Das PS+ Hilfe-System

Folgende Themen werden in diesem Hilfe-System behandelt:

EINLEITUNG

In einem modernen Programm darf die Möglichkeit zur Erstellung von Makros nicht fehlen. Daher wurde der PS+-Interpreter in GraphicWorks integriert. Mit seiner Hilfe können Sie Makros erstellen, die die Arbeit mit dem Programm sehr stark vereinfachen. Der Vorteil liegt darin, daß ein Makro vollständige Arbeitschritte wie schwierig zu erstellende grafische Objekte oder ständig wiederkehrende Texte ausgeben kann.
Dazu wird die Programmiersprache PS+ verwendet. Diese enthält einige Befehle aus der Seitenbeschreibungssprache PostScript und viele Befehle, die es in PostScript nicht in dieser Form gibt.
Diese Sprache ist auch für den Programmierlaien leicht zu erlernen, da die einzelnen Befehle, nicht zuletzt wegen ihres Namens, leicht zu verstehen sind.
Der PS+-Interpreter wird Ihnen insbesondere dann eine große Hilfe sein, wenn das zu zeichnende Objekt nur mathematisch beschrieben werden kann. Denn dann ist es schwierig, mit den normalen Zeichenfunktionen auszukommen.
Wenn Sie einmal damit begonnen haben, die Hilfe von Makros zu nutzen, werden Sie auf den PS+-Interpreter nicht mehr verzichten wollen.

DER EDITOR

Der Editor dient zur Eingabe von Makroquelltexten. Der Editor läßt sich wie ein einfaches Textverarbeitungssystem handhaben. Beim Interpretieren eines Makros werden hier die Fehlermeldungen ausgegeben. Zudem springt der Editor automatisch in die Zeile des Programms, die den Fehler hervorgerufen hat.

Makros unterbrechen
Makros können im Editor jederzeit durch Betätigen von *Esc* unterbrochen werden. Zu empfehlen ist das bei langen Makros, bei denen Sie am Anfang merken, daß sich ein Fehler eingeschlichen hat.
Ein einfacher, aber effektiver Trick beim Aufspüren von Fehlern besteht darin, nach und nach die korrekt ablaufenden Funktionen durch Einfügen von Prozentzeichen "außer Gefecht" zu setzen. Irgendwann besteht der Quelltext, der beim Aufruf den Fehler produziert, aus wenigen Zeilen, und es ist leicht, den "Störenfried" ausfindig zu machen.

Compilate
Fehlermeldungen werden beim Ausführen des Compilats nicht ausgegeben. Der Grund hierfür ergibt sich aus der höheren Geschwindigkeit des Makros, die u. a. auf einen Verzicht der Fehlerkontrolle beruht. Nur hierdurch ist die auf etwa das Fünffache gesteigerte Geschwindigkeit möglich.
Bevor Sie ein Makro compilieren, sollte das unbedingt im Interpreter ausgiebig getestet werden. Ein compiliertes Makro verfügt über keine Fehlerkontrolle, so daß im Extremfall GraphicWorks abstürzen kann.

FEHLERMELDUNGEN

In der Beschreibung der Funktionen der Menüleiste ist angeklungen, daß Makros mit dem Interpreter, aber auch mit dem Compiler aufgerufen werden können. Der Interpreter hat mehr die Aufgabe, Ihnen die Möglichkeit zu bieten, das Eingegebene direkt auszuprobieren. Beim Programmieren kann sich leicht ein kleiner Fehler einschleichen, und da ist es von Vorteil, wenn dieser direkt erkannt und behoben werden kann. Hierbei sind die angezeigten Fehlermeldungen hilfreich.
Die Meldungen, die Ihnen GraphicWorks immer liefert, wenn ein gravierender Fehler aufgetreten ist, sind im folgenden aufgeführt. Das Auftreten einer dieser Fehler hat sofort den Abbruch des Makros zur Folge

Stack voll
Der Stack kann bis zu 200 Elemente aufnehmen. Das reicht auch für anspruchsvolle Aufgaben aus. Sollte diese Fehlermeldung trotzdem erscheinen, handelt es sich meist um einen fehlerhaften Parameter, der z. B. beim Zeichnen eines Splines verwendet wird.
Typisch ist dieser Fehler auch, wenn versucht wird, einen Text zu interpretieren, der keine Befehlswörter beinhaltet.

Zu viele Schleifen
Es können bis zu zehn Schleifen ineinander verschachtelt werden. Sollte diese Zahl überschritten werden, erscheint die Fehlermeldung. Um Mißverständnissen vorzubeugen: Innerhalb des Programms können nacheinander beliebig viele Scheifen verwendet werden.

Strukturfehler
Strukturfehler treten vor allem beim Eingeben von Schleifen auf. Meistens vergißt man, alle Schleifen mit der geschweiften Klammer zu schließen.

Zu wenig Parameter
Werden vor Aufruf eines PS-Befehls zu wenig Parameter auf dem Stack abgelegt, erscheint diese Fehlermeldung. Oft resultiert dieser Fehler daraus, daß Sie zuerst den Befehl und erst dann die Parameter eingegeben haben.

Zu viele Variablen
Unter PS+ ist es möglich, bis zu 100 Variable gleichzeitig zu verwenden. Sollten Sie die 101. Variable definieren, tritt diese Fehlermeldung auf.

RETURN ohne CALL
Trifft der Programmzeiger während des Ablaufs auf das Kommando Return, springt er aus dem Unterprogramm in die nächsthöhere Ebene. Das gilt nur dann, wenn es sich um ein Unterprogramm handelt, das mit *Return* verlassen wird. Im Hauptprogramm hat das Kommando nichts zu suchen und führt unweigerlich zu dieser Fehlermeldung.

CALL ohne Procedure
Meistens basiert ein Erscheinen dieser Fehlermeldung auf einem simplen Tippfehler.

GraphicWorks sucht das angegebene Unterprogramm, und wenn das nicht gefunden werden kann, kommt es zum Abbruch des Makros.

ELSE ohne IF
Trifft GraphicWorks beim Ausführen eines Makros auf den Befehl ELSE außerhalb einer If-Abfrage, kommt es zum Abbruch des Makros mit dieser Fehlermeldung.

ENDIF ohne IF
Dieser Fehler tritt auf, wenn GraphicWorks bei einer Makroausführung auf das ENDIF-Kommando trifft, ohne daß zuvor das IF-Kommando verwendet wurde.

Division durch Null
Das ist eine Fehlermeldung, die in jedem Taschenrechner zu finden ist. Sie tritt immer auf, wenn Sie versuchen, eine Zahl oder eine Variable durch Null zu teilen. Das gilt auch für die Division durch eine Variable, die die Zahl Null enthält.

Illegales Kommando
Diese Fehlermeldung werden Sie hoffentlich nie zu Gesicht bekommen. Sie meldet nämlich, daß der Makro-Interpreter durch den (unsinnigen) Quelltext so verwirrt wurde, daß ein Abarbeiten des Makros nicht möglich ist. Bisher trat dieser Fehler noch nie auf. Aber dieses Programm versucht alle denkbaren Fehler zu melden und abzufangen.

DIE MENÜLEISTE

In der Menüleiste finden Sie folgende Einträge:

Datei:
Neu
Öffnen
Sichern
Sichern unter
Beenden

Bearbeiten:
Rückgängig
Ausschneiden
Kopieren
Einfügen
Löschen
Alles markieren

Programm:
Interpretieren
Compilieren

?:
Hilfe
Über PS+...

DER MENÜPUNKT "NEU"

Der in Arbeit befindliche Quelltext wird nach einer Sicherheitsabfrage verworfen, so daß Sie sich der Eingabe eines neuen Programmtextes widmen können.

