

EA uniTFT

INKL. KONTROLLER SSD1803A FÜR 4-/8-BIT, SPI, I²C

WELTNEUHEIT!



Abmessung:
124x78,5x12,2mm

DIE HMI-EINHEIT

- mit und ohne Touchpanel
- Analog-Resistiver Touchoder kapazitiver Multitouch
- Objektorientierter Bildschirmaufbau
- Objekteveränderung: Größe, Form, Farbe, Sichtbarkeit, Inhalt
- Vektorgrafiken, verlustfreies drehen und zoomen
- Alphablending, bewegte Objekte
- Zeichensätze, vektorisiert und Unicode
- Single supply 3,3 V
- 7 Kommunikationsinterfaces: USB, 2 x I²C, 2 x SPI, 2 x RS232
- 16 digitale frei definierbare I/O eingebaut, auf 128 erweiterbar
- 4 Analoge Eingänge
- Uhrzeitfunktion inklusive Batterie
- SD-Card als Datenspeicher für Bilder, Fonts, Menüs und Log-Daten

BESTELLBEZEICHNUNGEN

DISPLAYS

TFT 800x480 Pixel, weiße LED-Beleuchtung

Resistiver Touch, TFT 800x480 Pixel, weiße LED-Beleuchtung

Kapazitiver Multitouch, TFT 800x480 Pixel, weiße LED-Beleuchtung

ZUBEHÖR

Quick-Start resistiv: EA uniTFT050-TP und Programmierboard EA 93998 (siehe unten)

Quick-Start kapazitiv: EA uniTFT050-TC und Programmierboard EA 93998 (siehe unten)

Programmierboard: Lautsprecher, Resettaste Programmerinterfaces

MicoMatch THT, 26 poliger Gegenstecker zum einlöten

MicroMatch SMD, 26 poliger Gegenstecker zum einlöten

MicroMatch Flachband, 26 poliger Gegenstecker mit Quetschverbindung

EA uniTFT050-A

EA uniTFT050-TP

EA uniTFT050-TC

EA QUICKuni050-TP

EA QUICKuni050-TC

EA 93998-A

EA ????????

EA ????????

EA ????????

INHALTSVERZEICHNIS

Bestellbezeichnungen	1
Inhaltsverzeichnis	2
Revision	5
Allgemeines	6
Arbeitsweise des Displays	6
Objekte	6
Styles	6
Drawstyle und Linestyle	6
Textstyle	6
Buttonstyle	7
Styles und Objekte	7
Befehlssyntax	8
Befehlsparameter	8
Winkel	8
Kommentare	9
Objektbereich	9
Eingabe von Zahlen	10
Eingabe von Strings	10
Stringfile	10
Pfadangabe und Formatierung	12
Pfadangabe	12
Formatierte Strings	13
Datum und Uhrzeit	14
Bilder	15
Bildformate	15
Polylinie und Polygone	16
Segment	16
Indikatoren	16
Laufrichtungen und Kreisbögen	17
Anker	18
Gruppen	19
Kalkulation	20
Aktion und Animation	24
Aktion	24
Animation	24
Hardware	25
Pinbelegung	26
RS232	27
Datenformat	27
Baudraten	27
Applikationsbeispiele	28
RS232 V24 - Verbindung zu einem PC	28
RS485 - Bussystem	28
SPI	29
I ² C	31
USB	32
SD CARD	33
Video Eingang / Kamera	33
Analog Input	34
Pulsweitenmodulation (PWM)	35
Input/Output(I/O)	36
Pulsweitenmodulation (PWM)	36

Short Protokoll & Small Protokoll	37
Short Protokoll Befehle	38
Small Protokoll Befehle	40
Prüfsummenberechnung	41
Short Protokoll	41
Small Protokoll	42
Befehlsübersicht	43
Terminal	43
Bilder / Vektorgrafiken	45
Bildgröße	45
Bild/ Vektorgrafik einfügen	45
Animationstypen	46
Stylesheets	47
Farbverläufe und Linienmuster	47
Drawstyle	47
Linienstyle	49
Textstyle	50
Touchbuttonstyle	50
Zeichnen / Grafische Primitive	52
Segmenttypen	55
Strings und Zeichenkettenbefehle	56
Editbox	57
Touchobjekte / Touchfunktionen	59
Definition von Touchobjekten	59
Touchfunktionen	60
Bargraphen und Instrumente	62
Objektverwaltung	65
Definition von Objekten	65
Ändern von Objekten	66
Variablen / Register	71
Makros	73
Ausführung von Makros	73
Definition von Makroprozessen und Touchmakros	74
Typen für Analogmakros	76
Uhrzeit	77
Aktion	79
AktionsKurven und -Pfade	79
Aktion Definieren	79
Aktion	79
AktionsDefinitionen	81
Aktionstyp	82
Vordefinierte AktionsKurven	83
Keyboard	88
Peripherie - Schnittstellen	90
Analog	90
Pulsweitenmodulation (PWM)	91
Port	91
Video und Audio	92
RS232 Master Schnittstelle	92
Spi Master Schnittstelle	93
I ² c	95
Systemkommandos	96
Bootmenü und Touchabgleich durch Gesten	99

Filezugriff-1234567890	100
SD-Karte	100
Filetime	102
Rückmeldungen	103
Befehlsbeispiele	106
Aktion und Animation	107
Bild / Vektorgrafiken	110
Ein- und Ausgänge	111
Instrumente	112
Keyboard	114
String Zeichenkettenbefehle	115
Editbox	116
Style sheets	117
Terminal Befehle	121
Touchobjekte / Touchfunktionen	122
Uhrzeit	123
Zeichnen/ Grafische Primitive	124
Segmenttypen	127
Befehle zur Erzeugung der Beispielbilder	130
Aktion und Animation	130
Instrumente	135
Keyboard	139
String Zeichenkettenbefehle	140
Style sheets	141
Terminal Befehle	145
Touchobjekte / Touchfunktionen	146
Uhrzeit	147
Zeichnen / Grafische Primitive	148
Segmenttypen	151
Elektrische Spezifikation	158
Maßzeichnung	159

REVISION

Datum	Version / Firmware	Beschreibung
23.08.2016	0.9	First release

ALLGEMEINES

Die EA uniTFT-Serie ermöglicht mit den integrierten Befehlssatz eine ausgefeilte grafische Darstellung und intuitive Menüsteuerung. Dank dem [integrierten Befehlssatz](#) und der Windowsdesignsoftware EA uniSketch können nicht nur Elektronikspezialisten sondern z.B. auch Experten aus dem Bereich Design und Benutzerführung eine komplette HMI erstellen.

Die Displaymodule sind mit 3,3 V sofort betriebsbereit, die Ansteuerung erfolgt über die eingebauten [seriellen Schnittstellen](#).

Durch die objektorientierte "Programmierung", den weiten Befehlssatz und den bereits integrierten, aber erweiterbaren Unicode-Schriftsätzen wird "Time-to-Marked" ein Kinderspiel.

ARBEITSWEISE DES DISPLAYS

Die Darstellung auf dem Display erfolgt anhand den vom Benutzer übergebenen Befehlen. Jeder Gegenstand auf dem Display ist ein eigenes [Objekt](#) und kann manipuliert werden. Das Aussehen, also die Farbinformationen, Schrifttypen, Liniensärken usw. sind in [Styles](#) zusammengefasst.

OBJEKTE

Zur Erzeugung von verschiedenen Objekten gibt es die entsprechenden Befehle. Jedes Bild, jeder Text und jeder Button ist ein sogenanntes Objekt. Jedes Objekt muss mit einer Objekt-ID versehen werden, welche es eindeutig identifizierbar macht. Vergibt man eine Objekt-ID, dann wird bei erneuter Vergabe dieser Objekt-ID das vorherige Objekt überschrieben.

Somit ist bei der Verwendung von Objekten, die auf mehreren Bildschirmseiten vorkommen, Vorsicht geboten, sodass diese nicht überschrieben werden.

STYLES

Es gibt verschiedene Styles auf deren Basis Objekte dargestellt werden, wie z.B. Farbe, Linienstärke oder Transparenz.

Die entsprechenden Befehle und Beispiele können im Abschnitt [Stylesheets](#) eingesehen werden.

Genau wie bei den Objekten werden Styles in entsprechenden IDs gespeichert, welche auch überschrieben werden können. Die verschiedenen Stylegruppen haben ihre eigenen ID-Bereich, So kann ein Buttonstyle und ein Drawstyle mit der ID 1 zur gleichen Zeit nebeneinander existieren.

Die maximale Anzahl an Styles ist für jeden Bereich 255, weshalb bei der Verwendung von sehr vielen Styles die Überlegung nach einer lokalen Definition in Betracht gezogen werden muss.

DRAWSTYLE UND LINSTYLE

Außer bei Bildern wird jedes Objekt mit einer Umrandung und einer Füllung erzeugt. Der Drawstyle bestimmt eben jenes. Dabei können sowohl die [Füllung](#) als auch die [Umrandung](#) (Linie) definiert, komplett weggelassen oder transparent gestaltet werden. Neben einer einfachen Farbfüllung ist des Weiteren auch ein Farbverlauf möglich.

Bei der Linie ist es nicht möglich einen Farbverlauf zu definieren. Außerdem können sowohl ein Linienmuster (gestrichelt) definiert als auch die Enden abgerundet werden. Der Linestyle ist ein Bestandteil des Drawstyles, weshalb es sich für die Übersicht empfiehlt, beides stets gemeinsam anzugeben.

TEXTSTYLE

Der [Textstyle](#) muss definiert werden, wenn mit Strings gearbeitet werden soll. Diese beinhalten die Informationen über die verwendete Schriftart sowie deren Formatierung. Da ein String ebenfalls ein Objekt ist, wird

auch im Textstyle auf einen bestehenden Drawstyle referenziert, um die Schrift mit Füllung und Umrandung zu versehen. Aus Performancegründen empfehlen wir einen Drawstyle ohne Umrandung (Linie).

BUTTONSTYLE

Dem [Buttonstyle](#) liegen Text- und Drawstyles zur Grundlage. Um Touch-Buttons zu gestalten, welche bei gedrücktem Status eine andere Optik vorweisen als im gedrückten Zustand, werden unter Umständen mehrere Draw- und Textstyles benötigt.

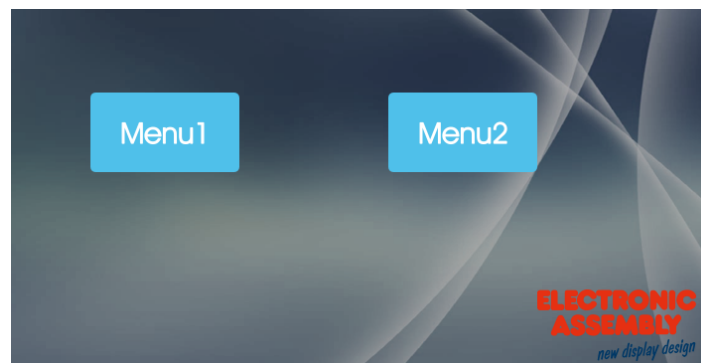
Ebenso werden Informationen wie Touchfeedback z.B. kurze Jingles bei Betätigen oder Tastenvergrößerung in diesem Style hinterlegt.

STYLES UND OBJEKTE

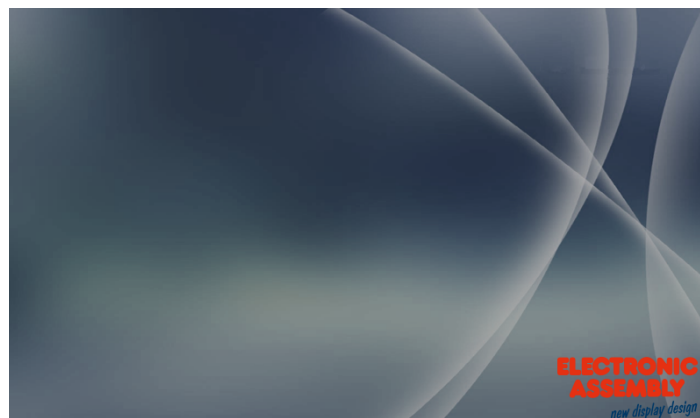
Um zwischen verschiedenen Bildschirmseiten wechseln zu können müssen immer erst alle vorhandenen Objekte [gelöscht](#) werden. Gerade in Menüs ist oft eine feste Struktur vorhanden, so müssen nur gezielt einzelne Objekte gelöscht werden andere bleiben vorhanden.

Nachfolgend ein Beispiel um die Arbeit mit Objekten zu verdeutlichen:

Nach dem Starten des Displays öffnet sich ein Hauptmenü mit einem Hintergrund(ID=100), Logo(ID=101) und zwei Touch Buttons(ID= 1,2) für die Wahl des Untermenüs. Des Weiteren ist ein Buttonstyle(Nr=1) für die Touch Buttons definiert.



Hintergrund, Logo und Buttonstyle sollen auch in den Untermenüs vorhanden sein. Um diese Objekte nicht zu überschreiben sollten die Objekt-ID's nicht weiter verwendet werden. Im Menü1 soll eine Überschrift mit entsprechendem Style sowie eine Linie erzeugt werden. Vorher werden noch alle Objekte gelöscht, die nicht mehr benötigt werden. In diesem Fall sind das die Objekte 1 bis 99. Das Löschen geschieht durch die Eingabe des Löschkommands für Objekte (→ [#ODI](#) 1-99). Das Ergebnis ist im folgendem Bild zu sehen.



Natürlich können die Objekte in Menü 1 und in Menü 2 gleichermaßen verwendet werden.

BEFEHLSSYNTAX

Ein Befehl beginnt immer mit '#'. Anschließend einer 3-stelligen Ziffernfolge - dem Befehlscode. Je nach Befehlscode werden weitere Parameter benötigt. Zur Trennung der Parameter muss einer der folgenden Zeichen verwendet werden:

- Leerzeichen
- Komma (,)
- Punkt (.)
- Semikolon (;) (nur und zwingend bei Stringende)
- Bereichsangabe mehrerer Objekte-ID's: '-' Zeichen: z.B. 1-5, statt 1,2,3,4,5.

Der Befehl muss immer durch einen LF (Line Feed) abgeschlossen werden. Ist der LF nicht vorhanden, wird der Befehl nicht ausgeführt.

Beispiel:

- Einzelner Befehl zur Erzeugung des Line Styles 1 `#CLS1 $3B7EAE,100,1,0,1(Line Feed)`
- Mehrere Befehle zur Erzeugung einer Animation von Objekt 1 `#AOA1 501,0 (Line Feed)`
`#AOT1 1,250,100,100 (Line Feed)`

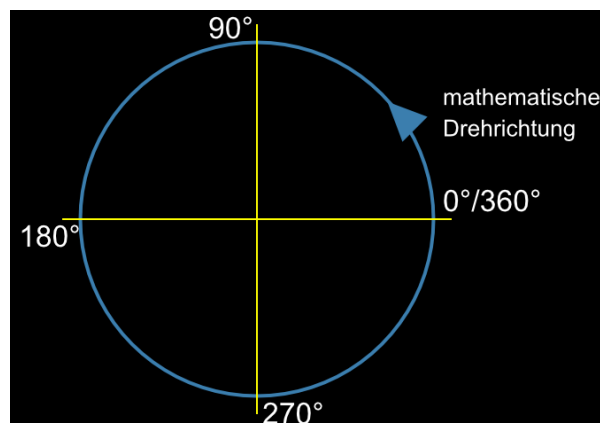
Es gibt Befehle, die sich auf Dateien beziehen. Jene werden ebenfalls in Hochkommata oder Anführungszeichen geschrieben. Diese Befehle haben als Bezugspunkt die Defaultordner und somit eine automatisierte [Pfadangabe](#) in sich hinterlegt.

BEFEHLSPARAMETER

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen übergeben werden.

WINKEL

Winkel werden im mathematischen Drehsinn angegeben, d.h. gegen den Uhrzeigersinn. Die Eingabe von negativen Winkeln ist ebenfalls möglich. Im folgendem Bild ist der Drehsinn und die Winkelverteilung zu erkennen.



Das Koordinatensystem erstreckt sich in einem Bereich von 800(x) * 480(y). Der Ursprung(0/0) liegt in der unteren linken Ecke.

KOMMENTARE

Es besteht die Möglichkeit Kommentare in die Befehlseingabe in Makrofiles einzufügen. Dadurch können Überlegungen zu Befehlsabläufen erläutert und ein besseres Verständnis sichergestellt werden. Ein Kommentar beginnt mit #- (Raute Minus) und gilt bis zum Zeilenende, d. h. sobald die Kommentarzeile einen Umbruch enthalten soll, muss die nächste Zeile ebenfalls mit #- beginnen, um den Kommentar fortzusetzen.

OBJEKTBEREICH

Bei Befehlen, welche die Eigenschaft besitzen ein oder mehrere Objekte(gekennzeichnet durch: Oject-id...) zu beeinflussen, kann die Angabe des Objektbereichs durch einen Bindestrich "-" angegeben werden. Im Beispiel des Abschnitt [Styles und Objekte](#) ist die Anwendung aufgezeigt.

EINGABE VON ZAHLEN

Eingabe	Definition
123	dezimale Übergabe als ASCII Zeichen
\$5A	hexadezimale Übergabe als ASCII Zeichen
%101001	binäre Übergabe als ASCII Zeichen
?x	Code von einem Zeichen (Unicode / ASCII)
R0..R255	Übergabe des Registerwertes
Q0..Q255	Registerwert indiziert übernehmen = R(R0..255).
(...)	Ergebnis des Kalkulationsstrings übernehmen
G len32 data...	Binäre Daten senden: G len 32bit binär gefolgt von binären Daten
!string!	Werte und Strings aus Stringdatei entnehmen, Ersetzung

EINGABE VON STRINGS

Werden Strings als Parameter verwendet gilt es zu beachten, dass diese in Anführungszeichen ("") oder Hochkommata (" ") stehen müssen und durch ein Semikolon (;) beendet werden. Sollte ein String der letzte Parameter im Befehlscode sein, muss kein Semikolon am Stringende gesetzt werden. Die maximale Länge eines Strings beträgt 255 Zeichen. Ein Zeilenumbruch innerhalb eines Strings wird mit dem Pipe-Zeichen '|' bzw. New Line '\n' realisiert.

- Beispiel: "string1"; 'string2'

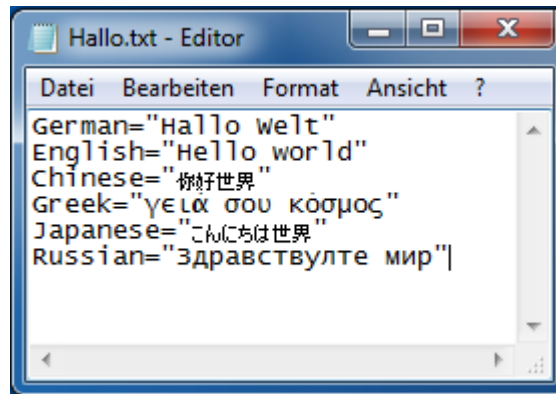
Eingabe	Definition
"Hello"32"World"	Eingabe eines Strings. Da kein ; vorhanden ist werden die Strings zusammengesetzt. Der Code dazwischen wird ebenso eingefügt. Ausgabe: Hello World
"Hello\nWorld" = "Hello World"	Mehrzeilige Eingabe eines Strings
S0..255	Stringregister übernehmen.
T0..255	Stringregister 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.

STRINGFILE

Stringfiles sind extern geschriebene Textfiles. Diese können als eine Art Datenbank von Strings verwendet werden. Auf die so gespeicherten Strings kann zugegriffen werden, indem der Name zum Aufrufen der Strings zwischen zwei Ausrufezeichen ! geschrieben wird. Es dürfen keine Anführungszeichen "" verwendet werden, da sonst nur der Aufruf als String abgebildet, jedoch nicht die gewünschte Stringfile geladen wird. Zur Verwendung dieser Funktion muss sich das erstellte Textfile lediglich in den *String*-Ordner der SD-Karte befinden und durch den Befehl zum Laden einer Stringfile ([#VFL](#)) geladen werden.

Zur Verdeutlichung folgt ein Beispiel:

Die extern erstellte und auf die SD-Karte kopierte Textdatei *Hallo.txt* sieht wie folgt aus.



Sie dient zur Verwendung eine Mehrsprachigkeit bezüglich der Aussage "Hallo Welt". Zuerst wird die Textfile durch [#VFL "Hallo"](#) geladen. Wird nun ein String durch den Befehl [#SSP 1, 1, 400, 240, 5 !German! "|" !English!](#) platziert, sieht das Resultat wie folgt aus.

Hallo Welt
Hello World

Das Pipe-Zeichen | ist für den Zeilenumbruch notwendig. Damit es für den auszugebenden String zählt, muss dieses in Anführungszeichen "" geschrieben werden. Ohne Verwendung des Stringfiles würde der oben stehende Befehl [#SSP 1, 1, 400, 240, 5 "Hallo Welt" "|" "Hello World"](#) lauten.

PFADANGABE UND FORMATIERUNG

PFADANGABE

Es gibt zwei Arten den Pfad anzugeben. Absolut und Relativ.

Bezeichnung	Beispiel
Pfadangabe absolut	</Ordner/Unterordner>
Pfadangabe relativ	<.../Unterordner>

Die absolute Pfadangabe sollte verwendet werden, um mit Dateien außerhalb des gesetzten Projektpfades zu arbeiten. Der Projektpfad, welcher durch den Befehl [#XPS](#) definiert wird, dient der Vereinfachung. Es entfällt die Angabe der übergeordneten Verzeichnisse. Durch die Eingabe von "P:" vor der Pfadangabe wird der Projektpfade automatisch eingefügt.

Soll nun auf eine Datei im gesetzten Projektordner "picture" zugegriffen werden, kann das absolut oder relativ geschehen, wie im folgenden Beispiel gezeigt wird:

Absolute Pfadangabe: </Ordner/Ordner/Ordner/Projekt/picture/Test.epg

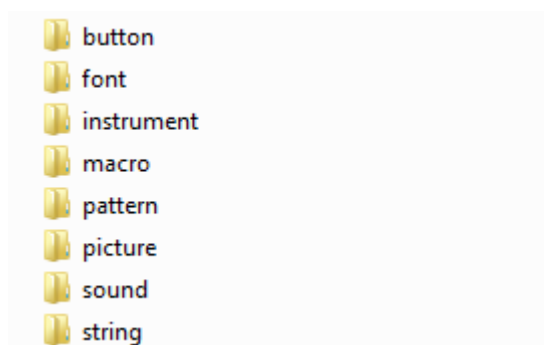
Relative Pfadangabe: <p:/picture/Test.epg → Der Projektpfad wurde mit " [#XPS](#) </Ordner/Ordner/Ordner/Projekt>" gesetzt

Die Groß- und Kleinschreibung ist bei der Pfadangabe zu berücksichtigen.

Noch eine Vereinfachung:

Es gibt eine Reihe von Defaultordnern, die automatisch im Projektordner erstellt werden und NIE verändert werden dürfen, da sonst Befehlsparameter, die automatisch auf diese Ordner zugreifen, nicht mehr funktionieren!

Diese Ordnerstruktur ist wie folgt definiert:



Beispiel:

[#PPP1](#), "Test"; 100, 100

Mit diesem Befehl wird das Bild "Test" an die Position 100,100 auf das Display geladen.

Eine alternative Eingabe wäre: [#PPP1](#), <P:/picture/Test.epg>, 100, 100 .

Gleiches geschieht bei Macros, Stringdateien(nicht Strings), Patterns, Buttons(Bilder), Sounds, Instrumenten und Fonts. Die Defaultordner dürfen niemals verändert werden. Unterordner in den Defaultordnern sind erlaubt, müssen aber im Dateinamen mit angegeben werden.

FORMATIERTE STRINGS

Formatstrings sind an die "printf" Ausgabefunktion von C angelehnt. Dadurch ist es möglich, einen String auszugeben, der zum Beispiel Zahlenwerte einer Kalkulation aufweist. Die maximale Länge des Formatstrings ohne eingefügte Werte beträgt 63 Zeichen.

Bezeichnung	Zeichen	Beschreibung
Integer	%d	Gibt Zahl als String aus.
Double	%f	Gibt Zahlen mit Nachkommastelle als String aus.
Hexadezimal	%x, %X	Gibt Hexadezimale Zahlen als String aus.

Ein Formatstring könnte wie folgt aussehen:

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

Mit entsprechenden Text- und Drawstyles sieht das Ergebnis folgendermaßen aus:

```
Integer: 3, Float: 3.14159, Hexadezimal: F
```

DATUM UND UHRZEIT

Dateformat ist eine spezielle Form des Formatstrings und beschreibt das Datum, inklusive Uhrzeit. Die nähere Erläuterung zum Dateformat kann [hier \(#WDF\)](#) gefunden werden.

In der folgenden Tabelle werden die Eingabebefehle für Dateformat aufgelistet.

Eingabe	Definition
%[]h	Stunde
%[]m	Minute
%[]s	Sekunde
%[]D	Tag
%[]M	Monat
%[]Y	Jahr
%{ }W	Name des Wochentags als String
%{ }N	Name des Monats als String
[] (optional)	0 = zwei Ziffern mit voranstehender 0 (Default) 1 = mindestens eine Ziffer ohne voranstehende Zeichen 2 = Zwei Ziffern mit voranstehenden Leerzeichen 4 = Vier Ziffern (Default für Jahre)
{ } (optional)	0...9 = Die ersten X Zeichen aus dem String anzeigen.

Beispiel:

#SDP

Datum und Uhrzeit
 Platzierung eines formatierten Datums

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

Thursday the 21. Apr 2016, 11:37

BILDER

Das Modul arbeitet intern mit einem speziellen Bildformat (*.epg). Die Umwandlung muss extern geschehen. Das Windowsprogramm EA uniSketch bietet die komfortabelste Möglichkeit die meisten Bildformate umwandeln zu lassen. Manche [Befehle](#) ermöglichen es, den Bildschirminhalt auf die SD-Karte zu speichern oder Bild-daten direkt über die serielle Schnittstelle zu übertragen. Hier stehen verschiedene [Bildformate](#) zur Verfügung.

BILDFORMATE

Die folgenden Bilderformate gelten für Screenshots/Hardcopies des Displays oder des Video-Inputs.

Eingabe	Definition	Speicherbedarf / Ausführungszeit
1	BMP 24Bit → True Color	sehr hoch
2	BMP 16Bit → High Color	hoch
3	BMP 8Bit grey → Graustufenbild	gering
11	epg 32Bit → True Color mit Alphakanal	sehr hoch
12	epg 16Bit → High Color	hoch
13	epg 8Bit grey → Graustufenbild	gering
21	epg 32Bit RLE compressed → siehe oben	hoch
22	epg 16Bit RLE compressed → siehe oben	mittel
23	epg 8Bit grey RLE compressed → siehe oben	sehr gering

POLYLINIE UND POLYGONE

Mit dem Befehl Polylinie ([#GPL](#)) und Polygon ([#GPF](#)) können nahezu beliebige Formen erzeugt werden. Jeder Abschnitt wird als Segment bezeichnet.

SEGMENT

Ein Segment ist allgemein ein Ausschnitt beziehungsweise ein Teil von etwas Ganzem.

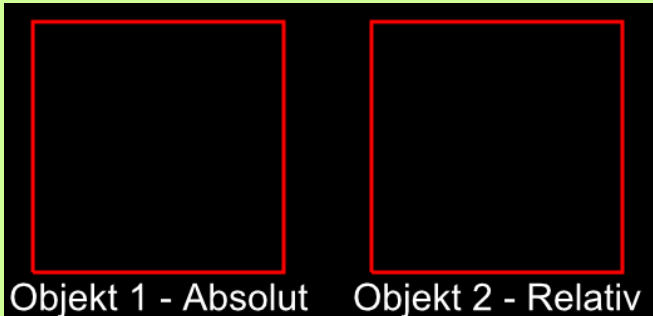
In der hiesigen Befehlsanwendung erfüllen die Segmente zwei primäre Aspekte. Zum einen können Grafiken und zum anderen Aktionspfade segmentweise erstellt werden. Dadurch erhalten frei wählbare Formen ihren Einzug in die Gestaltung des Layouts.

INDIKATOREN

Jedes Segment besitzt einen Indikator in Form eines Buchstabens, wodurch die Segmentart erkannt wird. Die Auflistung der Indikatoren befindet sich in der Befehlsübersicht [hier](#).

Eine Segmenteingabe beginnt immer mit "?" und anschließendem Hinzufügen des Indikators sowie der entsprechenden Parameter. Dabei können beliebig viele Segmente aneinander geknüpft werden. Des Weiteren besteht bei den Indikatoren ein Unterschied zwischen Groß- und Kleinschreibung, was die Unterscheidung zwischen absoluten und relativen Koordinaten darstellt.

Diesbezüglich ein Beispiel:

Erzeugung eines Quadrates mit absoluter und relativer Segmentangabe	
<pre>#GPP1, 1, 100, 100, ?V, 300, ?H, 300, ?V, 100, ?H, 100 #GPP2, 1, 370, 100, ?v, 200, ?h, 200, ?v, -200, ?h, -200</pre>	 <p>Objekt 1 - Absolut Objekt 2 - Relativ</p>

In diesem Beispiel werden durch das Verknüpfen horizontaler und vertikaler Linien zwei identische Quadrate erzeugt. Die Erzeugung von Objekt 1 erfolgt absolut, sprich es müssen die absoluten Koordinaten des Displays zur Positionierung der Linienpunkte angegeben werden. Daraus werden durch die Verbindung der Punktkoordinaten die entsprechenden Linien erzeugt. Bei Objekt 2 hingegen wird nur der Startpunkt absolut definiert. Bezogen auf diesen Startpunkt werden nun die Längen der jeweiligen Linien angegeben. Somit bezieht sich dieses Quadrat relativ auf den Startpunkt.

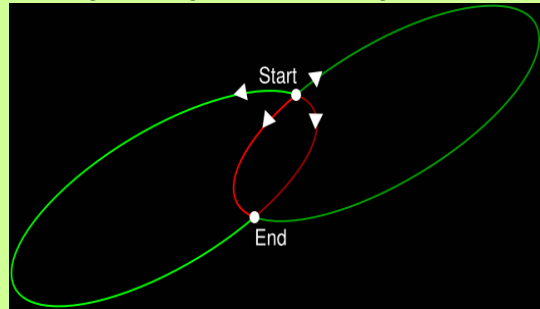
LAUFRICHTUNGEN UND KREISBÖGEN

Die Laufrichtung ist bei den kreisförmigen Segmenten zu beachten. Je nach Laufrichtung wird ein anderes Segment erzeugt.

In der unten stehenden Tabelle wird ein Segment als vollständige Ellipse mit Start- und Endpunkt definiert. Aufgrund der Angabe der Radien der Ellipse bestehen nun vier Möglichkeiten zur Verknüpfung von Start- und Endpunkt. Die vier Möglichkeiten unterscheiden sich bezüglich der beiden Laufrichtungen und der zwei möglichen Kreisbögen. Die kleinen Kreisbögen sind rot gekennzeichnet. Die beiden großen Kreisbögen wurden grün hervorgehoben. Die Laufrichtungen im Uhrzeigersinn werden dunkler dargestellt.

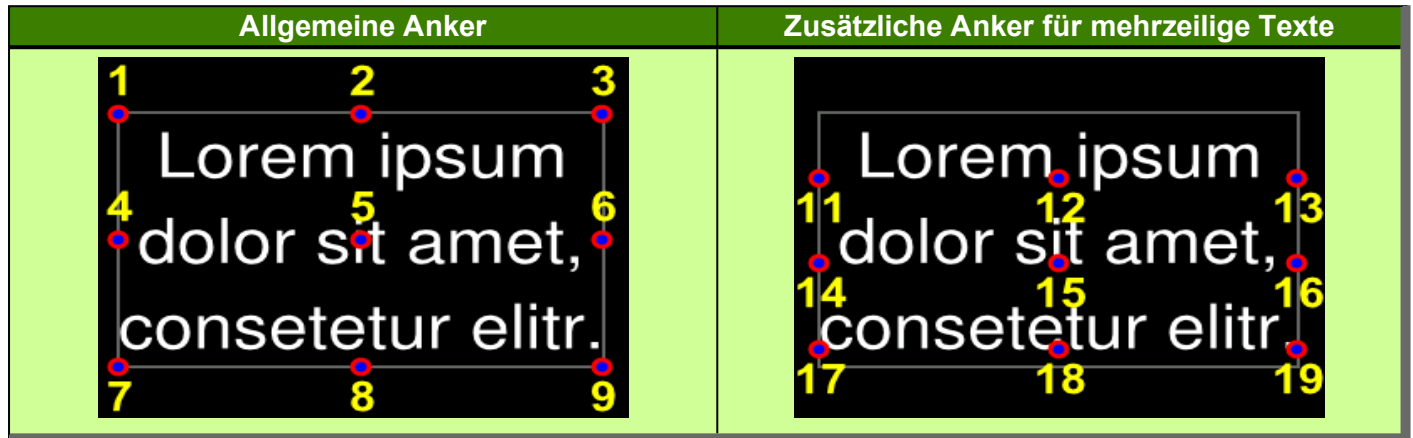
Bei Kreisen macht die Unterscheidung nach großen und kleinen Kreisbogen keinen Unterschied. Dies ist dem Fakt geschuldet, dass ein Kreis nur einen Radius vorzuweisen hat, wodurch der große und kleine Kreisbogen identisch sind.

