuiopü+\D\D"; "\5asdfghjklöä#\N";
"<yxcvbnm,.-"; " "
#KDB 5, 2, '°!"§$%&/()=?\8\C'; "\6QWERTZUIOPÜ*\D\D";
"\5ASDFGHJKLÖÄ'\N"; ">YXCVBNM;:_"; " "
```

Change labelling of the special keys

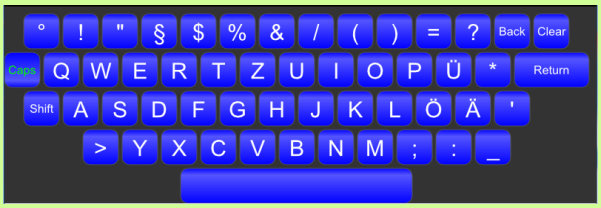
```
#KDL 5, 5 "Shift"; 6 "Caps"; 8 "Back"; 12 "Clear"; 13 "Return"
```

Place keyboard

```
#KPK 5, 0, 0, 7, 800
```

Fill in frames of the object group/keyboard

```
#CLD 42
#CFC 42, $FFFFFF, 20
#OFP 42, 10, 10, 5
```



CHARACTER STRING COMMANDS IN A STRING

#SDP, SEP, SEK, SEA

Date and time

Placing a formatted date

#SDP 1, 1, 50, 50, 17, "%W the %D. %3N %Y, %1h:%m"

Changing the formatted date

#SDC 1, (datetime(11, 37, 0, 21, 4, 2016))

Edit box in combination with a keyboard

Define text style for the keyboard

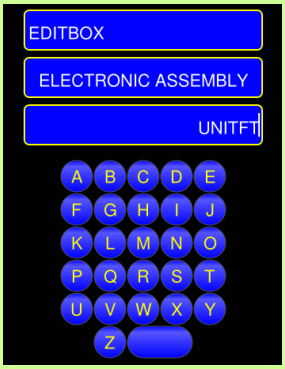
#CLD 1
#CFC 1, $FFFF00,100
#CTF 1, <I:Sans.evf>;30,1 11

Define filling for the keyboard for the pressed-down and not pressed down state

#CCR 1, 0, $0000FF, 100, 80, $5555FF, 100, 100, $0000FF, 100
#CFL 3, 1, 90
#CLS 3, $555555, 100, 1
#CFL 4, 1, 270
#CLS 4, $555555, 100, 1

Define touch button style

#CBT 2, 1, 1, 0, 0
#CBD 2, 3, 4, 100, 50, 25
#CBO 2, 0, 0, 150



Define and place keyboard

#KDC 9, 2, 2
#KDB 9, 1, "ABCDE";"FGHIJ"; "KLMNO"; "PQRST"; "UVWXY"; "Z "
#KPK 9, 400, 10, 8, 0, 250

Define border, filling and text style of the edit box

#CLS 20, $FFFF00, 100, 2
#CFC 20, $0000FF, 100
#CLD 21,
#CFC 21, $FFFFFF, 100
#CTF 21, <I:Sans.evf>;30, 2, 21
#CTF 22, <I:Sans.evf>;30, 1, 21
#CTF 23, <I:Sans.evf>;30, 0, 21

Place edit boxes

#SEP 6, 20, 400, 280, 8, 300, 50, 6, 21, -5, -3
#SEP 7, 20, 400, 340, 8, 300, 50, 6, 22, 0, -3
#SEP 8, 20, 400, 400, 8, 300, 50, 6, 23, 5, -3

Combine edit boxes and keyboard in one group

#OGA 91, 6-9

Allocate edit boxes to the keyboard and enable Edit Box 8

#SEK 9, 6-8
#SEA 8

Allocate touch function for selecting the edit boxes by touch control

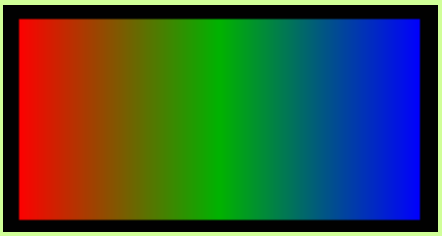
#TID 1, 6-8

STYLE SHEETS

Colour gradient and dash pattern(s)

Colour gradient

Define colour gradient
#CCR 1, 0,$FF0000,100, 50,$00FF00,70, 100,$0000FF,100
Define colour gradient as being linear
#CFL 1, 1,
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,100



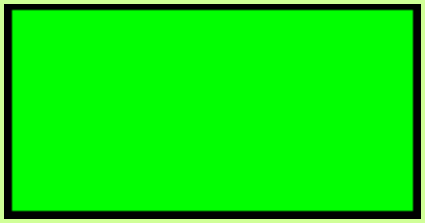
Dash pattern(s)

Define a dash pattern
#CDP 1, 0, 20,10, 5,10
Define line style
#CLS 1, $00FF00,100, 5,0 1
Refrain from defining a filling
#CFD 1
Draw a polyline
#GPL 1,1, 50,50,400,50

Fillings

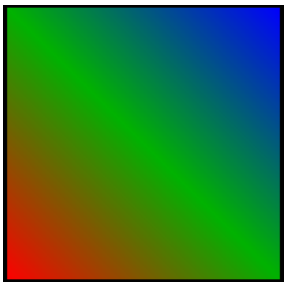
Monochrome filling