DER MENÜPUNKT "ÖFFNEN"

Nach Auswahl dieses Menüpunkts erscheint die Dateiauswahlbox, und Sie können den zu ladenden Quelltext auswählen. Der Quelltext trägt die Endung PSQ. Geladen werden können beliebige ANSI-Texte.
In GraphicWorks können Makroquelltexte, die mit BeckerDesign ST erstellt wurden, jederzeit eingelesen und ausgeführt werden. In Einzelfällen kann es zu Fehlfunktionen kommen. Das liegt daran, daß die deutschen Sonderzeichen auf dem PC anders definiert sind als auf dem Atari ST. Hier empfiehlt es sich, den Makrotext manuell auf ungewöhnliche Sonderzeichen zu durchsuchen und diese ggf. zu ersetzen.

DER MENÜPUNKT "SICHERN"

Wenn der Programmtext über einen Namen verfügt, können Sie ihn mit diesem Menüpunkt abspeichern. Anderenfalls ist der Menüpunkt inaktiv, und Sie müssen zunächst "Sichern unter" verwenden.

DER MENÜPUNKT "SICHERN UNTER"

Soll der Quelltext erstmals gespeichert werden, ist diese Funktion aufzurufen. Mit der Dateiauswahlbox bestimmen Sie den Namen und den Pfad der Datei.

DER MENÜPUNKT "BEENDEN"

Hiermit beenden Sie die Arbeit an einem PS-Programm. Der Quelltext geht nicht verloren, sondern kann nach erneutem Aufruf des Editors weiterbearbeitet werden, sofern Sie nicht auch das Hauptprogramm beenden.

DER MENÜPUNKT "RÜCKGÄNGIG"

Mit diesem Kommando läßt sich die zuletzt ausgeführte Aktion ungeschehen machen, z. B., wenn versehentlich gelöscht wurde.

DER MENÜPUNKT "AUSSCHNEIDEN"

Bevor diese Funktion aufgerufen werden kann, muß ein Teil des Quelltextes markiert sein, um diesen Textteil in die Zwischenablage zu verschieben.

DER MENÜPUNKT "KOPIEREN"

Soll der markierte Text nur in die Zwischenablage kopiert und nicht verschoben werden, ist der Aufruf dieser Funktion angebracht.

DER MENÜPUNKT "EINFÜGEN"

Mit diesem Kommando wird der Inhalt der Zwischenablage an der aktuellen Cursorposition eingefügt, sofern es sich um Text handelt.

DER MENÜPUNKT "LÖSCHEN"

Überflüssige Teile des Quelltextes lassen sich mit der Maus markieren und mit Hilfe dieses Menüpunkts entfernen.

DER MENÜPUNKT "ALLES MARKIEREN"

Lange Quelltexte können in einem Arbeitsgang markiert werden, um sie z. B. in die Zwischenablage zu kopieren.

DER MENÜPUNKT "INTERPRETIEREN"

Der Aufruf des Interpreters dient normalerweise zur Überprüfung, ob das geschriebene Programm die Erwartungen erfüllt. Die Ausführung der einzelnen Befehle erfolgt zwar in einem akzeptablen Tempo, für aufwendige Programme sollte man aber den Compiler aufrufen. Nachdem das Makro abgelaufen ist, erscheint der Editor wieder auf dem Monitor. Er verdeckt das Arbeitsfenster, so daß Sie das Ergebnis erst betrachten können, nachdem Sie das Editorfenster geschlossen haben. Das Editorfenster kann aber jederzeit bequem wieder aufgerufen werden, indem Sie in der GraphicWorks-Menüzeile im Menü *Makro* erneut auf *Editor* klicken. Der aktuelle Quelltext bleibt im Editor bis Sie GraphicWorks beenden oder der Strom ausfällt.

DER MENÜPUNKT "COMPILIEREN"

Beim Compilieren wird der Programmtext in eine Form überführt, die einen schnelleren Programmablauf zuläßt. Leider läßt sich das Compilat nicht mehr in den Editor laden und bearbeiten.
Wenn Sie den Compiler aufrufen, macht sich dieser direkt an die Arbeit und setzt den Quelltext Zeile für Zeile um. Hiernach erscheint die Dateiauswahlbox, in der Sie den Namen des herzustellenden Makros eingeben. Als Dateiendung ist .MAK vorgesehen.
Wenn Sie die Dateiauswahlbox mit *OK* schließen, wird das Compilat auf Diskette bzw. Festplatte geschrieben und liegt als lauffähiges Makro vor. Bitte beachten Sie, daß Sie den Quelltext separat abspeichern müssen, weil sich das Compilat nicht nachträglich bearbeiten oder gar erweitern läßt.

DER MENÜPUNKT "?"

Hilfe
Diesen Menüpunkt werden Sie vielleicht bereits kennen, da Sie so dieses Hilfe-System aufrufen können.

Über PS+...
Hier wird Ihnen die Version des PS+-Interpreters mitgeteilt. Zudem können Sie feststellen, daß der Editor denselben Autor wie das Hauptprogramm hat.

DIE PS+-BEFEHLE

Falls Sie bereits mit der Version 1.0 von GraphicWorks gearbeitet haben, werden Sie wissen, wie Sie ein Makro erstellen. Anderenfalls seien hier kurz einige grundsätzliche Tips aufgeführt:

Grundsätzliches

In die Version 2.0 sind einige neue Befehle integriert worden. Dadurch ist die Liste der vorhandenen Befehle umfangreicher geworden. Daher ist es sinnvoll, Ihnen zwei verschiedene Methoden anzubieten, wie Sie Hilfe zu einem Befehl erhalten können:

alphabetisch sortiert

thematisch sortiert

Sie haben also die Möglichkeit, Hilfe zu einem bestimmten Befehl zu erhalten oder den Namen des Befehls zu suchen, der die gewünschte Funktion beinhaltet.

GRUNDSÄTZLICHES

Die Handhabung des Interpreters ist relativ einfach. Dazu müssen Sie in der Hauptsache die für Ihr Problem benötigten Befehle kennen.
Die Eingabe eines Befehls macht aber zumeist dem Einsteiger die größten Probleme, da die Formulierung zwar einfach, aber ungewöhnlich ist.
Zur Erläuterung sei Ihnen hier die Vorgehensweise zum Zeichnen einer Linie gezeigt.
Zunächst müssen Sie den Startpunkt der Linie festlegen. Dazu definieren Sie die Position des Zeichenstifts mit dem Befehl moveto. Die Positionsangabe erfolgt *immer in Millimetern* und **vor** dem eigentlichen Befehl. Dies ist bei allen Befehlen der Programmiersprache der Fall. Wenn der Startpunkt 50mm vom oberen und 30mm vom linken Rand entfernt sein soll, geben Sie ein:

```
30 50 moveto
```

Als letztes muß noch der Enpunkt der Linie definiert werden. Dazu können Sie die tatsächlichen (absoluten) Koordinaten auf dem Papier mit Hilfe des Befehls lineto oder den jeweiligen Abstand in x- und y-Richtung vom Startpunkt mit Hilfe des Punkts rlineto angeben. Im ersten Fall lautet die zweite Zeile dann z. B.:

```
100 100 lineto
```

