Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

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).

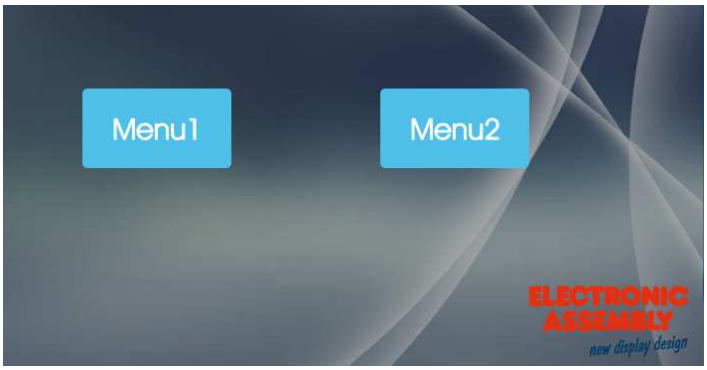
BUTTON STYLE

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.

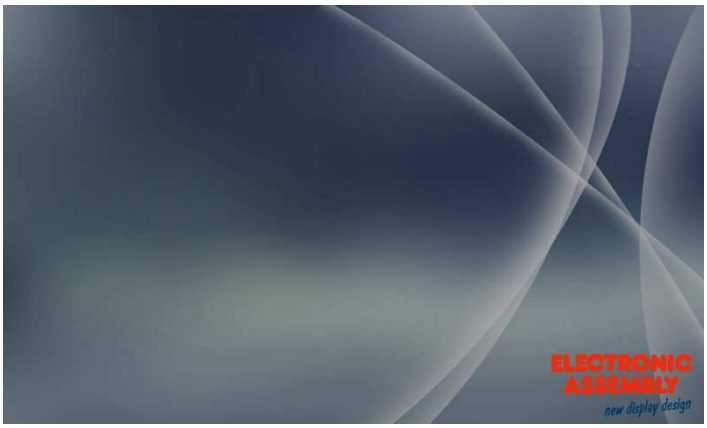
STYLES AND OBJECTS

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)

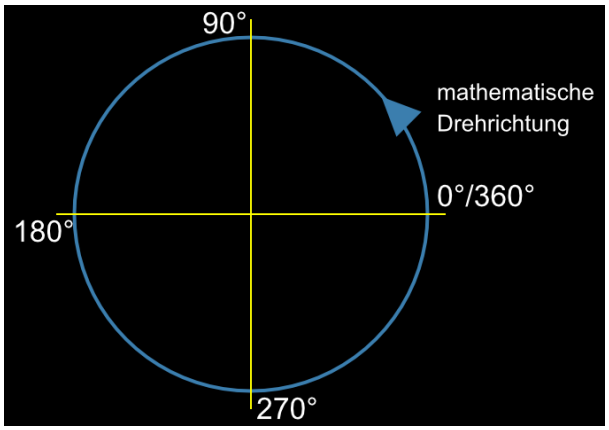
There are commands which relate to files. Those are likewise written in quotation marks or inverted commas. These commands have recorded in them, as their reference point, the default folder, and thus an automated path specification.

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'.

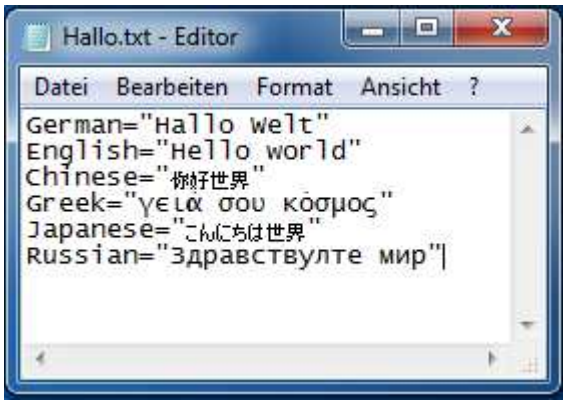
- Beispiel: *"string1" ; 'string2'*

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

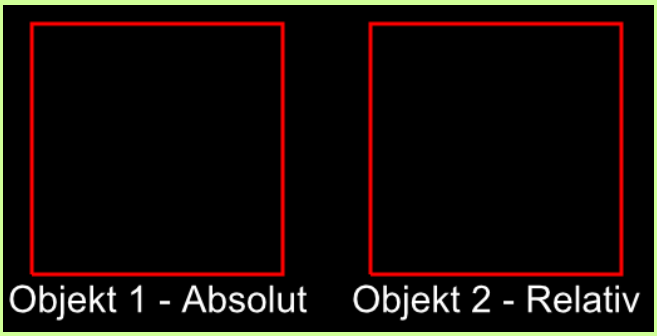
With the polyline (#GPL) and polygon (#GPF) commands, virtually all desired forms can be generated. Each section is designated a segment.