Define a background colour, wherein the alpha channel is optional
#CFC 1, $00FF00
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,100



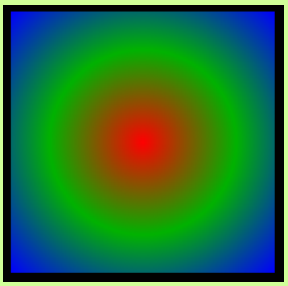
Linear colour gradient filling

Define colour gradient
#CCR 1, 0,$FF0000,100, 50,$00FF00,70, 100,$0000FF,100
Define colour gradient as being linear and inclined at an angle of 45°
#CFL 1, 1, 45
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,200



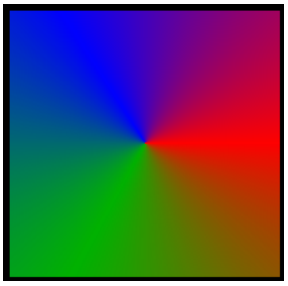
Radial colour gradient filling

Define colour gradient
#CCR 1, 0,$FF0000,100, 50,$00FF00,70, 100,$0000FF,100
Define colour gradient as being radial
#CFR 1, 1, 5000,5
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,200



Conical colour gradient filling

Define colour gradient
#CCR 1, 0,$FF0000,100, 33,$00FF00,70, 66,$0000FF,100, 100, $FF0000, 100
Define colour gradient as being conical
#CFK 1, 1, 5000,5
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,200



Pattern filling

Filling with a recurrent pattern
#CFP 1, "marmor02"
Refrain from defining any line style
#CLD 1
Create square
#GRR 1,1, 50,50,7,200,200

Linen style

Line colour and thickness

Green line with a thickness of 5 pixels

```
#CLS 1, $00FF00,100, 5
```

Line joint style

Refrain from defining any line style
```
#CLD 1
```
Define filling font
```
#CFC 1,$FFFFFF,100
```
Define text style
```
#CTF 1,<I:Sans.evf>,58,0,1,0,1,0
```
Refrain from defining an outline
```
#CFD 4
```
Define green line style as being angular
```
#CLS 4 $00FF00,100 10,0
```
Refrain from defining an outline
```
#CFD 5
```
Define green line style as being round
```
#CLS 5 $00FF00,100 10,1
```
Set integer register
```
#VRI 0, 50,50,20,50
```
Draw a polypath and place string
```
#GPL 1,4, R0,R3, (R0+R1),(R3+R2)rd, R0,(R3+2*R2)
#SSP 2,1, (R0+R1+20),(R3+R2),4, "Miter"
```
Set integer register
```
#VRI 3, (R3+3*R2)
```
Draw a polypath and place string
```
#GPL 3,5, R0,R3, (R0+R1),(R3+R2), R0,(R3+2*R2)
#SSP 4,1, (R0+R1+20),(R3+R2),4, "Round"
```

Text style

Text style

Creating a text style with the internal font “Sans”
```
#CTF 1,<I:Sans.evf>, 60,1,1
```
Place string
```
#SSP 1, 1, 200,50,7,"Hello World"
```

Button style

Touch button as an image

Define two images as a touch button

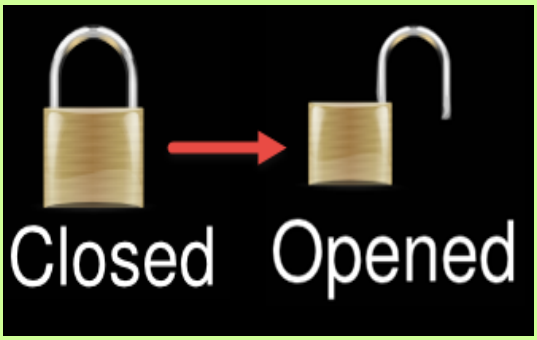
#CBP 1, "Padlock_closed"; "Padlock_opend"

Define text style for touch button

#CBT 1, 1,1,-10,-70

Define images for touch button

#TSP 1, 1, "Closed"; "Opened"; 400,240,7



Change touch button proportion upon pressing it