Wenn Sie also beide Zeilen untereinander schreiben und danach den Menüpunkt Interpretieren anwählen, wird eine Linie mit den Koordinaten x1=30, y1=50, x2=100 und y2=100 gezeichnet.
In einem richtigen Makro müssen Sie zuvor noch die entsprechenden Stiftparameter festlegen.
In größeren Makros verwenden Sie den "Stack". Der Stack ist eine Art Stapel, auf den einige Befehle ihre Daten legen. Neue Daten liegen immer oben und werden automatisch verwendet. Alle mathematischen Funktionen legen ihr Ergebnis auf dem Stack ab.
Sie können und werden zumeist mehrere Befehle direkt nacheinander in eine Zeile schreiben. Dabei müssen Sie dann beachten, daß die Befehle von links nach rechts abgearbeitet werden, dementsprechend müssen die zu jedem Befehl gehörenden Parameter angeordnet werden. Dazu ein Beispiel:
Sie wollen z. B. das Wort "Beispiel" an der Position x=100mm und y=100mm ausgeben. Umständlich geht dies so:

```
/ausgabetext (Beispiel) def
100 100 moveto
ausgabetext show
```

Zunächst übergeben Sie hier der Variablen "ausgabetext" das Wort "Beispiel". Danach verschieben Sie den Zeichenstift und geben zuletzt den Inhalt der Variablen aus.
Einfacher geht es so:

```
100 100 moveto (Beispiel) show
```

Diese eine Zeile erfüllt denselben Zweck, allerdings verzichtet sie auf die Verwendung von

Variablen. Auf diese Art und Weise können Sie auch Befehle ineinander verschachteln. Beispiel:

```
100 100 moveto
100 5 sub 2 mul show
```

Klammern helfen hier zu erkennen, warum die Zahl "190" ausgegeben wird. Denn die Befehle werden, wie gesagt, nacheinander abgearbeitet. Ohne die mathematische Reihenfolge zu verändern, können folgende Klammern gesetzt werden:

```
100 100 moveto
(((100 5) sub) 2) mul) show
```

Nun erkennen Sie die Reihenfolge:
1. 100 minus (sub) 5 = 95 (Zwischenergebnis)
2. 95 * (mul) 2 = 190
3. 190 ausgeben (show)

Manchmal reicht es aus, wenn Sie unverständliche Zeilen untereinander schreiben bzw. aufteilen. Dies dürfen Sie fast ohne Einschränkung. Sie dürfen natürlich keine Befehle in einzelne Buchstaben zerlegen. Im obigen Beispiel wäre eine mögliche und sinnvolle Auftrennung diese:

```
100 100 moveto   %(aktuelle Zeichenposition)
100 5 sub        %(schreibt "95" als obersten Wert auf den Stack)
2 mul            %(multipliziert den obersten Stackeintrag mit 2)
show             %(gibt den obersten Stackeintrag aus)
```

Denn spätestens hier erkennen Sie die Reihenfolge der Befehle, da die Zeilen nacheinander abgearbeitet werden. Jede Zeile liefert also ein Zwischenergebnis, wenn Sie dies bei umfangreicheren Formeln durchführen.
Achtung!
Das Programm beachtet keine "Punkt- vor Strichrechnung".
Nun sollten Sie in der Lage sein, einfache Makros zu erstellen. Mit etwas Übung werden Ihre Makros wahrscheinlich immer größer und funktionsgewaltiger.

THEMATISCHE SORTIERUNG

Zur leichteren Übersicht wurden die Befehle in mehrere Gruppen aufgeteilt. Wenn Sie einen bestimmten Befehl suchen, brauchen Sie nur die entsprechende Gruppe anzuwählen.

ARITHMETISCHE BEFEHLE

Arithmetikbefehle und Operatoren haben die Aufgabe, Variablen miteinander zu verknüpfen. Die Arithmetikbefehle sind die Rechenfunktionen, von denen Sie einige in den ersten Schulklassen kennengelernt haben. Es handelt sich um Rechenvorschriften, von den Grundrechenarten bis zu den Winkelfunktionen.

abs	Betragfunktion
acos	arcus Cosinus
add	Addition
asin	arcus Sinus
atn	arcus Tangens
cos	Cosinus
dec	Subtraktion
div	Division
even	Zahl gerade?
idiv	Integer-Division
inc	um eins erhöhen
log	natürlicher Logarithmus
log10	dekadischer Logarithmus
mod	Modulu-Verknüpfung
mul	Multiplikation
neg	Vorzeichenwechsel
odd	Zahl ungerade?
rnd	Zufallszahl
sin	Sinus
sqr	Quadratwurzel
sub	Subtraktion
tan	Tangens
!	Fakultät

BETRAGFUNKTION

abs
Syntax:
num abs

z. B.:
-12 abs *(Ergebnis: 12)*

Funktion:
Es wird der Betrag des Ausdrucks ermittelt. Dieser ist mit dem Zahlenwert ohne Vorzeichen identisch.

ARCUS COSINUS

Syntax:
num acos

z. B.:
0, 707 acos *(Ergebnis: 45°)*

Funktion: Es wird der Winkel des angegebenen Cosinuswerts ermittelt.

ADDITION

Syntax:
num1 num2 add

z.B:
12.5 13.1 add *(Ergebnis: 25.6)*

Funktion:
Beide Ausdrücke werden addiert und das Ergebnis auf den Stack gelegt.

ARCUSSINUS-FUNKTION

Syntax:
num asin

z. B.:
0.5 asin *(Ergebnis: 30)*

Funktion:
Es wird der zum Sinuswert passende Winkel ermittelt und auf den Stack gelegt.

ARCUSTANGENS-FUNKTION

Syntax:
num atn

z. B.:
1 atn *(Ergebnis: 45°)*

Funktion:
Der zum Tangenswert passende Winkel wird ermittelt und auf den Stack gelegt.

COSINUS-FUNKTION

Syntax:
num cos

z. B.:
30 cos *(Ergebnis: 0.866)*

Funktion:
Zum Winkel (360°=ganzer Kreis) wird der passende Cosinuswert ermittelt und auf den Stack gelegt.

SUBTRAKTION

Syntax:
num1 num2 dec

z. B.:
33 22 dec *(Ergebnis: 11)*

Funktion:
Der zweite Ausdruck wird vom ersten subtrahiert und das Ergebnis dem Stack zugeführt.

DIVISION

Syntax:
num1 num2 div

z. B.:
22 5 div *(Ergebnis: 4.4)*

Funktion:
Es wird der erste Ausdruck durch den zweiten dividiert. Das Ergebnis ist auf dem Stack wiederzufinden.

GERADE ZAHL?

Syntax:
num even

z. B.:
33 even *(Ergebnis: 0)*

Funktion:
Diese Funktion überprüft, ob die Zahl gerade ist. Als Ergebnis wird der logische Ausdruck wahr (1) bzw. unwahr (0) auf den Stack geschrieben.

INTEGER-DIVISION

Syntax:
num1 num2 idiv

z. B.:
22 5 idiv *(Ergebnis: 4)*

Funktion:
Diese Funktion führt wie div eine Division der beiden Ausdrücke durch. Das Ergebnis verfügt jedoch über keine Nachkommastellen.

VERGRÖSSERN UM EINS

Syntax:
num inc
z. B.:
23 inc *(Ergebnis: 24)*

Funktion:
Der Ausdruck wird um eins erhöht und auf den Stack geschrieben.

NATÜRLICHER LOGARITHMUS

Syntax:
num log

z. B.:
10 log *(Ergebnis: 2.303)*

Funktion:
Es wird der natürliche Logarithmus des Ausdrucks gebildet und auf den Stack geschrieben.

DEKADISCHER LOGARITHMUS

Syntax:
num log10

z. B.:
100 log10 *(Ergebnis: 2)*

Funktion:
Der dekadische Logarithmus des Ausdrucks wird gebildet und auf dem Stack abgelegt.

