

04 Einblick in PostScript

Behelfsskriptum WS 91/92 mit späteren Ergänzungen

Gliederung

04.1	Allgemeines zu Seitenbeschreibungssprachen.....	2
04.2	Typische Merkmale von PostScript.....	2
04.3	PostScript-Operatoren	6
04.4	PostScript-Standardfonts	15
04.5	Implementierungsbeschränkungen.....	16
04.6	Einleitende Beispiele "PS-01.PS" und "Regula.PS"	17
04.7	Beispiel FHM-Logo "FHM-Lo2.PS"	31
04.8	Beispiel FHM-Logo "FHM-Lo3.PS"	35
04.9	Spotfunktion für die klassische Punktform "Rpm90-kl.PS"	40
04.10	Spotfunktionssammlung "Spot-Fkt.PS"	43
04.11	Beispiel Moduldatei "Modul-0.PS"	51

04.1 Allgemeines zu Seitenbeschreibungssprachen

Seitenbeschreibungssprachen dienen letztlich dazu, auf einem Ausgabegerät (Drucker, Filmbelichter, Digitaldruckmaschine, ..) eine Bitmap der Ausgabeseite zu erzeugen. Dabei werden die Seitenbeschreibungsanweisungen von einem Interpreter, dem Raster Image Processor (RIP), interpretiert.

- Escape-Sequenzen, allgemein
- Esc/P (Epson. Escape/P)
- Interpress (Xerox)
- PCL (Hewlett & Packard. Page Control Language)
- GL (Hewlett & Packard. Graphic Language, für Plotter)
- PreScribe (Kyocera)
- PostScript (Adobe. Dynamische, d.h. programmierbare Seitenbeschreibungssprache)
- PDF (Adobe. Portable Document File Format)
- Die Geschichte von PostScript
 - 1976 Bei "Evans & Sutherland Computer" Entwicklung von "Design System" für CAD-Anwendungen. CAD = Computer Aided Design. Mitentwickler John Warnock.
 - 1978 Bei Rank Xerox Weiterentwicklung von "Design System" unter John Warnock und anderen zu "JaM". Später Aufteilung von "JaM" für CAD-Anwendungen und für Anwendungen in der grafischen Industrie (Warnock). Geburtsstunde von "Interpress".
 - 1982 **John Warnock** und **Chuck Geschke** gründen "**Adobe Systems**" und entwickeln "**PostScript™**".
 - 1985 Vorstellung von "PostScript". Späterer Zusatz: Level 1
 - 1990 Vorstellung von "PostScript Level 2"
 - 1996 Vorstellung von "PostScript Level 3"

04.2 Typische Merkmale von PostScript

- PostScript enthält alle erforderlichen Elemente zur Beschreibung der einzelnen Objekte einer Seite mit Text und Grafiken. Jedes Objekt kann beliebig skaliert und transformiert werden.
- Der Interpreter befindet sich im Ausgabegerät (Standard. Rasterimageprocessor RIP. Original Adobe oder kompatible Clones) oder im Rechner.