Change text orientation

#CTA 1,1

Generate colour gradients for touch button

#CCR 5, 0,$00007F,100, 50,$0000FF,100, 100,$00007F,100

#CCR 6, 0,$7F7F00,100, 50,$FFFF00,100, 100,$7F7F00,100

Create line styles for touch button

#CLS 5, $FFFFFF,100,1

#CLS 6,$FFFFFF,100,1

Rotate colour gradients by 90°

#CFL 5,5,90

#CFL 6,6,90

Define touch button style

#CBD 1, 5,6, 200,50,10

Define text style for touch button

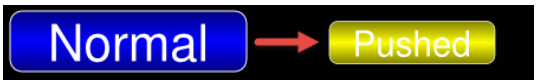
#CBT 1, 1,1

Change proportion in the pressed state

#CBO 1,0,0, 70

Define squarish touch button

#TSR 1, 1, "Normal"; "Pushed"; 400,240,5



TERMINAL COMMANDS

Termianl window

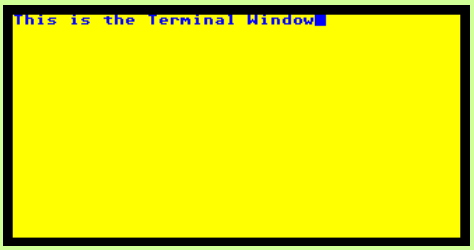
(Re-)define terminal window

Font colour blue, window colour yellow, non-transparent

#YDC $0000FF,100, $FFFF00,100

Small terminal window with 40 characters and 20 lines

#YDW 1, 30,30,7, 40,20



TOUCH CONTROL OBJECTS/TOUCH CONTROL FUNCTIONS

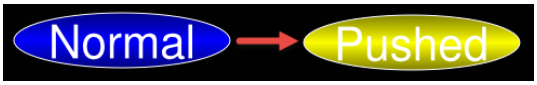
Touch control zones

Rectangular touch button

Change text orientation
#CTA 1,1
Define touch button colour gradients for normal and pressed down states
#CCR 5, 0,$00007F,100, 50,$0000FF,100, 100,$00007F,100
#CCR 6, 0,$7F7F00,100, 50,$FFFF00,100, 100,$7F7F00,100
Define touch button line style for normal and pressed down states
#CLS 5, $FFFFFF,100,1
#CLS 6, $FFFFFF,100,1
Define touch button colour gradients as being linear with 90° torsion
#CFL 5,5,90
#CFL 6,6,90
Define touch button style
#CBD 1, 5,6, 200,50,10
Define touch button text style
#CBT 1, 1,1
Define touch button change in proportion for pressed down state
#CBO 1,0,0,0
Set elliptical touch button to position 400,240.
#TBR 1, 1, "Normal"; "Pushed"; 400,240

Elliptical touch button

Change text orientation
#CTA 1,1
Define touch button colour gradients for normal and pressed down states
#CCR 5, 0,$00007F,100, 50,$0000FF,100, 100,$00007F,100
#CCR 6, 0,$7F7F00,100, 50,$FFFF00,100, 100,$7F7F00,100
Define touch button line style for normal and pressed down states
#CLS 5, $FFFFFF,100,1
#CLS 6, $FFFFFF,100,1
Define touch button colour gradients as being linear with 90° torsion
#CFL 5,5,90
#CFL 6,6,90
Define touch button style
#CBD 1, 5,6, 200,50,10
Define touch button text style
#CBT 1, 1,1
Define touch button change in proportion for pressed down state
#CBO 1,0,0,0
Set elliptical touch button to position 400,240.
#TSE 1, 1, "Normal"; "Pushed"; 400,240



TIME

#WGC

Object group as a clock

```
        Refrain from defining a line style and white filling for hour hand