Aufzeigen der Kreisbögen und Laufrichtungen	
#GPP1, 1, 390, 380, ?E0, 200, 70, 25, 340, 260	0: gegen Uhrzeigersinn kleiner Kreisbogen
#GPP2, 2, 390, 380, ?E1, 200, 70, 25, 340, 260	1: im Uhrzeigersinn kleiner Kreisbogen
#GPP3, 3, 390, 380, ?E2, 200, 70, 25, 340, 260	2: gegen Uhrzeigersinn großer Kreisbogen
#GPP4, 4, 390, 380, ?E3, 200, 70, 25, 340, 260	3: im Uhrzeigersinn großer Kreisbogen



ANKER

Alle Objekte können an eine Position(x,y) erzeugt werden. Mithilfe der Angabe der Ankernummern kann nun bestimmt werden, ob das Objekt z.B. mit der linken oberen Ecke(Anker 1) positioniert wird oder z.B. exakt in der Mitte(Anker 5). Bei mehrzeiligen Texten gibt es noch weitere Anker(11-19), die sich auf die Baseline beziehen. In der unten stehenden Tabelle ist ein String als Objekt zu sehen, um die Anker zu verdeutlichen:



Neben den oben stehenden Ankernummern gibt es außerdem Anker 0. Anker 0 ist ein spezielle Anker, welcher durch entsprechende Befehle beliebig gesetzt werden kann. Dadurch ist zum Beispiel ein Objekt in der Lage um einen gewünschten Punkt zu rotieren. Im nachfolgendem Beispiel wurde diesbezüglich ein Rundthermometer erzeugt.



Der Zeiger soll sich um den Mittelpunkt des Instrumentes drehen. Mit den allgemeinen Ankern, welche als kleine graue Quadrate dargestellt werden, kann der Zeiger ebenfalls um den Mittelpunkt des Thermometers rotieren. Die Standardanker 1 bis 9 sind hierfür nicht geeignet. Deshalb bestimmt man in diesem Fall einen individuellen Anker 0. Dieser kann Pixelgenau bestimmt werden und ist im Bild rot dargestellt. Die Rotation des Zeigers erfolgt nun um diesen Ankerpunkt.

GRUPPEN

Gruppen vereinfachen den gleichzeitigen Umgang mit mehreren Objekten. Die gewünschten Objekte werden in einer Objektgruppe zusammengefasst, behalten jedoch individuelle Funktionen und Definitionen bei. Der Vorteil liegt darin, dass ohne viel Aufwand alle Objekte der Gruppe gleichzeitig beeinflusst werden können, wie zum Beispiel eine Verschiebung, eine Rotation oder ein Ausblenden.

Neben den einfachen Objektgruppen gibt es des Weiteren auch Gruppen für Touch Schalter. Diese gewährleisten außerdem, dass nur ein Schalter der Gruppe aktiv ist, wodurch nicht jeder Schalter einzeln mit einer solchen Funktion definiert werden muss.

Achtung: Gruppen-ID's und Objekt-ID's befinden sich im gleichen Zahlenraum und überschreiben sich somit gegenseitig.

KALKULATION

Jeder numerische Parameter kann durch eine Kalkulation ersetzt werden. Die Kalkulation muss in Klammern () abgeschlossen werden.

Name	Befehl	Beschreibung	Integer	Float
mathematische Funktionen	+, -, *, /, ()	Arithmetische Funktionen	✓	✓
x (Betrag)	abs(x)	Betragsrechnung	✓	✓
x%y (Modulo)	mod(x,y)	Rechnung mit Rest	✓	
x^y (Potenz)	pow(x,y)	Rechnung mit Potenz	✓	✓
Wurzel	sqrt(var)	Berechnung Wurzel		✓
Logarithmus	log(var)	Berechnung Logarithmus		✓
Natürlicher Logarithmus	ln(var)	Berechnung natürlicher Logarithmus		✓
Grad in Rad	rad(deg)	Umrechnung Grad in Rad		✓
Rad in Grad	deg(rad)	Umrechnung Rad in Grad		✓
Sinus	sin(deg)	Berechnung Sinus		✓
Kosinus	cos(deg)	Berechnung Sinus		✓
Tangenz	tan(deg)	Berechnung Kosinus		✓
Arcussinus	asin(var)	Berechnung Arkussinus		✓
Arcuscosinus	acos(var)	Berechnung Arkuscosinus		✓
Arcustangens	atan(var)	Berechnung Arkustangens		✓
Arcustangens quadrantenrichtig	atan(y,x)	Berechnung Arkustangens quadrantenrichtig		✓
Zufallswert Bereich	rand(sv,ev)	Zufallswert in Wertebereich	✓	✓
Zufallswert 0 - ev	rand(ev)	Zufallswert größer 0	✓	✓
Zufallswert 0<= x<=1000	rand()	Zufallswert größer 0; kleiner 1000	✓	✓
Minima	min(a,b,c...)	Kleinster Wert	✓	✓
Maxima	max(a,b,c...)	Größter Wert	✓	✓
Durchschnitt	avg(a,b,c...)	Durchschnitt	✓	✓
Bit Operatoren	<<, >>, &, , ^, ~	Bit Operatoren	✓	
Logische Operatoren	<, >, <=, >=, !=, ==, &&, , !	Logische Operatoren	✓	✓
Erhöhen, vermindern	++,--	Erhöhen, Vermindern	✓	✓
Port (a=0..15) = 0..255	port(a)	Portwert	✓	✓
Portbit (a=0..127) = 0/1	bit(a)	Bitwert	✓	✓
Analogport (a=0..3) = 0..4095	analog(a)	Analogportwert	✓	✓

Name	Befehl	Beschreibung	Integer	Float
Aktuelles Datum in Sekunde	date()	Aktuelles Datum	✓	
Tage in Sekunde	date(D)	Aktueller Tag	✓	
Tag + Monat + Jahr (1932 - 2067) in Sekunde	date(D,M,Y)	Beliebige Datumsangabe	✓	
Aktuelle Uhrzeit in Sekunde	time()	Aktuelle Uhrzeit	✓	
Stunde in Sekunde	time(h)	Aktuelle Stunde	✓	
Stunde + min in Sekunde	time(h, m)	Aktuelle Stunde und Minute	✓	
Stunde + min + Sek in Sekunde	time(h, m, s)	Aktuelle Stunde, Minute und Sekunde	✓	
Aktuelle Zeit und Datum	datetime()	Aktuelles Datum mit Zeit	✓	
Stunde +min+sec + Tag+Monat+Jahr in Sekunde	datetime(h,m,s,D,M,Y)	Beliebige Datums- und Zeitangabe	✓	
Aktuelles Jahr	year()	Gespeichertes Jahr	✓	
Sekunden von 1.1.2000 0:00:00 in Jahr	year(a)	Jahr speichern	✓	
Aktueller Monat	month()	Gespeicherter Monat	✓	
Sekunden von 1.1.2000 0:00:00 in Monat	month(a)	Monat speichern	✓	
Aktueller Tag	day()	Gespeicherter Tag	✓	
Sekunden von 1.1.2000 0:00:00 in Tag	day(a)	Tag speichern	✓	
Aktueller Wochentag (0=Sonntag)	weekday()	Gespeicherter Wochentag	✓	
Sekunden von 1.1.2000 0:00:00 in Wochentag	weekday(a)	Wochentag speichern	✓	
Aktuelle Stunde	hour()	Gespeicherte Stunde	✓	
Sekunden von 1.1.2000 0:00:00 in Stunde	hour(a)	Stunde speichern	✓	
Aktuelle Minute	minute()	Gespeicherte Minute	✓	
Sekunden von 1.1.2000 0:00:00 in Minute	minute(a)	Minute speichern	✓	
Aktuelle Sekunde	second()	Gespeicherte Sekunde	✓	
Sekunden von 1.1.2000 0:00:00 in Sekunde	second(a)	Sekunde speichern	✓	
Globaler 10 ms timer	timer()	Globaler Zeitmesser 10ms	✓	
24 bit Farbe Rot Kanal	getR(x)	Farbanteil rot bestimmen	✓	
24 bit Farbe Grün Kanal	getG(x)	Farbanteil grün bestimmen	✓	
24 bit Farbe Blau Kanal	getB(x)	Farbanteil blau bestimmen	✓	
24 bit Farbe RGB	RGB(R, G, B)	Farbe komplett bestimmen	✓	

Name	Befehl	Beschreibung	Integer	Float
Gibt RGB von Ramp-nr aus	rampRGB(nr, offset)	Gibt Farbe von Farbverlauf aus	✓	✓
Gibt Transparenz von Ramp-nr aus	rampO(nr,offset)	Gibt Tansparenz von Farbverlauf aus	✓	✓
Displaybreite	scrW()	Breite des Displays	✓	✓
Displayhöhe	scrH()	Höhe des Displays	✓	✓
Videobreite	vidW()	Breite des Videos	✓	✓
Videohöhe	vidH()	Höhedes Videos	✓	✓
Anzahl Video Objekte	vidC()	Anzahl Videoobjekte	✓	✓
Objektbreite (ohne transform)	objW(id)	Breite des Objekts	✓	✓
Objekthöhe (ohne transform)	objH(id)	Höhe des Objekts	✓	✓
Objektposition x aktueller Anker relative Koordinaten zum Elternobjekt	objX(id)	X-Position des Objekts	✓	✓
Objektposition x beliebiger Anker Screenkoordinaten	objX(id, anchor)	X-Position des Objekts		
Objektposition y aktueller Anker relative Koordinaten zum Elternobjekt	objY(id)	Y-Position des Objekts	✓	✓
Objektposition y beliebiger Anker Screenkoordinaten	objY(id, anchor)	Y-Position des Objekts		
Objektskalierung Breite	objSW(id)	Skalierung der Objektbreite	✓	✓
Objektskalierung Höhe	objSH(id)	Skalierung der Objekthöhe	✓	✓
Objektscherung X	objSX(id)	Scherung in X-Richtung des Objekts	✓	✓
Objektscherung Y	objSY(id)	Scherung in Y-Richtung des Objekts	✓	✓
Objektrotation	objR(id)	Rotation des Objekts	✓	✓
Objekttransparenz	objO(id)	Transparenz des Objekts	✓	✓
Objektebene	objL(id)	Ebene des Objekts	✓	✓
Objekt Style Nummer	objC(id)	Style Nummer der Objekts	✓	✓
Aktueller Objektanker	objA(id)	Aktueller Anker des Objekts	✓	✓
Instrument Bar aktueller Wert	objIV(id)	Aktueller Instrumentenwert	✓	✓
Instrument Bar gezeichneter Wert	objID(id)	Gezeichneter Instrumentenwert	✓	✓
Instrument Bar Endwert	objIE(id)	Endwert der Insturments	✓	✓
Instrument Bar Startwert	objIS(id)	Startwert des Instruments	✓	✓
Gibt Länge des Aktionspfades aus	pathL(id)	Länge des Aktionspfades	✓	✓

Name	Befehl	Beschreibung	Integer	Float
Gibt relative X-Koordinate von Beginn des Aktionspfades aus	pathX(id,distance)	X-Position Pfadbeginn	✓	✓
Gibt relative Y-Koordinate von Beginn des Aktionspfades aus	pathY(id,distance)	Y-Position Pfadbeginn	✓	✓
Gibt Winkel der Tangente des Aktionspfades aus	pathR(id,distance)	Tangentenwinkel des Pfades	✓	✓
Gibt Touch Button/Schalter Status aus	butS(id)	Status Button oder Schalter	✓	✓
Gibt aktiven Touch Schalter der Radiogruppe aus	butR(groupid)	Aktiver Schalter in Gruppe	✓	✓
Gibt zuletzt gedrückte Taste/Schalter aus	butI()	Zuletzt gedrückter Schalter	✓	✓
Gibt Code der letzten Keyboard Taste aus	butC()	Zuletzt gedrückte Keyboard Taste	✓	✓
Gibt Länge des Strings aus	strL(nr)	Länge des Strings	✓	
Gibt ASCII-Code von Stringregister aus	strA(nr, pos)	ASCII-Code des gespeicherten Strings	✓	
Gibt Unicode von Stringregister aus	strU(nr, pos)	Unicode des gespeicherten Strings	✓	
Vergleich zwei Stringregister	strC(n1, n2)	Stringregister vergleichen	✓	
Numerischen String zu Wert konvertieren	strV(nr)	Numerischen String in beschriebene Wertigkeit konvertieren	✓	✓
Prüft ob Datei existiert (<name> in Stringregister nr)	fileE(nr)	Vorhandensein der Datei im Stringregister prüfen	✓	
Gibt Dateigröße aus (<name> in Stringregister nr)	fileS(nr)	Größe der Datei im Stringregister prüfen	✓	
Gibt Dateizeit aus(<name> in Stringregister nr)	fileT(nr)	Zeit der Datei im Stringregister prüfen	✓	
Gibt Dateieigenschaft aus(<name> in Stringregister nr)	fileA(nr)	Eigenschaft der Datei im Stringregister prüfen	✓	
Gibt Dateidatum aus(<name> in Stringregister nr)	fileD(nr)	Daum der Datei im Stringregister prüfen	✓	
Sekunde in FatTime 16Bit	fatT(Sekunde)		✓	
Sekunde in FatDate 16Bit	FatD(Sekunde)		✓	
Integer Kalkulation als float ausgeben	int(calculation)	Integerwert aus Float-Rechnung		✓
Float Kalkulation als integer ausgeben	float(calculation)	Floatwert aus Integer-Rechnung	✓	

AKTION UND ANIMATION

Aktionen und Animationen können genutzt werden, um Objekte beziehungsweise Grafiken auf dem Display zum "Leben zu erwecken". Dabei gibt es zwischen einer Aktion und einer Animation wesentliche Unterschiede.

AKTION

[Aktionen](#) werden verwendet, um Objekte zu beeinflussen. So können Objekte zum Beispiel verschoben oder deren Transparenz geändert werden. Demnach ist auch ein erscheinendes oder verschwindendes [Verhalten](#) definierbar. Werden durch eine Aktion die Parameter eines Objekts beeinflusst, dann bleiben diese auf dem neu zugeteilten Wert. Das heißt, wenn zum Beispiel ein Objekt verschwinden soll, dann wird es am Ende der Aktion gelöscht. Durch die Positionsänderung in Kombination mit den [Aktionspfaden](#) ermöglichen Aktionen auch das Verfolgen eines definierten Pfades. Aktionspfade bieten den Vorteil, dass Objekte darüber prozentual rotieren, skalieren, sich positionieren oder ihre Transparenz ändern können. Über die [Aktionskurven](#) kann der zeitliche Ablauf der Aktionen angepasst werden. Als Vorlagen stehen zum Beispiel bereits ein linearer Ablauf oder ein Ablauf mit Verzögerung beziehungsweise Beschleunigung zur Verfügung.

ANIMATION

Animationen gelten nur für Dateien vom Typ **GIF** sowie für Farbfüllungen und Linienmuster. Durch die verschiedenen [Animationstypen](#) können der Bildablauf von GIFs beeinflusst beziehungsweise Linienmuster und Farbfüllungen von Grafiken animiert werden. Für den zeitlichen Ablauf der Animation können, wie bereits bei den Aktionen bereits geschildert, die [Aktionskurven](#) zur besseren Anpassung genutzt werden.

HARDWARE

Das Displaymodul ist für 3,3 V Betriebsspannung ausgelegt und verfügt zur Kommunikation mit dem Host über folgende serielle Schnittstellen:

- [RS232](#)
- [SPI](#)
- [I²C](#)
- [USB](#)

Alle Schnittstellen sind gleichberechtigt und können parallel genutzt werden.

Drei weitere Schnittstellen RS232, SPI und I²C können zur Kommunikation mit weiteren Baugruppen verwendet werden. Diese sind als Master-Slave Schnittstellen ausgelegt und können durch diverse Befehle angesprochen werden.

Ebenso verfügt das Displaymodul über eine LED Hintergrundbeleuchtung deren Helligkeit per Befehl von 0 - 100 % variiert werden kann. Um die Lebensdauer von typ. 50.000 Stunden (halbe Helligkeit) zu optimieren sollte die Beleuchtung so oft als möglich gedimmt bzw. abgeschaltet werden.

Daten wie z.B. Bilder, Zeichensätze oder ganze Bildschirmdarstellungen, sowie Menüs können auf der integrierten [SD-Karte](#) abgelegt werden.

PINBELEGUNG

Im Folgenden ist eine Übersicht zur Pinbelegung des Displays dargestellt.

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	RS 232 TxD	O	RS232: Transmit Data	35	I/O 0.7	I/O	Input/Output 0.7 (Bit 7)
10	RS 232 RS485	O	RS232: Transmit Enable RS485	36	I/O 0.6	I/O	Input/Output 0.6 (Bit 6)
11	I ² C SDA	I/O	I ² C: Serial Data	37	I/O 0.5	I/O	Input/Output 0.5 (Bit 5)
12	I ² C SCL	I	I ² C: 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 negativ	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 positiv	42	A/D 2	I	Analogeingang 2
17	I/O 0.0	I/O	Input/Output 0.0 (Bit 0)	43	I ² C SCL	I	User I ² C: Clock Data
18	USB VBus		USB Supply Voltage	44	I ² C SDA	I/O	User I ² C: Serial Data
19	I/O 0.1	I/O	Input/Output 0.1 (Bit 1)	45	RS 232 TxD	I	User RS232: Transmit Data
20	Testmode SBUF	I O	PowerOn low: Testmode low: Daten im Sendbuffer	46	RS232 RxD	O	User RS232: Receive Data
21	PWM	O	PWM Ausgang	47	SPI CLK	O	User SPI: Serial Clock
22	DPROT	I	open (high): Protokoll aktiv low: Protokoll deaktiviert	48	SPI MISO	I	User SPI: Serial In
23	SND+	O	Lautsprecherausgang (8 Ohm)	49	SPI MOSI	O	User SPI: Serial Out
24	SND-	O	Lautsprecherausgang	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 ist ein Standard für eine serielle Schnittstelle. Die Übertragung erfolgt seriell asynchron. Die Daten werden also in ein Bitstrom gewandelt und übertragen. Es existiert keine gemeinsame Taktleitung, jeder Bus-teilnehmer muss also mit der selben Übertragungsrate (sogenannte Baudrate) arbeiten. RS232 ist eine Spannungsschnittstelle, die Dateninformationen werden also durch Spannungspegel übertragen. In der PC-Welt und Industriesteuerungen sind Pegel von +12V bzw. -12V als Standard definiert. Innerhalb von Platinen bzw. in Mikrocontrollersteuerungen wird mit 0V bzw. VDD (im Fall des EA uniTFT 3,3 V) gearbeitet. Um die Signalpegel anzupassen gibt es einige Möglichkeiten in Form von Levelshiftern.

RS232 besteht aus "hörenden" und "sprechenden" Leitungen, die zwischen den beiden Teilnehmern gekreuzt werden.

DATENFORMAT

Das Format für die Datenübertragung ist 8-N-1:



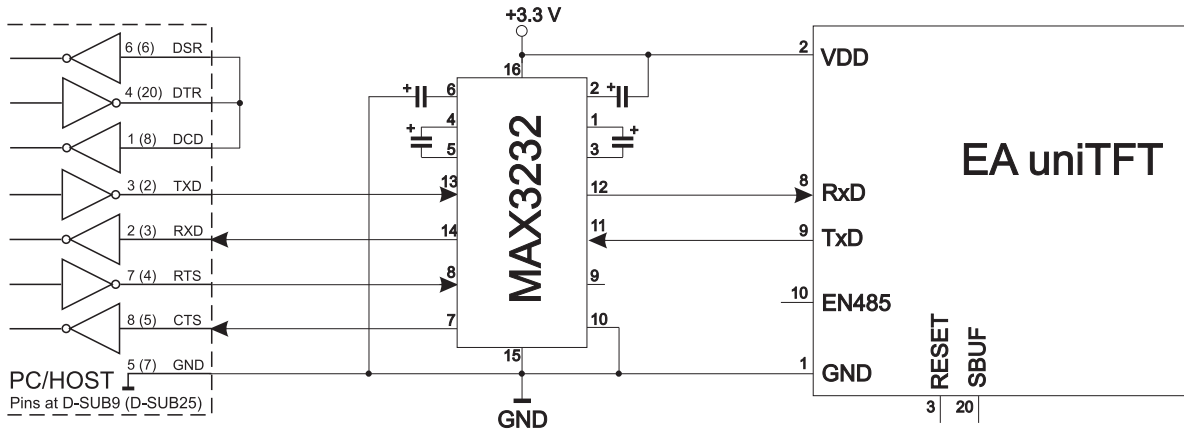
BAUDRATEN

In der folgenden Tabelle werden die möglichen Baudraten aufgelistet.

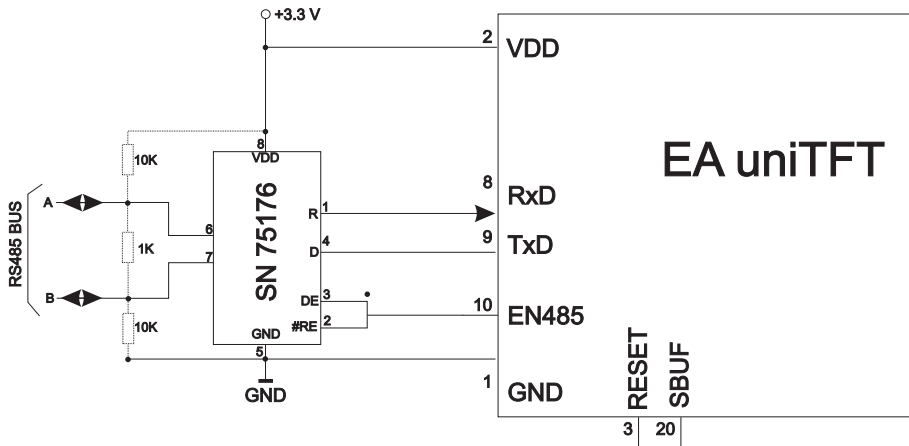
Baudrate	Fehler in %
9600	+0,04
19200	-0,08
38400	+0,16
57600	-0,08
115200	+0,64
230400	-0,80
460800	+2,08
921600	-3,68

APPLIKATIONSBEISPIELE

RS232 V24 - VERBINDUNG ZU EINEM PC



RS485 - BUSSYSTEM



SPI

Das **S**erial **P**eripheral **I**nterface ist ein Bussystem für eine serielle synchrone Datenübertragung zwischen verschiedenen ICs.

Der Bus besteht aus folgenden Leitungen

- MOSI (**M**aster **O**ut → **S**lave **I**n) auch SDO (Serial Data Out) oder DO
- MISO (**M**aster **I**n ← **S**lave **O**ut) auch SDI (Serial Data In) oder DI
- SCK (**S**erial **C**lock) - Schiebetakt
- SS (**S**lave **S**elect → Adressierung des Partners) auch CS (Chip Select)

SPI arbeitet mit einem bidirektionalem Übertragungsprinzip, es werden also zeitgleich Daten zwischen den Partner ausgetauscht. Jede Kommunikation wird vom Master mit Hilfe der SCK-Leitung bestimmt. Das Display arbeitet im Slave-Mode.

Das Protokoll für die Datenübertragung ist bei SPI nicht festgelegt, daher gibt es verschiedene Einstellmöglichkeiten. Diese werden durch die Parameter Clock Polarity, Clock Phase sowie Data Order festgelegt:

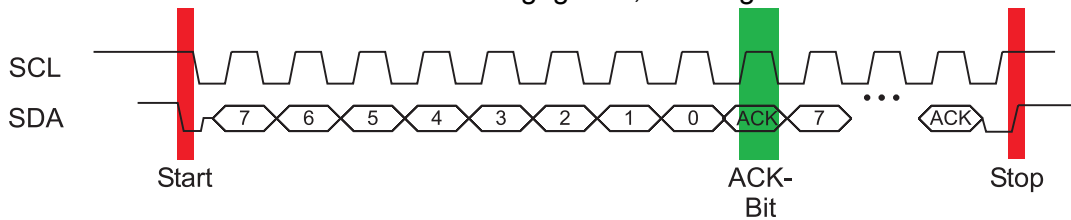
Mode	CLOCK Polarity	CLOCK Phase	DORD	
			DORD(0) MSB First	DORD(1) LSB First
0	Low (0)	erste Flanke (0)	SPI-Mode: 0 CPOL=0 (CLK idle state LOW) CPHA=0 (data valid at first edge) DORD=0 (MSB=send BIT7 first)	SPI-Mode: 0 CPOL=0 (CLK idle state LOW) CPHA=0 (data valid at first edge) DORD=1 (LSB=send BIT0 first)
1	Low (0)	zweite Flanke (1)	SPI-Mode: 1 CPOL=0 (CLK idle state LOW) CPHA=1 (data valid at second edge) DORD=0 (MSB=send BIT7 first)	SPI-Mode: 1 CPOL=0 (CLK idle state LOW) CPHA=1 (data valid at second edge) DORD=1 (LSB=send BIT0 first)
2	High (1)	erste Flanke (0)	SPI-Mode: 2 CPOL=1 (CLK idle state HIGH) CPHA=0 (data valid at first edge) DORD=0 (MSB=send BIT7 first)	SPI-Mode: 2 CPOL=1 (CLK idle state HIGH) CPHA=0 (data valid at first edge) DORD=1 (LSB=send BIT0 first)
3	High (1)	zweite Flanke (1)	SPI-Mode: 3 CPOL=1 (CLK idle state HIGH) CPHA=1 (data valid at second edge) DORD=0 (MSB=send BIT7 first)	SPI-Mode: 3 CPOL=1 (CLK idle state HIGH) CPHA=1 (data valid at second edge) DORD=1 (LSB=send BIT0 first)

I²C

I²C steht für **I**nter-**I**ntegrated **C**ircuit und ist ein von Phillips entwickelter serieller Datenbus. Der als Master-Slave-Bus konzipierte Bus benötigt 2 Signalleitungen:

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

Die elektrische Spezifikation sieht vor, dass beide Leitungen mit einem Pull-Up-Widerstand an VDD abgeschlossen werden, denn sämtliche an dem Bus angeschlossene Geräte haben Open-Collector-Ausgänge. Der Bustakt wird immer durch den Master vorgegeben, der die gesamte Kommunikation bestimmt:



Nach der Startbedingung folgt in einem Übertragungsprotokoll immer die Slaveadresse. Hierbei ist das Bit 0 das sogenannte R/W-Bit und bestimmt ob vom Slave gelesen (1) oder Daten übermittelt (0) werden sollen. Der Datenaustausch erfolgt bis der Master die Stopbedingung ausführt. Genauere Informationen sind in der I²C Spezifikation zu finden. Das Display arbeitet im Slave-Mode.

USB

Der **U**niversal **S**erial **B**us ist ein serielles Bussystem zur Verbindung mit einem Computer oder anderem Gerät. Er basiert auf einer differentiellen Datenübertragung. Die Bustopologie ist eine strikte Master-Slave-Kommunikation (Ausnahme: On the Go Geräte). Im Fall des EA uniTFT muss immer der PC/Master die Kommunikation leiten.

Das Modul verfügt über eine CDC Geräteklasse und meldet sich damit als virtuelle serielle COM-Schnittstelle am PC an:

Beschreibung	Wert
Geräteklasse	2
USB Vendor ID	0x2DA9
USB Product ID	0x2454
Gerätebeschreibung	EA uniTFT

Um das Displaymodul zu programmieren empfehlen wir die USB-Schnittstelle.

SD CARD

Das Modul verfügt über eine austauschbare SD-Karte(bis zu 32GB).



Der Speicher wird für sämtliche Projektdaten, wie z.B. verschiedene Styles, Objekte und Bilder verwendet. Der Speicher kann zusätzlich für Datenlogfunktionen zur Laufzeit verwendet werden.

Die SD-Karte muss mit FAT32 formatiert sein.

VIDEO EINGANG / KAMERA

Das Modul verfügt über einen analogen Videoeingang und unterstützt drei Farbsysteme: PAL (**P**hase-**A**lternating-**L**ine), Secam (**S**équentiel couleur à mémoire) und NTSC (**N**ational **T**elevision **S**ystems **C**ommittee). Das Modul wählt automatisch die richtigen Farbsystemeinstellungen.

Die meisten Digitalkameras und Videokameras verfügen über eine passende Schnittstelle, ebenso wie die meisten CMOS-Kameras. Die Auflösung ist dabei auf 720x576 (PAL und SECAM), 640x480 (NTSC) Pixel begrenzt. Der Eingang wird auch als Composite video signal input bezeichnet.

ANALOG INPUT

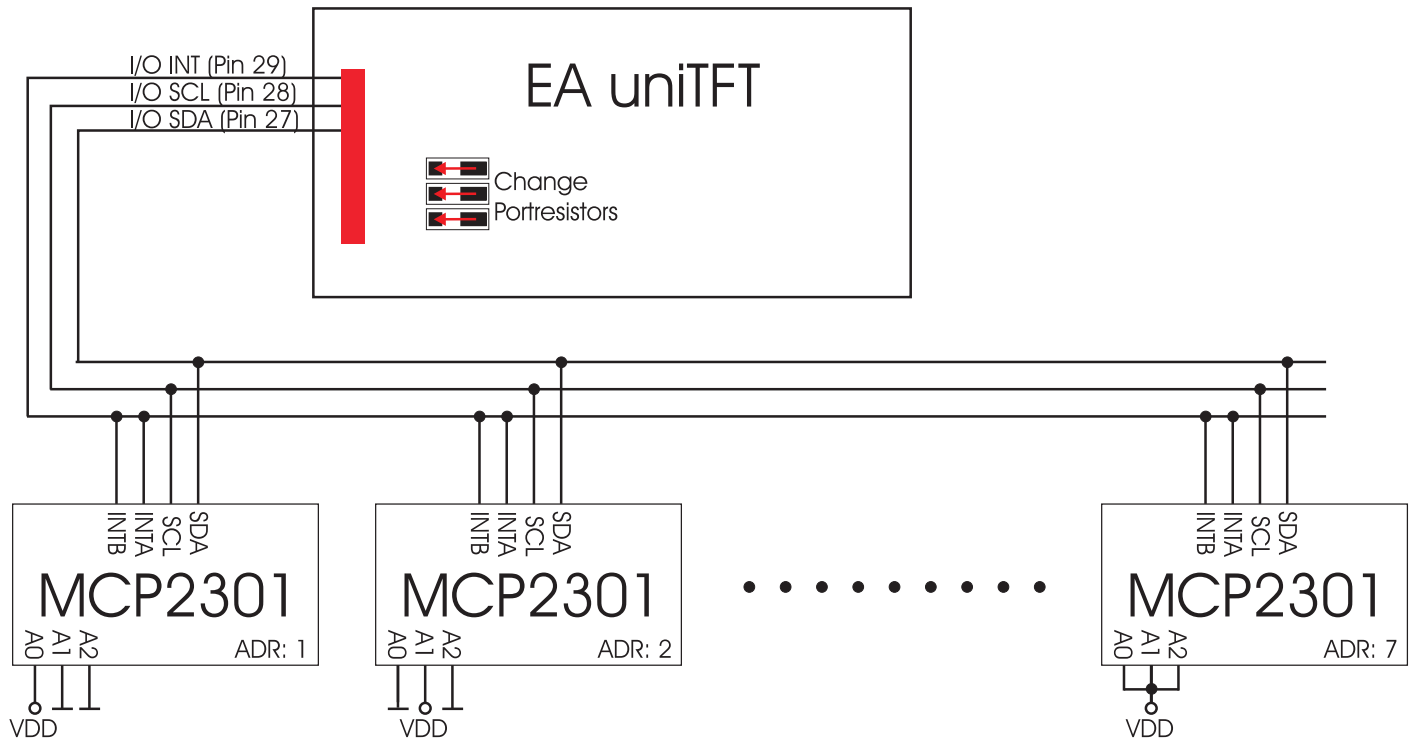
Das Modul verfügt über einen analogen Dateneingang mit einer Genauigkeit von 12 Bit. Durch einen Analog-Digital-Wandler werden analoge Signale erfasst und in digitale Signale umgerechnet. Dadurch ist das Modul in der Lage analoge Daten aufzunehmen, welche dann zur Bearbeitung bereit stehen.

PULSWEITENMODULATION (PWM)

Das Modul verfügt über die Möglichkeit der Datenumsetzung mittel Pulsweitenmodulation, kurz PWM. Dabei wird bei konstanter Frequenz das Tastverhältnis eines rechteckigen sowie zweiwertigen Impulses geändert beziehungsweise moduliert. Durch die Modulation ändert sich das Verhältnis zwischen An- und Ausschaltzeit und somit die Charakteristika des Ausgangssignals. Auf diese Art und Weise können elektromechanische Bauteile wie z.B. Motoren angesteuert werden. Die Variation der Tastverhältnisse sorgt dann für eine geringe Motordrehzahl bei kurzer Anschaltzeit oder eine hohe Motordrehzahl bei langer Anschaltzeit.

INPUT/OUTPUT(I/O)

Das Modul verfügt standardmäßig über 16 digital I/O. Durch das Umlegen der Widerstände für die [PINs](#) 27 (SDA), 28 (SCL), 29 (INT) gehen zwar drei I/O verloren, jedoch kann dadurch die Anzahl der gesamten I/O bis auf 128 erweitert werden. Im Folgenden Bild sind die drei Widerstände sowie das Umlegen aufgezeigt.



Die maximale Leistung des Bauteils beträgt 700mW . Das heißt, dass im Falle der Nutzung aller 16 Bausteine ohne Erweiterung maximal 200mA Strom pro Baustein fließen dürfen. Die maximale Strombelastung für einen einzelnen Pin liegt bei 25mA .

PULSWEITENMODULATION (PWM)

Das Modul verfügt über die Möglichkeit der Datenumsetzung mittel Pulsweitenmodulation, kurz PWM.