- Font-Technologie "**Type-1**". Bézier-Funktionen und Hinting. Durch die drohende Schrift-Monopolstellung entwickelten Apple, Microsoft und andere die alternative Font-Technologie "**True-Type**".
- Für PostScript-Dateien ist die Dateierweiterung (Extension) "**.PS**" Standard.
- PostScript ist eine **case-sensitive Sprache**, d.h. bezüglich der Groß-/Kleinschreibweise verbindlich. Von ganz wenigen (Spezial-) Operatoren abgesehen, werden PostScript-Operatoren klein geschrieben. Bei eigenen Bezeichnern kann die Schreibweise groß/klein beliebig gewählt werden, ist aber zwingend einheitlich zu benutzen.
- PostScript verwendet ausschließlich den **7-Bit-Ascii-Zeichensatz** (Zeichen Nr. 0 bis Nr. dez 127). Damit sind PostScript-Programme plattformunabhängig. Deutsche Umlaute und andere Sonderzeichen im Codebereich über dez 127 können nur durch eine Umkodierung des Zeichensatzes und durch Angabe des Oktal-Codes des Zeichens dargestellt werden. Da der Vorgang etwas aufwendig ist, lohnt es sich, dafür eine eigene Prozedur zu entwickeln, siehe späteres Beispiel "Modul-0.PS".
- Für die Erstellung eines PostScript-Programms benötigt man nur einen Text-Editor.
- PostScript ist völlig formatfrei, von der case-sensitiven Schreibweise abgesehen: Es gibt kein besonderes Trennzeichen für das Ende einer Anweisung. Als Trennzeichen dient alleine das Leerzeichen. Weitere Leerzeichen werden ignoriert, ebenso wie Tabulatorzeichen und Zeilenvorschub. Theoretisch kann ein PostScript-Programm aus einer einzigen Zeile bestehen. Die Verantwortung des Programmieres für ein lesbares Programm ist gefordert.
- Im Holzhacker-Primitivfall kann eine Ausgabe auf einem PostScript-Drucker von der Kommandozeile des DOS-Betriebssystems (Copy von der Console CON auf den Printer PRN) wie folgt erzeugt werden. Die Eingabe des "Programms" muß mit Strg+Z abgeschlossen werden.

COPY CON PRN

```
10 20 moveto
50 80 lineto
stroke
/Helvetica findfont 12 scalefont setfont
35 40 moveto
(FH Muenchen) show
showpage
```

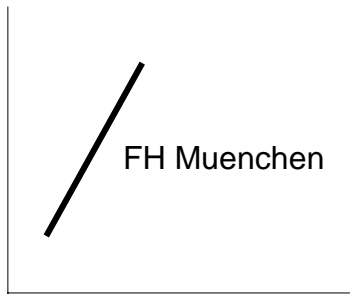
Strg+Z

Dieses "Programm" macht folgendes: Zunächst wird der (unsichtbare Druck-) Cursor mit "moveto" auf den Punkt P_1 ($x = 10$, $y = 20$) gesetzt. Von da aus wird mit "lineto" eine zunächst unsichtbare Linie zum Punkt P_2 ($x = 50$, $y = 80$) in der Standardeinstellung für Strichdicke und -art gezogen. Mit "stroke" wird Linie sichtbar gemacht. Als nächstes wird der Font "Helvetica" gesucht, auf 12 pt skaliert und gesetzt. Dann wird der Cursor auf den Punkt P_3 ($x = 35$, $y = 40$) gesetzt und der String, genauer die Stringkonstante "FH Muenchen" gedruckt. Mit "showpage" wird

die Seite schließlich gedruckt und ausgegeben. Der Koordinaten-Nullpunkt liegt bei PostScript links unten.

Die Standardeinheit in PostScript ist $1/72 \text{ inch} = 25,4 \text{ mm} / 72 = 0,3528 \text{ mm}$.

Die Seite sieht dann mit der linken unteren Blattbegrenzung etwa wie folgt aus:



Zur Stack-Arbeitsweise

PostScript arbeitet stackorientiert und in der Postfix-Notation, d.h. zuerst Operanden, dann Operator. Der Stack ist ein Stapelspeicher und arbeitet somit nach dem LIFO-Prinzip (**L**ast **I**n **F**irst **O**ut). Durch spezielle PostScript-Operatoren kann aber bei Bedarf die Reihenfolge der Operanden auf dem Stack geändert werden. Es existieren in PostScript mehrere Stacks (Operanden-Stack, Dictionary-Stack und Ausführungs-Stack).