MODULU-VERKNÜPFUNG

Syntax:
num1 num2 mod

z. B.:
23 5 mod *(Ergebnis: 3)*

Funktion:
Bei der Modulu-Verknüpfung wird der Rest einer Ganzzahldivision ermittelt und auf den Stack geschrieben.

MULTIPLIKATION

Syntax:
num1 num2 mul

z. B.:
10 3 mul *(Ergebnis: 30)*

Funktion:
Beide Ausdrücke werden miteinander multipliziert und das Ergebnis auf den Stack geschrieben.

VORZEICHENWECHSEL

Syntax:
num neg

z. B.:
125 neg *(Ergebnis: -125)*

Funktion:
Das Vorzeichen der Zahl wird gewechselt. Das Ergebnis kann dem Stack entnommen werden.

UNGERADE ZAHL?

Syntax:
num odd

z. B.:
12 odd *(Ergebnis: -1)*

Funktion:
Es wird geprüft, ob die auf dem Stack befindliche Zahl ungerade ist. In diesem Fall ist das Ergebnis -1.

ZUFALLSZAHL

Syntax:
num rnd

z. B.:
99 rnd *(Ergebnis: Zufallszahl zwischen 0 und 99)*

Funktion:
Die Funktion ermittelt eine Zufallszahl zwischen Null und dem anzugebenden Ausdruck *num*.

SINUS

Syntax:
num sin

z. B.:
30 sin *(Ergebnis: 0.5)*

Funktion:
Es wird der zum Winkel passende Sinuswert ermittelt und auf den Stack geschrieben.

QUADRATWURZEL

Syntax:
num sqr

z. B.:
81 sqr *(Ergebnis: 9)*

Funktion:
Die Funktion ermittelt die Quadratwurzel des Ausdrucks und führt das Ergebnis dem Stack zu.

SUBTRAKTION

Syntax:
num1 num2 sub

z. B.:
66 87 sub *(Ergebnis: -21)*

Funktion:
Beide Ausdrücke werden voneinander abgezogen und das Ergebnis auf dem Stack abgelegt.

TANGENS

Syntax:
num tan

z. B.:
60 tan *(Ergebnis: 1.732)*

Funktion:
Der zum Winkel (Werte kleiner als 90°) gehörige Tangens wird ermittelt.

OPERATOREN

Operatoren führen *logische Verknüpfungen* durch, bei denen es in erster Linie auf die einzelnen Bits der Zahl ankommt.

and	Bitweise Verknüpfung
eq	Identisch?
eqv	Gleichheit prüfen
ge	Größer oder gleich?
gt	Größer?
le	Kleiner oder gleich?
lt	Kleiner?
ne	Ungleich?
not	Wahrheitsprüfung
or	Oder-Verknüpfung
xor	Exklusives Oder

BITWEISES VERKNÜPFEN

Syntax:
num1 num2 and

z. B.:
12 8 and *(Ergebnis: 8)*

Funktion:
Die beiden angegebenen Zahlen oder Variablen werden bitweise miteinander verknüpft. Das Ergebnis wird auf dem Stack ausgegeben.

GLEICHHEITSPRÜFUNG

Syntax:
num1 num2 eqv

z. B.:
22 3 eqv *(Ergebnis: 10)*

Funktion:
Beide Ausdrücke werden bitweise miteinander auf Gleichheit überprüft. Ein Bit wird nur gesetzt, wenn es in beiden verknüpften Ausdrücken identisch ist. Es wird gesetzt, wenn es sowohl im ersten als auch im zweiten Ausdruck gesetzt ist. Gleiches gilt auch, wenn das entsprechende Datenbit in den beiden verglichenen Ausdrücken nicht gesetzt ist. Das Ergebnis der Verknüpfung wird auf den Stack geschrieben. Es ist mit dem identisch, das Sie erhalten, wenn auf das Ergebnis einer XOR-Verknüpfung eine NOT-Verknüpfung angewendet wird.

ODER-VERKNÜPFUNG

Syntax:
num1 num2 or

z. B.:
12 7 or *(Ergebnis: 15)*

Funktion:
Bei diesem Operator geht es um die ODER-Verknüpfung. Das Ergebnis wird auf dem Stack abgelegt.

WAHRHEITSPRÜFUNG

Syntax:
num not

z. B.:
-1 not *(Ergebnis: 0)*

Funktion:
Es wird der Wahrheitswert der Zahl umgekehrt. Als logisch wahr gilt einzig die Zahl -1, so daß diese als einzige das Ergebnis 0 liefert.

EXKLUSIVES ODER

Syntax:
num1 num2 xor

z. B.:
12 8 xor *(Ergebnis: 4)*

Funktion:
Beide Ausdrücke werden einer EXCLUSIV-ODER-Verknüpfung unterzogen. Hierbei werden nur Bits gesetzt, wenn sie in einem der beiden Ausdrücke gesetzt sind. Auch hier wird das Ergebnis auf den Stack gelegt.

IDENTISCH?

Syntax:
num1 num2 eq

z. B.:
12 12 eq *(Ergebnis: -1)*

Funktion:
Es wird ermittelt, ob beide Zahlen identisch sind, und das Ergebnis wird auf den Stack gelegt.

GRÖßER ODER GLEICH?

Syntax:
num1 num2 ge

z. B.:
13 12 ge *(Ergebnis: -1)*

Funktion:
Es wird ermittelt, ob der erste Ausdruck größer oder gleich dem zweiten Ausdruck ist, und das Ergebnis wird auf den Stack gelegt.

GRÖßER?

Syntax:
num1 num2 gt

z. B.:
11 12 gt *(Ergebnis: 0)*

Funktion:
Wenn der erste Ausdruck größer als der zweite ist, wird -1 als Ergebnis ausgegeben.

KLEINER ODER GLEICH?

Syntax:
num1 num2 le

z. B.:
7 7 le *(Ergebnis. -1)*

Funktion:
Falls der erste Ausdruck kleiner oder gleich dem zweiten Ausdruck ist, wird -1 als Ergebnis auf den Stack gelegt.

KLEINER ODER GLEICH?

Syntax:
num1 num2 lt

z. B.:
6 7 lt *(Ergebnis: 0)*

Funktion:
Es wird überprüft, ob der erste Ausdruck kleiner ist als der zweite.

UNGLEICH?

Syntax:
num1 num 2 ne

z. B.:
8 7 ne *(Ergebnis: -1)*

Funktion:
Es wird überprüft, ob die erste Zahl ungleich der zweiten ist. Ist dies der Fall, wird -1 auf dem Stack abgelegt, ansonsten wird die Zahl Null übergeben.