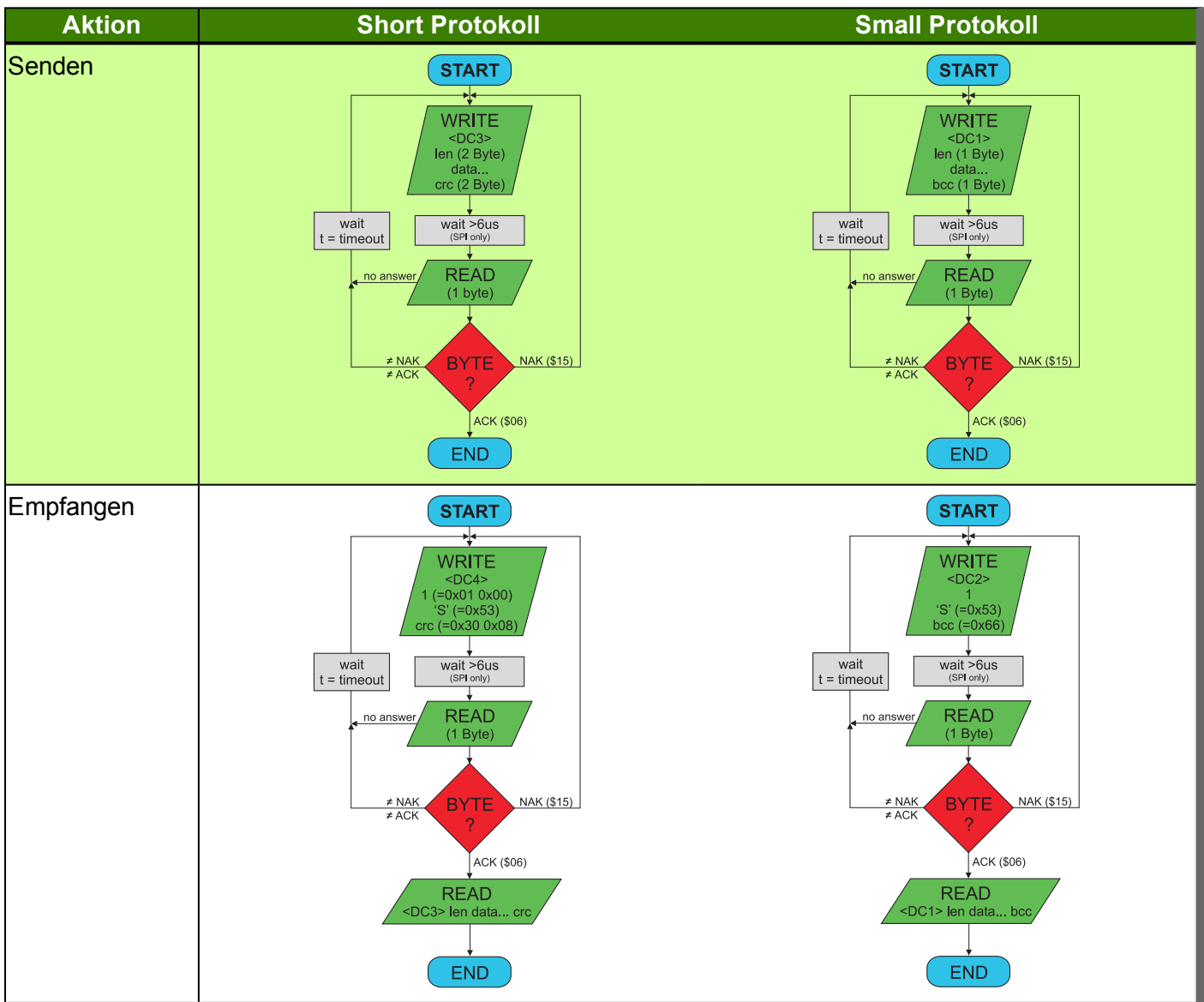
Dabei wird bei konstanter Frequenz das Tastverhältnis eines rechteckigen sowie zweiwertigen Impulses geändert beziehungsweise moduliert. Durch die Modulation ändert sich das Verhältnis zwischen An- und Ausschaltzeit und somit die Charakteristika des Ausgangssignals. Auf diese Art und Weise können elektromechanische Bauteile wie z.B. Motoren angesteuert werden. Die Variation der Tastverhältnisse sorgt dann für eine geringe Motordrehzahl bei kurzer Anschaltzeit oder eine hohe Motordrehzahl bei langer Anschaltzeit.

SHORT PROTOKOLL & SMALL PROTOKOLL

Das Protokoll ist für alle 4 Schnittstellenarten RS-232, SPI, I²C und USB identisch aufgebaut. Die Datenübertragung ist jeweils eingebettet in einen festen Rahmen mit Prüfsumme. Das EEA uniTFT quittiert dieses Paket mit dem Zeichen <ACK> (=06) bei erfolgreichem Empfang oder <NAK> (=15) bei fehlerhafter Prüfsumme oder Empfangspufferüberlauf. In jedem Fall wird bei <NAK> das komplette Paket verworfen und muss nochmal gesendet werden.

Hinweis: <ACK> muß eingelesen werden. Empfängt der Hostrechner keine Quittierung, so ist mindestens ein Byte verloren gegangen. In diesem Fall muss die eingestellte Timeoutzeit abgewartet werden, bevor das Paket komplett wiederholt wird. Die Anzahl (len) der Rohdaten pro Paket kann max. 2042Byte betragen. Befehle die grösser als 2042Byte (z.B. File schreiben #FWD ...) müssen auf mehrere Pakete aufgeteilt werden. Alle Daten in den Paketen werden nach korrektem Empfang vom EA uniTFT wieder zusammengefügt.

Aus Kompatibilitätsgründen wurde das aus der EA eDIP-Serie bekannte SmallProtokoll ebenso implementiert. Wir empfehlen aus Performancegründen bei größeren Datenmengen die Nutzung des Short Protokolls.



SHORT PROTOKOLL BEFEHLE

Befehl	Display	Byte										
		0	1	2	3	4	5	6	7	8	9	10
Befehle / Daten zum Display senden	Empfang	DC3 ¹	len ²	data	crc16 ³					
	Send	ACK										
Inhalt des Sendebuffers anfordern	Empfang	DC4	1	0	'S'	crc16						
	Send	ACK										
	Send	DC3	len	data	crc16					
Letztes Datenpaket wiederholen	Empfang	DC4	1	0	'R'	crc16						
	Send	ACK										
	Send	DC3	len	data	crc16					
Protokolleinstellungen	Empfang	DC4	5	0	'D'	packet size sendbuffer ⁴		timeout ⁵ 10 ms		crc16		
	Send	ACK										
Pufferinformationen anfordern	Empfang	DC4	1	0	'I'	crc16						
	Send	ACK										
	Send	DC4	4	0	Senbuffer bytes ready		Receivebuffer bytes free		crc16			
Protokollinformationen anfordern	Empfang	DC4	1	0	'P'	crc16						
	Send	ACK										
	Send	DC4	6	0	max. Packet size		akt. Packetsize		timeout 10 ms		crc16	
RS485 Adressierung	Empfang	DC4	3	0	'A'	Select / Deselect	adr	crc16				
	Send	ACK										
RS485 direction enable delay	Empfang	DC4	2	0	'T'	Verzögerung 10 us ⁶						
	Send	ACK										
Schnittstelle anfordern / freigeben	Empfang	DC4	2	0	'G'	0=Freigabe 1=Anfordern	crc16					
	Send	ACK										
	Send	DC4	1	0	aktiv ₇	crc16						
Breakkommand	Empfang	DC4	2	0	'C'	mask ⁸	crc16					
	Send	ACK										
Hardware Reset / Reboot	Empfang	DC4	2	0	'B'	option ⁹	crc16					

¹DC3 = 19d = 0x13; DC4 = 20d = 0x14

²len: Anzahl der Nutzendaten, ohne Prüfsumme, ohne DC3/4. Besteht aus 2 Byte, low Byte wird als erstes Übertragen

³crc: Checksumme über alle übertragenen Bytes, inklusive DC3/4 und Längenbytes. CRC16-CCITT mit Startwert 0xFFFF. Low Byte wird als erstes Übertragen

⁴Maximale Puffergröße ist 2042

⁵Wartezeit bis Protokollpaket verworfen wird. Die Standardeinstellung ist 200, also 2 Sekunden.

⁶Wartezeit bis das EA uniTFT den RS485-Richtungspin ändert. Standardwert ist 0, also sofort.

⁷aktiv: 0=ALLE, 1=RS232, 2=SPI, 3=I²C, 4=USB

⁸mask: 1=wait- command (#XXW) wird unterbrochen, 2=aktuelle Makros, 4=Sendbuffer löschen, 8=R-Receivebuffer löschen. Die einzelnen Bits können kombiniert werden.

⁹

option: 1=Testmode, 2=Disable PowerOnMacro, 3=Disable defaults 4=Bootmenü

SMALL PROTOKOLL BEFEHLE

Befehl	Display	Byte					
		0	1	2	3	4	5
Befehle / Daten zum Display senden	Empfang	DC1 ¹	len ²	data	bcc ³
	Send	ACK					
Inhalt des Sendebuffers anfordern	Empfang	DC2	1	'S'	bcc		
	Send	ACK					
	Send	DC1	len	data	bcc
Letztes Datenpaket wiederholen	Empfang	DC2	1	'R'	bcc		
	Send	ACK					
	Send	DC1	len	data	bcc
Protokolleinstellungen	Empfang	DC2	3	'D'	packet size send-buffer ⁴	timeout ⁵ 10 ms	bcc
	Send	ACK					
Pufferinformationen anfordern	Empfang	DC2	1	'I'	bcc		
	Send	ACK					
	Send	DC2	2	Senbuffer bytes ready	Receivebuffer bytes free	bcc	
Protokollinformationen anfordern	Empfang	DC2	1	'P'	bcc		
	Send	ACK					
	Send	DC2	3	max. Packet size	akt. Packetsize	timeout 10 ms	bcc
RS485 Adressierung	Empfang	DC2	3	'A'	Select / Deselect	adr	bcc
	Send	ACK					
RS485 direction enable delay	Empfang	DC4	2	'T'	Verzögerung 10 us ⁶		
	Send	ACK					
Schnittstelle anfordern / freigeben	Empfang	DC2	1	'G'	0=Freigabe 1=Anfordern		
	Send	ACK					
	Send	DC2	1	aktiv ⁷	bcc		
Breakkommand	Empfang	DC2	1	'C'	mask ⁸		
	Send	ACK					
Hardware Reset / Reboot	Empfang	DC2	1	'B'	option ⁹	bcc	

¹DC3 = 19d = 0x13; DC4 = 20d = 0x14

²len: Anzahl der Nutzdaten, ohne Prüfsumme, ohne DC3/4. Besteht aus 2 Byte, low Byte wird als erstes Übertragen

³bcc: Summe aller Bytes inklusive DC1/2 und Länge (Modulo 256)

⁴Maximale Puffergröße ist 2042

⁵Wartezeit bis Protokollpaket verworfen wird. Die Standardeinstellung ist 200, also 2 Sekunden.

⁶Wartezeit bis das EA uniTFT den RS485-Richtungspin ändert. Standardwert ist 0, also sofort.

⁷aktiv: 0=ALLE, 1=RS232, 2=SPI, 3=I²C, 4=USB

⁸mask: 1=wait-command (#XXW) wird unterbrochen, 2=aktuelle Makros, 4=Sendebuffer löschen, 8=R-Receivebuffer löschen. Die einzelnen Bits können kombiniert werden.

⁹option: 1=Testmode, 2=Disable PowerOnMacro, 3=Disable defaults 4=Bootmenü

PRÜFSUMMENBERECHNUNG

Im Folgenden zwei Beispielfunktionen wie eine Prüfsummenberechnung für das Short bzw. Small Protokoll ausgeführt werden können

SHORT PROTOKOLL

Für die Berechnung der Prüfsumme wird eine zyklische Redundanzprüfung (CRC) eingesetzt. Eine gängige und bekannte CRC-Prüfung ist die CRC-CCITT. Als Startwert wird 0xFFFF verwendet. Im folgenden sehen Sie eine typische C-Implementierung. Die Funktionen müssen extern aufgerufen werden. Die Prüfsumme muss mit dem Startwert vorbelegt werden.

```
//-----  
-  
//Funktion: buffer2crc16()  
//input: ptr Datum, ptr auf CRC, blocklänge  
//output: ---  
//Beschr: CRC-CCITT eines Speicherbereichs  
//-----  
-  
void buffer2crc16(UBYTE *dat, UINT16 *pCRC, UINT32 anz)  
{  
    while(anz--)  
        crc16(*dat++, pCRC);  
}  
  
//-----  
-  
//Funktion: sp_crc16()  
//input: Datum, ptr auf CRC  
//output: ---  
//Beschr: 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 PROTOKOLL

Für die Berechnung der Prüfsumme wird bei dieser Protokollart eine einfache Summenprüfung durchgeführt.

```
//-----  
-  
//Funktion: buffer2bcc()  
//input: ptr Datum, blocklänge  
//output: ---  
//Beschr: Byte bcc für einen Speicherbereich  
//-----  
-  
UBYTE buffer2bcc(UBYTE *dat, UBYTE anz)  
{  
    UBYTE bcc = 0;  
    while(anz--)  
        bcc += *dat++;  
  