Für den Programmierer ist der Operanden-Stack der wichtigste. Er ist 8 Byte breit und in der Tiefe implementierungsabhängig beschränkt, z.B. auf 500 Elemente. Alle Objekte die direkt in die 8-Byte-Breite passen, werden direkt darin abgelegt (Integer, Real, Boolean), bei größeren Objekten (Arrays, Strings usw.) wird ein Zeiger abgelegt. Um die Zeigerverwaltung braucht sich der Programmier nicht kümmern, er kann so tun, als würden die Objekte direkt auf dem Stack liegen. Die Objekte auf den Stack können völlig typverschieben sein, das gilt übrigens auch für Arrays, was bei streng typisierten Sprachen wie Pascal und C unvorstellbar ist. Der PostScript-Programmierer ist selbst dafür verantwortlich, dass auf das Objekt nur typgerechte Operatoren angewendet werden.

Die folgenden schematische Skizze zeigt unter Vorwegnahme der PostScript-Grundrechnungsoperatoren "add" (= addiere), "sub" (= subtrahiere), "mul" (= multipliziere) und "div" (= dividiere) die Funktionsweise des (Operanden-) Stacks mit einer einfachen arithmetischen Aufgabe:

Aufgabe: $((6 + 4) * 3 - 20) / 2$

Auf der Stackspitze *liege* vor der Berechnung das Objekt XX; darunter befinden sich evtl. auch noch weitere Objekte. Der Stack kann aber auch vollkommen leer sein.

Eingabe:		6	4	add	3	mul	20	sub	2	div
Stackspitze	XX	6	4	10	3	30	20	10	2	5
		XX	6	XX	10	XX	30	XX	10	XX
			XX		XX		XX		XX	

Eine Alternative für die gleiche Aufgabe:

Eingabe:		3	6	4	add	mul	-20	add	2	div
Stackspitze	XX	3	6	4	10	30	-20	10	2	5
		XX	3	6	3	XX	30	XX	10	XX
			XX	3	XX		XX		XX	
				XX						

Man beachte bei dieser Alternative, dass nach dem Minuszeichen kein Leerzeichen stehen darf!

Das Ergebnis 5 befindet sich in beiden Fällen nach der Berechnung auf der Stackspitze.

Die Stack-Operatoren

Wenn man von einer "üblichen" Programmiersprache kommt, erscheinen einige Operatoren zur Änderung der Reihenfolge der Objekte auf dem Stack mitunter etwas gewöhnungsbedürftig. Sie werden deshalb hier ausführlich dargestellt.

dup	vor	nach	Dupliziert oberstes Stack-Objekt
Beispiel: X Y dup	Y	Y	
	X	Y	
		X	
exch	vor	nach	Vertauscht die beiden obersten Stack-Objekte
Beispiel: X Y exch	Y	X	
	X	Y	
pop	vor	nach	Löscht das oberste Stack-Objekt
Beispiel: X Y pop	Y	X	
	X		
clear	vor	nach	Löscht <i>alle</i> Stack-Objekte
Beispiel: X Y clear	Y	. / .	Das Symbol ". / ." soll hier das Stack-Ende kennzeichnen
	X		
m n roll	vor	nach	Rollt die obersten <i>m</i> -Objekte <i>n</i> -mal
Beispiel: X Y Z 3 1 roll <i>m</i> = 3, <i>n</i> = +1	Z	Y	Bei negativem <i>n</i> wird in der anderen Richtung gerollt. Beispiel: X Y Z 3 -1 roll ergibt:
	Y	X	X
	X	Z	Z
			Y
n copy	vor	nach	Dupliziert die obersten <i>n</i> -Stack-Objekte
Beispiel: X Y 2 copy	Y	Y	
	X	X	
		Y	
		X	