ALPHABETISCH SORTIERTE BEFEHLE

!	Fakultät
#	Unterprogrammname
%	Kommentar
{}	Schleife
abs	Betragfunktion
acos	arcus Cosinus
add	Addition
alert	Dialogbox
and	Bitweise Verknüpfung
arc	Kreisbogen (Gegenuhrzeigersinn)
arcn	Kreisbogen (Uhrzeigersinn)
area	Polygon, Linienzug
asin	arcus Sinus
atn	arcus Tangens
box	Rechteck
call	Unterprogramm aufrufen
circle	Kreis
clear	Stack löschen
closepath	Linie (aktuelle Position)
cmycolor	Farbe (CMY-System)
cos	Cosinus
cursorx	Mauskoordinate (x)
cursory	Mauskoordinate (y)
curveto	Kurve
dec	Subtraktion
def	Variable definieren
dialog	Eingabebox
div	Division
dup	Stackelement duplizieren
e	Euler'sche Zahl
ellipse	Ellipse
else	Anderenfalls...
endcolor	Endfarbe Text (extrudiert)
endif	Bedingungsende
eq	Identisch?
eqv	Gleichheit prüfen
even	Zahl gerade?
exch	Stackelement vertauschen
exit	Schleife verlassen
fbox	Rechteck (gefüllt)
fcircle	Kreis (gefüllt)
fellipse	Ellipse (gefüllt)
findfont	Zeichensatz definieren
findpen	Zeichenstift definieren
fspline	Spline (gefüllt)
ge	Größer oder gleich?
getcursor	Mauskoordinaten
getdate	Datum
getkey	Tastencode
gettime	Uhrzeit
gt	Größer?

hlscolor	Farbe (HLS-System)
idiv	Integer-Division
if	Bedingungsanfang
inc	um eins erhöhen
inp	Tastendruck
input	Eingabefenster
le	Kleiner oder gleich?
lineto	Linie (absolut)
log	natürlicher Logarithmus
log10	dekadischer Logarithmus
lt	Kleiner?
mod	Modulu-Verknüpfung
moveto	Zeichenstift bewegen
mul	Multiplikation
ne	Ungleich?
neg	Vorzeichenwechsel
not	Wahrheitsprüfung
odd	Zahl ungerade?
or	Oder-Verknüpfung
pi	Kreiszahl Pi
pie	Kreisausschnitt (gefüllt)
pop	Stackelement löschen
rbox	Rechteck (relativ)
return	Unterprogramm abschließen
rfbox	Rechteck (gefüllt, relativ)
rgbcolor	Farbe (RGB-System)
rlineto	Linie (relativ)
rmoveto	Zeichenstift bewegen (relativ)
rnd	Zufallszahl
scalefont	Schrifthöhe
setborder	Umrandung
setdash	Linienstil
setfont	Zeichensatz
setgray	Grauwert
setpen	Zeichenstift
setshadow	Texttiefe (extrudiert)
setstyle	Schriftattribut
setwidth	Zeichenbreite
show	Textausgabe
sin	Sinus
spline	Spline
sqr	Quadratwurzel
startcolor	Anfangsfarbe Text (extrudiert)
sub	Subtraktion
tan	Tangens
textrotate	Textwinkel
timer	Timer
workb	Arbeitsbereich (Breite)
workh	Arbeitsbereich (Höhe)
xor	exklusives Oder

FAKULTÄT

Syntax:
num !

z. B.:
9 ! *(Ergebnis: 362880)*

Funktion:
Diese Funktion bildet die Fakultät der Zahl und legt diese auf dem Stack ab.

UNTERPROGRAMME

Syntax:
Name#
...
return

Nach der Zeile "Name#" würde das Unterprogramm mit dem Namen "Name" beginnen.
Ein Unterprogramm enthält zumeist Programmteile, die mehrfach innerhalb des Hauptprogramms Verwendung finden. Aufgerufen wird es aus dem Hauptprogramm des Makros durch:
call Name
Das Ende eines Unterprogramms definiert der Befehl return, wodurch wieder zum Hauptprogramm des Makros zurückgekehrt wird.

Wichtig ist, daß Sie beim Aufruf und bei der Definition des Namens die identische Groß- und Kleinschreibung verwenden, da ansonsten das Unterprogramm nicht gefunden wird.

KOMMENTARE

Syntax:
% Kommentar

z. B.:

```
% Dies ist eine Kommentarzeile
```

oder

```
call procedure        % Unterprogrammaufruf
```

Wenn Sie ein Makro schreiben, ist es oftmals sinnvoll, einige Kommentar einzufügen, um das Makro nach einigen Wochen noch zu verstehen. Eine Zeile, die am Anfang ein "%"-Zeichen enthält, wird vom Interpreter nicht ausgeführt. Sie können also ganze Kommentarzeilen einfügen.
Aber auch hinter einer fertigen Befehlszeile (s. o.) teilen Sie dem Interpreter mit Hilfe des "%"-Zeichens mit, daß alle weiteren Worte in dieser Zeile nicht zum Programm gehören. Diese Form des Kommentars ist sinnvoll, wenn Sie mit Hilfe eines kurzen Kommentars einzelne Zeilen erklären wollen.

SCHLEIFE

*Syntax:*Zahl { ... }

Funktion:
Die geschweiften Klammern umschließen einen Programmteil, der mehrfach durchlaufen werden soll. Die Zahl bzw. Variable vor der Klammer gibt Aufschluß darüber, wie oft dies geschehen soll.

BEDINGTE PROGRAMMAUSFÜHRUNG

Hier finden Sie alle Befehle, die es ermöglichen, Entscheidungen innerhalb des Programms zu treffen.

else	Anderenfalls...
endif	Bedingungsende
if	Bedingungsanfang

BEDINGUNGSANFANG

Syntax:
Bedingung if

z. B.:
-1 if
...
else
...
endif

Funktion:
Der If-Befehl untersucht nur, ob es sich bei der Bedingung um eine logisch wahre oder unwahre Aussage handelt. Nur wenn auf dem Stack 0 (die aus der Bedingung entstammt) zu finden ist, wird der Programmteil hinter dem Kommando nicht ausgeführt.

BEDINGUNGSENDE

Syntax:
Bedingung if
    ...
    else
    ...
    endif

Funktion:
Das Kommando ist unbedingt notwendig, um das Ende einer If-Bedingung zu markieren.

ELSE-BEFEHL

Syntax: Bedingung if
...
else
...
endif

Funktion:
Der Programmtext hinter dem Else-Befehl wird ausgeführt, wenn die Bedingung in der If-Abfrage nicht erfüllt ist.

Beispiel:
```
90 90 moveto
100 50 gt if
größer show
else
kleiner show
endif
(Programmende) alert
```

Das Beispiel untersucht, ob 100 größer als 50 ist. Da das der Fall ist, wird die Meldung "größer" ausgegeben und das Makro mit der Meldung "Programmende" beendet. Würden Sie 30 mit 50 vergleichen lassen, erhielten Sie die Meldung "kleiner", bevor das Ende durch das Dialogfenster gemeldet wird.

EIN- UND AUSGABE

Um in ein laufendes Programm Eingaben machen zu können, verfügt PS+ über Dialogboxen, die sich mit Hilfe spezieller Kommandos aufrufen lassen.

alert	Dialogbox
dialog	Eingabebox
getcursor	Mauskoordinaten
getkey	Tastencode
inp	Tastendruck
input	Eingabefenster
show	Textausgabe

DIALOGBOX

Syntax:

Hinweis$ **alert**

z. B.:

(Das ist ein|Test des PS-Interpreters!) **alert**

Funktion:

Es erscheint ein Hinweis auf dem Monitor. Dieser kann entweder einer String-Variablen entnommen oder direkt in Klammern davorgesetzt werden. Bei längerem Text empfiehlt es sich, die einzelnen Zeilen durch "|" voneinander zu trennen.

DIALOGFENSTER

*Syntax:*Frage$ Antwort$ dialog

z. B.:

(Ergebnis speichern?) (Ja|Nein|Hilfe) dialog

Funktion:
Es wird eine Dialogbox mit bis zu fünf Schaltflächen erzeugt. Auch hier gilt, daß statt der String-Variablen eingeklammerter Text verwendet werden kann. Das Trennen der bis zu fünf Textzeilen erfolgt mit "|". Das gleiche Zeichen wird auch benutzt, um die bis zu fünf Schaltfächen zu definieren.
Das Anklicken einer der Schaltflächen bewirkt das Schließen des Dialogfelds. Welche Schaltfläche angeklickt wurde, kann der auf dem Stack abgelegten Zahl entnommen werden. Wenn Sie "&" bei Angabe des in die Schaltflächen einzutragenden Texts verwenden, ist diese Schaltfläche die aktivierte, ansonsten ist immer die erste aktiviert und wird durch *Return* bzw. *Eingabe* automatisch ausgewählt.