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

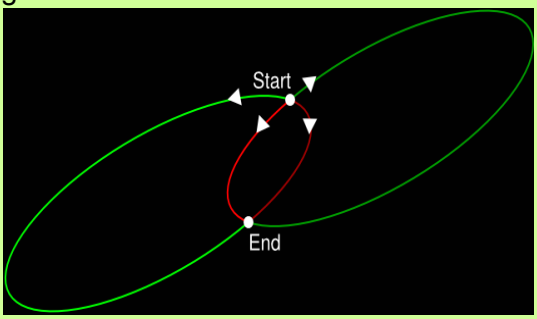
Each segment contains an indicator in the form of a character whereby the type of segment is recognised. The list of indicators can be found here in the command overview.
A segment entered always begins with "?", and the indicator, as well as the corresponding parameters, are added subsequently. In that 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.

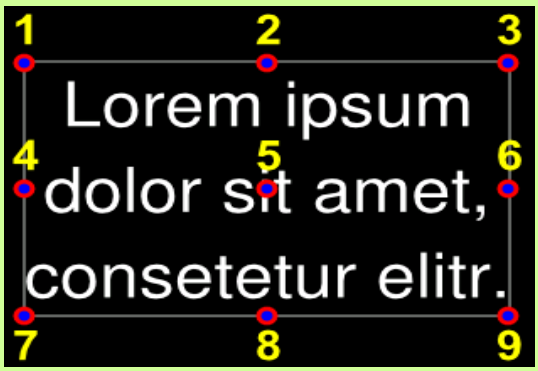
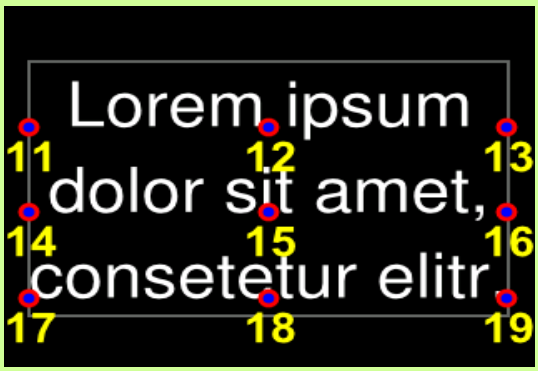
DIRECTIONS OF MOVEMENT AND ARCS

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.

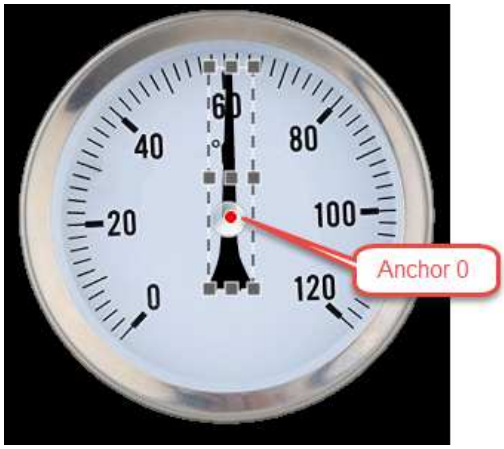
Highlighting the arcs and directions of movement	
#GPP1, 1, 390, 380, ?E0, 200, 70, 25, 340, 260 #GPP2, 2, 390, 380, ?E1, 200, 70, 25, 340, 260 #GPP3, 3, 390, 380, ?E2, 200, 70, 25, 340, 260 #GPP4, 4, 390, 380, ?E3, 200, 70, 25, 340 ,260	0: Small anti-clockwise arc 1: Small clockwise arc 2: Large anti-clockwise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