n index	vor	nach	Dupliziert das <i>n</i> -te Objekt auf die Stack-Spitze
Beispiel: X Y Z 2 index	X	Z	Wichtig: Der Stack-Index beginnt mit 0 und nicht mit 1!
	Y	X	
	Z	Y	
		Z	
count	vor	nach	Zählt Stack-Objekte und legt Wert auf die Stack-Spitze
Beispiel: clear X Y count	Y	2	Das Symbol "/*." steht hier für Stack-Ende. Zählung der Objekte ab 1.
	X	Y	
	/*.	X	
		/*.	
mark	vor	nach	Legt Markierung (Mark-Objekt) auf Stackspitze
Beispiel: X Y mark	Y	mark	Anwendung mit counttomark und cleartomark
	X	Y	
		X	
counttomark	vor	nach	Zählt Objekte bis zum obersten Mark-Objekt
Beispiel: X mark Y Z counttomark	Z	2	Anwendung mit mark
	Y	Z	
	mark	Y	
	X	mark	
		X	
cleartomark	vor	nach	Löscht alle Objekte von der Stackspitze bis zur Markierung und auch diese.
Beispiel: X Y mark U V cleartomark	V	Y	Anwendung mit mark
	U	X	
	mark		
	Y		
	X		

04.3 PostScript-Operatoren und Fehlermeldungen

Es folgt eine Kurzbeschreibung *aller* PostScript-Operatoren in Anlehnung an Adobe PostScript Handbuch (roter Band, 2. Auflage). Die Kurzbeschreibung soll lediglich einen Überblick über den Leistungsumfang von PostScript gewähren. Für konkrete Anwendungen ist die ausführliche Beschreibung in dem erwähnten Handbuch notwendig.

Zur Syntax der Operatorenbeschreibung:

<i>Operand₁ Operand₂ ... Operand_n Operator Resultat₁ Resultat₂ ... Resultat_m</i>
--

Die Punktfolge ... symbolisiert beliebige Wiederholungen. Vor der Ausführung liegt *Operand_n* zuoberst auf dem Stack, nach der Ausführung dagegen *Resultat_m*. Die Operanden oder die Ergebnisse können je nach Operator auch fehlen.

Verwendete Symbole in der Operatorenbeschreibung:

<i>int</i>	Integer-Objekt
<i>real</i>	Real-Objekt
<i>num</i>	beliebiges Zahlenobjekt (Integer oder Real)
<i>any</i>	beliebiges Objekt
<i>proc</i>	PostScript-Prozedur (ausführbares Array)
<i>bool</i> <i>int</i>	entweder Boolean- oder Integer-Objekt
<i>font</i>	Dictionary
‡	Boden des Stacks (Stack-Ende)

Hinzu kommen weitere Symbole, die aber selbsterklärend sind.

Die Stack-Operatoren

<i>any</i>	pop	Löscht oberstes Element
<i>any</i> ₁ <i>any</i> ₂	exch <i>any</i> ₁ <i>any</i> ₂	Exchange. Vertauscht die beiden obersten Stack-Elemente
<i>any</i>	dup <i>any</i> <i>any</i>	Dupliziert oberstes Element
<i>any</i> ₁ ... <i>any</i> _n <i>n</i>	copy <i>any</i> ₁ ... <i>any</i> _n <i>any</i> ₁ ... <i>any</i> _n	Dupliziert die obersten <i>n</i> Elemente
<i>any</i> _n ... <i>any</i> ₀ <i>n</i>	index <i>any</i> _n ... <i>any</i> ₀ <i>any</i> _n	Dupliziert beliebiges Element
<i>any</i> _{n-1} ... <i>any</i> ₀ <i>n</i> <i>j</i>	roll <i>any</i> _(j-1) mod <i>n</i> ... <i>any</i> ₀ <i>any</i> _{n-1} ... <i>any</i> _j mod <i>n</i>	Rotiert <i>n</i> Elemente <i>j</i> mal
‡ <i>any</i> ₁ ... <i>any</i> _n	clear ‡	Löscht Stack komplett
‡ <i>any</i> ₁ ... <i>any</i> _n	count ‡ <i>any</i> ₁ <i>any</i> _n <i>n</i>	Zählt Stack-Elemente
	mark <i>mark</i>	Legt <i>mark</i> -Objekt auf Stack
<i>mark</i> <i>obj</i> ₁ ... <i>obj</i> _n	cleartomark	Löscht bis <i>mark</i>
<i>mark</i> <i>obj</i> ₁ ... <i>obj</i> _n	counttomark <i>mark</i> <i>obj</i> ₁ ... <i>obj</i> _n <i>n</i>	Zählt Objekte bis oberst. Mark-Objekt