    return bcc;  
}
```

BEFEHLSÜBERSICHT

Das Modul wird über eine Reihe von Grafikbefehlen gesteuert. Sie erlauben den Aufbau eines Bildschirms sowohl als Makro, als auch über die seriellen Schnittstellen.

In den folgenden Tabellen wird jeder Befehl mit den notwendigen und optionalen Parametern erklärt.

TERMINAL

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Fenster und Schriftart definieren	#YDW PosX(old Pos), PosY(old Pos), Anchor(7), Columns(scrW()/Font width), Rows(scrH()/Font height), Font-nr(2)	Definiert das Terminalfenster durch die Schriftart Font-nr (1=8x8;2=8x16), Spaltenanzahl Columns , Reihenanzahl Rows , Position PosX , PosY und den Anker Anchor . Nach dem Reset ist die Position in der linken unteren Ecke (0,0), der Anker steht auf 7. Wird bei der Neudefinition des Fensters die Position nicht mit angegeben, bleibt diese erhalten. Beispiel siehe S. 121
Farbe einstellen	#YDC Text-RGB, Text-Opacity(100), Background-RGB(0x00000000), Background-Opacity(0)	Definiert die Farbe des Terminalfensters bezüglich Farbe und Transparenz von Text Text-RGB (z. B: \$FF0000 für Rot), Text-Opacity und Hintergrund Background-RGB , Background-Opacity . Beispiel siehe S. 121
Terminal Ebene	#YDL Layer	Auswahl der Terminal Ebene Layer (0= Hinten; 1= Vorne)
Terminal an/aus	#YDO On/Off, Visibility (=On/Off)	Der Parameter On/Off (0=aus; 1=ein) bestimmt ob das Terminalfenster an- oder ausgeschaltet ist. Im angeschalteten Zustand wird die Ausgabe gespeichert, kann allerdings durch Visibility (0= unsichtbar; 1= sichtbar) ausgeblendet sowie bei Bedarf wieder eingeblendet werden.
Cursor blinken an/aus	#YCB Cursor	Blinken des Cursors bestimmen: Cursor 0=aus; 1=ein
Cursor positionieren	#YCP Column, Row(keine Änderung)	Positioniert den Cursor des Terminalfensters anhand der Spalte Column und der Zeile Row .
Cursorposition sichern	#YCS	Speichert die aktuelle Cursorposition.
Cursorposition wiederherstellen	#YCR	Stellt die gespeicherte Cursorposition wieder her.

Bezeichnung	Befehlscode	Beschreibung
String ausgeben	#YPA 'String'	Gibt die Zeichenkette ' String ' im Terminalfenster aus. Dieser Befehl kann z.B. für die Ausgabe der Stringregister genutzt werden.
Terminal löschen	0x12(FF, Formfeed)	Löscht das Terminal und setzt den Cursor in die obere linke Ecke.
Uhrzeit/Datum ausgeben	#YPD DateFormat(#WDF), Date32(actual time)	Gibt die Uhrzeit im Terminalfenster aus. Die Anzeige kann durch das Datumsformat DateFormat geändert sowie deren Inhalt durch die Änderung der Zeit und des Datums Date32 überschrieben werden. Informationen zu DateFormat gibt es hier .
String formatiert ausgeben	#YPF Formatstring, Value1...	Ausgabe einer formatierten Zeichenkette. Der Formatstring gibt die Zeichenkette an. Die Werte werden anschließend durch (mehrere) Value(s) angegeben. Informationen zum Formatstring gibt es hier .
Konfiguration ausgeben	#YPI	Ausgabe der Displayparameter, wie z.B. Breite und Höhe sowie Interfaceinstellungen im Terminal.
Versionsstring ausgeben	#YPV	Gibt die aktuelle Version des Displays aus.

BILDER / VEKTORGRAFIKEN

Das Modul arbeitet intern mit einem speziellen Bildformat (*.epg). Die Umwandlung muss extern geschehen. Das Windowsprogramm EA uniSketch bietet die komfortabelste Möglichkeit die meisten Bildformate umwandeln zu lassen. Manche [Befehle](#) ermöglichen es, den Bildschirminhalt auf die SD-Karte zu speichern oder Bild-daten direkt über die serielle Schnittstelle zu übertragen. Hier stehen verschiedene [Bildformate](#) zur Verfügung.

BILDGRÖßE

Wird die Breite **Width** ODER die Höhe **Height** mit 0 angegeben, dann wird diese Größe proportional angepasst.

Werden die Breite **Width** UND die Höhe **Height** mit 0 angegeben, dann wird das Bild in originaler Größe(Pixel) erzeugt.

BILD/ VEKTORGRAFIK EINFÜGEN

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Bild platzieren	#PPP Object-id, 'Picturename', PosX, PosY, Anchor(1), Width(0), Height(0), Angle (0)	Definiert als Objekt Object-id die Bilddatei ' Picturename ' und platziert es in Abhängigkeit der Position PosX , PosY , des Ankers Anchor (0...9), der Bildbreite Width (Pixel), der Bildhöhe Height (Pixel) und des Winkel Angle (0...360). Die Groß- und Kleinschreibung der Bilddatei muss beachtet werden. Beispiel siehe S. 110
Bildanimation ändern	#PPA Object-id, AnimationType (0), Time, Bild-nr	Ändert im Objekt Object-id die vorliegende Animation durch die Wahl des neuen Animationstypen AnimationType . Die Ablaufzeit Time (hs) und das Endbild Bild-nr sind nur für den Animationstypen AnimationType = 7 relevant. Hierbei wird die Zeit definiert, in welcher bis zur gesetzten Bildnummer animiert wird. Die Voraussetzungen für Animationen sind hier beschrieben. Beispiel siehe S. 110
Videoinput anzeigen	#PVP Object-id, PosX, PosY,Anchor(1), Width(0), Height(0), Angle(0)	Definiert das Bild des Videoinputs als Object-id und platziert es in Abhängigkeit der Position PosX , PosY , des Ankers Anchor (0...9), der Bildbreite Width (Pixel), der Bildhöhe Height (Pixel) und des Winkels Angle (0...360). Beispiel siehe S. 110

ANIMATIONSTYPEN

Animationen beziehen sich auf Objekte vom Ursprungstyp **.GIF** oder animierte Verläufe.

Bezeichnung	Parameter	Beschreibung
Stop	0	Definiert einen Animationsstop.
Zyklisch	1	Definiert eine zyklische Animation.
Zyklisch rückwärts	2	Definiert eine zyklische Animation in entgegengesetzte Richtung.
Pendeln	3	Definiert eine pendelnde Animation.
Pendeln rückwärts	4	Definiert eine pendelnde Animation in entgegengesetzte Richtung.
Einmalig	5	Definiert eine einmalige Animation.
Einmalig rückwärts	6	Definiert eine einmalige Animation in entgegengesetzter Richtung.
Einmalig fragmentarisch	7	Definiert ein Animationsausschnitt.

STYLESHEETS

Allgemeine Informationen zu den Styles befinden sich [hier](#).

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

FARBVERLÄUFE UND LINIENMUSTER

Bezeichnung	Befehlscode	Beschreibung
Farbverlauf definieren	#CCR Ramp-nr, Offset, RGB, Opacity, alle Parameter wiederholen	Definiert einen Verlauf Ramp-nr (1...255) mit den Farben RGB (z.B: \$FF0000 für Rot) und dazugehöriger Transparenz Opacity (0...100). Der Offset Offset (0...100) bestimmt prozentual, an welchem Punkt sich die nächste Farbe befindet. Es können maximal 10 Stützpunkte in einem Verlauf definiert werden. Beispiel siehe S. 117
Farbverlauf animieren	#CAC Ramp-nr, AnimationType(1), Time (100)	Animiert den Farbverlauf Ramp-nr mit dem Animationstypen AnimationType (1...7) über die Zeit Time (hs). Animationstypen
Linienmuster definieren	#CDP Dash-nr, Dashphase, Dash, Space, Dash, Space ...	Definiert ein Linienmuster Dash-nr (1...10) in Abhängigkeit von der Strichlänge Dash (Pixel) und Freiraumlänge Space (Pixel). Der Offset Dashphase (0...100) bestimmt prozentual den Startpunkt, ab dem das Linienmuster gezeichnet wird. Es können maximal 10 Kombinationen für Linienlänge und Freiraum in einem Linienmuster definiert werden. Beispiel siehe S. 117
Linienmuster animieren	#CAD Dash-nr, AnimationType(1), Time(100)	Animiert das Linienmuster Dash-nr mit dem Animationstypen AnimationType (1...7) über die Zeit Time (hs). Animationstypen

DRAWSTYLE

Bezeichnung	Befehlscode	Beschreibung
Keine Füllung	#CFD Drawstyle-nr	Definiert einen Drawstyle Drawstyle-nr ohne Füllung. Die Füllung ist damit transparent.

Bezeichnung	Befehlscode	Beschreibung
Füllfarbe definieren	#CFC Drawstyle-nr, RGB, Opacity(100)	Definiert einen Drawstyle Drawstyle-nr mit einer einfachen Farbfüllung RGB (z.B: \$FF0000 für Rot) sowie deren Transparenz Opacity (0...100). Beispiel siehe S. 118
Linearer Farbverlauf	#CFL Drawstyle-nr, Ramp-nr, Angle(0)	Definiert einen Drawstyle Drawstyle-nr mit linearem Farbverlauf Ramp-nr und dessen Winkel Angle (0...360). Zur Verwendung dieses Befehls muss zuvor ein Farbverlauf definiert worden sein. Beispiel siehe S. 118
Radialer Farbverlauf	#CFR Drawstyle-nr, Ramp-nr, FocusX (5000), FocusY(0)	Definiert einen Drawstyle Drawstyle-nr mit radialem Farbverlauf Ramp-nr . Die Fokusposition FocusX, Y (0...100) definiert prozentual den Ausgangspunkt des Verlaufs. Wird der Fokuspunkt FocusX mit einer Pixelzahl größer 5000 angegeben, dann kann FocusY als Anker gesetzt werden. Zur Verwendung dieses Befehls muss zuvor ein Farbverlauf definiert worden sein. Beispiel siehe S. 118
Konischer Farbverlauf	#CFK Drawstyle-nr, Ramp-nr, FocusX (5000), FocusY(0), Direction	Definiert einen Drawstyle Drawstyle-nr mit konischem Farbverlauf Ramp-nr . Die Fokusposition FocusX, Y (0...100) definiert prozentual den Ausgangspunkt des Verlaufs. Die Laufrichtung Direction (0= gegen Uhrzeigersinn; 1= Uhrzeigersinn) bestimmt die Verlaufsrichtung. Wird der Fokuspunkt FocusX mit Pixelzahl größer 5000 angegeben, dann kann FocusY als Anker gesetzt werden. Zur Verwendung dieses Befehls muss zuvor ein Farbverlauf definiert worden sein. Beispiel siehe S. 118
Füllung mit Muster	#CFP Drawstyle-nr, 'Patternname', Size(0), Angle(0), Repeat/Reflekt(0), FocusX (5000), FocusY(0), AnchorP(5)	Definiert einen Drawstyle-nr mit einer Musterung ' Patternname ' in Abhängigkeit der Größe Size (proportional), des Winkels Angle (0...360) und der Darstellung Repeat/Reflekt (Repeat= 0; Reflekt= 1). Die Fokusposition FocusX, Y (0...100) definiert prozentual den Ausgangspunkt des Verlaufs. Wird der Fokuspunkt FocusX mit Pixelzahl größer 5000 angegeben, dann kann FocusY als Anker Anchor gesetzt werden. Der Musteranker AnchorP (0...9) setzt den Bezugspunkt des Musters. Beispiel siehe S. 119

Bezeichnung	Befehlscode	Beschreibung
Farbverlaufswinkel ändern	#CFA Drawstyle-nr, Angle	Ändert den Farbverlaufswinkel Angle des Drawstyles Drawstyle-nr .
Farbverlauf ändern	#CFG Drawstyle-nr, Ramp-nr,	Ändert den Farbverlauf Ramp-nr des Drawstyles Drawstyle-nr . Zur Verwendung dieses Befehls muss zuvor ein Farbverlauf definiert worden sein.
Füllfokus ändern	#CFF Drawstyle-nr, FocusX, FocusY, AnchorP(No Change)	Ändert die Fokusposition FocusX,Y des Drawstyles Drawstyle-nr sowie den Musteranker AnchorP bei einer Füllung mit Muster.
Muster ändern	#CFN Drawstyle-nr, 'Patternname'	Ändert die Musterung ' Patternname ' des Drawstyles Drawstyle-nr . Zur Verwendung dieses Befehls muss zuvor eine Musterung definiert worden sein.
Muster Größe ändern	#CFS Drawstyle-nr, Size	Ändert proportional die Größe Size der Musterung des Drawstyles Drawstyle-nr .
Muster Darstellung ändern	#CFT Drawstyle-nr, Repeat/Reflect	Ändert die Darstellung Repeat/Reflect des Füllmusters des Drawstyles Drawstyle-nr .

LINIENSTYLE

Bezeichnung	Befehlscode	Beschreibung
Keine Linie definieren	#CLD Drawstyle-nr	Definiert einen Drawstyle Drawstyle-nr ohne Linienstyle. Die Linie ist damit unsichtbar.
Linienstyle definieren	#CLS Drawstyle-nr, RGB, Opacity(100), Width(1), Joint(0), Dash-nr(0)	Definiert einen Drawstyle Drawstyle-nr mit einer Umrandung in Abhängigkeit von deren Farbe RGB (z.B: \$FF0000 für Rot), Transparenz Opcaity (0..100), Breite Width (Pixel), Abrundung Joint (eckig = 0; rund = 1) und Liniemuster Dash-nr . Zur Verwendung des Liniemusters muss dieses zuvor definiert worden sein. Beispiel siehe S. 119
Linienfarbe ändern	#CLC Drawstyle-nr, RGB, Opacity(No Change)	Ändert die Farbe RGB (z.B: \$FF0000 für Rot) und Transparenz Opacity (0..100) des Drawstyles Drawstyle-nr . Zur Verwendung des Befehls muss zuvor eine Farbe definiert wurden sein.
Linienbreite ändern	#CLW Drawstyle-nr, Width	Ändert die Breite Width der Linie des Drawstyles Drawstyle-nr .
Linienende ändern	#CLE Drawstyle-nr, Joint	Ändert die Linienabrundung Joint (eckig = 0; rund = 1) der Linie des Drawstyles Drawstyle-nr . Beispiel siehe S. 119

Bezeichnung	Befehlscode	Beschreibung
Linienmuster ändern	#CLP Drawstyle-nr, Dash-nr	Ändert das Linienmuster Dash-nr des Drawstyles Drawstyle-nr . Zur Verwendung des Linienmusters muss dieses zuvor definiert worden sein.

TEXTSTYLE

Bezeichnung	Befehlscode	Beschreibung
Textstil definieren	#CTF Textstyle-nr, 'Fontname', Size(20 *.evf / 0 *.epf), Align(0), Drawstyle-nr(1), Italic(0), SpaceL(0), SpaceC(0)	Definiert einen Textstyle Textstyle-nr mit einer Schriftart ' Fontname ' in Abhängigkeit von deren Größe Size , Ausrichtung Align (0=links; 1=zentriert; 2=rechts), Drawstyle Drawstyle-nr , Kursivwinkel Italic (-45°...45°), Zeilenabstand SpaceL (Pixel) und Zeichenabstand SpaceC (Pixel). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Aus Performancegründen wird empfohlen auf eine Outline (#CLD) zu verzichten. Beispiel siehe S. 119
Font ändern	#CTN Textstyle-nr, 'Fontname'	Ändert die Schriftart ' Fontname ' des Textstyles Textstyle-nr .
Textgröße ändern	#CTS Textstyle-nr, Size	Ändert die Größe Size des Textstyles Textstyle-nr .
Textausrichtung ändern	#CTA Textstyle-nr, Align	Ändert die Ausrichtung Align (0=links; 1=zentriert; 2=rechts) des Textstyles Textstyle-nr .
Textzeichensti ändern	#CTC Textstyle-nr, Drawstyle-nr	Ändert den Drawstyle Drawstyle-nr des Textstyles Textstyle-nr .
Textneigung ändern	#CTI Textstyle-nr, Italic	Ändert den Kursivwinkel Italic des Textstyles Textstyle-nr .
Textabstand ändern	#CTG Textstyle-nr, SpaceL, SpaceC(No Change)	Ändert den Zeilenabstand SpaceL und Zeichenabstand SpaceC des Textstyles Textstyle-nr . Es sind auch negative Abstände möglich.

TOUCHBUTTONSTYLE

Bezeichnung	Befehlscode	Beschreibung
Touchbutton Bild	#CBP Buttonstyle-nr, 'Buttonname'normal, 'Buttonname'down(normal), SizeX(0), SizeY(0)	Definiert einen Buttonstyle Buttonstyle-nr mit einem Bild für ungedrückten Zustand ' Buttonname '(normal) und gedrückten Zustand ' Buttonname '(down) in Abhängigkeit der Bildgröße SizeX , SizeY (Pixel). Beispiel siehe S. 120

Bezeichnung	Befehlscode	Beschreibung
Touchbutton Zeichenstil	#CBD Buttonstyle-nr, 'Drawstyle-nr'normal, 'Drawstyle-nr'down(normal), Width, Height, Radius	Definiert einen Buttonstyle Buttonstyle-nr mit einem Drawstyle für ungedrückten Zustand Drawstyle-nr (normal), Drawstyle für gedrückten Zustand Drawstyle-nr (down), Buttonbreite Width (Pixel), Buttonhöhe Height (Pixel) und Eckenradius Radius (Pixel). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 120
Touchbutton Textstil	#CBT Buttonstyle-nr, 'Textstyle-nr'normal, 'Textstyle-nr'down, OffsetX, OffsetY	Definiert einen Buttonstyle Buttonstyle-nr mit Textstyle für ungedrückten Zustand Textstyle-nr (normal), Textstyle für gedrückten Zustand Textstyle-nr (down) und einer Textverschiebung OffsetX , OffsetY (Pixel).
Proportion Touchbutton gedrückt ändern	#CBO Buttonstyle-nr, OffsetX(0), OffsetY(OffsetX), Size(0), Angle(0)	Definiert einen Buttonstyle Buttonstyle-nr für den gedrückten Zustand mit einer Verschiebung OffsetX , OffsetY (Pixel), Größenänderung Size (proportional) und Winkeländerung Angle (0...360). Beispiel siehe S. 120
Touchbutton deaktiviert	#CBG Buttonstyle-nr, DisableR(-30), DisableG(DisableR), DisableB(DisableR)	Definiert einen deaktivierten/ausgegrauten Buttonstyle Buttonstyle-nr in Abhängigkeit einer Subtraktion der Farbanteile Rot DisableR (-100...0), Grün DisableG (-100...0)und Blau DisableB (-100...0).
Touchbutton Sound	#CBS Buttonstyle-nr, 'Soundname'(alten löschen)	Definiert eine Sound im Buttonstyle Buttonstyle-nr mit der Audiodatei ' Soundname '. Wird keine Audiodatei angegeben, dann wird der alte Sound gelöscht. Wird für Buttonstyle eine 0 eingegeben, dann ist der Standardsound gewählt.

ZEICHNEN / GRAFISCHE PRIMITIVE

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Rechteck	#GRR Object-id, Drawstyle-nr, PosX, PosY, Anchor, Width, Height(same as Width), Radius(0), BorderWidth(0), Angle(0)	Definiert als Objekt Object-id ein Rechteck in Abhängigkeit des Drawstyles Drawstyle-nr , der Position PosX , PosY , des Ankers Anchor (0...9), der Breite Width (Pixel), der Höhe Height (Pixel), des Radius für die Ecken Radius (Pixel), der Randbreite BorderWidth (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 124
Polylinie zeichnen	#GPL Object-id, Drawstyle-nr, PosX, PosY, PosX2, PosY2, PosX3, PosY3, ...	Definiert als Objekt Object-id eine Polylinie in Abhängigkeit des Drawstyles Drawstyle-nr und den entsprechenden Positionspunkten PosX , PosY . Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 124
Polygon(gefüllt)	#GPF Object-id, Drawstyle-nr, PosX, PosY, PosX2, PosY2, PosX3, PosY3, ..., autoclose polygon	Definiert als Objekt Object-id ein gefülltes Polygon in Abhängigkeit des Drawstyles Drawstyle-nr und den Positionspunkten PosX , PosY . Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Das Polygon wird nach dem letzten Positionswert automatisch abgeschlossen. Beispiel siehe S. 124
Polylinie/Polygon erweitern	#GPA Object-id, PosX, PosY, PosX2, PosY2, ...	Fügt zusätzliche Positionen PosX , PosY zum Objekt Object-id von #GPF und #GPL hinzu. Beispiel siehe S. 124
Polypfad Segment	#GPP Object-id, Drawstyle-nr, PosX, PosY, Segment1, Segment2, ...	Definiert als Objekt Object-id ein gefülltes Polygon in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY und den einzelnen Segmenten Segment . Diese ermöglichen auch runde Formen. Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 125

Bezeichnung	Befehlscode	Beschreibung
Polygon(geometrisch)	#GGP Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius, NumberCorners, BorderWidth(0), Angle(0)	Definiert als Objekt Object-id ein Polygon in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des Radius Radius (Pixel), der Anzahl der Ecken NumberCorners , der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 125
Polygon(Stern)	#GGS Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2, CountPoints, BorderWidth(0), Angle(0)	Definiert als Objekt Object-id ein sternförmiges Polygon in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des äußeren Radius Radius1 (Pixel), des inneren Radius Radius2 (Pixel), der Anzahl an Spitzen CountPoints , der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 125
Kreis/Ellipse	#GET Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), BorderWidth(0), Angle(0)	Definiert als Objekt Object-id ein kreis- bzw. ellipsenförmiges Polygon in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des äußeren Radius Radius1 (Pixel), des inneren Radius Radius2 , der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 126
Kreisbogen	#GEA Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), StartAngle(0), StopAngle(360), BorderWidth(0), Angle(0)	Definiert als Objekt Object-id ein gebogenes Polygon in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des äußeren Radius Radius1 (Pixel), des inneren Radius Radius2 , des Startwinkels StartAngle (0...360), des Endwinkels StopAngle (0...360), der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 126

Bezeichnung	Befehlscode	Beschreibung
Gerader Schnitt	#GES Object-id, Drawstyle-nr, PosX, PosY, Acnhor, Radius 1, Radius2(Radius1), StartAngle(0), StopAngle(360), Angle (0)	Definiert als Objekt Object-id einen geschnittenen Kreis in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des Längenradius Radius1 (Pixel), des Höhenradius Radius2 , des Startwinkels StartAngle (0...360), des Endwinkels StopAngle (0...360), der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 126
Ausschnitt	#GEP Object-id, Drawstyle-nr, PosX, PosY, Anchor, Radius1, Radius2(Radius1), StartAngle(0), StopAngle(360), Angle (0)	Definiert als Objekt Object-id ein Polygonausschnitt in Abhängigkeit des Drawstyles Drawstyle-nr , den Positionspunkten PosX , PosY , des Ankers Anchor (0...9), des äußeren Radius Radius1 (Pixel), des inneren Radius Radius2 , des Startwinkels StartAngle (0...360), des Endwinkels StopAngle (0...360), der Randbreite Borderwith (0...100) und des Winkels Angle (0...360). Zur Verwendung des Befehls muss zuvor ein Drawstyle definiert worden sein. Beispiel siehe S. 126

SEGEMENTTYPEN

Allgemeine Informationen zu den Segmenten befinden sich [hier](#).

Bezeichnung	Parameter	Beschreibung
Horizontale Linie	?H, x	Definiert die Position x für einen horizontalen Linienpunkt. Beispiel siehe S. 127
Vertikale Linie	?V, y	Definiert die Position y für einen vertikalen Linienpunkt. Beispiel siehe S. 127
Freie Linie	?L, x, y	Definiert die Position x, y für nächsten Linienpunkt. Beispiel siehe S. 127
Kreisbogen	?Ct, r, x, y	Definiert die Position x, y für den nächsten Punkt des Kreisbogens sowie dessen Radius r (Pixel). Beispiel siehe S. 128
Ellipsenbogen	?Et, rx, ry, Angle, x, y	Definiert die Position x, y für den nächsten Punkt der Ellipse sowie deren Radien rx, ry (Pixel) und Winkel Angle . Beispiel siehe S. 128
Quadratische Bezierkurve	?Q, c1x, c1y, x, y	Definiert die Positionen c1x, c1y sowie x, y und für die quadratische Bezierkurve. Beispiel siehe S. 128
Leichte Quadr. Bezierkurve	?R, x, y	Definiert die Position x, y für eine Ergänzung zur quadratische Bezierkurve. Spiegelt c1x und c1y des vorherigen Segments, d. h. dieser Befehl eignet sich nicht für nur ein Segment. Beispiel siehe S. 128
Kubische Bezierkurve	?S, c1x, c1y, c2x, c2y, x, y	Definiert die Positionen c1x, c1y und c2x, c2y sowie x, y , für die kubische Bezierkurve. Beispiel siehe S. 129
Leichte Kub. Bezierkurve	?T, c2x, c2y, x, y	Definiert die Positionen c2x, c2y und x, y für eine Ergänzung zur kubischen Bezierkurve. Spiegelt c1x und c1y des vorherigen Segments, d. h. dieser Befehl eignet sich nicht für ein einzelnes Segment. Beispiel siehe S. 129
Pfad schließen	?Z	Definiert das Pfadende, d. h. der nächste Punkt ist der Startpunkt des ersten Segments. Beispiel siehe S. 129
Sprung (Neuer Subpfad)	?M x, y	Definiert einen neuen Startpunkt für das darauffolgende Segment. Beispiel siehe S. 129

STRINGS UND ZEICHENKETTENBEFEHLE

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Zeichenkette platzieren	#SSP Object-id, Textstyle-nr, PosX, PosY, Anchor, 'String'	Definiert als Objekt Object-id einen 'String' und platziert diesen in Abhängigkeit des Textstyles Textstyle-nr , der Position PosX , PosY und des Ankers Anchor . Beispiel siehe S. 115
Zeichenkette ändern	#SSC Object-id, String	Ändert den im Objekt Object-id zugeteilten String in einen neuen String 'String' .
Formatierte Zeichenkette platzieren	#SFP Object-id, Textstyle-nr, PosX, PosY, Anchor, 'Formatstring'; Value1...	Definiert als Objekt Object-id einen 'Formatstring' und platziert diesen in Abhängigkeit des Textstyles Textstyle-nr , der Position PosX , PosY und des Ankers Anchor . Der Inhalt des Strings wird unter anderem durch die darauffolgenden Werte Value bestimmt. Beispiel siehe S. 115
Formatierte Zeichenkette ändern	#SFC Object-id, Value1...	Ändert den Wert Value des formatierten Strings vom Objekt Object-id .
Kalkulationsstring platzieren (autoupdate)	#SAP Object-id, Textstyle-nr, PosX, PosY, Anchor, 'Formatstring'; Value1(Calculation), Value2	Definiert als Objekt Object-id einen 'Formatstring' und platziert diesen in Abhängigkeit des Textstyles Textstyle-nr , der Position PosX , PosY und des Ankers Anchor . Der Inhalt des Strings wird unter anderem durch die darauffolgenden kalkulierten Werte Value bestimmt. Bei einer Veränderung der Kalkulation des ersten Wertes Value1 wird die Ausgabe erneuert. Beispiel siehe S. 115
Autoupdate des Kalkulationsstrings ändern	#SAC Object-id, (ChangeCalculation)	Aktualisiert die Update-Kalkulation ChangeCalculation für die automatische Ausgabe im Objekt Object-id . Die Werteberechnung des Kalkulationsstrings wird dabei nicht geändert. Die geänderte Kalkulation hat immer den Datentyp Integer. Beispiel siehe S. 115

Bezeichnung	Befehlscode	Beschreibung
Datum-/Zeitstring platzieren	#SDP Object-id, Textstyle-nr, PosX, PosY, Anchor, DateFormat (Date), Date32 (actual time)	Definiert als Objekt Object-id einen String und platziert diesen in Abhängigkeit des Textstyles Textstyle-nr , der Position PosX , PosY und des Ankers Anchor . Der Inhalt des Strings wird durch die Wahl des Datumformates DateFormat(#WDF) , der aktuelle Zeit sowie des aktuellen Datums Date32 bestimmt. Die Zeit wird automatisch aktualisiert. Der letzte optionale Parameter Date32 kann weggelassen werden, falls nicht die aktuelle Zeit ausgegeben werden soll.
Datum-/Zeitstring ändern	#SDC Object-id, Date32(actual time)	Ändert die Zeit- bzw. Datumsangabe Date32 des Zeitstrings Object-id .

EDITBOX

Die Editbox ist eine editierbare Box für Strings. Anwendung findet diese z. B. in Kombination mit dem Keyboard. Der Defaultstring ist die erste sichtbare Ausgabe der Editbox und liegt vor, sobald die entsprechende Funktion aufgerufen wird.

Bezeichnung	Befehlscode	Beschreibung
String Editbox platzieren	#SEP Object-id, Drawstyle-nr, PosX, PosY, Anchor Width, Height, Radius, Textstyle, OffsetX, OffsetY	Definiert als Objekt Object-id eine editierbare Stringbox und platziert diese in Abhängigkeit des Drawstyles Drawstyle-nr , der Position PosX , PosY und des Ankers Anchor . Breite Width , Höhe Height , Rundung der Ecken Radius , Textstyle Textstyle Abstand vom seitlichen Rand OffsetX und Offset in Y Richtung OffsetY müssen ebenfalls für die Erstellung angegeben werden. Beispiel siehe S. 115
Defaultstring an Editbox senden	#SED Object-id, Strings...	Definiert als Objekt Object-id einen Defaultstring String und sendet diesen an die Editbox.
Code/String an Editbox senden	#SEC Object-id, Codes/Strings...	Definiert als Objekt Object-id Codes oder Strings Code/String und sendet diese an die Editbox.

Bezeichnung	Befehlscode	Bschreibung
Keyboard mit Editbox verknüpfen	#SEK Keyboard-id, Object-id...	Verknüpft eine Tastatur Keyboard-id mit einer oder mehreren Editboxen Object-id . Dadurch kann mit einer Tastatur in mehreren Editboxen gearbeitet werden. Beispiel siehe S. 115
Editbox für Keyboard aktivieren	#SEA Object-id	Aktiviert eine Editbox Object-id für die damit verknüpfte Tastatur. Dadurch kann eine der zur Tastatur zugewiesenen Editboxen zur Bearbeitung aktiviert werden. Es ist jeweils nur eine Editbox zum aktiv. Object-id= 0 deaktiviert alle Editboxen. Beispiel siehe S. 115

TOUCHOBJEKTE / TOUCHFUNKTIONEN

Bei Touchbereichen stehen zwei Arten der Betätigung zur Auswahl:

Touch Tasten (**B**) = Tastend, sie springen in den nicht gedrückten Zustand sobald die Betätigung beendet ist.

Touch Schalter (**S**) = Schaltend, bei jeder Betätigung wird der Zustand einmal gewechselt

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

DEFINITION VON TOUCHOBJEKTEN

Bezeichnung	Befehlscode	Beschreibung
Rechteck als Touchbereich	#TBR Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, #TSR Anchor(5), Width(Buttonstyle), Height (Buttonstyle)	Der rechteckige Touchbereich Object-id wird mit dem Buttonstyle Buttonstyle-nr , den beiden Texten für die Zustände ungedrückt 'Text'(normal) und gedrückten Zustand Text(down) gezeichnet. Die Breite Width(Pixel) und Höhe Height(Pixel) wird aus dem Buttonstyle übernommen, außer es ist eine andere Größe gewünscht. Zur Verwendung dieses Befehls muss zuvor ein Buttonstyle definiert worden sein. Beispiel siehe S. 122
Ellipse/Kreis als Touchbereich	#TBE Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, #TSE Anchor (5), Width(Buttonstyle), Height (Buttonstyle)	Der elliptische Touchbereich Object-id wird mit dem Buttonstyle Buttonstyle-nr , den beiden Texten für die Zustände ungedrückt 'Text'(normal) und gedrückten Zustand Text(down) gezeichnet. Die Breite Width(Pixel) und Höhe Height(Pixel) wird aus dem Buttonstyle übernommen, außer es ist eine andere Größe gewünscht. Zur Verwendung dieses Befehls muss zuvor ein Buttonstyle definiert worden sein. Beispiel siehe S. 122
Bild als Touch Taste/Schalter definieren	#TBP Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down), PosX, PosY, #TSP Anchor (5), SizeX(Buttonstyle), SizeY (Buttonstyle)	Ein Bild wird als Touchobjekt Object-id verwendet. Die Eigenschaften werden im Buttonstyles Buttonstyle-nr hinterlegt. Der Text für ungedrückten Zustand 'Text'(normal) und für gedrückten Zustand Text(down) sowie die Positionierung PosX , PosY müssen mit angegeben werden. Der Anker Anchor , die Breite SizeX(Pixel) und die Höhe SizeY(Pixel) können auch aus dem Buttonstyle übernommen werden. Zur Verwendung dieses Befehls muss zuvor ein Buttonstyle definiert worden sein. Der Befehl #CBP wird vorausgesetzt. Beispiel siehe S. 122

Bezeichnung	Befehlscode	Beschreibung
Objekt als Touch Taste/Schalter umdefinieren	#TBO Object-id, Buttonstyle-nr, 'Text'(normal), 'Text'(down) #TSO	Wandelt ein beliebiges bestehendes Objekt Object-id in ein Touch Element um. Aus dem Buttonstyles Buttonstyle-nr werden alle zusätzlich benötigten Informationen für die Erstellung eines Touchbereichs entnommen. Der Text für ungedrückten Zustand 'Text' (normal) und der Text für gedrückten Zustand Text (down) werde ebenfalls mit angegeben. Zur Verwendung dieses Befehls muss zuvor ein Buttonstyle definiert worden sein.

TOUCHFUNKTIONEN

Bezeichnung	Befehlscode	Beschreibung
Touchstatus ändern	#TCS Touchstate, Object-id ...	Ändert von einem oder mehreren Touchobjekten Object-id den Touch-Zustand Touchstate (0= gedrückt; 1= ungedrückt; 2= deaktiviert).
Touch Beschriftung ändern	#TCL Object-id, 'Text'(normal), 'Text'(down)	Ändert vom Touchobjekt Object-id den Touch Text Text (normal/down).
Disable/Enable Touch	#TCE State, Object-id ...	Aktiviert(State = 1) oder deaktiviert(State = 0) ein oder mehrere Touchobjekte Object-id .
Touch Antworten	#TCR Signal, Filter, Object-id ...	Bestimmung des Verhaltens von einem oder mehreren Touchobjekten Object-id . Signale Signal welche Aktionen ein Senden bzw. Makro auslösen können sind: 1= ungedrückt; 2= gedrückt; 3= gedrückt und ungedrückt; 4 = ziehen; 7 = gedrückt, ungedrückt und ziehen Der Filter Filter bietet die Möglichkeit das Senden je nach Makrodefinition zu blockieren: 0= nur senden wenn kein Makro definiert ist; 1 = immer senden
Objekt eine Touchreaktion zuweisen	#TID Mask, Object-id ...	Weist einem oder mehreren Objekten Object-id eine Aktion Mask (1= intern; 2= bewegen; 4= göße, 8= rotieren) zu. Mask=1 ist für spezielle Touchobjekte wie z.B. Bargraphen oder Zeigerinstrumente vorgesehen. Die Aktion geschieht intern im Objekt. Mask= 2 bedeutet, dass sich das Objekt durch den Bediener auf dem Bildschirm verschieben lässt, Mask= 4 lässt eine Größenänderung des Objekts zu, Mask= 8 definiert ein Rotieren um die eigene Achse.

Bezeichnung	Befehlscode	Beschreibung
Status Touch Schalter abfragen	#TQS Object-id ...	Sendet den Status eines oder mehrerer Touchobjekte Object-id . Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#TQS)
Radiogruppe erstellen	#TRA Group-id, Object-id...	Speichert in einer Gruppe Group-id mehrere Touch Schalter Object-id . Informationen zu Gruppen befinden sich hier.
Aktiven Schalter der Radio-Gruppe abfragen	#TQR Group-id ...	Sendet den momentan aktiven Touch Schalter einer oder mehrerer Radio-Gruppen Group-id . Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#TQR)

BARGRAPHEN UND INSTRUMENTE

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Bargraph Rechteck	#IBR Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Width, Height, Radius(0), Startvalue(0), Endvalue(100), Course(1), Angle(0)	Definiert einen rechteckigen Bargraphen als Objekt Object-id in Abhängigkeit des Drawstyles für den Vordergrund Drawstyle-nr Front , des Drawstyles für den Hintergrund Drawstyle-Back , der Position PosX, PosY , des Ankers Anchor(0...9) , der Breite des Bargraphen Width(Pixel) , der Höhe des Bargraphen Height(Pixel) , des Radius zum Abrunden der Ecken Radius , des Startwertes Startvalue , des Endwertes Endvalue , der Laufrichtung Course (0=rechts → links; 1=links → rechts) und des Winkels Angle(0...360) . Beispiel siehe S. 112
Bargraph Dreieck	#IBT Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Width, Height, Tip(0), Startvalue(0), Endvalue(100), Course(1), Angle(0)	Definiert einen dreieckigen Bargraphen als Objekt Object-id in Abhängigkeit des Drawstyles für den Vordergrund Drawstyle-Front , des Drawstyles für den Hintergrund Drawstyle-Back , der Position PosX, PosY , des Ankers Anchor(0...9) , der Breite des Bargraphen Width(Pixel) , der Höhe des Bargraphen Height(Pixel) , der Position für die Spitze Tip(0=Bot; 1=Top; 2=Mid) , des Startwertes Startvalue , des Endwertes Endvalue , der Laufrichtung Course(0=groß→klein; 1=klein→groß) und des Winkels Angle(0...360) . Beispiel siehe S. 112

Bezeichnung	Befehlscode	Beschreibung
Bargraph Bogen	#IBA Object-id, Drawstyle-Front, Drawstyle-Back, PosX, PosY, Anchor, Radius, Borderwidth, Startangle, Stopangle, Startvalue(0), Endvalue(100), Direction(1)	Definiert einen gebogenen Bargraphen als Objekt Object-id in Abhängigkeit des Drawstyles für den Vordergrund Drawstyle-Front , des Drawstyles für den Hintergrund Drawstyle-Back , der Position PosX , PosY , des Ankers Anchor (0...9), des Radius zur Krümmung des Bargraphen Radius , der Bargraphenhöhe als Randbreite des definierten Kreises Borderwith (Pixel), des Startwinkels Startangle (0...360), des Endwinkels Stopangle (0...360), des Startwertes Startvalue , des Endwertes Endvalue und der Laufrichtung Direction (0=gegen Uhrzeigersinn; 1=Uhrzeigersinn). Beispiel siehe S. 112
Instrumentbild platzieren	#IPP Object-id, "Instrumentname", PosX, PosY, Anchor, Width(0), Height(0), Startvalue(0), Endvalue(100), Angle(0)	Definiert die Instrumentendatei " Instrumentname " als Object-id in Abhängigkeit der Position PosX , PosY , des Ankers Anchor (0...9), der Breite des Instruments Width (Pixel), der Höhe des Instruments Height (Pixel), des Startwertes Startvalue , des Endwertes Endvalue und des Winkels Angle (0...360). Beispiel siehe S. 113
Drehinstrument definieren	#IGM Group-id, Indicator-id, Startangle, DeltaAngle, Startvalue(0), Endvalue(100)	Definiert eine Objektgruppe Group-id als Drehinstrument. Zur Verwendung müssen Drehobjekt Indicator-id , Startwinkel Startangle und Drehwinkel DeltaAngle bestimmt werden. Startwert Startvalue und Endwert Endvalue bestimmen den Wertebereich. Beispiel siehe S. 112
Slider definieren	#IGS Group-id, Path-id, Slide-id, Turn(0), Startvalue(0), Endvalue(100)	Definiert eine Objektgruppe Group-id als Slider. Zur Verwendung müssen Pfad des Sliders Path-id und Slide-Objekt Slide-id bestimmt werden. Verdrehung bei Pfadumlauf Turn is eine optional Angabe. Startwert Startvalue und Endwert Endvalue bestimmen den Wertebereich. Beispiel siehe S. 113
Wert setzen	#IVS Object-id, Current, Time(0), ActionCurve(0)	Setzt das Instrument Object-id auf einen Wert Current . Die Animation der Werteeinstellung wird durch die Dauer Time (hs) und die Aktionskurve ActionCurve (1...10) bestimmt.

Bezeichnung	Befehlscode	Beschreibung
Wert automatisch ändern	#IVA Object-id, BarCalculation, Time(0), ActionCurve(0)	Definiert im Instrument oder Bargraphen Object-id einen Wert als Kalkulation BarCalculation . Die Animation der Wert-einstellung wird durch die Dauer Time (hs) und die Aktionskurve ActionCurve (1...10) bestimmt. Beispiel siehe S. 113
Berechnung aktualisieren	#IVC Object-id, ChangeCalculation(bar calculation)	Stößt im Instrument oder Bargraphen Object-id eine Neuberechnung ChangeCalculation an. Die Kalkulation ist immer vom Datentyp Integer.

OBJEKTVERWALTUNG

Allgemeine Informationen zu den Objekten befinden sich [hier](#).

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

DEFINITION VON OBJEKTEN

Bezeichnung	Befehlscode	Beschreibung
Objekt sichtbar	#OVV Object-id, Object-id...	Macht ein oder mehrere Objekte Object-id sichtbar. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Objekt unsichtbar	#OVI Object-id, Object-id...	Macht ein oder mehrere Objekte Object-id unsichtbar. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Freien Anker(0) setzen Display	#OAS Anchor0X, Anchor0Y, Object-id, Object-id...	Der freie Anker Anchor = 0 von einem oder mehreren Objekten Object-id wird durch die Angabe seiner Position Anchor0X , Anchor0Y absolut auf dem Display gesetzt.
Freien Anker(0) setzen relativ	#OAO Anchor0X, Anchor0Y, Object-id, Object-id...	Der freie Anker Anchor = 0 von einem oder mehreren Objekten Object-id wird durch die Angabe seiner Position Anchor0X , Anchor0Y relativ zum Objekt gesetzt. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Aktiven Anker setzen	#OAA Anchor, Object-id, Object-id...	Bestimmt die aktiven Anker Anchor (0-9) von einem oder mehreren Objekten. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

Bezeichnung	Befehlscode	Beschreibung
Objektgruppe erstellen	#OGA Group-id, Object-id, Object-id...	Definiert eine Objektgruppe Group-id , bestehend aus mehreren Objekten Object-id . Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's. Informationen zu Gruppen befinden sich hier .
Objektrahmen zeichnen	#OFP Drawstyle-nr, addL/R, addT/B, Object-id, Object-id...	Erzeugt einen Rahmen um ein Objekt Object-id in Abhängigkeit des Drawstyles Drawstyle-nr und der Addition von Pixeln zur Größenänderung des Rahmens links/rechts addL/R sowie oben/unten addT/B . Ist die Object-id = 0 , wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Objekt löschen	#ODI Object-id, Object-id...	Löscht ein oder mehrere Objekte Object-id . Beträgt die Eingabe für das Objekt Object-id = 0 , werden alle Objekte gelöscht. Da nur Objekte betroffen sind, bleiben alle Styles bestehen. Ist die Object-id = 0 , wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