MAUSKOORDINATEN

*Syntax:*getcursor

Funktion:
Das Kommando ermittelt die aktuelle Mausposition und übergibt diese den Systemvariablen cursorx und cursory.

TASTENCODE

*Syntax:*getkey

Funktion:
Mit diesem Befehl wird der ANSI-Code der betätigten Taste ermittelt. Wird keine Taste gedrückt, während der Befehl ausgeführt wird, ist eine Null auf dem Stack zu finden.

TASTENDRUCK

*Syntax:*inp

Funktion:
Das Kommando unterbricht die Programmausführung so lange, bis eine Taste betätigt wird. Der ANSI-Code der Taste ist anschließend dem Stack zu entnehmen.

EINGABEFENSTER

*Syntax:*Frage$ input

z. B.:

(Wie geht|es ihnen?) input

Funktion:
Es wird eine Eingabebox erzeugt, in der eine Zeichenkette oder eine Zahl eingegeben werden kann. Nachdem die Box mit *Return* geschlossen worden ist, ist diese Zeichenkette auf dem Stack wiederzufinden.

TEXTAUSGABE

*Syntax:*Text$ show

z. B.:

Das ist ein Test! show

Funktion:
An der aktuellen Position des Zeichenstifts wird der angegebene Text ausgegeben. Zur Textausgabe verwendet PS die mit setstyle, setfont, scalefont und setwidth eingestellten Parameter.

VARIABLEN

Hier finden Sie alle Befehle, die sich für die Variablenverwaltung und den Stack eignen.

clear	Stack löschen
def	Variable definieren
dup	Stackelement duplizieren
exch	Stackelement vertauschen
getdate	Datum
gettime	Uhrzeit
pop	Stackelement löschen
timer	Timer

STACK LÖSCHEN

*Syntax:*clear

Funktion:
Alle auf dem Stack abgelegten Daten werden mit diesem Kommando gelöscht.

VARIABLE DEFINIEREN

Syntax:/Variable Zahl def

z. B.:
/Anzahl 100 def

Funktion:
Es wird eine Variable mit dem angegebenen Zahlenwert definiert.

oder:

Syntax:/Variable (Text) def

z. B.:
/Wohnort (47799 Krefeld) def

Funktion:
Der Variablen wird hier ein Text zugewiesen.

STACK LÖSCHEN

*Syntax:*dup

Funktion:
Das oberste Element auf dem Stack wird dupliziert, es ist also gleich zweimal hintereinander vorhanden.

STACKELEMENT VERTAUSCHEN

*Syntax:*exch

Funktion:
Die beiden obersten Elemente des Stacks werden miteinander vertauscht.

STACK LÖSCHEN

*Syntax:*getdate

Funktion:
Es wird das aktuelle Datum ermittelt und auf den Stack gelegt.

UHRZEIT

*Syntax:*gettime

Funktion:
Diese Funktion ermittelt die aktuelle Uhrzeit und legt diesen Wert auf dem Stack ab.

STACKELEMENT LÖSCHEN

*Syntax:*pop

Funktion:
Das oberste Stackelement wird gelöscht.

TIMER

*Syntax:*timer

Funktion:
Das Kommando legt den Wert des Timers auf den Stack. Meist wird das Kommando zum Messen von kurzen Zeitunterschieden gebraucht.

GRAFIKBEFEHLE

Die grafischen Befehle ermöglichen die Hauptaufgabe des PS+-Interpreters:

arc	Kreisbogen (Gegenuhrzeigersinn)
arcn	Kreisbogen (Uhrzeigersinn)
area	Polygon, Linienzug
box	Rechteck
circle	Kreis
closepath	Linie (aktuelle Position)
curveto	Kurve
ellipse	Ellipse
fbox	Rechteck (gefüllt)
fcircle	Kreis (gefüllt)
fellipse	Ellipse (gefüllt)
fspline	Spline (gefüllt)
lineto	Linie (absolut)
moveto	Zeichenstift bewegen
pie	Kreisausschnitt (gefüllt)
rbox	Rechteck (relativ)
rlineto	Linie (relativ)
rfbox	Rechteck (gefüllt, relativ)
rmoveto	Zeichenstift bewegen (relativ)
spline	Spline

KREISBOGEN

*Syntax:*xpos ypos radius winkel1 winkel2 arc

z. B.:

100 100 25 200 300 arc

Funktion:
Es wird ein Kreisbogen mit dem angegebenen Radius gezeichnet. Da der Bogen immer gegen den Uhrzeigersinn gezeichnet wird, können Sie durch Vertauschen der beiden Winkelangaben festlegen, welches Segment gezeichnet werden soll.

KREISBOGEN IM UHRZEIGERSINN

*Syntax:*xpos ypos radius winkel1 winkel2 arc

z. B.:

100 200 50 200 300 arc

Funktion:
Es wird ein Kreisbogen im Uhrzeigersinn gezeichnet. Auch hier gilt, daß ein Vertauschen der beiden Winkelangaben das Segment verändert.

POLYGON

*Syntax:*linienzahl area

z. B.:

```
100     120
180     150
200     50
100     120
4 area
```

Funktion:
Dieser Befehl zeichnet einen gefüllten Linienzug (Polygon). Die Positionen der einzelnen Polygonstützpunkte (max. 100 Stück) müssen vor Aufruf dieser Funktion auf dem Stack abgelegt werden.

RECHTECK

*Syntax:*xpos ypos box

z. B.:
100 200 box

Funktion:
Diese Funktion zeichnet ein Rechteck ausgehend von der Position des Zeichenstifts zur angegebenen Position. Nachdem das Rechteck gezeichnet wurde, befindet sich der Zeichenstift an der mit diesem Befehl zu übergebenden Position.

KREIS

*Syntax:*radius circle

z. B.:

100 circle

Funktion:
Es wird an der aktuellen Position des Zeichenstifts ein Kreis gezogen. Der Zeichenstift befindet sich nach Abschluß des Zeichnens in der Kreismitte.

LINIE ZUR AKTUELLEN POSITION

*Syntax:*closepath

Funktion:
Der Befehl veranlaßt das Zeichnen einer Linie von der aktuellen Position des Zeichenstifts zu der zuletzt mit moveto oder rmoveto angesprungenen Position.

KURVE

*Syntax:*xpos1 ypos1 xpos2 ypos2 curveto

Funktion:
Es wird ausgehend von der aktuellen Cursorposition eine Kurve zum Endpunkt *xpos2 ypos2* gezeichnet. Die Kurve durchläuft nicht den durch das erste Koordinatenpaar bestimmten Punkt. Das Koordinatenpaar gibt die Stärke der Krümmung an.

ELLIPSE

*Syntax:*radiusx radiusy ellipse

z. B.:

100 200 ellipse

Funktion:
Es wird eine Ellipse mit den angegebenen Radien um den Ort gezeichnet, an dem sich der Zeichenstift befindet.

GEFÜLLTES RECHTECK

*Syntax:*x y fbox

z. B.:
100 200 fbox

Funktion:
Das Kommando veranlaßt das Zeichnen eines gefüllten Rechtecks von der Position des Zeichenstifts zur angegebenen Position. Dort befindet sich danach auch der Zeichenstift.

GEFÜLLTER KREIS

*Syntax:*radius fcircle

z. B.:

25 fcircle

Funktion:
In dem Ort, an dem sich der Zeichenstift befindet, wird ein gefüllter Kreis gezeichnet. Der Zeichenstift verbleibt im Zentrum des Kreises.