Die arithmetischen Operatoren

<i>num</i> ₁ <i>num</i> ₂	add <i>sum</i>	<i>num</i> ₁ plus <i>num</i> ₂
<i>num</i> ₁ <i>num</i> ₂	sub <i>difference</i>	<i>num</i> ₁ minus <i>num</i> ₂
<i>num</i> ₁ <i>num</i> ₂	mul <i>product</i>	<i>num</i> ₁ mal <i>num</i> ₂
<i>num</i> ₁ <i>num</i> ₂	div <i>quotient</i>	<i>num</i> ₁ dividiert durch <i>num</i> ₂
<i>int</i> ₁ <i>int</i> ₂	idiv <i>quotient</i>	Integerdivision: <i>int</i> ₁ idiv <i>int</i> ₂
<i>int</i> ₁ <i>int</i> ₂	mod <i>remainder</i> (restwert)	Integer-Modulo: <i>int</i> ₁ mod <i>int</i> ₂
<i>num</i> ₁	neg <i>num</i> ₂	Negierung
<i>num</i> ₁	abs <i>num</i> ₂	Absolutwert
<i>num</i> ₁	ceiling <i>num</i> ₂	Nächstgrößere ganze Zahl

<i>num</i> ₁	floor <i>num</i> ₂	Nächstkleinere ganze Zahl
<i>num</i> ₁	round <i>num</i> ₂	Runden
<i>num</i> ₁	truncate <i>num</i> ₂	Abschneiden
<i>num</i>	sqrt <i>real</i>	Quadratwurzel
<i>angle</i>	sin <i>real</i>	Sinus. Winkel in Grad
<i>angle</i>	cos <i>real</i>	Cosinus. Winkel in Grad
<i>num</i> ₁ <i>num</i> ₂	atan <i>angle</i>	Arcustangens in Grad (real)
<i>num</i>	ln <i>real</i>	Natürlicher Logarithmus
<i>num</i>	log <i>real</i>	Dezimallogarithmus
<i>base</i> <i>exponent</i>	exp <i>real</i>	Potenzfunktion <i>base</i> ^{<i>exponent</i>}
	rand <i>int</i>	Zufallszahl
<i>int</i>	srand	Zufallszahlengenerator starten
	rrand <i>int</i>	Startwert für Zufallszahlengenerator

Die Array-Operatoren

<i>int</i>	array <i>array</i>	Erstellt Array der Länge <i>int</i>
	[<i>mark</i>	Beginn einer Array-Definition
<i>mark</i> <i>obj</i> ₀ ... <i>obj</i> _{<i>n</i>-1}] <i>array</i>	Ende einer Array-Definition
<i>array</i>	length <i>int</i>	Zahl der Elemente in <i>array</i>
<i>array</i> <i>index</i>	get <i>any</i>	Hole Array-Element
<i>array</i> <i>index</i> <i>any</i>	put	Array-Element einfügen
<i>array</i> <i>index</i> <i>count</i>	getintervall <i>subarray</i>	Sub-Array holen
<i>array</i> ₁ <i>index</i> <i>array</i> ₂	putintervall	Sub-Array einfügen
<i>array</i>	aload <i>array</i> ₀ ... <i>array</i> _{<i>n</i>-1} <i>array</i>	Alle Elemente von <i>array</i> aus Stack legen
<i>any</i> ₀ ... <i>any</i> _{<i>n</i>-1} <i>array</i>	astore <i>array</i>	Stackelemente in <i>array</i> schreiben
<i>array</i> ₁ ... <i>array</i> _{<i>n</i>}	copy <i>subarray</i> ₂	Elemente aus <i>array</i> ₁ nach <i>array</i> ₂ kopieren
<i>array</i> <i>proc</i>	forall	<i>proc</i> für alle Elemente in <i>array</i> ausführen