ÄNDERN VON OBJEKTEN

Bezeichnung	Befehlscode	Beschreibung
Position ändern absolut	#OPA PosX, PosY, Object-id, Object-id...	Absolute Positionsänderung PosX, PosY von einem oder mehreren Objekten Object-id . Ist die Object-id = 0 , wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Position ändern relativ	#OPR PosX, PosY, Object-id, Object-id...	Relative Positonsänderung PosX, PosY von einem oder mehreren Objekten Object-id . Ist die Object-id = 0 , wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

Bezeichnung	Befehlscode	Beschreibung
Größe ändern absolut	#OSA Width, Height, Object-id, Object-id...	Ändert die Breite Width und Höhe Height von einem oder mehreren Objekten Object-id absolut. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Größe ändern relativ	#OSR Width, Height, Object-id, Object-id...	Ändert die Breite Width und Höhe Height von einem oder mehreren Objekten Object-id relativ. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Scherung ändern absolut	#OHA ShearX, ShearY, Object-id, Object-id...	Ändert die Scherung ShearX , ShearY (Grad) von einem oder mehreren Objekten Object-id absolut. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Scherung ändern relativ	#OHR ShearX, ShearY, Object-id, Object-id...	Ändert die Scherung ShearX , ShearY (Grad) von einem oder mehreren Objekten Object-id relativ. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Drehung ändern absolut	#ORA Angle, Object-id, Object-id...	Ändert den Winkel Angle von einem oder mehreren Objekten Object-id absolut. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Drehung ändern relativ	#ORR Angle, Object-id, Object-id...	Ändert den Winkel Angle von einem oder mehreren Objekten Object-id relativ. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

Bezeichnung	Befehlscode	Beschreibung
Transparenz ändern absolut	#OOA Opacity, Object-id, Object-id...	Ändert die Transparenz Opacity von einem oder mehreren Objekten Object-id absolut. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Transparenz ändern relativ	#OOR Opacity, Object-id, Object-id...	Ändert die Transparenz Opacity von einem oder mehreren Objekten Object-id relativ. Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Draw-/Text-/Buttonstyle ändern	#OCS Style-nr, Object-id, Object-id...	Ändert den verwendeten Style Style-nr (Drawstyle-nr, Textstyle-nr oder Buttonstyle-nr) von einem oder mehreren Objekten Object-id . Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.
Farbe ändern	#OCC AddR, AddG, AddB, Object-id, Object-id...	Ändert die Farbe RGB (z.B: \$FF0000 für Rot) von einem oder mehreren Objekten Object-id durch Addition der Farbanteile Rot AddR (-100...100), Grün AddG (-100...100) und Blau AddB (-100...100). Ist die Object-id = 0, wird der Befehl auf alle Objekte angewendet. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

Bezeichnung	Befehlscode	Beschreibung
Ebenen ändern absolut	#OLA Nr, Object-id, Object-id...	<p>Ändert absolut die Objektebene Nr von einem oder mehreren Objekten Object-id. Die angegebenen Objekte werden nacheinander der ausgewählten Objektebene zugeordnet, d. h. das letzte Objekt befindet sich auf der gewünschten Objektebene und die anderen Objekte darunter. Ist unter der gewählten Objektebene "kein Platz" mehr, werden die anderen Objekte darüber platziert. Ebene 1 ist dabei die unterste Ebene. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.</p> <p>Beispiel:</p> <p>#OLA 1, 1, 2, 3 Objekt 3 befindet sich auf Objektebene 1. Objekt 2 auf Ebene 2 und Objekt 1 auf Ebene 3. Da Ebene 1 die unterste Ebene ist und darunter nichts platziert werden kann, werden die Objekte 1 und 2 auf den darüberliegenden Ebenen platziert.</p> <p>#OLA 2, 1, 2, 3 Objekt 3 befindet sich auf Objektebene 2. Objekt 2 auf Ebene 3 und Objekt 1 auf Ebene 1. Da unter Ebene 2 nur eine weitere Ebene existiert, wird Objekt 1 unter Ebene 2 und Objekt 2 darüber platziert.</p> <p>#OLA 99, 1, 2, 3 Objekt 3 befindet sich auf Objektebene 99. Objekt 2 auf Ebene 98 und Objekt 1 auf Ebene 97.</p>

Bezeichnung	Befehlscode	Beschreibung
Ebenen ändern relativ	#OLR Difference, Object-id, Object-id...	Ändert relativ die Objektebene von einem oder mehreren Objekten Object-id . Durch Die Addition oder Subtraktion der gewünschten Ebenenzahl Difference wird die Objektebene für das ausgewählte Objekt um diesen Wert geändert. Demnach sind auch negative Zahlen möglich. Positive Zahlen setzen das Objekt auf eine höhere und negative Zahlen auf eine niedrigere Ebene. Es kann eine Objektreihe ("ID1 - ID10") angegeben werden. Die Reihenfolge der Bearbeitung richtet sich nach der Angabe der beiden Objekt-ID's.

VARIABLEN / REGISTER

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Stringdatei laden	#VFL 'Stringfilename'	Lädt die Stringdatei ' Stringfilename '. Es können maximal 8 Stringdateien mit insgesamt maximal 1000 Strings geladen werden. Unter ' Stringfile ' sind Details über deren Verwendung zu finden
Stringdatei löschen	#VFD 'Stringfilename'(Alle Stringdateien löschen)	Löscht die ausgewählte bzw. alle Stringdateien ' Stringfilename '.
Stringdatei Strings zählen	#VFC	Gibt die Anzahl der Zeichenketten aller geladenen Stringdateien aus. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#VFC)
Stringregister setzen	#VSS String-id, 'String', String(id + 1), String(id + 2)	Speichert einen String ' String ' im Stringregister String-id . Weitere Eingaben von Strings werden in den darauffolgenden Registern gespeichert. Die maximale Länge beträgt 255 Zeichen.
Stringregister ab Position setzen	#VSP String-id, pos, 'String'	Speichert einen String ' String ' im Stringregister String-id ab der Position pos . So können mehrere Strings in einem zusammengefasst werden.
Stringregister Date/Time	#VSD String-id, DateFormat(#WDF), Date32(datetime())	Schreibt anhand des Datumsformats DateFormat im Stringregister String-id die Uhrzeit und das Datum Date32 .
Stringregister Formatstring	#VSF String-id, Formatstring (printf), Value1, Value2...	Speichert eine formatierte Zeichenkette ' Formatstring ' im Stringregister String-id unter der Angabe seiner Werte Value . Unter ' Formatierte Strings ' sind Details über die Fromatangaben zu finden.
Stringregister von Objekt kopieren	#VSO String-id, Object-id, Objekt(id+1), Objekt(id+2)	Schreibt einen mit dem Objekt Object-id verknüpften String in den Stringregister String-id .

Bezeichnung	Befehlscode	Beschreibung
Stringregister lesen(ASCII)	#VSA String-id, String-id	Liest das Stringregister String-id und gibt dieses über die Schnittstelle als ASCII-Code aus. Das Auslesen mehrerer Register ist möglich. Sind mehrere Stringregister in einer Reihenfolge kann mit dem Zeichen '-' ein Bereich definiert werden. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#VSA)
Stringregister lesen (Unicode)	#VSU String-id, String-id	Liest das Stringregister String-id und gibt dieses über die Schnittstelle als Unicode aus. Das Auslesen mehrerer Register ist möglich. Sind mehrere Stringregister in einer Reihenfolge kann mit dem Zeichen '-' ein Bereich definiert werden. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#VSU)
Register setzen (Integer)	#VRI Register-id, Integer, Integer(id + 1)	Speichert einen Integerwert, der auch eine Kalkulation sein kann Integer im Register Register-id . Weitere Angaben von Integerwerten werden in den darauffolgenden Registern gespeichert.
Register setzen (Float)	#VRF Register-id, Float, Float(id + 1)	Speichert eine Gleitkommazahl oder Kalkulation Float im Register Register-id . Weitere Angaben von Gleitkommazahlen werden in den darauffolgenden Registern gespeichert.
Register lesen Int/Float	#VRG Register-id, Register-id, Register-id	Liest das Register von Integerwerten oder Gleitkommazahlen Register-id und gibt diesen über die Schnittstelle aus. Das Auslesen mehrerer Register ist möglich. Sind mehrere Register in einer Reihenfolge kann mit dem Zeichen '-' ein Bereich definiert werden. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#VRG)

MAKROS

Allgemeine Informationen zu Makros befinden sich hier.

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

AUSFÜHRUNG VON MAKROS

Bezeichnung	Befehlscode	Beschreibung
Makro ausführen	#MRN "Makroname"	Das Makro " Makroname "(Dateiname) wird ausgeführt.
Makro mit Bedingung ausführen	#MRC Condition, "Makroname" true; "Makroname" false(Kein Makro)	Das Makro " Makroname "(true) wird ausgeführt, wenn die Bedingung Condition erfüllt ist. Andernfalls wird ' Makroname ' (false) ausgeführt.
Portmakro ausführen	#MRP Port	Das Portmakro für den Port Port (0-15) wird ausgeführt.
Bitmakro ausführen	#MRB Bit, Edge	Das Bitmakro für das Bit Bit (0-127) mit der Flanke Edge (0= fallend; 1= steigend) wird ausgeführt.
Touchmakro ausführen	#MRT Object-id, TouchState	Das Touchmakro, welches durch die Touchfunktion TouchState (1= nicht gedrückt; 2= gedrückt; 4= ziehen) im Touchobjekt Object-id definiert ist, wird ausgeführt.
Makrofile vorzeitig beenden	#MFE Condition(wahr)	Beendet das Makro vorzeitig, wenn die Bedingung Condition erfüllt ist.
Makrofile Zeilen überspringen	#MFS Condition(wahr), Lines(1)	Überspringt Zeilen Lines im Makro, wenn die Bedingung Condition erfüllt ist.
Makrofile Marke setzen	#MFM Marker-nr(0)	Setzt im Makrofile eine Markierung Marker-nr (0-9).
Makrofile zu Marke springen	#MFJ Condition, Marker-nr(0)	Springt zur vorher definierten Markierung Marker-nr (0-9), wenn die Bedingung Condition erfüllt ist.
Makro Definitionen löschen	#MCD Mask	Alle Makrodefinitionen löschen. Durch Mask wird festgelegt welche Gruppe von Makros gelöscht wird (1=Second; 2= Prozess; 4= Port; 8= Bit; 16= Analog; 32= Touch; 64= Aktion). Die Addition der einzelnen Flags in Mask ergibt ein kombiniertes Löschen der jeweiligen Makrogruppen. So werden mit der Eingabe von Mask =3 Sekunden- und Prozessmakros gelöscht.

DEFINITION VON MAKROPROZESSEN UND TOUCHMAKROS

Bezeichnung	Befehlscode	Beschreibung
Makro Prozess	#MPD Prozess-nr, Time, "Makroname"; , Startnr(Keine), Endnr(Startnr), Type(1)	Definiert ein Prozessmakro mit der Prozessnummer Prozess-nr (1-10). Das Makro " Makroname " wird im Abstand einer Zeit Time (hs) aufgerufen. Optional können mehrere Makros ausgeführt werden. Hierfür muss der " Makroname " mit mindestens einer Ziffer enden. Das Start- Startnr und Endmakro Endnr werden als optionale Parameter übergeben und mit einem AnimationsTypen Type (1-6) versehen. Werden zum Beispiel Makroname= Photo, Startnr= 1, Time= 100 und Endnr= 6 verwendet, dann wird im Sekundentakt "Photo1"→"Photo2"→...→Photo6 aufgerufen.
Konditional Makro	#MPC Prozess-nr, Conditiontime, Condition, "Makroname"; , Startnr(Keine), Endnr(Startnr), Type(1)	Definiert ein Prozessmakro mit der Prozessnummer Prozess-nr (1-10). Im Intervall Conditiontime (hs) wird abgefragt, ob die Bedingung Condition erfüllt ist. Ist Condition =TRUE(wahr), dann wird das Makro " Makroname " ausgeführt. Es besteht die Möglichkeit mehrere Makros aufzurufen. Dazu werden Start- und Endmakro, sprich Startnr und Endnr , angegeben und mit einem Animationstypen Type (1-6) versehen, siehe #MPD .
Automatisch wechselndes Makro	#MPA Prozess-nr, Calculationtime, Change, "Makroname"; , Startnr(Keine), Endnr(Startnr), Type(1)	Definiert ein Prozessmakro mit der Prozessnummer Prozess-nr (1-10). Das Makro " Makroname " wird nur bei Wertänderungen der Kalkulation Change aufgerufen. Die Kalkulation ist immer vom Datentyp Integer. Die Zeit Calculationtime (hs) bestimmt das Intervall der Abfrage des Kalkulationsergebnisses. Hat sich das aktuelle Ergebnis im Vergleich zum vorherigen Ergebnis geändert, dann wird das Makro " Makroname " ausgeführt. Es besteht die Möglichkeit mehrere Makros aufzurufen. Dazu werden Start- und Endmakro, sprich Startnr und Endnr , angegeben und mit einem Animationstypen Type (1-6) versehen, siehe #MPD .

Bezeichnung	Befehlscode	Beschreibung
Prozesszeit ändern	#MPT Prozess-nr, Time	Ändert die Zeit Time (hs) nach der ein Makro im Prozessmakro Prozess-nr aufgerufen wird. Zur Verwendung dieses Befehls muss bereits ein Prozessmakro definiert sein.
Port Makro definieren	#MHP Port, "Makroname"(Altes Makro löschen)	Ändert sich der Status der I/O- Pins des Ports Port (0-15) wird das Makro " Makroname " aufgerufen. Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.
Bit Makro definieren	#MHB Bit, Edge, "Makroname"(Altes Makro löschen)	Definiert für den Portpin Pin (0-127) und der Flanke Edge (0= fallend; 1= steigend) das Makro " Makroname ". Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.
Analog Makro definieren	#MHA Channel, AnalogType, "Makroname" (Altes Makro löschen)	Definiert im Kanal Channel (0-3) das Makro " Makroname ". Dabei muss der <u>Typ</u> des Analogmakros AnalogType berücksichtigt werden. Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.
RTC Sekunden Makro definieren	#MDS , "Makroname" (Altes Makro löschen)	Ruft jede Sekunde das Makro " Makroname " auf. Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.
Aktion Endmakro definieren	#MDA Object-id, "Makroname" (Altes Makro löschen)	Definiert im Objekt Object-id ein Endmakro. Das Makro " Makroname " wird erst nach Beendigung der Aktion des Objekts Object-id ausgeführt. Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.
Touch Makro definieren	#MDT Object-id/Group-id, "Makroname down" (Altes Makro löschen); "Makroname up" (Altes Makro löschen), "Makroname drag" (Altes Makro löschen)	Definiert im Objekt Object-id oder in der Objektgruppe Group-id eine Touchfunktion. Dabei werden Makros " Makroname " für gedrückten(down), nicht gedrückten(up) und gehaltenene(drag) Zustand angegeben. Das entsprechende Objekt Object-id muss dafür als Touch Objekt definiert sein. Wird der Befehl ohne " Makroname " ausgeführt, wird die derzeitige Makrodefinition gelöscht.

TYPEN FÜR ANALOGMAKROS

Es wird die eingestellte Hysterese des Analogkanals beachtet. Für Die Einstellung der Grenzen und der Hysterese ist [hier](#) beschrieben.

Bezeichnung	Parameter	Beschreibung
Änderung	0	Das Makro wird bei jeglicher Wertänderungen aufgerufen.
Verminderung	1	Das Makro wird bei Wertminderungen aufgerufen.
Steigerung	2	Das Makro wird bei Wertsteigerungen aufgerufen.
Unterschreiten untere Grenze	3	Das Makro wird beim Unterschreiten der unteren Grenze aufgerufen.
Überschreiten untere Grenze	4	Das Makro wird beim Überschreiten der unteren Grenze aufgerufen
Unterschreiten obere Grenze	5	Das Makro wird beim Unterschreiten der oberen Grenze aufgerufen.
Überschreiten obere Grenze	6	Das Makro wird beim Überschreiten der oberen Grenze aufgerufen
Verlassen des Grenzfensters	7	Das Makro wird beim Verlassen des durch die beiden Grenzen vorgegebenen Fensters aufgerufen.
Eintritt ins Grenzfenster	8	Das Makro wird beim Eintreten in das durch die beiden Grenzen vorgegebenen Fenster aufgerufen.

UHRZEIT

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Uhrzeit setzen	#WTD Hour, Min(Alte Werte), Sec(Alte Werte), Day(Alte Werte), Month(Alte Werte), Year(Alte Werte)	Setzt die Uhrzeit durch die Angabe der Zeit mittels Stunde Hour , Minute Min und Sekunde Sec sowie des Datums mittels Tag Day , Monat Month und Jahr Year . Die RTC ist mit einer Knopfzelle vom Typ 364 gepuffert.
Uhrzeit binär seriell senden	#WSB Date32(Aktuelle Zeit)	Sendet die aktuelle Uhrzeit bzw. die im Parameter angegebene Sekundenvariable Date32 . Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#WSB)
Uhrzeit Format festlegen	#WDF DateFormat	Legt die Formatausgabe der Uhrzeit und des Datums fest. Dazu muss das Datumsformat DateFormat als formatierter String (Beispiel) angegeben werden. Die Befehle zur Datumsformatierung werden hier aufgelistet.
Uhrzeit print seriell ASCII	#WSA DateFormat(#WDF), Date32(Aktuelle Zeit)	Stellt die aktuelle oder als Date32 angegebene Uhrzeit anhand des Datumsformats DateFormat als ASCII-Code in den Sendebuffer. Die Befehle zur Datumsformatierung werden hier aufgelistet. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#WSA)
Uhrzeit print seriell Unicode	#WSU DateFormat(#WDF), Date32(Aktuelle Zeit)	Stellt die aktuelle oder als Date32 angegebene Uhrzeit anhand des Datumsformats DateFormat als Unicode in den Sendebuffer. Die Befehle zur Datumsformatierung werden hier aufgelistet. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#WSU)
Monatsnamen setzen	#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)	Definiert die Monatsnamen ' Jan ' - ' Dec '. Nicht angegebene Monatsnamen bleiben unangetastet.

Bezeichnung	Befehlscode	Beschreibung
Wochentag setzen	#WDW 'Mon'(Monday), 'Tue'(Tuesday), 'Wed'(Wednesday), 'Thu'(Thursday), 'Fri'(Friday), 'Sat'(Saturday), 'Sun'(Sunday)	Definiert die Wochentagsnamen ' Mon ' - ' Sun '. Nicht angegebene Tage bleiben unangetastet.
Objekgruppe als Uhr definieren	#WGC Group-id, IndicatorH-id, IndicatorM-id, IndicatorS-id	Definiert eine Objektgruppe Group-id als Uhr. Zur Verwendung des Stunden- IndicatorH-id , Minuten- IndicatorM-id oder wahlweise Sekundenzeigers IndicatorS-id müssen Objekte aus der Gruppe diesen zugewiesen werden. Beispiel siehe S. 123

AKTION

Allgemeine Informationen zu Aktionen befinden sich [hier](#).

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

AKTIONSKURVEN UND -PFADE

Bezeichnung	Befehlscode	Beschreibung
Aktionskurve definieren	#ACD ActionCurve-Nr, X1, Y1, X2, Y2	Die Aktionskurve beschreibt den zeitlichen Verlauf einer Aktion, z.B. das Beschleunigen und Abbremsen. Definiert wird eine Aktionskurve ActionCurve-Nr (1... 10) durch die Positionen X1 , 2 (0... 100) und Y1 , 2 (-200...300). Siehe vordefinierte Aktionskurven für Informationen über Defaultkurven und genauere Erklärungen.
Aktionspfad definieren	#APD ActionPath-nr, StartX, StartY, Segment1, Segment2,...	Definiert einen Aktionspfad Actionpath (1... 10) durch die Position Xstart , Ystart und die darauffolgenden Segmente . Der Aktionspfad beschreibt den Weg auf dem die Aktion stattfinden soll.

AKTION DEFINIEREN

Bezeichnung	Befehlscode	Beschreibung
Aktion definieren	#ADC Start/End	Start/End (Finish= 0; Start= 1) Befehle, die zeitlich nach #ADC1 stehen, werden erst ausgeführt, wenn durch #ADC0 die Definition der Aktion beendet wird. Dadurch können mehrere Objekte oder Aktionen definiert werden, die gleichzeitig erzeugt werden beziehungsweise beginnen sollen.

AKTION

Bezeichnung	Befehlscode	Beschreibung
Absolute Aktion definieren	#AOA Object-id, Action, Action2, ...	Definiert eine Aktion Action für ein Objekt Object-id absolut. Siehe Aktionsdefinition für eine Erläuterung der möglichen Aktionen.

Bezeichnung	Befehlscode	Beschreibung
Relative Aktion definieren	#AOR Object-id, Action, Action2, ...	Definiert eine Aktion Action für ein Objekt Object-id relativ. Siehe Aktionsdefinition für eine Erläuterung der möglichen Aktionen.
Aktionstyp definieren	#AOT Object-id, ActionType, TotalTime(100), DelayStart(0), DelayEnd(0)	Definiert einen Aktionstyp ActionType für das zugewiesene Objekt Object-id in Abhängigkeit von der gesamten Laufzeit TotalTime (hs), der Startverzögerung DelayStart (hs) und der vorzeitigen Beendigung DelayEnd (hs). Für diesen Befehl muss zuvor eine Aktion für das Objekt Object-id bestimmt worden sein. Siehe Aktionstyp für eine Erläuterung der unterschiedlichen Aktionstypen.
Aktion stoppen	#AOS Object-id, Stop(0), Next(0)	Definiert einen Aktionsstop Stop (0= jetzt; 1= Aktionsbeginn; 2= Aktionsende) und Folgeschritt Next (0= Normal stoppen; 1= Sprung; 2= Objekt löschen) für das zugewiesene Objekt Object-id . Für diesen Befehl muss zuvor eine Aktion für das Objekt Object-id bestimmt worden sein.

AKTIONSDEFINITIONEN

Bezeichnung	Parameter	Beschreibung
Position Curve	101-110 PosX, PosY	Definiert die Position PosX , PosY . Das Objekt bewegt sich in einer Geraden zu dem angegebenen Punkt. Beispiel siehe S. 107
Position Curve Pfad	151-160 Actionpath-nr, Offset	Das Objekt folgt dem Aktionspfad Actionpath-nr beziehungsweise wird auf diesem platziert. Der Startwert Offset (0...100) bestimmt prozentual den Startpunkt auf dem Aktionspfad. Beispiel siehe S. 107
Scale Curve	201-210 ScaleX, ScaleY	Definiert die Größe ScaleX , ScaleY prozentual auf die Originalgröße des Objekts. Beispiel siehe S. 107
Scale Curve Pfad	251-260 Actionpath-nr, Offset	Das Objekt skaliert über den Aktionspfad Actionpath-nr . Der Startwert Offset (0...100) bestimmt prozentual die Startgröße auf dem Aktionspfad. Beispiel siehe S. 107
Rotation Curve	301-310 Angle	Definiert den Rotationswinkel Angle . Negative Winkel für Drehung im Uhrzeigersinn. Beispiel siehe S. 108
Roation Curve Pfad	351-360 Actionpath-nr, Offset	Das Objekt rotiert über den Aktionspfad Actionpath-nr . Der Startwert Offset (0...100) bestimmt prozentual den StartWinkel auf dem Aktionspfad. Beispiel siehe S. 108
Shear Curve	401-410 ShearX Shear Y	Definiert eine Scherung ShearX , ShearY in Grad. Beispiel siehe S. 108
Shear Curve Pfad	451-460 Actionpath-nr, Offset	Objekt schert über den Aktionspfad Actionpath-nr . Der Startwert Offset (0...100) bestimmt prozentual die Start-scherung auf dem Aktionspfad.
Opacity Curve	501-510 Opacity	Definiert die Transparenz Opacity (0...100). Beispiel siehe S. 109
Coloroffset Curve	601-610 Red, Green, Blue	Definiert eine Farbaddition der Farbanteile Rot Red (-100...100), Grün Green (-100...100) und Blau Blue (-100...100). Beispiel siehe S. 109

AKTIONSTYP

Beispiel zur Betrachtung der Aktionstypen

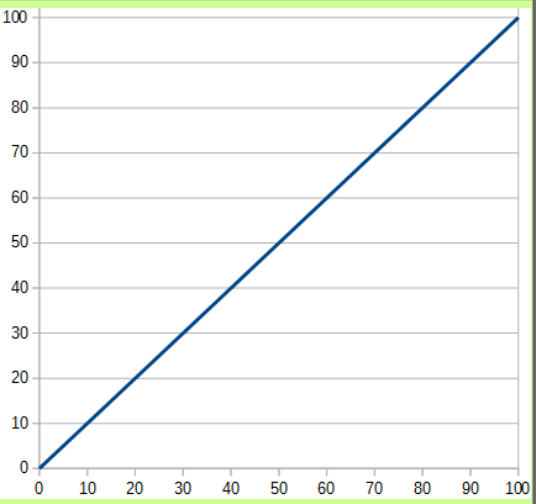
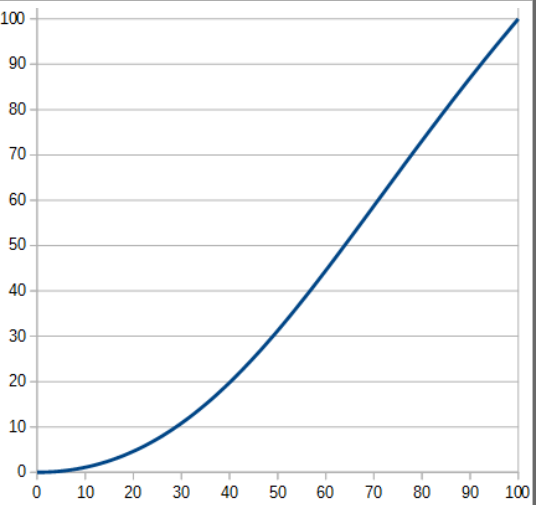
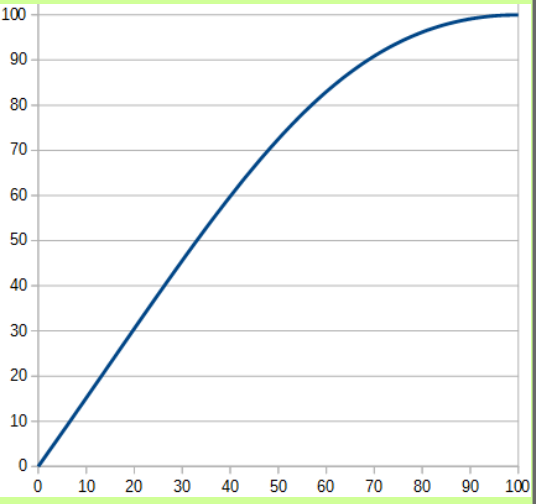
#AOT id, X, 100

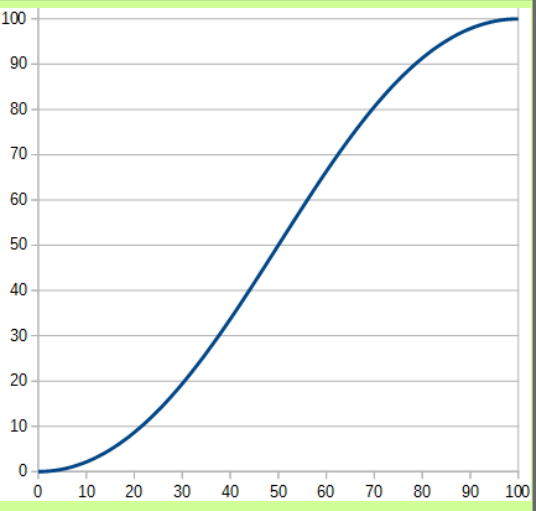
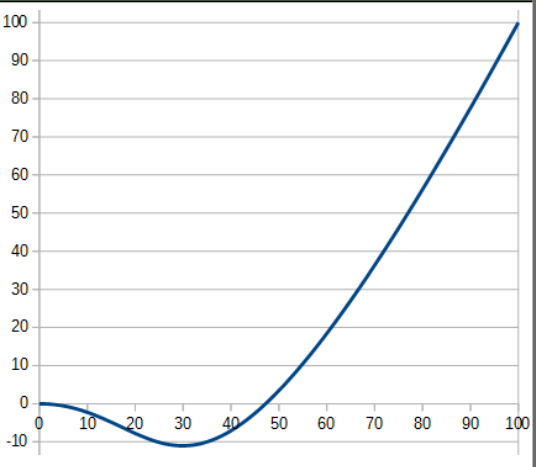
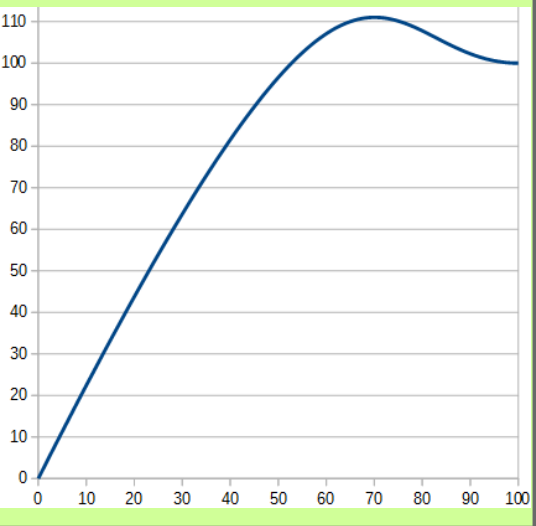
Der Parameter "X" entspricht den Parametern der Aktionstypen der nachfolgenden Tabelle.

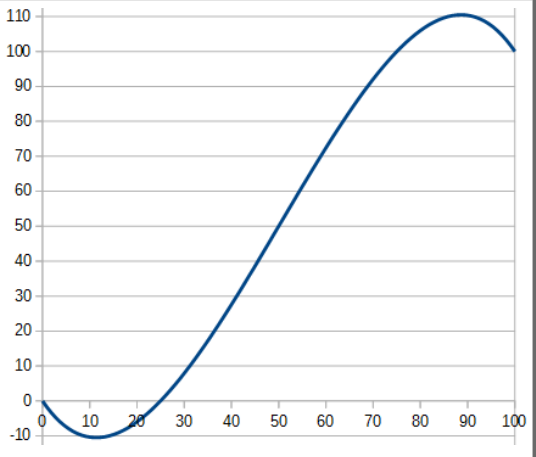
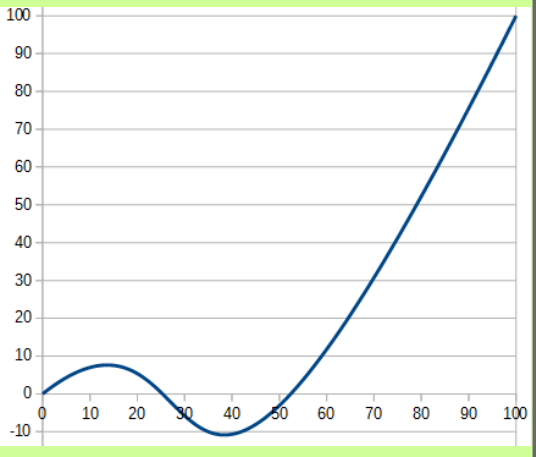
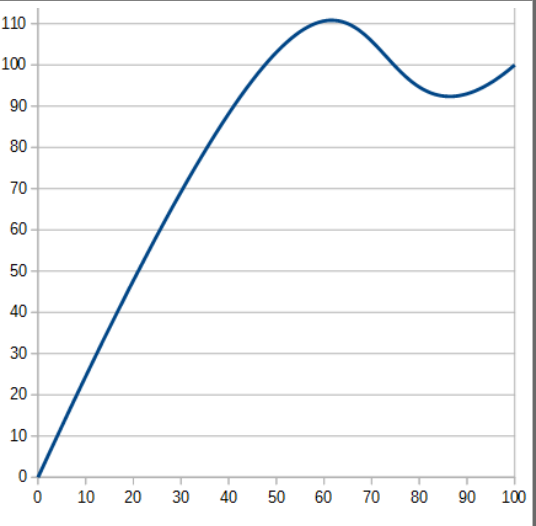
Bezeichnung	Parameter	Beschreibung
Erscheinend (appear)	1	Definiert ein erscheinendes Verhalten für die ausgewählte Aktion. Das Objekt wird an seinem Erzeugungspunkt(400,240) in Abhängigkeit der Aktionsdefinition erzeugt. Das heißt, dass es von der angegebenen Position (1000,600) unter Berücksichtigung der angegebenen Größe(25%) einfliegen wird und auf seine Originalgröße skaliert.
Verschwindend (disappear)	2	Definiert ein verschwindendes Verhalten für die ausgewählte Aktion. Der Ablauf läuft entgegengesetzt zum Erscheinen ab, sprich das Objekt wird sich von der Erzeugungsposition zur angegebenen Position mit entsprechender Größenänderung bewegen. Das Objekt wird anschließend gelöscht.
Unsichtbar (invisible)	3	Definiert ein ausblendendes Verhalten für die ausgewählte Aktion. Der Ablauf gleicht dem verschwindendem Verhalten, jedoch wird das Objekt nach der Aktion nicht gelöscht, sondern unsichtbar. Dadurch muss es für spätere Verwendungen nicht erneut definiert werden.
Ändernd (change)	4	Definiert ein einmaliges Ändern der Parameter für die ausgewählte Aktion. Der Ablauf gleicht dem verschwindendem Verhalten, jedoch wird das Objekt nach der Aktion nicht gelöscht.
Zyklisch (cyclic)	5	Definiert ein zyklisches Verhalten für die ausgewählte Aktion. Der Ablauf entspricht dem änderndem Verhalten, jedoch wird dieser stetig wiederholt.
Pendeln (ping-pong)	6	Definiert ein pendelndes Verhalten für die ausgewählte Aktion. Der Ablauf ähnelt dem zyklischen Verhalten, jedoch mit der Unterscheidung, dass das Objekt sich nicht stetig von Erzeugungspunkt zur angegebene Position bewegt, sondern stetig zwischen diesen mit der angegebenen Größenskalierung pendelt.

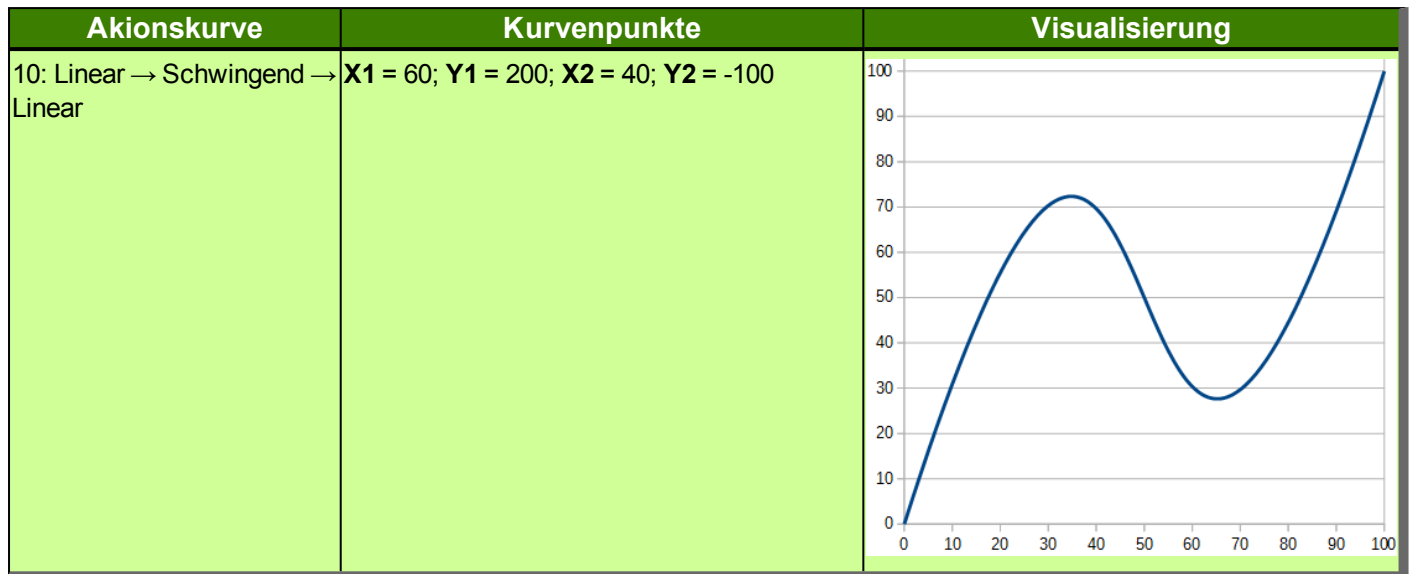
VORDEFINIERTE AKTIONSKURVEN

Bei den Aktionskurven kann der Verlauf der Aktion definiert werden. Auf der X-Achse befindet sich der zeitliche Verlauf von 0 bis 100%. Die Y-Achse gibt den zugeordneten Wert an. Es handelt sich um eine kubische Spline mit 2 Stützpunkten. Das Windowsprogramm EA uniSketch bietet Hilfe bei der Erstellung eigener Kurven.

Akionskurve	Kurvenpunkte	Visualisierung
1: Linear	X1 = 10; Y1 = 10; X2 = 90; Y2 = 90	
2: Beschleunigend → Linear	X1 = 40; Y1 = 0; X2 = 60; Y2 = 40	
3: Linear → Abbremsend	X1 = 40; Y1 = 60; X2 = 60; Y2 = 100	

Akionskurve	Kurvenpunkte	Visualisierung
4: Beschleunigend → Linear → Abbremsend	X1 = 40; Y1 = 0; X2 = 60; Y2 = 100	
5: Unterschwingend → Linear	X1 = 30; Y1 = 0; X2 = 30; Y2 = -60	
6: Linear → Überschwingend	X1 = 70; Y1 = 160; X2 = 70; Y2 = 100	

Akionskurve	Kurvenpunkte	Visualisierung
7: Unterschwingend→ Linear → Überschwingend	X1 = 30; Y1 = -60; X2 = 70; Y2 = 160	
8: Schwingend → linear	X1 = 40; Y1 = 40; X2 = 20; Y2 = -100	
9: Linear → Schwingend	X1 = 80; Y1 = 200; X2 = 60; Y2 = 60	



KEYBOARD

Zur Erzeugung von speziellen Tasten, wie z. B. *Shift*, *Backspace* oder *CAPSLOCK*, müssen im "ButtonString" entsprechende Parameter eingesetzt werden. Damit die Parameter ihrer Einsatz finden muss vor diesen ein *Backslash* '\' gesetzt werden.

Der Befehl "ButtonString" beschreibt die Reihenfolge der Tasten innerhalb einer Zeile. Um mehrere Zeilen einzugeben müssen diese mit der Eingabe von '|' oder ';' am Zeilenende getrennt werden.

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Parameter	Beschreibung
\1: (code 1)	Zeige Keyboard Nr. 1.
\2: (code 2)	Zeige Keyboard Nr. 2.
\3: (code 3)	Zeige Keyboard Nr. 3.
\4: (code 4)	Zeige Keyboard Nr. 4.
\5: (code 5)	Umschalt schaltet auf Keyboard Nr. 2. Nach Eingabe einer Taste erfolgt die automatische Rückkehr zu Keyboard Nr. 1.
\6: (code 6)	Feststell zwischen Keyboard Nr. 1 und Nr. 2 umschalten.
\7: (code 7)	Entfernen löscht das Zeichen rechts der Schreibmarke.
\8: (code 8)	Löschen / BACKSPACE löscht das Zeichen links der Schreibmarke.
\A: (code 10)	<i>In neue Zeile springen; reserviert für zukünftige Verwendung</i>
\B: (code 11)	Einfügen / Überschreiben schaltet zwischen Einfügen und Überschreiben um.
\C: (code 12)	CLEAR löscht die gesamte Eingabe.
\D: (code 13)	Enter / Return bestätigt die Eingabe und setzt die Eingabe als neuen Defaultstring (#SED).
\E: (code 14)	Pfeiltaste links
\F: (code 15)	Pfeiltaste rechts
\G: (code 16)	Pfeiltaste hoch; <i>reserviert für zukünftige Verwendung</i>
\H: (code 17)	Pfeiltaste runter; <i>reserviert für zukünftige Verwendung</i>
\I: (code 18)	Pos1 springt zum Anfang der Eingabe.
\J (code 19)	Ende springt zum Ende der Eingabe.
\L: (code 21)	Abbruch bricht die Eingabe ab und kehrt zum Defaultstring (#SED) zurück.
\N: (code 23)	Platzhalter für eine unbenutzte und noch nicht definierte Taste.
\O..\V: (code 24 .. 31)	8 x Funktionstasten

Bezeichnung	Befehlscode	Beschreibung
Tasten definieren	#KDB Group-id, Nr, "ButtonStringLine1", "ButtonStringLine2"...	Definiert als Objektgruppe Group-id die Tastenreihe der Tastatur " ButtonStringLine " in Abhängigkeit der Keyboard Nummer Nr und der Reihenfolge der Tasten. Es können weitere Tastenreihen angegeben werden. Wird die gleiche Taste mehrmals hintereinander eingegeben, dann ändert sich proportional die Breite der erzeugten Taste. Dies gilt auch für Leerzeichen. Beispiel siehe S. 114
Alternative Tastenbeschriftung definieren	#KDL Group-id, Code, "ButtonText"	Ändert in der Objektgruppe Group-id bei den speziellen Tasten Code die Beschriftung " ButtonText ". Beispiel siehe S. 114
Styles definieren	#KDC Group-id, Gap, ButtonStyle1 normal, ButtonStyle 1 special, Button Style2 normal, ButtonStyle2 special...	Definiert für die Objektgruppe Group-id den Tastenabstand Gap , den Buttonstyle für normale ButtonStyle normal und spezielle Tasten ButtonStyle special . Es können maximal fünf Buttonstyle- Paare eingegeben werden. Die Zahlen des Buttonstyles geben die Tastenlänge an, für die der Style wirkt. Beispiel siehe S. 114
Keyboard platzieren	#KPK Group-id, PosX, PosY, Anchor, Width, Height	Positioniert die Objektgruppe Group-id in Abhängigkeit der Position, PosX , PosY , des Ankers Anchor und der Breite Width . Die Höhe Height ist eine optionale Angabe. Beispiel siehe S. 114

PERIPHERIE - SCHNITTSTELLEN

Das Modul besitzt verschiedene Schnittstellen:

- 4 [Analogeingänge](#) (Genauigkeit 12 Bit)
- 1 [PWM](#) Ausgang (Genauigkeit 16 Bit)
- [Videoeingang](#) (PAL/SECAM Analog)
- [Audioausgang](#) (8 Ohm Lautsprecher max. 1 W)
- 16 digitale [I/O](#), erweiterbar auf bis zu 128 (max. 20 mA)
- 3 Serielle Schnittstellen [RS232](#), [SPI](#), [I²C](#)

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

ANALOG

Bezeichnung	Befehlscode	Beschreibung
Analogwandler kalibrieren	#HAC	Führt die Kalibrierung des Analogwandlers aus.
Analogeingang Hysterese	#HAH Channel, Hysteresis (1)	Definiert die Hysterese des Analogeingangs Channel (0-3). Durch die Angabe von Hysterese Hysteresis werden nicht alle Wertänderungen ausgegeben, sondern nur Wertänderungen ab der angegebenen Größe. Ohne die Angabe der Hysterese werden sämtliche Wertänderungen ausgegeben, sprich Hysteresis = 1.
Analogeingang Grenzen	#HAL Channel, Limit, Limit2	Definiert für den Analogeingang Channel (0-3) eine obere und untere Grenze Limit , Limit 2 .
Analogeingang lesen	#HAR Channel(0), Number(4)	Liest den Analogeingang Channel (0-3) ein. Die Anzahl Number (1-4) gibt die Zahl der zu lesenden Eingänge an. Beispiel: Channel = 1, Number = 2 führt zum Einlesen der Analogeingänge 1 und 2 und deren Ausgabe über die serielle Schnittstelle. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HAR)

PULSWEITENMODULATION (PWM)

Bezeichnung	Befehlscode	Beschreibung
PMW Ausgang Frequenz	#HFO Frequenz32Bit , OnTime(keine Änderung), TotalTime(keine Änderung)	Definiert die Frequenz Frequency (2Hz-1MHz) des PWM Ausgangs. Interne Teiler führen dazu, dass nicht jede Frequenz genau einstellbar ist. Das Verhältnis zwischen An- und Auszeit wird durch die beiden Parameter OnTime / TotalValue bestimmt. Beispiel siehe S. 111
PWM Ausgang Tastverhältnis	#HFD OnTime, TotalTime(keine Änderung)	Änder das Verhältnis zwischen An- und Auszeit OnTime / TotalValue .

PORT

Bezeichnung	Befehlscode	Beschreibung
Porteinstellung	#HPC Port, I/O, ...	Einstellung für den gesamten Port (0-15). I/O (0-0xFF) definiert dabei das jeweilige Portbit als Eingang (1) oder Ausgang (0). Eine weitere Angabe von I/O definiert den nächst höherwertigen Port . Beispiel siehe S. 111
Ausgänge setzen	#HPW Port, PortState, ...	Ausgänge des Bausteins Port in den Zustand PortState (0-255) bringen. Weitere Angaben von PortState beschreiben den nächsten Portbaustein. Beispiel siehe S. 111
Eingänge lesen	#HPR Port(0), Number(2)	Liest den Portbaustein Port (0-15) ein. Mit der Anzahl Number (1-16) können mehrere Portbausteine gelesen werden. Als Standard sind zwei Portbausteine vorhanden. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HPR)
Information Portbaustein	#HPI	Rückgabe der angeschlossenen Portbausteine. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HPI)
Portpineinstellung	#HBC Pin, Output/Input, ...	Bestimmt, ob der Portpin Pin (0-127) als Eingang oder Ausgang Output/Input (0=Ausgang; 1=Eingang) definiert wird. Eine weitere Angabe von Output/Input definiert den nächst höherwertigen Pin .

Bezeichnung	Befehlscode	Beschreibung
Portpin Ausgang setzen	#HBW Pin, BitState, ...	Beschreibt den Zustand BitState (0, 1, 2= invert) des Ausgangsportpins Pin (0-127). Eine weitere Angabe von BitState definiert den Zustand des nächst höherwertigen Pins .
Portpin Eingang lesen	#HBR Pin(0), Number(16)	Liest den Portpin Pin (0-127) ein. Mit der Anzahl Number (1-128) können mehrere Portpins gelesen werden. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HBR)

VIDEO UND AUDIO

Bezeichnung	Befehlscode	Beschreibung
Audio Sound abspielen	#HTP "Soundname"	Spielt die Sounddatei mit dem Namen Soundname ab. Es können nur *.esd-Files abgespielt werden. Diese Files werden am einfachsten mit dem Windowstool EA uniSketch erstellt.
Audio Sound stoppen	#HTS	Stoppt das Abspielen der Sounddatei.
Video Input Fenster	#HVB Left(0), Top(0), Width(720), Height (576)	Fenstereinstellung des Video Inputs mit linken Left und oberen Top Offset sowie Breite Width und Höhe Height .

RS232 MASTER SCHNITTSTELLE

Bezeichnung	Befehlscode	Beschreibung
Sende ASCII	#HRA "string"	Sendet einen Text string als ASCII-Code über die RS232 Masterschnittstelle.
Sende Unicode	#HRU "string"	Sendet einen Text string als Unicode über die RS232 Masterschnittstelle.
Sende Binary	#HRS len32Bit, Data	Sendet binäre Daten Data mit der Länge len über die RS232 Masterschnittstelle. Die Länge muss kleiner als 1kB sein.
Sende Datei	#HRF <Filename>	Sendet eine Datei <Filename> über die RS232 Masterschnittstelle.
Empfangene Daten	#HRR len32Bit (1024)	Liest Länge len der über RS232 empfangenen Daten ein. Die maximale Länge ist 1 KB. Ist die Länge größer als die verfügbaren Daten, werden nur diese gesendet. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HRR)

Bezeichnung	Befehlscode	Beschreibung																		
RS232 Parameter	#HRP Baudrate32Bit	Baudrateneinstellung Baudrate der Master RS232 Schnittstelle mit 8-N-1. Baudraten:																		
		<table border="1"> <thead> <tr> <th>Baudrate</th> <th>Fehler in %</th> </tr> </thead> <tbody> <tr> <td>9600</td> <td>+0,04</td> </tr> <tr> <td>19200</td> <td>-0,08</td> </tr> <tr> <td>38400</td> <td>+0,16</td> </tr> <tr> <td>57600</td> <td>-0,08</td> </tr> <tr> <td>115200</td> <td>+0,64</td> </tr> <tr> <td>230400</td> <td>-0,80</td> </tr> <tr> <td>460800</td> <td>+2,08</td> </tr> <tr> <td>921600</td> <td>-3,68</td> </tr> </tbody> </table>	Baudrate	Fehler 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
		Baudrate	Fehler 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 SCHNITTSTELLE

Bezeichnung	Befehlscode	Beschreibung
Sende ASCII	#HSA "string"	Sendet einen Text string als ASCII-Code über die SPI Masterschnittstelle.
Sende Unicode	#HSU "string"	Sendet einen Text string als Unicode über die SPI Masterschnittstelle.
Sende Binary	#HSS len, Data	Sendet binäre Daten Data mit der Länge len über die SPI Masterschnittstelle. Die Länge muss kleiner als 1kB sein.
Sende Datei	#HSF <Filename>	Sendet eine Datei <Filename> über die SPI Masterschnittstelle.
Empfangene Daten	#HSR len32Bit	Liest Länge len der über SPI empfangenen Daten ein. Die maximale Länge ist 1 KB. Ist die Länge größer als die Verfügbaren Daten, werden nur diese gesendet. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HSR)
Einstellung CS-Pin	#HSC State	Einstellung der CS-Leitung: State (0=low (manually); 1=high (manually); 2=a-automatic)).

Bezeichnung	Befehlscode	Beschreibung
SPI Parameter	#HSP Frequency32Bit (in Hz), SPI Mode, DataOrder	Definiert die Parameter Frequenz Frequency (15,6kHz-1MHz), Modus SPI Mode (0-3) und Datenreihenfolge Dataorder (0= MSB (Most Signiifcant Bit)) der SPI. Die Clockfrequenz stellt sich auf die Frequency ein, wobei interne Teiler dazu führen, dass die tatsächliche Frequenz immer gleich oder langsamer ist. Informationen zur SPI-Spezifiaktion befinden sich hier .

I²C

Bezeichnung	Befehlscode	Beschreibung
Sende ASCII	#HIA "string"	Sendet einen Text string als ASCII-Code über die I ² C Masterschnittstelle.
Sende Unicode	#HIU "string"	Sendet einen Text string als Unicode über die I ² C Masterschnittstelle.
Sende Binary	#HIS len32Bit, Data	Sendet binäre Daten Data mit der Länge len über die I ² C Masterschnittstelle. Die Länge muss kleiner als 1kB sein.
Sende Datei	#HIF <Filename>	Sendet eine Datei <Filename> über die I ² C Masterschnittstelle.
Empfangene Daten	#HIR len32Bit	Liest Länge len der über I ² C empfangenen Daten ein. Die maximale Länge ist 1 KB. Ist die Länge größer als die Verfügbaren Daten, werden nur diese gesendet. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#HIR)
I ² C Parameter	#HIP Adress8Bit, Frequenz32Bit	Definiert die Parameter Adresse Adress und Frequenz Frequency (3,9kHz-1MHz) für die I ² C Übertragung. Informationen zur I ² C-Spezifikation befinden sich hier .

SYSTEMKOMMANDOS

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Befehlscode	Beschreibung
Projektpfad definieren	#XPS </Absolute path>	Definiert den Projektpfad. Informationen zur Pfadangabe gibt es hier .
Projektpfad auslesen	#XPG	Schreibt den aktuellen Projektpfad in den Sendebuffer. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#XPG).
Version senden	#XIV	Schreibt den Versionsstring in den Sendebuffer. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#XIV).
Display Info	#XID	Schreibt die Displayparameter, wie z. B. Breite und Höhe in den Sendebuffer. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#XID)
SD-Card Info	#XIS	Schreibt Informationen über die SD-Karte in den Sendebuffer. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#XIS)
RAM Info	##XIR	Schreibt die RAM-Speichergöße und Auslastung in den Sendebuffer. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#XIR)
Displayausrichtung	#XCO Angle	Ändert den Winkel Angle (0, 90, 180, 270) für die Ausrichtung des Displays.