#CLD    11
#CFC    11, $FFFFFF, 100
        Refrain from defining a line style and grey filling for minute hand
#CLD    12
#CFC    12, $E0E0E0, 100
        Refrain from defining a line style and red filling for second hand
#CLD    13
#CFC    13, $FF0000, 100
        Create colour gradient
#CCR    9, 0,$0,100 100, $A0A0A0, 100
        Define colour gradient and no line style for connecting point
#CFR    14, 9
#CLD    14
        Set integer register
#VRI    1,(scrW()/2),(scrH()/2)
        Place graphic of the dial
#PPP    5, "Clock3"; R1, R2, 5, 200
        Place the three hands and connecting point
#GPF    6, 11, 0, 0, 4, 75, 8, 0
#GPF    7, 12, 0, 0, 4, 95, 8, 0
#GRR    8, 13, 0, 0, 7, 2, 95
#GET    9, 14, R1, R2, 5, 7
        Define free anchors for hands
#OAO    4, 15, 6-7
#OAO    1, 20, 8
        Change the position of the hands absolutely
#OPA    R1, R2, 6-8
        Create object group
#OGA    10, 5-9
        Define object group as a clock
#WGC    10, 6, 7,8
```

DRAWING/GRAPHIC PRIMITIVES

Graphic drawing objects

Rounded rectangle with border

Define line style

#CLS 1, $00FF00,100, 5

Define filling

#CFC 1, $FF0000,100

Rounded rectangle with border, a type of picture frame

#GRR 1,1, 400,240,5, 200,100,20, 15

Polyline - Santa Claus house

Define line style

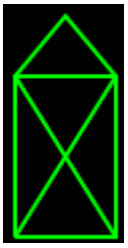
#CLS 1, $00FF00,100, 2

Refrain from defining a filling

#CFD 1

Draw a Santa Claus house with the aid of a polypath

#GPL 1,1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,130, 20,100, 70,20



Filled-in polyline - trapezium

Define line style

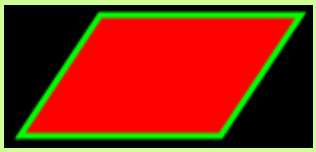
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

Draw a filled-in trapezium with the aid of a polypath

#GPF 1,1, 20,20, 60,80, 160,80, 120,20



Extend a polyline - Santa Claus house

Define line style

#CLS 1, $00FF00,100, 2

Refrain from defining a filling

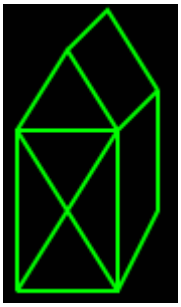
#CFD 1

Draw a Santa Claus house with polypath from lines

#GPL 1,1, 20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,140, 20,100, 70,20

Extend the Santa Claus house by the 3rd dimension

#GPA 1, 1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,130, 20,100, 70,20



Polypath - Draw a car

Define line style

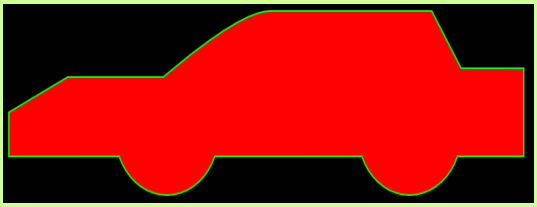
#CLS 1, $00FF00,100, 2

Define filling

#CFC 1, $FF0000,100

Draw a car using path segments

#GPP 1,1, 50,200, ?v-50, ?h150, ?c0,70,130,0, ?h200, ?c0,70,130,0, ?h90, ?v100, ?h-85, ?l-40,65, ?h-220, ?q-40,0,-145,-75, ?h-130, ?z