Die Dictionary-Operatoren

<i>dict</i>	begin	Dictionary mit Platz für <i>n</i> Elemente definieren
<i>dict</i>	length <i>int</i>	Zahl der Einträge in <i>dict</i>
<i>dict</i>	maxlength <i>int</i>	Maximalzahl der Einträge in <i>dict</i>
<i>dict</i>	begin	<i>dict</i> auf Dictionary-Stack legen
	end	Oberstes Element des Dictionary-Stack löschen
<i>key</i> <i>value</i>	def	<i>key</i> mit <i>value</i> im aktuellen Dictionary assoziieren
<i>key</i>	load <i>value</i>	<i>key</i> in allen Dictionaries suchen
<i>key</i> <i>value</i>	store	Oberste Definition von <i>key</i> ersetzen
<i>dict</i> <i>key</i>	get <i>any</i>	Wert von <i>key</i> in <i>dict</i> finden
<i>dict</i> <i>key</i> <i>value</i>	put	<i>key</i> mit <i>value</i> in <i>dict</i> assoziieren
<i>dict</i> <i>key</i>	known <i>bool</i>	<i>key</i> in <i>dict</i> definiert?
<i>key</i>	where <i>dict</i> <i>true</i> oder <i>false</i>	Dictionary finden, in dem <i>key</i> definiert ist
<i>dict</i> ₁ <i>dict</i> ₂	copy <i>dict</i> ₂	Inhalt von <i>dict</i> ₁ nach <i>dict</i> ₂ kopieren

<i>array proc</i>	forall	<i>proc</i> für jedes Element in <i>dict</i> ausführen
	errordict dict	errordict auf den Operanden-Stack legen
	systemdict dict	systemdict auf Operanden-Stack legen
	userdict dict	userdict auf Operanden-Stack legen
	currentdict dict	Aktuelles Dictionary auf Operanden-Stack legen
	countdictstack int	Elemente des Dictionary-Stacks zählen
<i>array</i>	dictstack subarray	Dictionary-Stack in <i>array</i> kopieren

Die String-Operatoren

<i>int</i>	string string	String der Länge <i>int</i> definieren
<i>string</i>	length int	Länge von <i>string</i>
<i>string index</i>	get int	Indiziertes Element holen
<i>string index any</i>	put	Element in <i>string</i> schreiben
<i>string index count</i>	getintervall substring	Substring aus count Elementen beginnend bei <i>index</i> holen
<i>string₁ index string₂</i>	putintervall	Substring, beginnend bei <i>index</i> , durch <i>string₂</i> ersetzen
<i>string₁ string₂</i>	copy substring₂	<i>string₁</i> nach <i>string₂</i> kopieren
<i>string proc</i>	forall	<i>proc</i> für alle Elemente in <i>string</i> ausführen
<i>string seek</i>	anchorsearch post match true oder false	Beginnt <i>string</i> mit <i>seek</i> ?
<i>string seek</i>	search post match pre true oder false	<i>seek</i> in <i>string</i> suchen
<i>file</i>	token any true oder false	Liest token von Beginn von <i>string</i>