Virtuelle Displayauflösung definieren	#XCV Width(Displaywidth), Height(Displayheight), Touchwidth(Width), Touchheight(Height)	Definiert die virtuelle Displayauflösung durch Breite Width , Höhe Height , Touchbreite Touchwidth und Touchhöhe Touchheight . Dadurch ist eine automatische Größenanpassung von Programmen auf kleine und große Displays möglich.

Bezeichnung	Befehlscode	Beschreibung
Displayhelligkeit definieren	#XCB Brightness, Time, Flash	Definiert die Helligkeit Brightness (0-100) des Displays unter Einbeziehung der Änderungszeit Time (hs). Der Wert kann dauerhaft gespeichert werden: Flash 0= nicht speichern; 1= speichern.
RS232 Slave Parameter	#XCR Baudrate, RS485 Adress(no change), Flash(0)	Definiert die Parameter für den RS232 Slave: Baudrate (default = 115200) und RS485 Adress (default = 7). Der Wert kann im Flash Flash (0= nicht speichern; 1= speichern) gespeichert werden. Das Format für die Datenübertragung ist 8-N-1. Die zur Verfügung stehenden Baudraten sind hier aufgelistet. Der Wert kann dauerhaft gespeichert werden: Flash 0= nicht speichern; 1= speichern.
SPI Slave Parameter	#XCS SPI-Mode, DataOrder(no change), Flash(0)	Einstellung des SPI-Modes (0-3) (default = 3) und Bitreihenfolge DataOrder (0= MSB First; 1= LSB first) (default= 0). Informationen zum SPI-Mode befinden sich hier . Der Wert kann dauerhaft gespeichert werden: Flash 0=nicht speichern; 1=speichern.
I ² C Slave Adress	#XCI Adress8Bit, Flash(0)	I ² C Slave Adresse setzen 8-Bit Adress (default 0xDE). Der Wert kann dauerhaft gespeichert werden: Flash 0= nicht speichern; 1= speichern.
ASCII-String an Sendbuffer	#XSA 'String'	Sendet die Zeichenkette ' String ' als ASCII-String zum Sendebuffer. Dieser Befehl ist für die Ausgabe der Stringregister gedacht.
Unicode-String an Sendbuffer	#XSU 'String'	Sendet die Zeichenkette ' String ' als Unicode-String zum Sendebuffer. Dieser Befehl ist für die Ausgabe der Stringregister gedacht.
Hardcopy an Datei	#XHF <File>, PictureFormat(1), x(0), y(0), Anchor (7), width (display width), height (display height)	Screenshot im Format PictureFormat und den Abmessungen x, y, Anchor, width, height unter dem Pfad und Namen File speichern. Informationen zum Pictureformat befinden sich hier .
Hardcopy an Serial Interface	#XHS PictureFormat(1), x1(0), y1(0),Anchor (7), width (display width - 1), height (display height -1)	Screenshot im Format PictureFormat und den Abmessungen x, y, width, height in den Sendebuffer stellen. Informationen zum Pictureformat befinden sich hier .

Bezeichnung	Befehlscode	Beschreibung								
Hardcopy des Videobildes an Datei	#XVF <File>, , PictureFormat	Standbild des Videofensters im Format PictureFormat unter dem Pfad und Namen File speichern. Informationen zum Pictureformat befinden sich hier .								
Hardcopy Video an Serial Interface	#XVS PictureFormat	Standbild des Videofensters im Format PictureFormat in den Sendebuffer stellen. Informationen zum Pictureformat befinden sich hier .								
Programmablauf unterbrechen	#XXW Wait	Stoppt den Programmablauf für die Dauer der Wartezeit Wait (hs). Dieser Befehl ist für Debugzwecke geeignet. Das Modul arbeitet in dieser Zeit keine weiteren Befehle ab.								
Touch Abgleich (nur resistiv)	#XXT	Startet den Abgleich des resistiven Touches. Der Nutzer muss 2 vorgegebene Punkte auf dem Bildschirm berühren.								
Firmware reBoot (HW-Reset)	#XFB Option	Der Befehl führt einen kompletten Neustart des Moduls aus. Der Parameter Option wählt den Bootmodus aus: <table border="0"> <tr> <td>1 Testmode</td> <td>Parameter und Informationen des Moduls auf dem Display anzeigen.</td> </tr> <tr> <td>2 Disable PowerOnMacro</td> <td>Deaktiviert den Start des PowerOnMacros (start.emc).</td> </tr> <tr> <td>3 Disable Default</td> <td>Deaktiviert alle Standardmakros. Es werden keine Defaulteinstellungen geladen.</td> </tr> <tr> <td>4 Bootmenu</td> <td>Alle auf der SD-Karte befindlichen Projekte werden aufgelistet und können gestartet werden.</td> </tr> </table>	1 Testmode	Parameter und Informationen des Moduls auf dem Display anzeigen.	2 Disable PowerOnMacro	Deaktiviert den Start des PowerOnMacros (start.emc).	3 Disable Default	Deaktiviert alle Standardmakros. Es werden keine Defaulteinstellungen geladen.	4 Bootmenu	Alle auf der SD-Karte befindlichen Projekte werden aufgelistet und können gestartet werden.
1 Testmode	Parameter und Informationen des Moduls auf dem Display anzeigen.									
2 Disable PowerOnMacro	Deaktiviert den Start des PowerOnMacros (start.emc).									
3 Disable Default	Deaktiviert alle Standardmakros. Es werden keine Defaulteinstellungen geladen.									
4 Bootmenu	Alle auf der SD-Karte befindlichen Projekte werden aufgelistet und können gestartet werden.									

BOOTMENÜ UND TOUCHABGLEICH DURCH GESTEN

Zum Zeitpunkt des Neustarts können via zwei verschiedener Gesten sowohl der resistive Touchabgleich(#XXT) als auch das Bootmenü(#XFB 4) aufgerufen werden.

Dazu muss direkt beim Start des Displays(max 1 Sekunde danach) dieses mit einem Finger bis zur Bestätigung gedrückt werden. Nach einer weiteren Sekunde wird eine Nachricht eingeblendet, ob ein Touchabgleich stattfinden oder das Bootmenü aufgerufen werden soll.

Soll der Touchabgleich stattfinden, so muss der Finger kurz angehoben und dann wieder auf das Display gedrückt werden, bis die Aufforderung zum Abgleich kommt. Dies ist das Drücken zweier Punkte auf dem Display.

Soll hingegen das Bootmenü starten, so muss der Finger kurz angehoben und anschließend zwei mal in kurzem zeitlichen Abstand auf das Display gedrückt werden.

Soll nichts von beidem geschehen, so kann der Finger entweder auf dem Display verweilen oder dieser vom Display genommen werden.

Das Bootmenü ist in der nachstehenden Abbildung zu sehen.



Das Bootmenü teilt sich in drei Bereiche auf.

Links stehen die verschiedenen Projekte, welche durch Anwählen gestartet werden können..

Unten rechts können alles Makros deaktiviert, der Testmode gestartet oder der Aufruf des Bootmenüs abgebrochen werden.

Oben rechts wird per Druck die Wahl eines einmaligen(Once) oder eines permanenten(Save) Aufrufs der entsprechenden Auswahl gewählt.

FILEZUGRIFF-1234567890

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

SD-KARTE

Die Möglichkeiten der Pfadangabe sind [hier](#) aufgelistet.
Allgemeine Informationen zur SD-Karte befinden sich [hier](#).

Bezeichnung	Befehlscode	Beschreibung
Arbeitspfad setzen	#FDS <Path>	Definiert den Arbeitspfad Path . Bei relativen Pfadangaben wird von diesem Ordner ausgegangen.
Arbeitspfad zurücklesen	#FDG	Auslesen des Arbeitspfades. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FDG)
Verzeichnis komplett auflisten	#FDR <Path> (Arbeitspfad)	Auflisten der im Pfad Path vorhandener Dateien und Ordner. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FDR)
Verzeichnis komplett auflisten im ASCII Format	#FDA <Path> (Arbeitspfad)	Auflisten der im Pfad Path vorhandener Dateien und Ordner. Die Ausgabe erfolgt im ASCII-Format. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FDA)
Verzeichnis Ordner auflisten	#FDL <Path> (Arbeitspfad)	Auflisten der im Path vorhandenen Ordner. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FDL)
Verzeichnis erstellen	#FDC <Path>	Neues Verzeichnis im Arbeitsverzeichnis Path erstellen.
Verzeichnis löschen	#FDD <Path>, Content(0)	Inhalt Content (0= nur Inhalt; 1= Ordner und Inhalt) im Arbeitsverzeichnis Path löschen.
Datei zum Schreiben öffnen / erstellen	#FWO <Path + Filename>, Pos (0)	Datei mit dem Namen Filename unter optionaler Angabe des Pfades Path an der Position Pos zum Schreiben öffnen. Wenn die Datei nicht vorhanden ist wird diese Erstellt.
Datei mit Inhalt beschreiben	#FWD Anz, Data	Beschreibt die geöffnete und beschreibbare Datei mit der Anzahl Anz binärer Werte Data .

Bezeichnung	Befehlscode	Beschreibung
Datei mit (ASCII)Inhalt beschreiben	#FWA "String"	Der String String wird in die geöffnete Datei geschrieben.
Datei mit (Unicode)Inhalt beschreiben	#FWU "String"	Der Unicode-String String wird in die geöffnete Datei geschrieben.
Schreibbare Datei schließen	#FWC	Schließt die im Moment geöffnete und beschreibbare Datei.
Datei lesen öffnen	#FRO <Path + Filename>, Pos (0)	Datei mit dem Namen Filename unter optionaler Angabe des Pfades Path an der Position Pos zum Lesen öffnen.
Datei Inhalt lesen	#FRD Anz (gesamtes File)	Liest den Inhalt der geöffneten und lesbaren Datei. Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FRD)
Datei lesen schließen	#FRC	Schließt die geöffnete und lesbare Datei.
File löschen	#FFD <Path + Filename>	Datei löschen
Datei/Verzeichnis Info	#FFI <Path (Arbeitsverzeichnis)>	Sendet Informationen über das aktuelle Verzeichnis / Datei Path . Im Kapitel Antworten/Rückmeldungen ist eine genauere Beschreibung zur Systemantwort zu finden. (#FFI)
Datei/Verzeichnis Name ändern	#FFR <Path>, <NewName>, Replace(0)	Ändert den Namen der Datei oder des Verzeichnisses am Ort Path mit der neuen Bezeichnung NewName . Das Verhalten falls eine andere Datei mit selben Namen vorhanden ist wird durch Replace(0= nur Ändern wenn nicht bereits vorhanden; 1= immer ändern → Andere Datei wird gelöscht) definiert.
Datei/Verzeichnis kopieren	#FFC <PathOld>, <PathNew>, Replace(0)	Kopiert die Datei oder das Verzeichnis vom Ursprungsort PathOld zum neuen Ort PathNew . Das Verhalten bei Existenz mit selben Namen wird durch Replace(0= kopieren, falls Datei nicht bereits vorhanden ist; 1= überschreiben) definiert.
Datei/Verzeichnis verschieben	#FFM <PathOld>, <PathNew>, Replace(0)	Verschiebt die Datei oder das Verzeichnis vom Ursprungsort PathOld zum neuen Ort PathNew . Das Verhalten bei Existenz mit selben Namen wird durch Replace(0= verschieben, falls Datei nicht bereits vorhanden ist; 1= überschreiben) definiert.

Bezeichnung	Befehlscode	Beschreibung								
Datei/Verzeichnis Zeitstempel	#FFT <Path>, time, date	Setzt eine neue Zeit für die angegebene Datei Path . Der Zeitstempel setzt sich aus time und date zusammen. Eine Erläuterung zur Filetime finden Sie siehe S. 102 .								
Datei/Verzeichnis Eigenschaften ändern	#FFA <Path>, Fileattribute	Setzt neue Attribute für die angegebene Datei Path . Der Parameter Fileattribute ist folgendermaßen aufgebaut: <table style="width: 100%; border: none;"> <tr> <td>Schreibgeschützt</td> <td style="text-align: right;">0x01</td> </tr> <tr> <td>Versteckt</td> <td style="text-align: right;">0x02</td> </tr> <tr> <td>System</td> <td style="text-align: right;">0x04</td> </tr> <tr> <td>Archiv</td> <td style="text-align: right;">0x20</td> </tr> </table> Die Flags können durch einfache Bitveroderung kombiniert werden.	Schreibgeschützt	0x01	Versteckt	0x02	System	0x04	Archiv	0x20
Schreibgeschützt	0x01									
Versteckt	0x02									
System	0x04									
Archiv	0x20									
Speichern einer uniTFT- Datei	#FSO <Path>, File-Data	Speichert eine Objektdatei des uniTFT's (z.B. *.epg) in den angegebenen Pfad Path .								
Dateinamen in Stringregister laden	#FNF <Path> (Arbeitsverzeichnis)	Speichert aus dem Zielordner Path die Anzahl in Register R1 und die Namen der Dateien in S1, S2, ... Sanz. Befinden sich zum Beispiel im Zielordner die beiden Verzeichnisse "Projekt1" und "Projekt2" sowie die Datei "start.emc", wird im Register R1 die Zahl "1" gespeichert und im Stringregister S1 der String "start".								
Verzeichnis in Stringregister laden	#FND <Path> (Arbeitsverzeichnis)	Speichert aus dem Zielordner Path die Anzahl in Register R1 und die Namen der Ordner in S1, S2, ... Sanz. Befinden sich zum Beispiel im Zielordner die beiden Verzeichnisse "Projekt1" und "Projekt2" sowie die Datei "start.emc", wird im Register R1 die Zahl "2", im Stringregister S1 der String "Projekt1" und S2 der String "Projekt2" gespeichert.								

FILETIME

Die Filetime setzt sich aus dem Datum und der Zeit zusammen. Im FAT-Dateisystem hat die Zeitangabe eine Auflösung von 2 s, gerechnet aber dem 1.1.1980 um 00:00:00 Uhr.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Datum	Jahr, gezählt ab 1980 (0..127)						Monat (1..12)			Tag (1..31)						
Zeit	Stunde (0..23)					Minute (0..59)					Sekunde/2 (0..29)					

Kurzes Beispiel für die Umrechnung der Zeit:

Datum = ((Jahr-1980)<<9) + (Monat<<5) + Tag;

Zeit = (Stunde<<11) + (Minute<<5) + (Sekunde>>1);

RÜCKMELDUNGEN

Allgemeine Informationen zu Makros befinden sich hier.

Parameter, die in den Befehlstabellen **GRAU** geschrieben sind, gelten als optionale Angaben und besitzen zum Teil Defaultwerte. Die Defaultwerte stehen in Klammern hinter den entsprechenden Parametern. **SCHWARZ** geschriebene Parameter müssen hingegen mit Werten beschrieben werden. Befehle, die zur Änderung von Parametern beitragen können nur dann Verwendung finden, wenn die entsprechenden Parameter vorher bereits definiert wurden. Manche Befehle ermöglichen die Eingabe mehrerer Objekt-ID's. Sind diese Objekte in numerischer Reihenfolge, kann mit dem '-' Zeichen der Bereich angegeben werden, z.B. 1-5, statt 1,2,3,4,5.

Bezeichnung	Returncode	Beschreibung
Sende Version	#XIV String, 0	Gibt die aktuelle Version als String zurück. (←XIVEA uniTFT V1.0)
Display Info	#XID Width 16Bit, Height 16Bit, Colordepth 8Bit, Touch 8Bit, Videowidth 16Bit, Videoheight 16Bit	Gibt die Weite Width , Höhe Height , Farb-tiefe Colordepth (16Bit/24Bit), Touch-funktion Touch (0 = kein Touch; 3 = resistiv; 7 = kapazitiv; 15 = PC-Maus) und die Abmessungen des Videobildes durch dessen Breite Videowidth sowie Höhe Videoheight zurück. Kommt bei Touch der Wert 1 (Top-Bottom) oder 2 (Left-Right) zurück, dann fehlt der jeweils andere Richtungs-gwert.
RAM Info Byte	#XIR Gesamt RAM 32Bit, Frei RAM 32Bit	Rückgabe des Gesamt- und freien Spei-chers im RAM. Sollte kein Speicher mehr frei sein, müssen Objekte gelöscht werden.
SD Card Info in KB	#XIS Gesamt SD 32Bit, Frei SD 32Bit	Rückgabe des Gesamt- und freien Spei-chers auf der SD-Karte in KB.
Uhrzeit binär	#WSB Hour 16Bit, Min 16Bit, Sec 16Bit, Day 16Bit, Month 16Bit, Year 16Bit, Week-day 16Bit	Gibt die Uhrzeit und das Datum in Stunde Hour , Minute Min , Sekunde Sec , Tag Day , Monat Month , Jahr Year und Wochentag Weekday (0=Sonntag) zurück.
Uhrzeit ASCII	#WSA String, 0=Ende	Gibt die Uhrzeit und das Datum als String in ASCII-Code zurück. (z.B. "#WSA08:30:36\nTuesday, 03. May 2016")
Uhrzeit Unicode	#WSU UnicodeString, 0=Ende	Gibt die Uhrzeit und das Datum als String in Unicode-Code zurück. (z.B. "#WSU08:30:36\nTuesday, 03. May 2016")
Projektpfad erhalten	#XPG String, 0	Gibt den Projektpfad als String zurück.
Hardcopy an Serial Interface	#XHS Bitmap Header, Data	Sendet einen Screenshot an die serielle Schnittstelle.
Arbeitsverzeichnis erhalten	#FDG String,0, 0=Ende	Gibt das Arbeitsverzeichnis als String aus.

Bezeichnung	Returncode	Beschreibung
Verzeichnis lesen	#FDR NameString , 0, Filesize 32Bit, File-attribute8Bit, Modified Time 16Bit, Modified Date16Bit, ..., ..., 0=ENDE	Liest den Inhalt des Verzeichnisses mit Namen Namestring , Größe Filesize und Eigenschaften Filattribute aus. Die Eigenschaften können durch Bitveroderung kombiniert werden. Schreibgeschützt 0x01 Versteckt 0x02 System 0x04 Archiv 0x20
Verzeichnis(ASCII) lesen	#FDA SizeString, Datestring, Timestring, Namestring, 0x0D, 0x0A, ..., ..., 0=ENDE	Liest den Inhalt des Verzeichnisses als ASCII-Code mit Größe Sizestring , Datum Datestring , Zeit Timestring und Namen Namestring aus.
Verzeichnisse auflisten	#FDL Namestring, 0, Namestring, ..., 0=ENDE	Listet die Verzeichnisse mit Namen Namestring auf.
Verzeichnis Info	#FFI NameString , 0, Filesize 32Bit, File-attribute8Bit, Modified Time 16Bit, Modified Date16Bit, ..., ..., 0=ENDE	Gibt Informationen über das Verzeichnis mit Namen Namestring , Größe Filesize und Eigenschaften Filattribute aus. Die Eigenschaften können durch Bitveroderung kombiniert werden. Schreibgeschützt 0x01 Versteckt 0x02 System 0x04 Archiv 0x20
Dateiinhalt lesen	#FRD Len 32Bit, Data	Sendet den Inhalt der zum lesen geöffneten Datei Data mit der Länge Len .
Anzahl Portbausteine	#HPI Teilnehmer 16Bit	Information welcher Portbaustein angeschlossen ist. Teilnehmer ist Bitcodiert. Bit0 = Adresse 0 .. Bit7 = Adresse 7..
Port(Byte) lesen	#HPR Portnr 8Bit, Number 8Bit, Value 8Bit	Der Portzustand Value wird gesendet. Portnr gibt den Port an ab dem gelesen wird, Number die Anzahl der zu lesenden Ports.
Port(Bit) lesen	#HBR Bitnr 8Bit, Number 8Bit, Value 8Bit	Der Pinzustand Value wird gesendet. Bitnr gibt den Portpin an ab dem gelesen wird, Number die Anzahl der zu lesenden Pins.
Analoginput lesen	#HAR Channel 8Bit, Number 8Bit, Value 16Bit	Der Analogwert Value wird gesendet. Channel gibt den Analogkanal an ab dem gelesen wird, Number die Anzahl der zu lesenden Kanäle.
Master RS232 Daten	#HRR Len 32Bit, Data	Rückgabe der über die Master-RS232 empfangenen Daten Data mit der Länge Len .

Bezeichnung	Returncode	Beschreibung
Master SPI Daten	#HSR Len 32Bit, Data	Rückgabe der über die Master-SPI empfangenen Daten Data mit der Länge Len .
Master I ² C Daten	#HIR Len 32Bit, Data	Rückgabe der über die Master-I ² C empfangenen Daten Data mit der Länge Len .
Stringdatei Anzahl	#VFC Number 16Bit	Gibt die Anzahl Number der Strings der Stringdatei aus.
Stringregister ASCII	#VSA Number 16Bit, StrLen 16Bit, String	Gibt den String String des Stringregister mit der Nummer Number und der Stringlänge StrLen als ASCII-String aus.
Stringregister Unicode	#VSU Number 16Bit, StrLen 16Bit, UnicodeString	Gibt den String String des Stringregister mit der Nummer Number und der Stringlänge StrLen als Unicodestring aus.
Register lesen	#VRG Number 16Bit, Type 16Bit, Value 32Bit	Gibt den Inhalt Value des Registers Number zurück. Type bestimmt dabei ob der Wert ein 'I' = Integer oder 'F'=Float ist.
Status Touch Schalter abfragen	#TQS Object-id 16Bit, Status 16Bit	Gibt den Status Status (1=Up; 2=Down) des Touch-Schalters Object-id aus.
Aktiven Schalter der Radio-Gruppe abfragen	#TQR Group-id 16Bit, Object-id 16Bit	Gibt die Object-id des aktiven Touch-Schalters der Radiogruppe Group-id aus.
Touch Code Keyboard ausgeben	#TGK Object-id 16Bit, Code 16Bit	Gibt den Code Code der Touchtaste des Keyboards Object-id aus.
Touch Value Instrument ausgeben	#TQI Object-id 16Bit, Value 16Bit	Gibt den Wert Value des Instruments Object-id aus.

BEFEHLSBEISPIELE

Nachfolgend werden die meisten grafischen Befehle anhand kurzer Beispiele veranschaulicht.

AKTION UND ANIMATION

Position Curve, Scale Curve

Aktionspfad mit Positionsänderung

Gerade zeichnen um Aktionspfad zu visualisieren

#GPL 1, 2, 100, 100, 700, 200

Kreis mit Füllung einfügen

#GET 2, 1, 100, 100, 5, 60

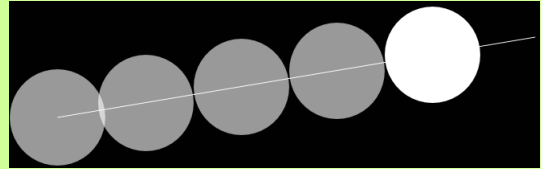
Aktion bewegen über Pfad 1

#AOA 2, 101, 700, 200

Aktionstyp und Aktionszeit definieren

#AOT 2, 5, 500

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 131](#).



Elliptischer Aktionspfad mit Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

#GET 2, 1, 400, 80, 5, 60

Aktionspfad Ellipse und Linie definieren. Der Pfad entspricht Obj. 1

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

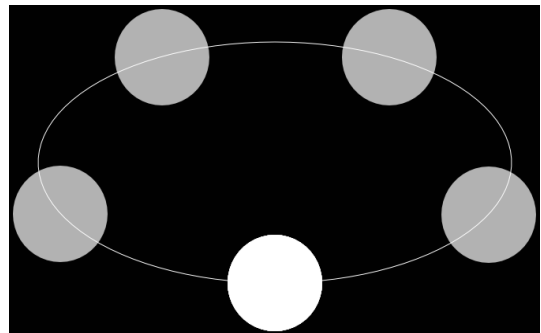
Aktion bewegen über Pfad 1

#AOA 2, 151, 1, 0

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 133](#).



Größenänderung

Kreis mit Füllung einfügen

#GET 1, 1, 200, 200, 5, 50

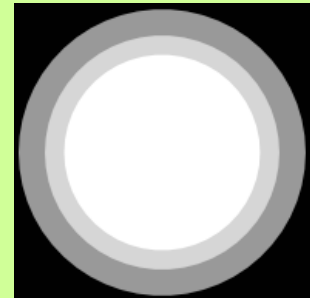
Aktion Größe verdoppeln.

#AOA 1, 201, 200, 200

Aktionstyp und Aktionszeit definieren

#AOT 1, 5, 500

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 131](#).



Größen- und Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

#GET 2, 1, 400, 80, 5, 60

Aktionspfad Ellipse definieren. Der Pfad entspricht Obj. 1

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

Aktionspfad Größenänderung

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

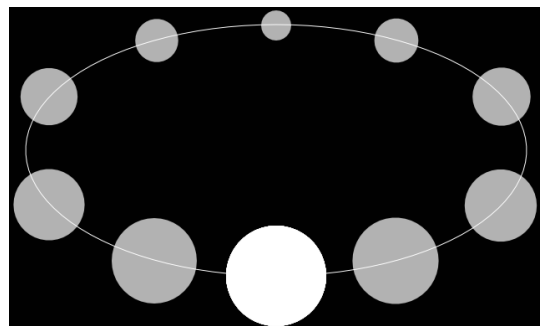
Aktion Bewegung und Größenänderung

#AOA 2, 151, 1, 0, 251, 2, 0

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 132](#).



Rotation Curve, Shear Curve

Drehung

Rechteck waagrecht definieren

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

Aktion Drehung 90° gegen den Uhrzeigersinn.

#AOA 1, 301, 90

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 132](#).

Rotations- und Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

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

Aktionspfad Ellipse definieren. Der Pfad entspricht Obj. 1

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

Aktionspfad Größenänderung

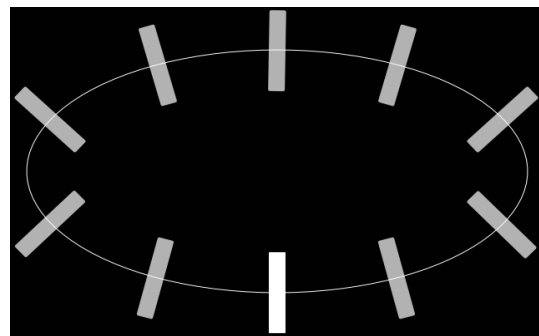
#APD 2, 100, 100 ?L, 30, 30 ?L, 100, 100

Aktion Rotierend am Pfad entlang Bewegen

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

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 134](#).

Verzerren

Quadrat definieren

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

Aktion Shearing 40° auf beiden Achsen

#AOA 1, 401, 40, 40

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 134](#).

Opacity Curve, Coloroffset Curve

Deckkraft

Quadrat definieren

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

Aktion Alphakanal ändern

#AOA 1, 501, 20

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 130](#).



Farbtransformation

Quadrat definieren

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

Aktion Farbtransformation. Rot- und Grünanteil entfernen

#AOA 1, 501, -100,-100

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 130](#).



BILD / VEKTORGRAFIKEN

Bilder / Vektorgrafiken

Plazierung eines (Vektor-)Bildes

Tiger platzieren, Größe proportional skalieren

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



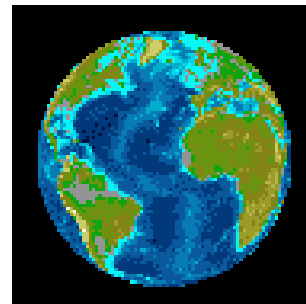
Plazierung einer Animation mit Typänderung

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



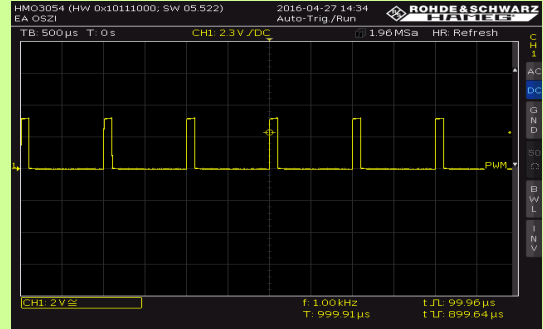
EIN- UND AUSGÄNGE

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 0 (Bit 0-7)	Port 1 (Bit 8-15)
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>

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

Port 0 (Bit 0-7)	Port 1 (Bit 8-15)
<input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>

INSTRUMENTE

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

Bargraph mit abgerundeten Ecken

Bargraph mit Grünverlauf als Vordergrund, inklusive des weißen Rahmens. Hintergrund einfaches Blau.

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

Bargraph auf den Wert 70 einstellen

#IVS 1, 70

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 136](#).



Triangulärer Bargraph

Bargraph mit Grünverlauf als Vordergrund, inklusive des weißen Rahmens. Hintergrund einfaches Blau.

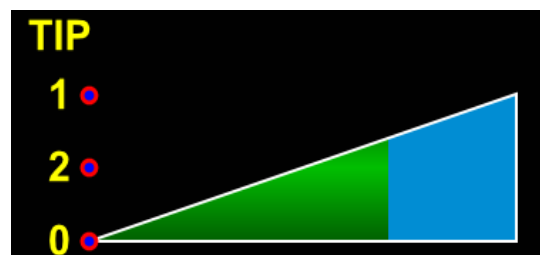
Als Hinweis wurden die 3 möglichen Positionen der Spitze verdeutlicht

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

Bargraph auf den Wert 70 einstellen

#IVS 1, 70

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 135](#).



Bogen - Bargraph

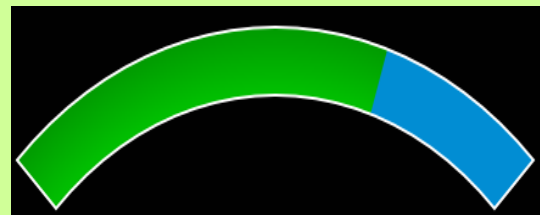
Bargraph mit Grünverlauf im Uhrzeigersinn als Vordergrund, inklusive des weißen Rahmens. Hintergrund einfaches Blau.

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

Bargraph auf den Wert 70 einstellen

#IVS 1, 70

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 137](#).

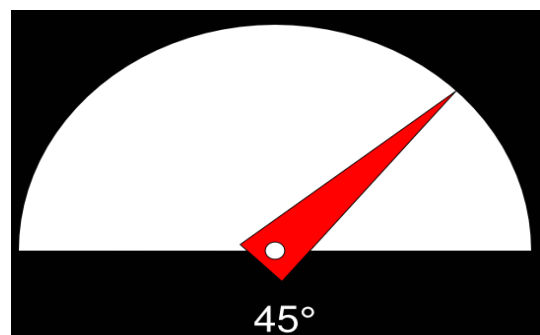


Objektgruppe als Drehinstrument

Definiert Objektgruppe 2 als Drehinstrument.

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

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 137](#).

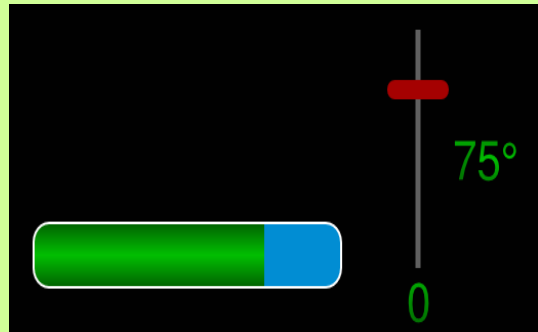


#IBR, IBT, IBA, IGM, IGS, IPP, IVA,

Objektgruppe als Slider

Definiert Objektgruppe 3 als Slider

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 138](#).

Wattmeter als Instrument

Instrumentenfile plazieren; Originalbildgröße Startwert=0, Endwert=500

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

Bargraph auf 70% einstellen (0,7x500)

#IVS 1, 350



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 136](#).

Bargraph Wert Update

Rechteckiger Bar, mit Start- und Endvalue auf Sekunden angepasst

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

Bar automatisch mit der Sekunde ändern

#IVA 1,(second())



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 136](#).

KEYBOARD

#KDB, KDL, KDC, KPK

Objektgruppe als Tastenfeld

Objektgruppe 5 als Tastenfeld definieren

Erzeugen der des Tastenfeldes für Keyboard1

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

Erzeugen der des Tastenfeldes für Keyboard2

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 139](#).

Tastenbeschriftung der speziellen Tasten ändern

Beschriftung der speziellen Tasten in Objektgruppe 5 ändern

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

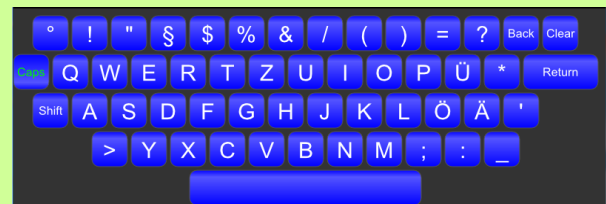


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 139](#).

Buttonstyle für Tasten definieren

Buttonstyle für normale und spezielle Tasten der Länge 1 definieren

#KDC 5, 1, 2



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 139](#).

Objektgruppe als Keyboard platzieren

Platziert Objektgruppe 5 als Keyboard

#KPK 5, 0, 0, 7, 800



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 139](#).

STRING ZEICHENKETTENBEFEHLE

#SSP, SFP, SAP

Zeichenkette platzieren

Platzierung einer einfachen Zeichenkette

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

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

Int: 3, Float: 3.140, Hex: 0x0D

Formatierte sich selbst erneuernde Zeichenkette platzieren

Sobald sich der Analogeingang 0 ändert wird diese Zeichenkette neu dargestellt

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

Analog In 0: 1.65 V

Update-Kalkulation ändern

Sobald sich der Analogeingang 0 ändert wird diese Zeichenkette neu dargestellt

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

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

#SAC 1, (second())

Analog In 0: 1.65 V

Datum und Uhrzeit

Platzierung eines formatierten Datums

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

Thursday the 21. Apr 2016, 11:37

Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 140](#).

Zuweisung von drei Editboxen auf eine Tastatur

Editboxen 6-8 der Tastatur 9 zuweisen

#SEK 9, 6-8

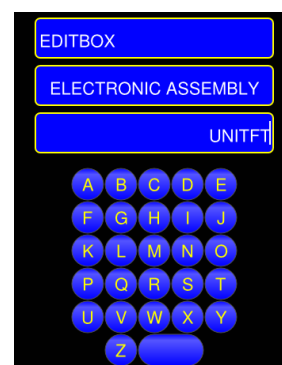


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 140](#).

Aktivierung Editbox

Aktiviert Editbox 8, die oberste der drei Editboxen

#SEA 8



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 140](#).

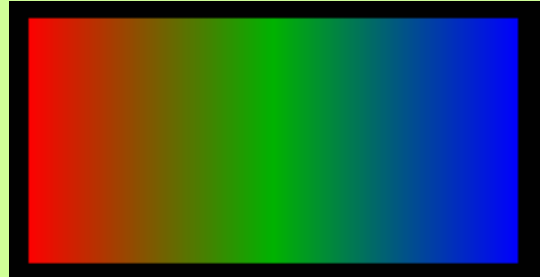
STYLE SHEETS

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 141](#).

Linienmuster

Erstellung eines Linienmusters (Dash pattern),

[#CDP](#) 1, 0, 20,10, 5,10



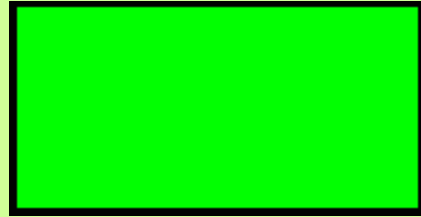
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 141](#).

Füllungen

Einfarbige Füllung

Füllfarbe definieren, der Alphakanal ist optional,

`#CFC 1, $00FF00`

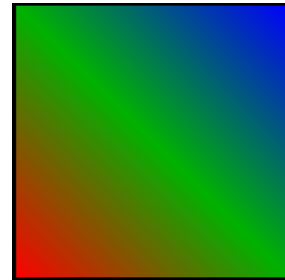


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 142](#).

Lineare Farbverlaufsfüllung

Gradientenverlauf als linearen Verlauf nutzen - um 45° geneigt

`#CFL 1, 1, 45`

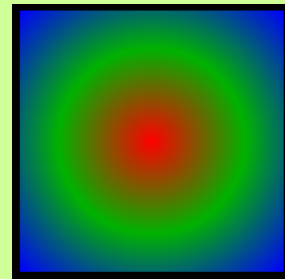


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 142](#).

Radiale Farbverlaufsfüllung

Gradientenverlauf als radialen Verlauf nutzen. Anker 5 als Fokuspunkt nutzen

`#CFR 1, 1, 5000,5`



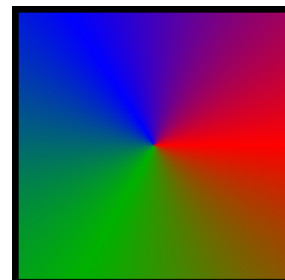
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 142](#).

Konische Farbverlaufsfüllung

Gradientenverlauf als konischen Verlauf nutzen.

Anker 5 als Fokuspunkt nutzen. Die Drehrichtung ist im Uhrzeigersinn

`#CFK 1, 1, 5000,5`



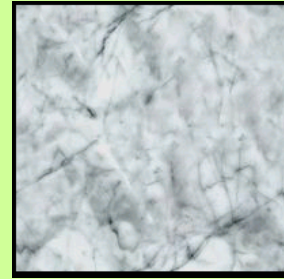
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 142](#).

Füllungen

Musterfüllung

Füllung mit sich selbst wiederholendem Muster

#CFP 1, "marmor02"



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 143](#).

Linienverbindung

Linienverbindung mit abgerundeten Ecken (Round = 1)

#CLE 1, 1

Linienverbindung mit spitzen Ecken (Miter = 0)

#CLE 2, 0



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 143](#).

Textstyle

Textstyle

Erstellung eines Textstyles mit dem internen Font "Sans"

#CTF 1,<!Sans.evf>, 60,1,1

Hello World

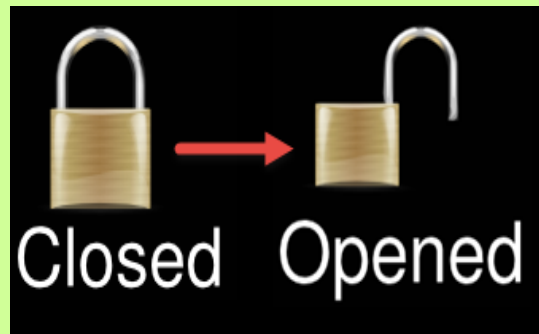
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 143](#).

Buttonstyle

Touchbutton als Bild

Zwei Bilder als Touchbuttonstyle definieren

#CBP 1, "Padlock_closed"; "Padlock_opend"

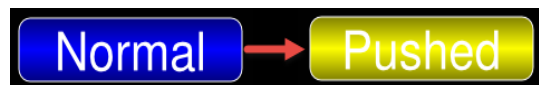


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 144](#).

Touchbuttonstyle für geometrische Objekte

Touchbuttonstyle mit 200 Breite, 50 Höhe und Eckenradius 50

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 144](#).

Touchbutton Proportion bei Drücken ändern

Touchbutton soll seine Größe auf 70% im
gedrückten Zustand ändern

#CBO 1, 0,0, 70



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 144](#).

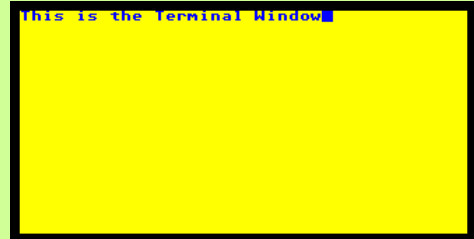
TERMINAL BEFEHLE

Terminalfenster

Terminalfenster (neu) definieren

Kleines Terminalfenster mit 40 Zeichen und 20 Zeilen

[#YDW](#) 1, 30,30,7, 40,20



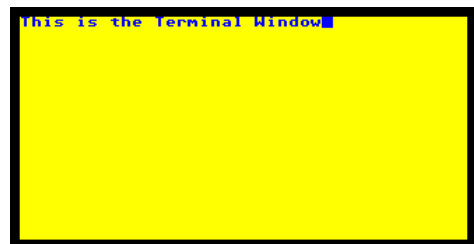
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 145](#).

Terminalfenster Vordergrund- Hintergrundfarbe

Kleines Terminalfenster mit 40 Zeichen und 20 Zeilen

Schriftfarbe Blau, Fensterfarbe, Gelb, ohne Transparenz

[#YDC](#) \$0000FF,100, \$FFFF00,100



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 145](#).

TOUCHOBJEKTE / TOUCHFUNKTIONEN

Touchbereiche

Rechteckiger Touchbutton

Touchschalter auf Position 400,240 setzen.

Außenmasse sind durch den Buttonstyle definiert

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 146](#).

Elliptischer Touchbutton

Touchschalter auf Position 400,240 setzen.

Außenmasse sind durch den Buttonstyle definiert

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

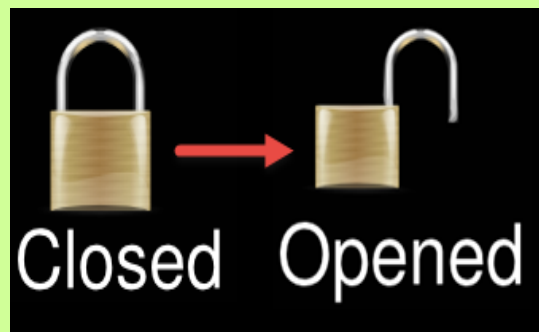


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 146](#).

Touchbutton als Bild

Zwei Bilder als Touchbutton nutzen

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



UHRZEIT

#WGC

Objektgruppe als Uhr

Objektgruppe 10 als Uhr definieren

Objekte 6, 7 und 8 bilden die Uhrzeiger

[#WGC](#) 10, 6, 7, 8



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 147](#).

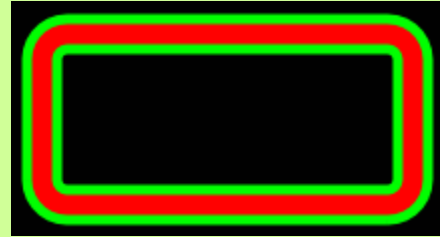
ZEICHNEN/ GRAFISCHE PRIMITIVE

Grafische Zeichenobjekte

Abgerundetes Rechteck mit Rand

Abgerundetes Rechteck mit Rand, eine Art Bilderrahmen

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

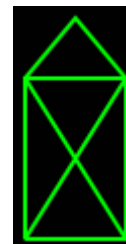


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 148](#).

Polylinie - Nikolaushaus

Mithilfe eines Polypfades ein Nikolaushaus zeichnen

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

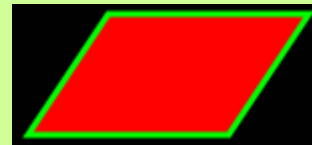


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 148](#).

gefüllte Polylinie - Trapez

Mithilfe eines Polypfades eingefülltes Trapez zeichnen

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

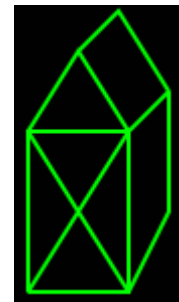


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 148](#).

Polylinie erweitern - Nikolaushaus

Das Nikolaushaus um die 3 Dimension erweitern

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 148](#).

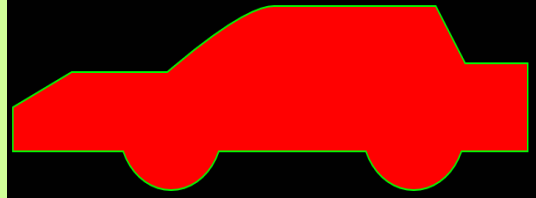
Grafische Zeichenobjekte**Polypfad - Auto zeichnen**

Ein Auto mit Pfadsegmenten zeichnen

#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

Verlauf des relativen Polypfad:

Vom Startpunkt wird die Linie vertikal 50 Pixel nach unten gezeichnet. Anschließend folgt eine horizontale Linienführung nach rechts. Zwischen den drei horizontalen Segmenten werden die Räder durch Kreissegmente gezeichnet. Die Räder werden so gezeichnet, dass deren Breite 130 Pixel beträgt. Es folgt eine vertikale Linienführung um 100 Pixel nach oben sowie eine vertikale Linie um 85 Pixel nach links. Für die Heckscheibe wird ein Liniensegment genutzt, welches den Endpunkt auf 40 Pixel links und 65 Pixel oberhalb der letzten Position setzt und mittel einer Linie verbindet. Anschließend folgt eine horizontale Linie mit 220 Pixel Länge nach links. Für die Frontscheibe wird ein quadratisches Bezier-Segment verwendet. Danach folgt eine horizontale Linie mit 130 Pixeln nach links sowie der Pfadabschluss.

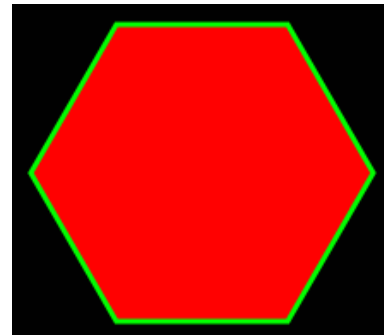


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 148](#).

6-Eck

6 eckiges Polygon um 90° gedreht

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 149](#).

Stern mit 5 Spitzen

Gefüllten Stern zeichnen

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



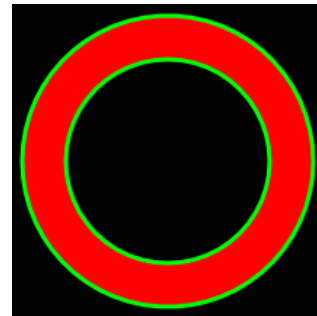
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 149](#).

Grafische Zeichenobjekte

Reifen

Kreis mit Loch

[#GET](#) 1, 1 400,240,5, 100,100, 30



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 149](#).

Grafische Zeichenobjekte

Kreisbogen

Ein C als Kreisbogen zeichnen

[#GEA](#) 1, 1 400,240,5, 100,100, 45,-45, 30



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 149](#).

Kreissegmente geschnitten

Zwei geschnittene Kreissegmente zeichnen

[#GES](#) 1, 1, 400, 200, 5, 100, 100, 90, 0

[#GES](#) 2, 1, 450, 200, 5, 100, 100, 180, 60

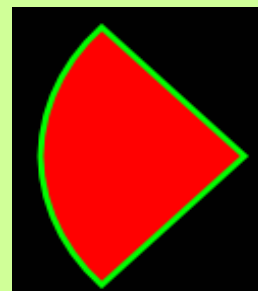


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 150](#).

Kuchenstück

Ein Kuchenstück zeichnen

[#GEP](#) 1, 1 400,240,5, 100,100, 135,225



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 150](#).

SEGEMENTTYPEN

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

Polypfad Horizontal

Horizontale Linie als Polypfad erzeugen

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 151](#).

Polypfad Vertikal

Vertikale Linie als Polypfad erzeugen

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

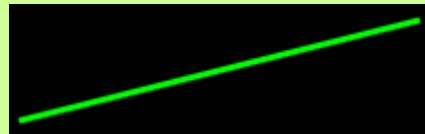


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 151](#).

Polypfad Linie

Linie als Polypfad erzeugen

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



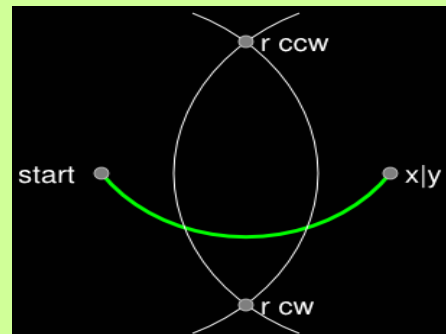
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 151](#).

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

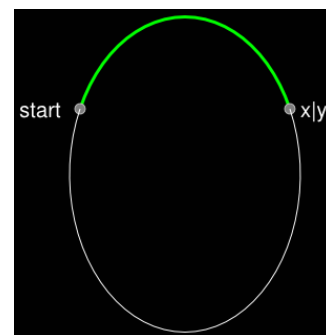


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 152](#).

Polypfad Ellipse

Ellipse durch die beiden Punkte

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



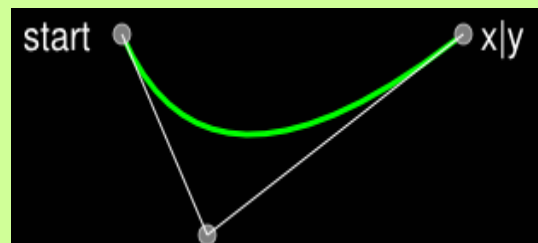
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 152](#).

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

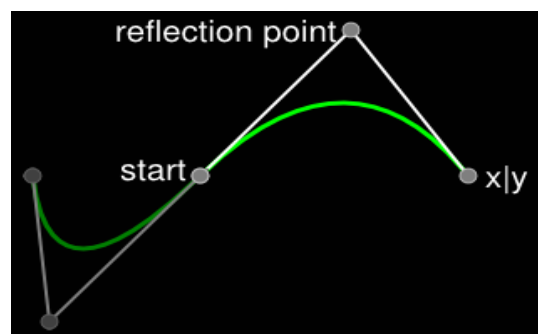


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 153](#).

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



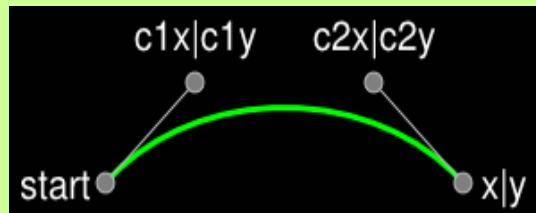
Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 154](#).

Segmente T, S(pline)

Polypfad Spline

Spline zeichnen

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

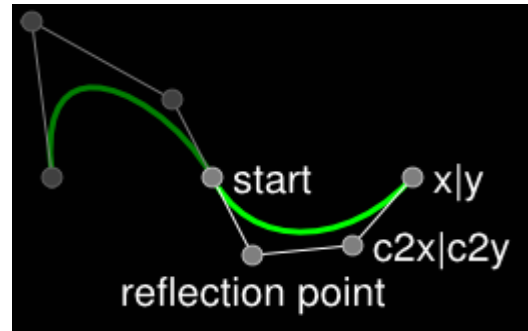


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 155](#).

Polypfad geglättete kubische Bézierkurve

Die geglättete kubische Bézierkurve übernimmt von dem vorhergegangenen 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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 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

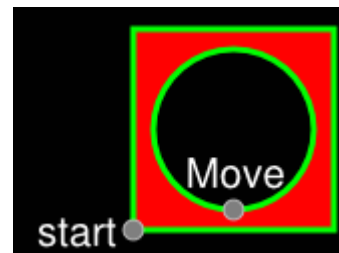


Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 157](#).

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



Für die gesamte Befehlseingabe des Beispielbildes [siehe S. 157](#).

BEFEHLE ZUR ERZEUGUNG DER BEISPIELBILDER

Nachfolgend werden die kompletten Befehle aufgezeigt, welche für die Erstellung der Beispielbilder nötig sind.

AKTION UND ANIMATION

Opacity Curve, Coloroffset Curve

Deckkraft

Quadrat definieren

[#GRR](#) 1, 1, 100,100,5, 80,80,5
Aktion Alphakanal ändern


[#AOA](#) 1, 501, 20
Aktionstyp und Aktionszeit definieren

[#AOT](#) 1, 4, 500
Weiße Füllung definieren

[#CFC](#) 4, \$FFFFFF, 60
Rechtecke erzeugen

[#GRR](#) 2, 4, 200,100,5, 80,80,5

[#GRR](#) 3, 1, 300,100,5, 80,80,5



Farbtransformation

[#CFC](#) 4, \$7F7FFF, 60
Quadrat definieren


[#GRR](#) 1, 1, 100,100,5, 80,80,5
Aktion Farbtransformation. Rot- und Grünanteil entfernen

[#AOA](#) 1, 501, -100,-100
Aktionstyp und Aktionszeit definieren

[#AOT](#) 1, 4, 500
Restliche Quadrate definieren

[#GRR](#) 2, 4, 200,100,5, 80,80,5

[#GRR](#) 3, 1, 300,100,5, 80,80,5

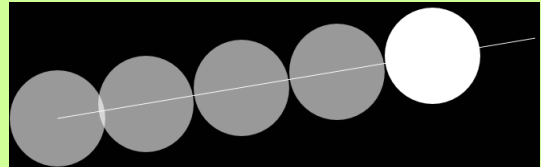


Position Curve, Scale Curve

Aktionspfad mit Positionsänderung

Gerade zeichnen um Aktionspfad zu visualisieren

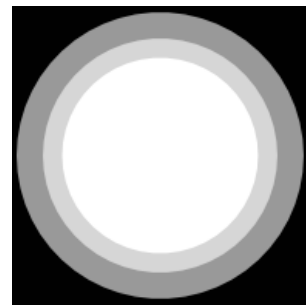
```
#GPL 1, 2, 100,100, 700,200
      Kreis mit Füllung einfügen
#GET 2, 1, 100, 100, 5, 60
      Aktion bewegen über Pfad 1
#AOA 2, 101, 700, 200
      Aktionstyp und Aktionszeit definieren
#AOT 2, 5, 500
      Integer Register setzen
#VRI 0, 3, 1,10
      Weiße Füllung definieren
#CFC 4, $FFFFFF, 60
      Kreise mit Füllung sowie deren Aktionstyp und Aktionszeit definieren
#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
```



Größenänderung

Kreis mit Füllung einfügen