Graphic drawing objects

Hexagon

Define line style

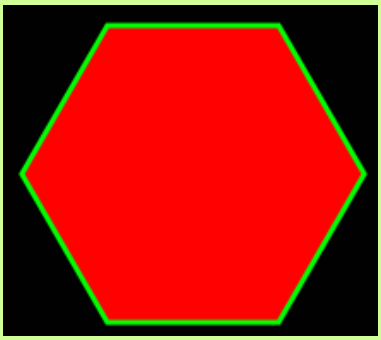
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

A hexagon rotated by 90°

#GGP 1, 1 400,240,5, 100,6,0, 90



Five-point star

Define line style

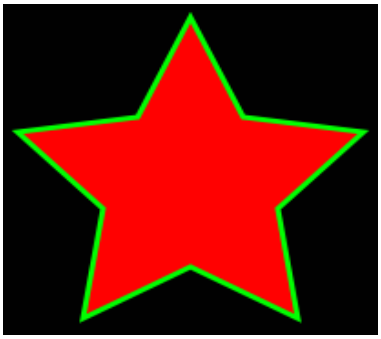
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

Draw a filled-in star

#GGS 1, 1 400,240,5, 100,50, 5



Tyres

Define line style

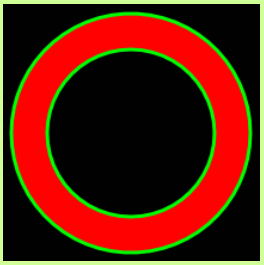
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

Draw a circle with a hole

#GET 1, 1 400,240,5, 100,100, 30



Arc of a circle

Define line style

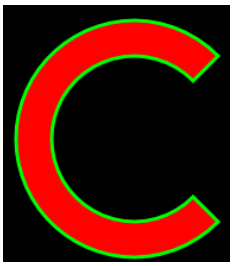
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

Draw a letter “C” as an arc of a circle

#GEA 1, 1 400,240,5, 100,100, 45,-45, 30



Graphic drawing objects

Cut segments of a circle

Define line style

#CLS 1, $00FF00, 100, 3

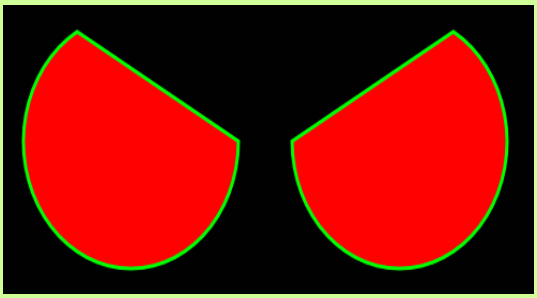
Define filling

#CFC 1, $FF0000, 100

Draw two cut segments of a circle

#GES 1, 1, 400, 200, 5, 100, 100, 90, 0

#GES 2, 1, 450, 200, 5, 100, 100, 180, 60



Piece of cake

Define line style

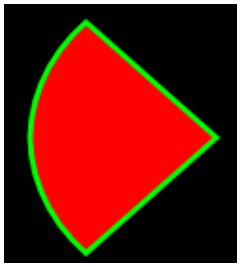
#CLS 1, $00FF00,100, 3

Define filling

#CFC 1, $FF0000,100

Draw a piece of cake

#GEP 1, 1 400,240,5, 100,100, 135,225



SEGEMENT TYPES

Segments H(orizontal), V(ertical), L(inear)

Horizontal polypath

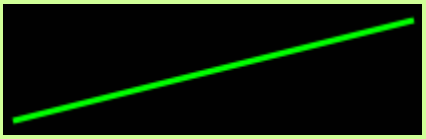
Define line style
#CLS 1, $00FF00,100, 3
Refrain from defining an outline
#CFD 1
Generate a horizontal line as a polypath
#GPP 1, 1, 10, 50, ?H 300

Vertical polypath

Define line style
#CLS 1, $00FF00,100, 3
Refrain from defining a filling
#CFD 1
Generate a vertical line as a polypath
#GPP 1, 1, 50, 50, ?V 150

Line polypath

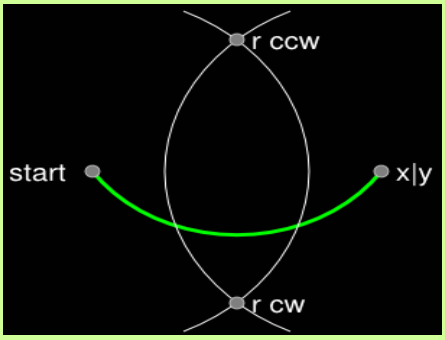
Define line style
#CLS 1, $00FF00,100, 3
Refrain from defining an outline
#CFD 1
Generate a line as a polypath
#GPP 1, 1, 50,50, ?L 250,100



Segmente C(ircle), E(llipse)

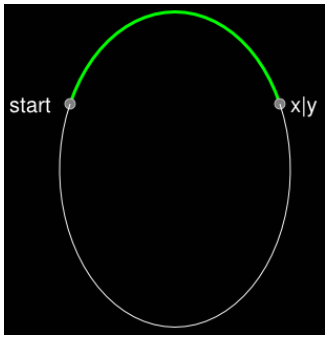
Circle segment polypath

```
        Define green line style
#CLS    5, $00FF00,100, 3
        Refrain from defining a filling
#CFD    5
        Define grey line style
#CLS    6, $BFBFBF,100, 1
        Define a grey filling
#CFC    6, $7F7F7F,100
        Draw a polypath as a circle segment
#GPP    1, 5, 200,200,?C0,120,400,200
        Create dots/circles
#GET    2, 6, 200,200,5, 5
#GET    3, 6, 400,200,5, 5
#GET    6, 6, 300,311,5, 5
#GET    7, 6, 300,89,5, 5