GEFÜLLTE ELLIPSE

*Syntax:*radiusx radiusy fellipse

z. B.:

100 200 fellipse

Funktion:
Hier zeichnet GraphicWorks eine ausgefüllte Ellipse an dem Ort, an dem der Zeichenstift zu finden ist.

GEFÜLLTES SPLINE

*Syntax:*stützpunktanzahl fspline

z. B.:

```
100     120
180     150
200     50
100     120
4 fspline
```

Funktion:
Bevor dieser Befehl aufgerufen wird, müssen diverse Koordinatenpaare auf dem Stack abgelegt werden. Die bis zu 99 Koordinatenpaare stellen die Spline-Stützpunkte dar. Es wird ein ausgefülltes Spline gezeichnet.

LINIE

*Syntax:*xpos ypos lineto

z. B.:
100 200 lineto

Funktion:
Das Kommando veranlaßt das Zeichnen einer Linie von der Position des Zeichenstifts zum angegebenen Punkt.

ZEICHENSTIFT BEWEGEN

*Syntax:*xpos ypos moveto

z. B.:

100 200 moveto

Funktion:
Der Zeichenstift wird an die angegebene Position bewegt.

GEFÜLLTES KREISSEGMENT

*Syntax:*xpos ypos radius winkel1 winkel2 pie

z. B.:

100 200 50 200 300 pie

Funktion:
Es wird ein ausgefülltes Kreissegment gezeichnet, das Ähnlichkeit mit einem Tortenstück hat. Durch Vertauschen der Winkelangaben läßt sich festlegen, welches Segment entgegen dem Uhrzeigersinn gezeichnet werden soll.

RECHTECK (RELATIV)

*Syntax:*breite höhe rbox

z. B.:

100 200 rbox

Funktion:
Dieser Befehl zeichnet eine Box, deren Eckpunkte nicht absolut, sondern relativ angegeben werden. Also nicht die Lage des gegenüberliegenden Eckpunkts wird auf dem Blatt angegeben, sondern wie weit dieser von der Position des Zeichenstifts entfernt ist.

LINIE (RELATIV)

*Syntax:*abstandx abstandy rlineto

z. B.:

100 200 rlineto

Funktion:
Es wird eine Linie zu der relativ angegebenen Position gezeichnet. Auch hier gilt, daß angegeben wird, wieweit der Endpunkt in X- und Y-Richtung vom Ausgangspunkt entfernt ist.

RECHTECK (GEFÜLLT, RELATIV)

*Syntax:*breite höhe rfbox

z. B.:
100 200 rfbox

Funktion:
Mit diesem Kommando wird ein ausgefülltes Rechteck gezeichnet. Das Koordinatenpaar gibt die Höhe und Breite der Box an. Es wird hier also mit relativen Werten gearbeitet.

ZEICHENSTIFT RELATIV BEWEGEN

*Syntax:*abstandx abstandy rmoveto

z. B.:

```
100 30 rmoveto
```

Funktion:
Der Zeichenstift wird um die angegebenen Werte verschoben: relative Ortsangaben.

SPLINE

*Syntax:*Koordinatenpaare Anzahl spline

z. B.:

```
100     120
180     150
200     50
100     120
4 spline
```

Funktion:
Mit den bis zu 100 auf dem Stack abgelegten Koordinatenpaaren wird ein Spline gezeichnet. Der Zeichenstift befindet sich an der Position des letzten angegebenen Koordinatenpaars. Bitte beachten Sie, daß hinter den Koordinatenpaaren deren Anzahl auf dem Stack abgelegt wird, weil ansonsten eine Fehlermeldung erscheint.

PROGRAMMSTRUKTUR

Hier finden Sie alle Befehle, die sich für den strukturierten Aufbau eines Programms eignen.

#	Unterprogramm definieren
{}	Schleife
call	Unterprogramm aufrufen
exit	Schleife verlassen
return	Unterprogramm abschließen

UNTERPROGRAMM AUFRUFEN

*Syntax:*Unterprogramm call

Funktion:
Es wird das genannte Unterprogramm aufgerufen und die Programmausführung dorthin verlegt.

UNTERPROGRAMM ABSCHLIESSEN

*Syntax:*return

Funktion:
Ein Unterprogramm muß mit *return* abgeschlossen werden, weil ansonsten die Programmausführung nicht mehr fortgeführt werden kann.

Beispiel: /zahl 50 def

Addieren call
90 90 moveto
zahl show

Addieren#
/zahl 100 zahl add def
return

SCHLEIFE VERLASSEN

*Syntax:*zahl exit

z. B.:
-1 exit

Funktion:
Wird dem EXIT-Befehl die Zahl -1 übergeben, dann wird die Programmschleife verlassen, in der sich das Programm derzeit befindet. So ist es auch beim Beispielmakro, das mit der Ausgabe der Variablen x beendet wird, sobald x größer ist als die Blattbreite.

Beispiel:
```
/x 0 def
10000 {
/x x 1 add def
x workb gt exit
}
100 100 moveto
x show
```

DEFINITIONSBEFEHLE

Definitionsbefehle ermöglichen die Einstellung der Parameter für Text, Farben und Linienstilen.

cmycolor	Farbe (CMY-System)
endcolor	Endfarbe Text (extrudiert)
findfont	Zeichensatz definieren
findpen	Zeichenstift definieren
hlscolor	Farbe (HLS-System)
rgbcolor	Farbe (RGB-System)
scalefont	Schrifthöhe
setborder	Umrandung
setdash	Linienstil
setfont	Zeichensatz
setgray	Grauwert
setpen	Zeichenstift
setshadow	Texttiefe (extrudiert)
setstyle	Schriftattribut
setwidth	Zeichenbreite
startcolor	Anfangsfarbe Text (extrudiert)
textrotate	Textwinkel

FARBE EINSTELLEN (CMY)

*Syntax:*cyan magenta yellow cmycolor

z. B.:

25 25 25 cmycolor

Funktion:
Das Kommando stellt die angegebenen Farbanteile an der Gesamtfarbe ein (in %).

FARBE EINSTELLEN (HLS)

*Syntax:*grad hell satt hlscolor

z. B.:

359 50 50 hlscolor

Funktion:
Es wird die im HLS-System anzugebende Farbe eingestellt.

FARBE EINSTELLEN (CMY)

*Syntax:*rot grün blau rgbcolor

z. B.:

50 50 50 rgbcolor

Funktion:
Es werden die Farbanteile entsprechend des RGB-Systems eingestellt.

SCHRIFTHÖHE EINSTELLEN

*Syntax:*Höhe scalefont

z. B.:
15 scalefont

Funktion:
Die Schrifthöhe wird auf den angegebenen Wert festgelegt.

OBJEKT UMRANDEN

*Syntax:*Zahl setborder

z. B.:

-1 setborder

Funktion:
Mit diesem Kommando wird die Umrandung von ausgefüllten Objekten (z. B. Kreise oder Rechtecke) ein- bzw. ausgeschaltet.

LINIENSTIL EINSTELLEN

*Syntax:*Zahl setdash

z. B.:
1 setdash

Funktion:
Es wird der angegebene Linienstil aufgerufen. Hierbei gelten folgende Parameter:

1 durchgehende Linie
2 unterbrochene Linie
3 kurze unterbrochene Linien
4 Punktlinie
5 lange unterbrochene Linien
6 Strich-Punkt-Linie 1
7 Strich-Punkt-Linie 2
8 Strich-Punkt-Linie 3
9 Strich-Punkt-Linie 4

FONT WÄHLEN

*Syntax:*Zahl setfont

z. B.:
3 setfont

Funktion:
Es wird der Zeichensatz mit der angegebenen Nummer eingestellt. Sollte die Nummer keinen Zeichensatz definieren können, weil an dieser Position keiner geladen wurde, wird der Standardzeichensatz (Nummer eins) aktiviert. Angewendet wird dieser Befehl meist in Zusammenhang mit dem Befehl findfont.