```
#GET 1, 1, 200, 200, 5, 50
      Aktion Größe verdoppeln.
#AOA 1, 201, 200, 200
      Aktionstyp und Aktionszeit definieren
#AOT 1, 5, 500,0
      Weiße Füllung definieren
#CFC 4, $FFFFFF, 60
      Integer Register setzen
#VRI 0, 3, 500, 1,300
      Kreise erzeugen und deren Aktionstyp sowie Aktionszeit definieren
#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

Größen- und Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

#GET 2, 1, 400, 80, 5, 60

Aktionspfad Ellipse definieren. Der Pfad entspricht Obj. 1

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

Aktionspfad Größenänderung

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

Aktion Bewegung und Größenänderung

#AOA 2, 151, 1, 0, 251, 2, 0

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400

Integer Register setzen

#VRI 0, 3, 1, 10

Weißer Füllung definieren

#CFC 4, \$FFFFFF, 70

Marke Makrofile setzen

#MFM

Punkte/Kreise erzeugen

#GET R0, 4, 400, 80, 5, 60

Aktion Bewegung und Größenänderung

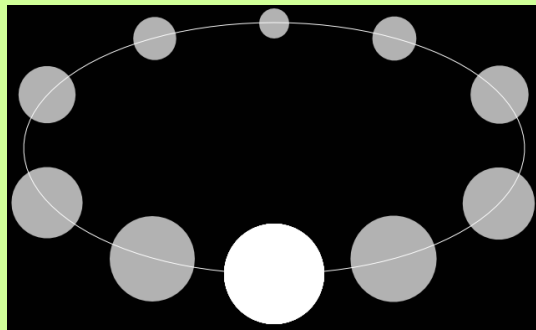
#AOA R0, 151, 1, (R1*R2), 251, 2, (R1++*R2)

Aktionstyp und Aktionszeit definieren

#AOT R0, 1, 400

Zurück zur Makrofile Marke springen

#MFJ (R0++<20)



Drehung

Rechteck waagrecht definieren

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

Aktion Drehung 90° gegen den Uhrzeigersinn.

#AOA 1, 301, 90

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500

Integer Register setzen

#VRI 0, 2, 1, 100

Weißer Füllung definieren

#CFC 4, \$FFFFFF, 70

Marke Makrofile setzen

#MFM

Rechtecke erzeugen

#GRR R0, 4, 50, 100, 4, 300, 40, 5

Aktion Drehung 90° gegen den Uhrzeigersinn.

#AOA R0, 301, 90

Aktionstyp und Aktionszeit definieren

#AOT R0, 1, 400

Zurück zur Makrofile Marke springen

#MFJ (R0++<20)



Position Curve, Scale Curve

Elliptischer Aktionspfad mit Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

#GET 2, 1, 400, 80, 5, 60

Aktionspfad Ellipse und Linie definieren. Der Pfad entspricht Obj. 1

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

Aktion bewegen über Pfad 1

#AOA 2, 151, 1, 0

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400

Weißer Füllung definieren

#CFC 4, \$FFFFFF, 70

Integer Register setzen

#VRI 0, 3, 1, 20

Kreise mit Aktionstyp und Aktionszeit definieren

#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

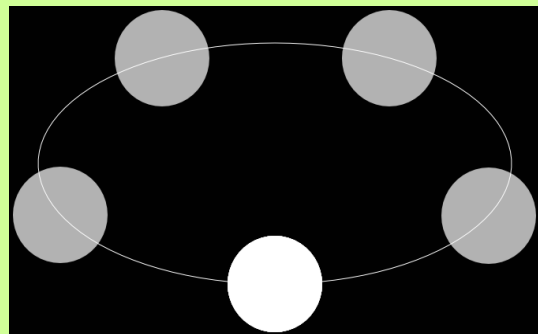
#AOA R0, 151, 1, (R1++*R2)

#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

Rotations- und Positionsänderung

Polypfad Ellipse einfügen um Aktionspfad zu visualisieren

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

Kreis mit Füllung einfügen

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

Aktionspfad Ellipse definieren. Der Pfad entspricht Obj. 1

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

Aktion Rotierend am Pfad entlang Bewegen

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

Aktionstyp und Aktionszeit definieren

#AOT 2, 1, 400

Integer Register setzen

#VRI 0, 3, 1, 10

Weißer Füllung definieren

#CFC 4, \$FFFFFF, 70

Marke Makrofile setzen

#MFM

Rechtecke erzeugen

#GRR R0, 4, 400, 80, 5, 20, 100, 2

Aktion Rotierend am Pfad entlang Bewegen

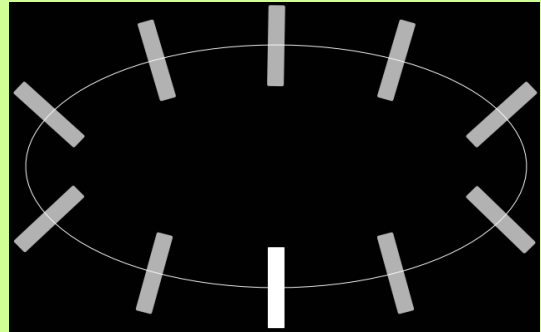
#AOA R0, 151, 1, (R1*R2), 351, 2, (R1++*R2)

Aktionstyp und Aktionszeit definieren

#AOT R0, 1, 400

Zurück zur Makrofile Marke springen

#MFJ (R0++<20)



Verzerren

Quadrat definieren

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

Aktion Shearing 40° auf beiden Achsen

#AOA 1, 401, 40, 40

Aktionstyp und Aktionszeit definieren

#AOT 1, 4, 500

Integer Register setzen

#VRI 0, 2, 1, 200

Weißer Füllung definieren

#CFC 4, \$FFFFFF, 70

Marke Makrofile setzen

#MFM

Rechtecke erzeugen

#GRR R0, 4, 100, 100, 5, 80, 80, 5

Aktion Shearing 40° auf beiden Achsen

#AOA R0, 401, 40, 40

Aktionstyp und Aktionszeit definieren

#AOT R0, 4, (500+R1*R2), (R1++*R2)

Zurück zur Makrofile Marke springen

#MFJ (R0++<3)



INSTRUMENTE

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

Triangulärer Bargraph

Grünen Farbverlauf definieren

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

Keine Umrandung definieren

#CLD 1

Einfarbige blaue Füllung definieren

#CFC 2, \$008dd3, 100

Linienstyle für Umrandung definieren

#CLS 2, \$ffffff, 100, 2

Farbverlauf als linear mit 90° Verdrehung definieren

#CFL 1, 1, 90,

Bargraph mit Grünverlauf als Vordergrund. Hintergrund einfaches Blau.