        Draw a grey arc
#GEA    4, 2, 200,200,0, 150,150 -65, 65
#GEA    5, 2, 400,200,0, 150,150, 115, -115
        Change text size
#CTS    1, 28
        Place strings
#SSP    8, 1, 180,200,6, "start"
#SSP    9, 1, 310,311,4, "r ccw"
#SSP    10, 1, 310, 89,4, "r cw"
#SSP    11, 1, 410,200,4, "x\|y"
```



Ellipse polypath

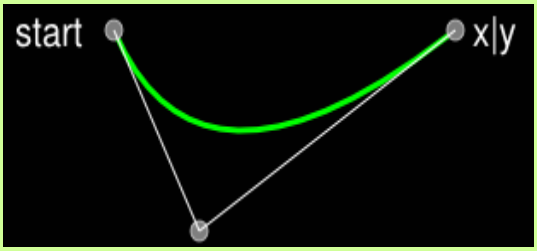
```
        Define green line style
#CLS    5, $00FF00,100, 3
        Refrain from defining a filling
#CFD    5
        Define grey line style
#CLS    6, $BFBFBF,100, 1
        Define a grey filling
#CFC    6, $7F7F7F,100
        Set integer register
#VRI    0,110,150
        Polypath ellipse green and grey
#GPP    1, 5, 200,300,?E1,R0,R1,0,400,300
#GPP    4, 2, 200,300,?E2,R0,R1,0,400,300
        Set dots/circles
#GET    2, 6, 200,300,5, 5
#GET    3, 6, 400,300,5, 5
        Change text size
#CTS    1, 28
        Set strings
#SSP    8, 1, 180,300,6, "start"
#SSP    11, 1, 410,300,4, "x\|y"
```



Segmente Q(uadratische Bézier Kurve), R

Polypfad quadratische Bézierkurve

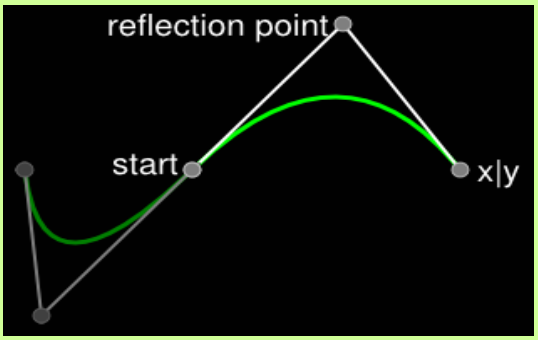
```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Keine Umrandung definieren
#CFD    5
        Grauen Linienstyle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Integer Register setzen
#VRI    0,250,200
        Quadratische Bezierkurve zeichen
#GPP    1, 5, 200,300,?QR0,R1,400,300
        Punkte/Kreise setzen
#GET    2, 6, 200,300,5, 5
#GET    3, 6, 400,300,5, 5
#GET    4, 6, R0,R1,5, 5
        Geraden zeichnen
#GPL    5, 2, 200,300, R0,R1
#GPL    6, 2, 400,300, R0,R1
        Textgröße ändern
#CTS    1, 28
        Strings setzen
#SSP    8, 1, 180,300,6, "start"
#SSP    11, 1, 410,300,4, "x\|y"
```



Segmente Q(uadratische Bézier Kurve), R

Polypfad quadratische Bézierkurve mit Hilfspunkt

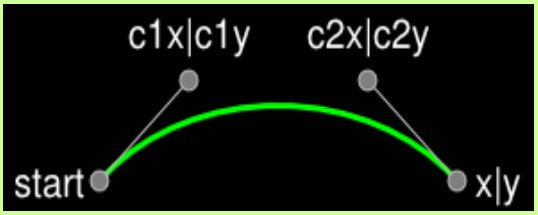
```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Keine Füllung definieren
#CFD    5
        Grauen Linienstyle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Integer Register setzen
#VRI    0,210,200,300,300,460
        Grünen Polypfad zeichnen
#GPP    1, 5, 200,R3,?QR0,R1,R2,R3, ?RR4,R3
        Grauen Polypfad zeichnen
#GPL    2, 2,200,R3, R0,R1, R2,R3, (2*R2-R0),(2*R3-R1), R4,R3
        Punkte/Kreise erzeugen
#GET    3, 6, 200,300,5, 5
#GET    4, 6, R0,R1,5, 5
#GET    6, 6, R2,R3,5, 5
#GET    7, 6, R2,R3,5, 5
#GET    8, 6, R4,300,5, 5
#GET    9, 6, (2*R2-R0),(2*R3-R1),5, 5
        Schwarze Füllung mit 50% Deckkraft definieren