GRAUWERT EINSTELLEN

*Syntax:*Prozentzahl setgray

z. B.:

33 setgray

Funktion:
Es wird der angegebene Grauwert eingestellt. Hierbei muß auf die Darstellung von Farben verzichtet werden. Ein Grauwert von 100% steht für schwarz, ein Wert von 0% für weiß.

STIFTGRÖSSE EINSTELLEN

*Syntax:*Zahl setpen

z. B.:
2 findpen setpen

Funktion:
Es wird der angegebene Zeichenstift aktiviert.

SCHRIFTATTRIBUTE WÄHLEN

*Syntax:*stil setstyle

z. B.:
normal setstyle

Funktion:
Das Kommando stellt ein Schriftattribut ein. Hierbei kann einer der folgenden Parameter verwendet werden, der vor Aufruf dieses Kommandos auf den Stack gelegt werden muß. Um das eingestellte Attribut auszuschalten, genügt es, das Kommando wieder mit dem verwendeten Parameter aufzurufen.
Folgende Attribute sind möglich:

normal	normale Proportionalschrift
bold	Fettschrift
kursiv	Kursivschrift
mirror	gespiegelte Schrift
fixed	fester Buchstabenabstand
high	hochgestellte Schrift
deep	tiefgestellte Schrift
shadow	extrudierte Schrift

Für extrudierte Schrift sind die Befehle startcolor, endcolor und setshadow zu beachten. Allgemein sind die Parameter für findfont, scalefont, setfont, setstyle, setwidth und textrotate sorgfältig einzustellen. Denn sonst werden die im Hauptprogramm aktiven Parameter übernommen. Diese Tatsache können Sie ausnutzen, um mit einem Makro dieselbe Schrift zu erzeugen, die auch sonst in einer Zeichnung verwendet wurde.

SCHRIFTBREITE EINSTELLEN

*Syntax:*breite setwidth

z. B.:
30 setwidth

Funktion:
Das Kommando stellt die Zeichenbreite auf 1 bis 250% des Standardwerts ein.

SCHREIBWINKEL EINSTELLEN

*Syntax:*Winkel textrotate

z. B.:
230 Textrotate

Funktion:
Das Kommando legt den Winkel fest, mit dem ein Text durch den Befehl "show" ausgegeben werden soll.

FARBVERLAUFSENDE (TEXT)

*Syntax:*rot grün blau endcolor

z. B.:

50 50 50 endcolor

Funktion:
Es werden die Farbanteile für die Endfarbe bei extrudiertem Text entsprechend dem RGB-System eingestellt.

FARBVERLAUFSANFANG (TEXT)

*Syntax:*rot grün blau startcolor

z. B.:

0 100 50 startcolor

Funktion:
Es werden die Farbanteile für die Startfarbe bei extrudiertem Text entsprechend dem RGB-System eingestellt.

ZEICHENSATZ SUCHEN

*Syntax:*name$ findfont

z. B.:

(Times New Roman) findfont setfont

Funktion:
Es wird die Nummer ermittelt, unter der der angegebene Zeichensatz geladen wird. Die Nummer läßt sich dem Stack entnehmen und mit setfont (s. o.) zum Aktivieren des Zeichensatzes verwenden. Ist der genannte Zeichensatz nicht geladen, wird die Zahl Null ermittelt. Einige Systemzeichensätze haben einen Namen, der aus mehreren Worten besteht. Diese Namen müssen Sie in Klammern setzen (s. o.), damit er vom Programm gefunden werden kann. Sonst wird nur das letzte Wort im Namen des Zeichensatzes erkannt. Im obigen Beispiel würde die Zeile

Times New Roman findfont setfont

den Zeichensatz "Roman" suchen und somit den falschen Zeichensatz finden und einstellen.

STIFTBREITE SUCHEN

*Syntax:*Stärke findpen

z. B.:

0.15 findpen setpen

Funktion:
Das Kommando ermittelt die Nummer des Stifts, der über die angegebene Breite verfügt. Das Ergebnis wird auf dem Stack abgelegt. Sollte der passende Stift nicht ermittelt werden, wird derjenige ermittelt, der der angegebenen Linienstärke am nächsten kommt.

EXTRUSIONSPARAMETER

*Syntax:*winkel tiefe setshadow

z. B.:

45 30 setshadow

Funktion:
Das Kommando stellt die Parameter für extrudierten Text ein. Der erste Parameter gibt die Extrusionsrichtung an, möglich sind hier Winkel in 45°-Schritten. Der zweite Wert gibt die Extrusionstiefe an, möglich sind Werte von 1%-100%.

SYSTEMVARIABLEN

In einem Makro ist es häufig notwendig, die Mausposition zu ermitteln oder die genaue Zeichenfläche erkennen zu können. Diese Werte und die vorhandenen mathematischen Konstanten finden Sie bei den Systemvariablen:

workb	Arbeitsbereich (Breite)
workh	Arbeitsbereich (Höhe)
e	Euler'sche Zahl
pi	Kreiszahl Pi
cursorx	Mauskoordinate (x)
cursory	Mauskoordinate (y)

ARBEITSBEREICH (BREITE)

*Syntax:*workb

z. B.:

/breite workb def

Funktion:
Hinter dieser Variablen steckt nichts anderes als die Breite des verfügbaren Arbeitsbereichs. Durch Verändern des Zahlenwerts wird der Bereich verändert, der bearbeitet werden kann. Voreingestellt ist die aktuelle Breite. Zwar lassen sich die Variaben "workb" und "workh" auch mit Werten belegen, die einen größeren Arbeitsbereich definieren als den des eingestellten Blattformats, sinnvoll ist dies aber nicht.

ARBEITSBEREICH (HÖHE)

*Syntax:*workh

z. B.:

/höhe workh def

Funktion:
"Workh" gibt die Höhe des Arbeitsbereichs an. Dieser Zahlenwert kann verändert werden, wodurch sich die verfügbare Arbeitsfläche ändert. Zwar lassen sich die Variaben "workb" und "workh" auch mit Werten belegen, die einen größeren Arbeitsbereich definieren als den des eingestellten Blattformats, sinnvoll ist dies aber nicht.

EULER'SCHE ZAHL

*Syntax:*e

z. B.:

e log show (Ergebnis: 1)

Bei der Zahl "e" geht es um die Euler'sche Zahl mit dem Wert 2.71828... Sie ist die Basis des natürlichen Logarithmus log.

KREISZAHL PI

*Syntax:*pi

Die Zahl "Pi" ist auch als Kreiszahl bekannt und hat einen Wert von 3.141593... Sie benötigen sie eventuell zur Umrechnung, da daß Programm bei den trigonometrischen Funktionen nur mit Gradzahlen rechnet.

MAUSKOORDINATE (X)

*Syntax:*cursorx

z. B.:

```
getcursor
/x cursorx def
```

Funktion:
Diese Variable enthält die X-Position des Mauszeigers innerhalb des Arbeitsblatts. Mit jedem Aufruf von getcursor wird der Wert der Variablen erneut bestimmt.

MAUSKOORDINATE (Y)

*Syntax:*cursory

z. B.:

getcursor
/y cursory def

Funktion:
Diese Variable enthält die Y-Position des Mauszeigers auf dem Arbeitsblatt, nachdem getcursor aufgerufen wurde.