Als Hinweis wurden die 3 möglichen Positionen der Spitze verdeutlicht

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

Bargraph auf den Wert 70 einstellen

#IVS 1, 70

Linienstyle für Punkte definieren

#CLS 5, \$FF0000, 100, 3

Blaue Füllung für Punkte definieren

#CFC 5, \$0000FF, 100

Gelbe Füllung für Schrift definieren

#CFC 6, \$FFFF00, 100

Keine Umrandung für Schrift definieren

#CLD 6

Textstyle definieren

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

Integer Register setzen

#VRI 0, 1

String Register setzen

#VSS 1, "1"

String Register setzen

#VSS 4, "2"

String Register setzen

#VSS 7, "0"

Marke für Makrofile setzen

#MFM

Punkte erzeugen

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

Nummerierung erzeugen

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

Integer Register setzen

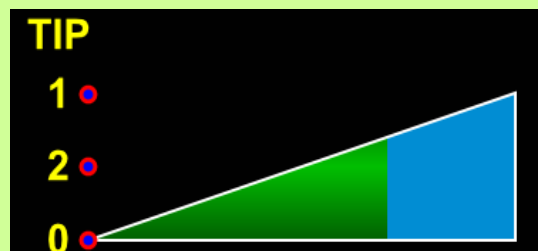
#VRI 0, (R0+3)

Befehl um wieder zur Marke des Makrofile zu springen

#MFJ (R0<9)

Nummerierung setzen

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



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

Bargraph mit abgerundeten Ecken

Grünen Farbverlauf definieren

#CCR 1, 0, \$005f00, 100, 50, \$00Bf00, 100, 100, \$005f00, 100
Keine Umrandung definieren

#CLD 1
Einfarbige blaue Füllung definieren

#CFC 2, \$008dd3, 100
Linienstyle für Umrandung definieren

#CLS 2, \$ffffff, 100, 2
Farbverlauf als linear mit 90° Verdrehung definieren

#CFL 1, 1, 90,
Rechteckiger Bar, mit Start- und Endvalue auf Sekunden angepasst

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15
Bar automatisch mit der Sekunde ändern

#IVS 1,70



Bargraph Wert Update

Grünen Farbverlauf definieren

#CCR 1, 0, \$005f00, 100, 50, \$00Bf00, 100, 100, \$005f00, 100
Keine Umrandung definieren

#CLD 1
Einfarbige blaue Füllung definieren

#CFC 2, \$008dd3, 100
Linienstyle für Umrandung definieren

#CLS 2, \$ffffff, 100, 2
Farbverlauf als linear mit 90° Verdrehung definieren

#CFL 1, 1, 90,
Rechteckiger Bar, mit Start- und Endvalue auf Sekunden angepasst

#IBR 1, 1, 2, 150, 70, 4, 300, 50, 15,0,59
Bar automatisch mit der Sekunde ändern

#IVA 1,(second())
Einfarbige weiße Füllung für Beschriftung definieren

#CFC 3, \$FFFFFF, 100
Keine Umrandung für die Schrift definieren

#CLD 3
Textstyle ändern

#CTC 1,3
Platzieren des kalkulierten Strings in Objektmittle

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



Wattmeter als Instrument

Projektpfad absolut angeben

#XPS </Projekt>

Instrumentenfile plazieren; Orginalbildgröße Startwert=0, Endwert=500

#IPP 1, "Wattmeter"; 400,200,5,0,0,0,500
Bargraph auf 70% einstellen (0,7x500)

#IVS 1, 350



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

Bogen - Bargraph

Grünen Farbverlauf definieren

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

Keine Umrandung definieren

#CLD 1

Einfarbige blaue Füllung definieren

#CFC 2, \$008dd3, 100

Linienstyle für Umrandung definieren

#CLS 2, \$ffffff, 100, 2

Farbverlauf als radial definieren

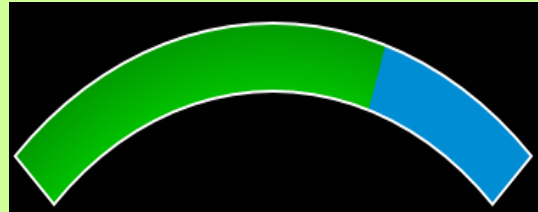
#CFR 1, 1

Bargraph mit Grünverlauf im Uhrzeigersinn als Vordergrund. Hintergrund einfaches Blau.

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

Bargraph auf den Wert 70 einstellen

#IVS 1, 70



Objektgruppe als Drehinstrument

Linienstyle und Füllung Zeiger definieren

#CLS 2, \$000000

#CFC 2, \$FF0000

Linienstyle und Füllung Punkt und Scheibe definieren

#CLS 3, \$000000

#CFC 3, \$FFFFFF

Scheibe und Zeiger erzeugen

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

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

Zeiger drehen, freien Anker setzen und platzieren

#ORA 90, 2

#OAO 30, 20, 2

#OPA 300, 116, 2

Weißes Punkt erzeugen

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

Gruppe erzeugen und diese als Drehinstrument definieren

#OGA 5, 1, 2, 3, 4

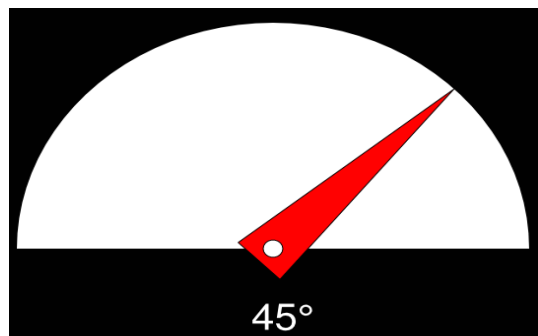
#IGM 5, 2, 180, -180, -90, 90

Touchfunktion zuweisen

#TID 1, 5

Kalkulierten String platzieren

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



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

Objektgruppe als Slider

Grünen Farbverlauf definieren

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

Kein Linienstyle und linear um 90° verdrehten Farbverlauf definieren

#CLD 1

#CFL 1, 1, 90

Blau Füllung und weißen Linienstyle für Bargraph definieren

#CFC 2, \$008dd3, 100

#CLS 2, \$ffffff, 100, 2

Linienstyle für Slider-Pfad definieren

#CLS 3, \$606060, 100, 5

Kein Linienstyle und rote Füllung für Slider definieren

#CLD 4

#CFC 4, \$A50000, 100

Slider mit Pfadlinie platzieren

#GPL 1, 3, 425, 60, 425, 245

#GRR 2, 4, 425, 60, 5, 60, 15, 15, 0

Gruppe erzeugen und als Slider definieren

#OGA 3, 1, 2

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

Bargraph platzieren

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

Touchfunktion zuweisen

#TID 1, 3

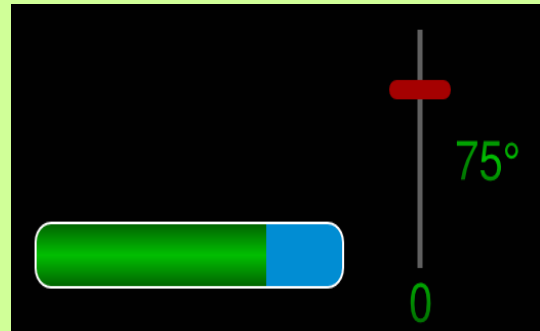
Kalkulierten und festen String platzieren

#SAP 5, 1, 490, 145, 5, "%d°"; (objIV(3))

#SSP 6, 1, 425, 35, 5, "0"

automatische Wertänderung Bargraph

#IVA 4, (objIV(3))



KEYBOARD

#KDB, KDL, KDC, KPK

Objektgruppe als Keyboard platzieren

Linienstyles, Farbverlauf und Füllungen für Tasten definieren

```
#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
```

Textstyles definieren

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

Textstyles für Touchbttons definieren

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

Drawstyle und Proportionsänderung der Touchbuttons definieren

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

Keyboardstyle definieren

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

Tastenfelder definieren

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

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

Beschriftung der speziellen Tasten ändern

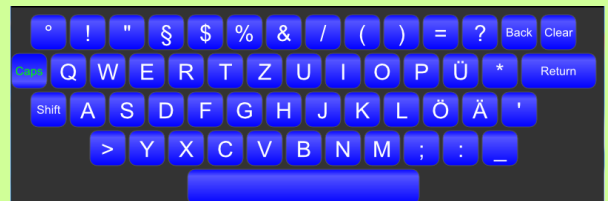
```
#KDL 5, 5 "Shift"; 6 "Caps"; 8 "Back"; 12 "Clear"; 13 "Return"
```

Keyboard platzieren

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

Rahmen der Objektgruppe / Keyboard füllen

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



STRING ZEICHENKETTENBEFEHLE

#SDP, SEP, SEK, SEA

Datum und Uhrzeit

Platzierung eines formatierten Datums

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

Änderung des formatierten Datums

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

Thursday the 21. Apr 2016, 11:37

Editbox in Kombination mit einem Keyboard

Textstyle Tastatur definieren

#CLD 1

#CFC 1, \$FFFF00, 100

#CTF 1, <l:Sans.evf>;30, 1 11

Füllung Tastatur für gedrückten und ungedrückten Zustand definieren

#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

Touchbuttonstyle definieren

#CBT 2, 1, 1, 0, 0

#CBD 2, 3, 4, 100, 50, 25

#CBO 2, 0, 0, 150

Keyboard definieren und platzieren

#KDC 9, 2, 2

#KDB 9, 1, "ABCDE"; "FGHIJ"; "KLMNO"; "PQRST"; "UVWXY"; "Z "

#KPK 9, 400, 10, 8, 0, 250

Umrandung, Füllung und Textstyle Editbox definieren

#CLS 20, \$FFFF00, 100, 2

#CFC 20, \$0000FF, 100

#CLD 21,

#CFC 21, \$FFFFFF, 100

#CTF 21, <l:Sans.evf>;30, 2, 21

#CTF 22, <l:Sans.evf>;30, 1, 21

#CTF 23, <l:Sans.evf>;30, 0, 21

Editboxen platzieren

#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

Editboxen und Tastatur als Gruppe zusammenfassen

#OGA 91, 6-9

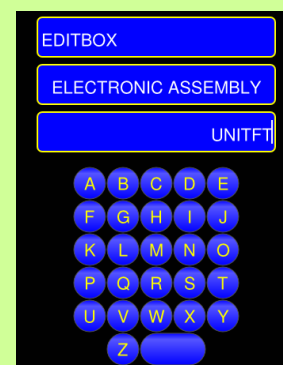
Editboxen der Tastatur zuweisen und Editbox 8 aktivieren

#SEK 9, 6-8

#SEA 8

Touchfunktion zur Auswahl der Editboxen per Touch zuweisen

#TID 1, 6-8



STYLE SHEETS

Farbverlauf und Linienmuster

Farbverlauf

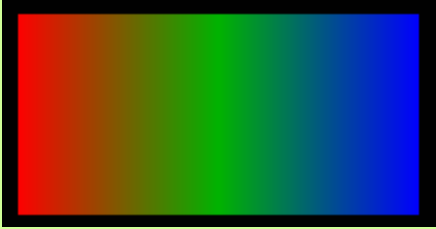
Farbverlauf definieren

#CCR 1, 0,\$FF0000,100, 50,\$00FF00,70, 100,\$0000FF,100
Farbverlauf als linear definieren

#CFL 1, 1,
Keinen Linienstyle definieren

#CLD 1
Quadrat erzeugen

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



Linienmuster


Definieren eines Linienmusters (Dash pattern),

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

#CLS 1, \$00FF00,100, 5,0 1
Keine Füllung definieren

#CFD 1
Polylinie zeichnen

#GPL 1,1, 50,50,400,50



Füllungen

Einfarbige Füllung

Füllfarbe definieren, der Alphakanal ist optional,

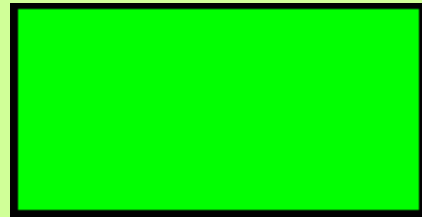
#CFC 1, \$00FF00

Keinen Linienstyle definieren

#CLD 1

Quadrat erzeugen

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



Lineare Farbverlaufsüllung

Farbverlauf definieren

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

Farbverlauf als linear und um 45° geneigt definieren

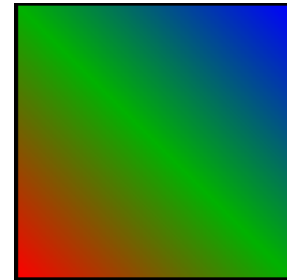
#CFL 1, 1, 45

Keinen Linienstyle definieren

#CLD 1

Quadrat erzeugen

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



Radiale Farbverlaufsüllung

Farbverlauf definieren

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

Farbverlauf als radial definieren

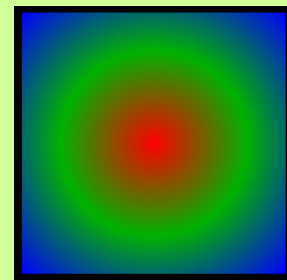
#CFR 1, 1, 5000, 5

Keinen Linienstyle definieren

#CLD 1

Quadrat erzeugen

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



Konische Farbverlaufsüllung

Farbverlauf definieren

#CCR 1, 0, \$FF0000, 100, 33, \$00FF00, 70, 66, \$0000FF, 100, 100, \$FF0000, 100

Farbverlauf als konisch definieren

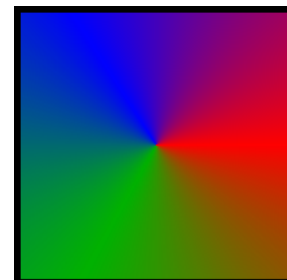
#CFK 1, 1, 5000, 5

Keinen Linienstyle definieren

#CLD 1

Quadrat erzeugen

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



Musterfüllung

Füllung mit sich selbst wiederholendem Muster

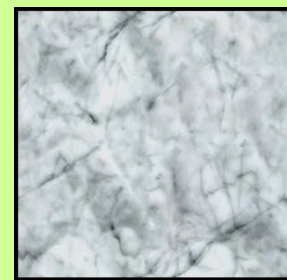
#CFP 1, "marmor02"

Keinen Linienstyle definieren

#CLD 1

Quadrat erzeugen

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

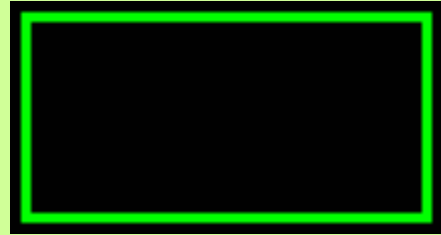


Liniestyle

Liniensfarbe und -dicke

grüne Linie mit 5 Pixel dicke

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



Linienverbindung

Keinen Liniestyle definieren

#CLD 1

Füllung Schrift definieren

#CFC 1, \$FFFFFF,100

Textstyle definieren

#CTF 1, <!:Sans.evf>,58,0,1,0,1,0

Keine Umrandung definieren

#CFD 4

Grünen Liniestyle definieren eckig

#CLS 4 \$00FF00,100 10,0

Keine Umrandung definieren

#CFD 5

Grünen Liniestyle definieren rund

#CLS 5 \$00FF00,100 10,1

Integer Register setzen

#VRI 0, 50,50,20,50

Polypfad zeichnen und String platzieren

#GPL 1,4, R0,R3, (R0+R1),(R3+R2)rd, R0,(R3+2*R2)

#SSP 2,1, (R0+R1+20),(R3+R2),4, "Miter"

Integer Register setzen

#VRI 3, (R3+3*R2)

Polypfad zeichnen und String platzieren

#GPL 3,5, R0,R3, (R0+R1),(R3+R2), R0,(R3+2*R2)

#SSP 4,1, (R0+R1+20),(R3+R2),4, "Round"



Textstyle

Textstyle

Erstellung eines Textstyles mit dem internen Font "Sans"

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

String platzieren

#SSP 1, 1, 200,50,7, "Hello World"

Hello World

Buttonstyle

Touchbutton als Bild

Zwei Bilder als Touchbuttonstyle definieren

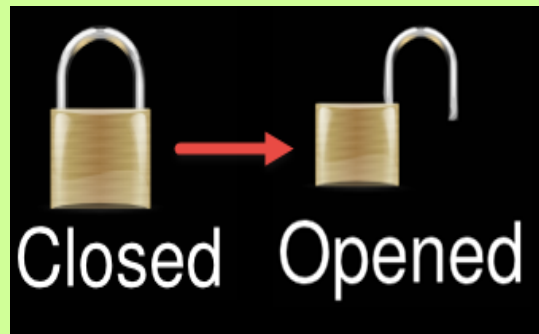
#CBP 1, "Padlock_closed"; "Padlock_opend"

Textstyle Touchbutton definieren

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

Bilder Touchbutton definieren

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



Touchbutton Proportion bei Drücken ändern

Textausrichtung ändern

#CTA 1,1

Farbverläufe für Touchbutton erzeugen

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

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

Linienstyles für Touchbutton erzeugenT

#CLS 5, \$FFFFFF, 100, 1

#CLS 6, \$FFFFFF, 100, 1

Farbverläufe um 90° drehen

#CFL 5,5,90

#CFL 6,6,90

Touchbuttonstyle definieren

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

Textstyle Touchbutton definieren

#CBT 1, 1, 1

Proportion des gedrückten Zustandes ändern

#CBO 1,0,0, 70

Rechteckigen Touchbutton definieren

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



TERMINAL BEFEHLE

Terminalfenster

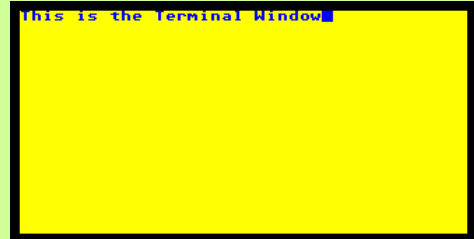
Terminalfenster (neu) definieren

Schriftfarbe Blau, Fensterfarbe, Gelb, ohne Transparenz

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

Kleines Terminalfenster mit 40 Zeichen und 20 Zeilen

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



TOUCHOBJEKTE / TOUCHFUNKTIONEN

Touchbereiche

Rechteckiger Touchbutton
Textausrichtung ändern

#CTA 1,1
Touchschalter Farbverläufe für *normal* und *gedrückt* definieren

#CCR 5, 0,\$00007F,100, 50,\$0000FF,100, 100,\$00007F,100
#CCR 6, 0,\$7F7F00,100, 50,\$FFFF00,100, 100,\$7F7F00,100
Touchschalter Linienstyle für *normal* und *gedrückt* definieren

#CLS 5, \$FFFFFF,100,1
#CLS 6, \$FFFFFF,100,1
Touchschalter Farbverläufe als linear mit 90° Verdrehung definieren


#CFL 5,5,90
#CFL 6,6,90
Touchschalter Buttonstyle definieren

#CBD 1, 5,6, 200,50,10
Touchschalter Textstyle definieren

#CBT 1, 1,1
Touchschalter Proportionsänderung für *gedrückt* definieren

#CBO 1,0,0,0
Elliptischen Touchschalter auf Position 400,240 setzen.

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



Elliptischer Touchbutton
Textausrichtung ändern

#CTA 1,1
Touchschalter Farbverläufe für *normal* und *gedrückt* definieren

#CCR 5, 0,\$00007F,100, 50,\$0000FF,100, 100,\$00007F,100
#CCR 6, 0,\$7F7F00,100, 50,\$FFFF00,100, 100,\$7F7F00,100
Touchschalter Linienstyle für *normal* und *gedrückt* definieren

#CLS 5, \$FFFFFF,100,1
#CLS 6, \$FFFFFF,100,1
Touchschalter Farbverläufe als linear mit 90° Verdrehung definieren


#CFL 5,5,90
#CFL 6,6,90
Touchschalter Buttonstyle definieren

#CBD 1, 5,6, 200,50,10
Touchschalter Textstyle definieren

#CBT 1, 1,1
Touchschalter Proportionsänderung für *gedrückt* definieren

#CBO 1,0,0,0
Elliptischen Touchschalter auf Position 400,240 setzen.

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



UHRZEIT

#WGC

Objektgruppe als Uhr

Kein Linienstyle und weiße Füllung für Stundenzeiger definieren

#CLD 11

#CFC 11, \$FFFFFF, 100

Kein Linienstyle und graue Füllung für Minutenzeiger definieren

#CLD 12

#CFC 12, \$E0E0E0, 100

Kein Linienstyle und rote Füllung für Sekundenzeiger definieren

#CLD 13

#CFC 13, \$FF0000, 100

Farbverlauf erstellen

#CCR 9, 0,\$0,100 100, \$A0A0A0, 100

Farbverlauf und keinen Linienstyle für Verbindungspunkt definieren

#CFR 14, 9

#CLD 14

Integer Register setzen

#VRI 1,(scrW()/2),(scrH()/2)

Bild des Ziffernblattes platzieren

#PPP 5, "Clock3"; R1, R2, 5, 200

Die drei Zeiger und Verbindungspunkt platzieren

#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

Freie Anker für Zeiger definieren

#OAO 4, 15, 6-7

#OAO 1, 20, 8

Position der Zeiger absolut ändern

#OPA R1, R2, 6-8

Objektgruppe erstellen

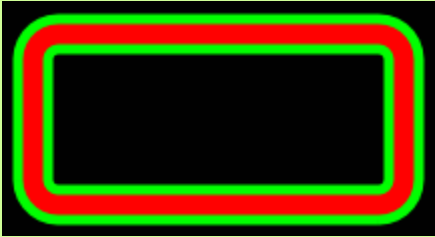
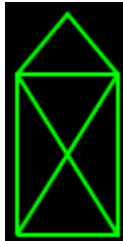
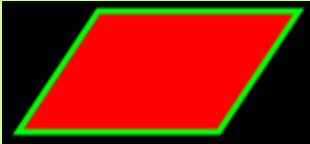
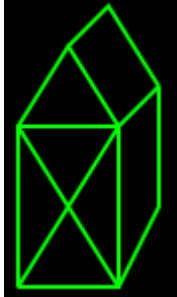
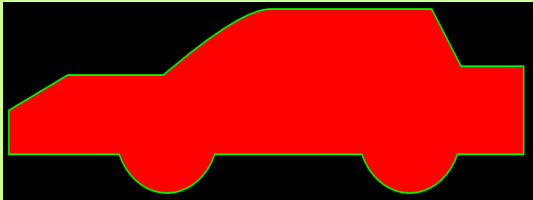
#OGA 10, 5-9

Objektgruppe als Uhr definieren

#WGC 10, 6, 7, 8



ZEICHNEN / GRAFISCHE PRIMITIVE

Grafische Zeichenobjekte	
<p>Abgerundetes Rechteck mit Rand</p> <p>Liniestyle definieren</p> <p>#CLS 1, \$00FF00,100, 5</p> <p>Füllung definieren</p> <p>#CFC 1, \$FF0000,100</p> <p>Abgerundetes Rechteck mit Rand, eine Art Bilderrahmen</p> <p>#GRR 1,1, 400,240,5, 200,100,20, 15</p>	
<p>Polylinie - Nikolaushaus</p> <p>Liniestyle definieren</p> <p>#CLS 1, \$00FF00,100, 2</p> <p>Keine Füllung definieren</p> <p>#CFD 1</p> <p>Mithilfe eines Polypfades ein Nikolaushaus zeichnen</p> <p>#GPL 1,1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,130, 20,100, 70,20</p>	
<p>gefüllte Polylinie - Trapez</p> <p>Liniestyle definieren</p> <p>#CLS 1, \$00FF00,100, 3</p> <p>Füllung definieren</p> <p>#CFC 1, \$FF0000,100</p> <p>Mithilfe eines Polypfades eingefülltes Trapez zeichnen</p> <p>#GPF 1,1, 20,20, 60,80, 160,80, 120,20</p>	
<p>Polylinie erweitern - Nikolaushaus</p> <p>Liniestyle definieren</p> <p>#CLS 1, \$00FF00,100, 2</p> <p>Keine Füllung definieren</p> <p>#CFD 1</p> <p>Nikolaushaus mit Polypfad aus Linien zeichnen</p> <p>#GPL 1,1, 20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45,140, 20,100, 70,20</p> <p>Das Nikolaushaus um die 3 Dimension erweitern</p> <p>#GPA 1, 1,20,20, 20,100, 70,100, 70,20, 20,20, 70,100, 45, 130, 20,100, 70,20</p>	
<p>Polypfad - Auto zeichnen</p> <p>Liniestyle definieren</p> <p>#CLS 1, \$00FF00,100, 2</p> <p>Füllung definieren</p> <p>#CFC 1, \$FF0000,100</p> <p>Ein Auto mit Pfadsegmenten zeichnen</p> <p>#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</p>	

Grafische Zeichenobjekte

6-Eck

Liniestyle definieren

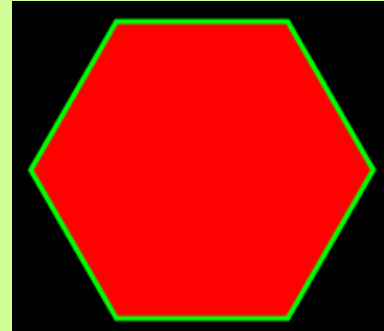
#CLS 1, \$00FF00,100, 3

Füllung definieren

#CFC 1, \$FF0000,100

6 eckiges Polygon um 90° gedreht

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



Stern mit 5 Spitzen

Liniestyle definieren

#CLS 1, \$00FF00,100, 3

Füllung definieren

#CFC 1, \$FF0000,100

Gefüllten Stern zeichnen

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



Reifen

Liniestyle definieren

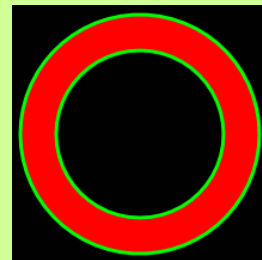
#CLS 1, \$00FF00,100, 3

Füllung definieren

#CFC 1, \$FF0000,100

Kreis mit Loch zeichnen

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



Kreisbogen

Liniestyle definieren

#CLS 1, \$00FF00,100, 3

Füllung definieren

#CFC 1, \$FF0000,100

Ein C als Kreisbogen zeichnen

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



Grafische Zeichenobjekte

Kreissegmente geschnitten

Liniestyle definieren

#CLS 1, \$00FF00, 100, 3

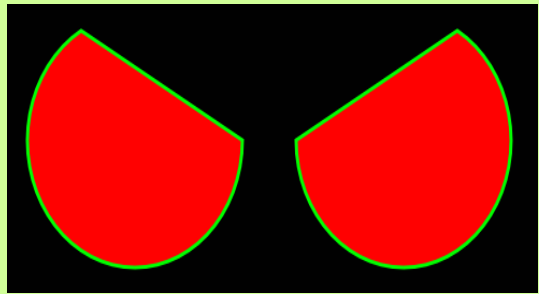
Füllung definieren

#CFC 1, \$FF0000, 100

Zwei geschnittene Kreissegmente zeichnen

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

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



Kuchenstück

Liniestyle definieren

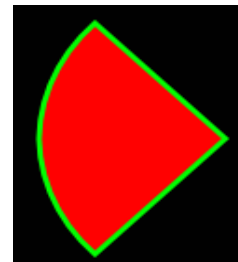
#CLS 1, \$00FF00, 100, 3

Füllung definieren

#CFC 1, \$FF0000, 100

Ein Kuchenstück zeichnen

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



SEGEMENTTYPEN

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

Polypfad Horizontal

Liniestyle definieren

#CLS 1, \$00FF00,100,3

Keine Umrandung definieren

#CFD 1

Horizontale Linie als Polypfad erzeugen

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



Polypfad Vertikal

Liniestyle definieren

#CLS 1, \$00FF00,100,3

Kein Füllung definieren

#CFD 1

Vertikale Linie als Polypfad erzeugen

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



Polypfad Linie

Liniestyle definieren

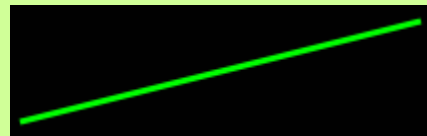
#CLS 1, \$00FF00,100,3

Keine Umrandung definieren

#CFD 1

Linie als Polypfad erzeugen

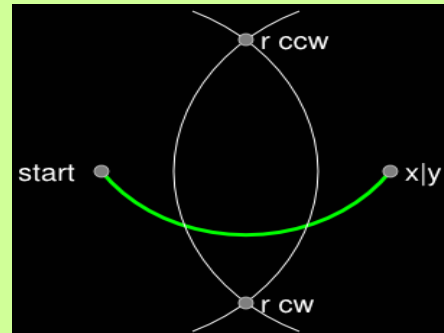
#GPP 1, 1, 50,50, ?L 250,100



Segmente C(ircle), E(llipse)

Polypfad Kreissegment

Grünen Linienstyle definieren
#CLS 5, \$00FF00,100, 3
 Keine Füllung definieren
#CFD 5
 Grauen Linienstyle definieren
#CLS 6, \$BFBFBF,100, 1
 Grau Füllung definieren
#CFC 6, \$7F7F7F,100
 Polypfad als Kreissegment zeichnen
#GPP 1, 5, 200,200,?C0,120,400,200
 Punkte/Kreise erzeugen
#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
 Graue Kreisbögen zeichnen
#GEA 4, 2, 200,200,0, 150,150 -65, 65
#GEA 5, 2, 400,200,0, 150,150, 115, -115
 Textgröße ändern
#CTS 1, 28
 Strings platzieren
#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"



Polypfad Ellipse

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 0,110,150
 Polypfad Ellipse Grün und Grau
#GPP 1, 5, 200,300,?E1,R0,R1,0,400,300
#GPP 4, 2, 200,300,?E2,R0,R1,0,400,300
 Punkte/Kreise setzen
#GET 2, 6, 200,300,5, 5
#GET 3, 6, 400,300,5, 5
 Textgröße ändern
#CTS 1, 28
 Strings setzen
#SSP 8, 1, 180,300,6, "start"
#SSP 11, 1, 410,300,4, "x|y"



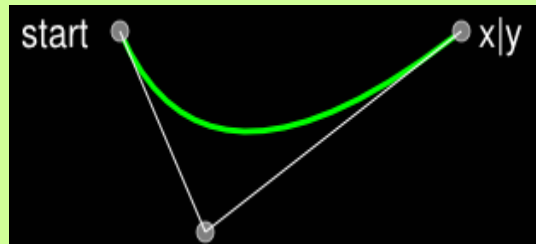
Segmente Q(uadratische Bézier Kurve), R

Polyfad 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 zeichnen
#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"

```



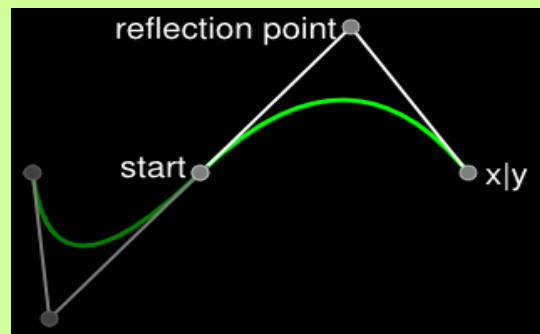
Segmente Q(uadratische Bézier Kurve), R

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"

```



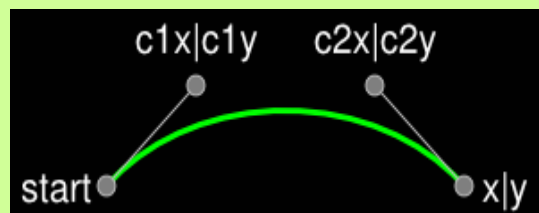
Segmente T, S(pline)

Polyfad 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"

```



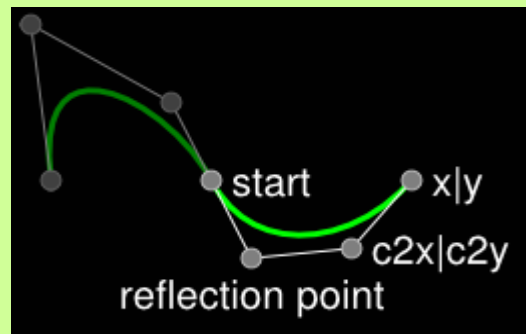
Segmente T, S(pline)

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"

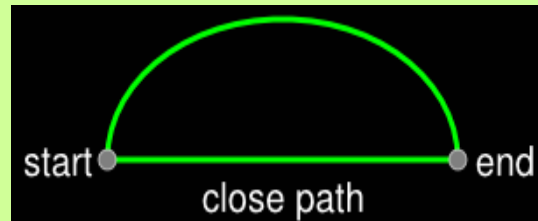
```



Segmente Z, M

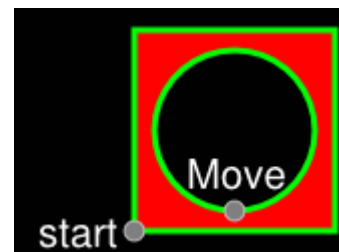
Polypfad schließen

Grünen Linienstyle definieren
#CLS 5, \$00FF00,100, 3
 Keine Füllung definieren
#CFD 5
 Grauen Linienstyle 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 definieren
#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"



ELEKTRISCHE SPEZIFIKATION

Value	Condition	min.	typ.	max.	Unit
operating temperature		-20		+70	°C
storage temperature		-30		+80	°C
storage humidity				90% RH	@ 60°
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	VDD*0.3	V
input high voltage (except USB, I/O)		VDD*0.7	VDD	VDD+0.3	V
output low voltage (except USB, I/O)		-	0	0.4	V
output high voltage (except USB, I/O)		VDD-0.5	VDD	-	V
input/output low voltage I/O		0	0	VDD*0.2	V
input/output high voltage I/O		VDD*0.8	-	VDD	V
output current I/O				25	mA
Input pull-up resistor I ² C				1	MΩ
Brightness	w./o. Touch	700			cd/m ²
	with capacitive Touch				
	with resistive Touch				

MAßZEICHNUNG