#CFC    4, $000000, 50
        Rechteck erzeugen zum Abdunkeln
#GRR    5, 4, 195,(R1-5),7, (R2-R1+10), (R3-R1+10)
        Textgröße ändern
#CTS    1, 28
        Strings platzieren
#SSP    10, 1, (R2-10),(R3+5),6, "start"
#SSP    11, 1, (2*R2-R0-10),(2*R3-R1),6, "reflection point"
#SSP    12, 1, (R4+10),R3,4, "x\|y"
```



Segmente T, S(pline)

Polypfad Spline

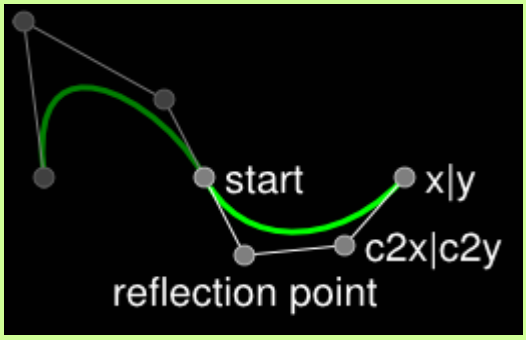
```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Keine Füllung definieren
#CFD    5
        Grauen Linienstyle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Spline zeichnen
#GPP    1, 5, 100,50,?S150,100,250,100,300,50
        Geraden Zeichnen
#GPL    2, 6, 100,50,150,100
#GPL    3, 6, 300,50,250,100
        Punkte/Kreise erzeugen
#GET    4, 6, 150,100,5, 5
#GET    5, 6, 250,100,5, 5
#GET    6, 6, 100, 50,5, 5
#GET    7, 6, 300, 50,5, 5
        Textgröße ändern
#CTS    1, 28
        Strings platzieren
#SSP    8, 1, 90, 50,6, "start"
#SSP    9, 1, 150,110,8, "c1x\|c1y"
#SSP    10, 1, 250,110,8, "c2x\|c2y"
#SSP    11, 1, 310, 50,4,"x\|y"
```



Segmente T, S(pline)

Polypfad geglättete kubische Bézierkurve

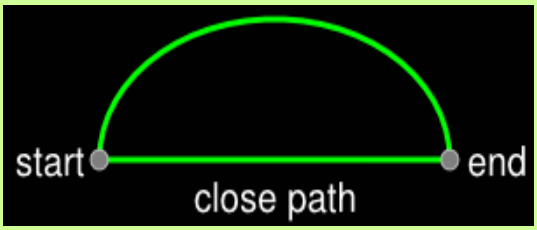
```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Keine Füllung definieren
#CFD    5
        Grauen Linienstyle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Integer Register setzen
#VRI    0, 200,200, -10,80, 60,40, 80,0, 70,-35, 100,0
        Grünen Polypfad erzeugen
#GPP    1, 5, R0,R1,?sR2,R3,R4,R5,R6,R7, ?tR8,R9,R10,R11
        Polypfad aus Geraden erzeugen
#GPL    9, 2, R0,R1, (R0+R2),(R1+R3), (R0+R4),(R1+R5), (R0+R6),
        (R1+R7), (R0+2*R6-R4),(R1-R5), (R0+R6+R8),(R1+R7+R9),
        (R0+R6+R10),(R1+R7+R11)
        Punkte/Kreise erzeugen
#GET    2, 6, R0,R1,5, 5
#GET    3, 6, (R0+R2),(R1+R3),5, 5
#GET    4, 6, (R0+R4),(R1+R5),5, 5
#GET    5, 6, (R0+R6),(R1+R7),5, 5
#GET    6, 6, (R0+2*R6-R4),(R1-R5),5, 5
#GET    7, 6, (R0+R6+R8),(R1+R7+R9),5, 5
#GET    8, 6, (R0+R6+R10),(R1+R7+R11),5, 5
        Schwarze Füllung mit 50% Deckkraft definieren
#CFC    4, $000000, 50
        Rechteck erzeugen zum Abdunkeln
#GRR    20, 4, (R0-5+R2),(R1-5),7, (R6-R2+10),(R1+R3+5)
        Textgröße ändern
#CTS    1, 28
        Strings platzieren
#SSP    10, 1, (R0+R6+10),(R1+R7),4, "start"
#SSP    11, 1, (R0+2*R6-R4),(R1-R5-5),2, "reflection point"
#SSP    12, 1, (R0+R6+R8+10),(R1+R7+R9),4, "c2x\|c2y"
#SSP    13, 1, (R0+R6+R10+10),(R1+R7+R11),4, "x\|y"
```



Segmente Z, M

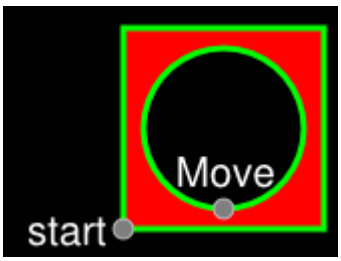
Polypfad schließen

```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Keine Füllung definieren
#CFD    5
        Grauen Linensytle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Integer Register setzen
#VRI    0, 200,200, 100,70,200,0
        Polypfad mit Pfadstop deinieren
#GPP    1, 5, R0,R1,?e1,R2,R3,0,R4,R5, ?Z
        Punkte/Kreise zeichnen
#GET    2, 6, R0,R1,5, 5
#GET    3, 6, (R0+R4),(R0+R5),5, 5
        Textgröße ändern
#CTS    1, 28
        Strings platzieren
#SSP    10, 1, (R0-10),(R1),6, "start"
#SSP    11, 1, (R0+R4+10),(R0+R5),4, "end"
#SSP    12, 1, (R0+R4/2),(R1-5),2, "close path"
```



Polypfad Sprung

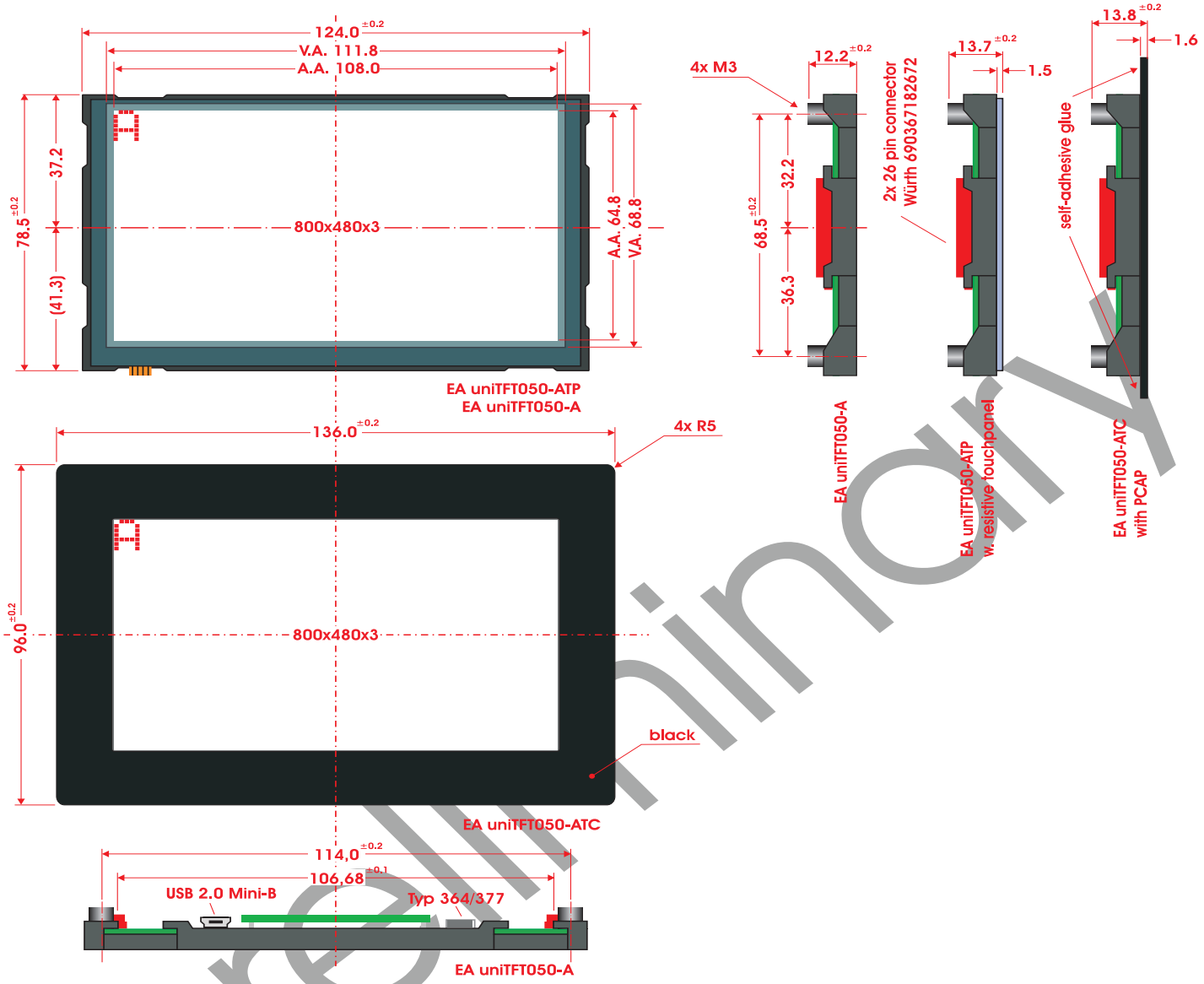
```
        Grünen Linienstyle definieren
#CLS    5, $00FF00,100, 3
        Rote Füllung definieren
#CFD    5, $FF0000,100
        Grauen Linienstyle definieren
#CLS    6, $BFBFBF,100, 1
        Graue Füllung definieren
#CFC    6, $7F7F7F,100
        Integer Register setzen
#VRI    0, 200,200, 100,100, 10
        Quadrat mit Kreis in der Mitte zeichnen
#GPP    1, 5, R0,R1,?hR2, ?vR3, ?h(-R2), ?Z, ?M(R0+R2/2),(R1+R4), ?c1,
        ((R3-2*R4)/2), 0,(R3-2*R4), ?c1, ((R3-2*R4)/2), 0,(2*R4-R3)
        Punkte/Kreise setzen
#GET    2, 6, R0,R1,5, 5
#GET    3, 6, (R0+R2/2),(R1+R4),5, 5
        Textgröße ändern
#CTS    1, 28
        Strings setzen
#SSP    4, 1, (R0-10),(R1),6, "start"
#SSP    5, 1, (R0+R2/2),(R1+R4+5),8, "Move"
```



ELECTRICAL SPECIFICATION

Value	Condition	min.	typ.	max.	Unit
Operating temperature		-20		+70	°C
Storage temperature		-30		+80	°C
Storage humidity				90% RH	@ 60°C
Operating voltage		3.1	3.3	3.5	V
Supply current	Backlight 100%		750		mA
	Backlight 0%		350		mA
Input low voltage (except USB, I/O)		-0.3	0	0.3*VDD	V
Input high voltage (except USB, I/O)		VDD*0.7		VDD+0.3	V
Output low voltage (except USB, I/O)		-	-	0.4	V
Output high voltage (except USB, I/O)		VDD-0.5	-	-	V
Input low voltage I/O		-	-	0.2*VDD	V
Input high voltage I/O		0.8*VDD	-	-	V
Output low voltage I/O		-	-	0.6*VDD	V
Output high voltage I/O		VDD-0.7	-	-	V
Output current I/O		-	-	25	mA
		-	-	125	mA (total)
I²C-bus pull-up				1	MΩ
Brightness	w./o. Touchpanel		700		cd/m²
	with Capactive Touchpanel				
	with Resistive Touchpanel				

DIMESIONS



ELECTRONIC ASSEMBLY GmbH
Zeppelinstraße 19
D-82205
Gilching Germany

Fon: +49 (0)8105-7780 90
Fax: +49 (0)8105-7780 99
e-Mail: info@lcd-module.de
Web: www.lcd-module.de

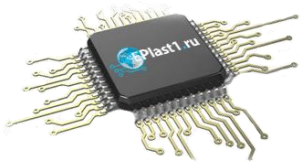
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.