Die logischen und Vergleichsoperatoren. Abk: *str* = string

<i>any₁ any₂</i>	eq bool	Gleich? =
<i>any₁ any₂</i>	ne bool	Ungleich? <>
<i>num₁ str₁ num₂ str₂</i>	ge bool	Größer oder gleich? >=
<i>num₁ str₁ num₂ str₂</i>	gt bool	Größer als? >
<i>num₁ str₁ num₂ str₂</i>	le bool	Kleiner oder gleich? <=
<i>num₁ str₁ num₂ str₂</i>	lt bool	Kleiner als? <
<i>bool₁ int₁ bool₂ int₂</i>	and bool₃	Logisches oder bitweises AND
<i>bool₁ int₁ bool₂ int₂</i>	not bool₂ int₂	Logisches oder bitweises NOT
<i>bool₁ int₁ bool₂ int₂</i>	or bool₃ int₃	Logisches oder bitweises OR
<i>bool₁ int₁ bool₂ int₂</i>	xor bool₃ int₃	Logisches oder bitweises XOR
	true true	Wert 'true' auf Stack legen
	false false	Wert 'false' auf Stack legen
<i>int₁ shift</i>	bitshift int₂	<i>int₁</i> um <i>shift</i> -Positionen bitweise schieben. Wenn <i>shift</i> >= 0 nach links schieben, sonst nach rechts

Die Steuerungsoperatoren

<i>any</i>	exec	Operanden auf Ausführungsstack legen und unmittelbar ausführen
<i>bool proc</i>	if	Falls <i>bool</i> wahr, <i>proc</i> ausführen
<i>bool proc₁ proc₂</i>	ifelse	Falls <i>bool</i> wahr, <i>proc₁</i> ausführen, sonst <i>proc₂</i>
<i>initial increment limit proc</i>	for	<i>proc</i> mit Werten von <i>initial</i> bis <i>limit</i> mit Schrittweite <i>increment</i> ausführen
<i>int proc</i>	repeat	<i>proc</i> <i>int</i> -mal ausführen
<i>proc loop</i>	loop	<i>proc</i> endlos oft ausführen
	exit	Innerste Schleife abbrechen
	stop	stopped -Kontext beenden
<i>any</i>	stopped <i>bool</i>	Kontext etablieren, um stop abfangen zu können
	countexecstack <i>int</i>	Elemente des Exec-Stacks zählen
<i>array</i>	execstack <i>subarray</i>	Exec-Stack nach <i>array</i> kopieren
	quit	Interpreter anhalten
	start	Wird beim Interpreter-Start-Up ausgeführt

Die Typ-, Attributs- und Konversionsoperatoren

<i>any</i>	type <i>name</i>	Typ von <i>any</i> ermitteln
<i>any</i>	cvlit <i>any</i>	Objekt literal machen
<i>any</i>	cvx <i>any</i>	Objekt ausführbar machen
<i>any</i>	xcheck <i>bool</i>	<i>any</i> ausführbar?
<i>array file string</i>	executeonly <i>array</i>	Zugriff auf 'ausführbar' beschränken
<i>array dict file string</i>	noaccess <i>array</i>	Jeden Zugriff verbieten
<i>array dict file string</i>	readonly <i>array</i>	Zugriff auf 'Lesen' beschränken
<i>array dict file string</i>	rcheck <i>bool</i>	Lesezugriff erlaubt?
<i>array dict file string</i>	wcheck <i>bool</i>	Schreibzugriff erlaubt?
<i>num string</i>	cvi <i>integer</i>	In Integer konvertieren
<i>string</i>	cvn <i>name</i>	In Name konvertieren
<i>num string</i>	cvr <i>real</i>	In Real konvertieren
<i>num radix string</i>	cvrs <i>substring</i>	In String zur Basis <i>radix</i> konvertieren
<i>any string</i>	cvs <i>substring</i>	In String konvertieren

Die File-Operatoren

<i>string₁ string₂</i>	file <i>file</i>	Datei <i>string₁</i> mit Zugriff <i>string₂</i> öffnen
<i>file</i>	closefile	Datei schließen
<i>file</i>	read <i>byte true</i> oder <i>false</i>	Ein Zeichen aus <i>file</i> lesen
<i>file int</i>	write	Ein Zeichen in <i>file</i> schreiben
<i>file string</i>	readhexstring <i>substring bool</i>	Hex-Zeichen aus <i>file</i> in <i>string</i> lesen
<i>file string</i>	writehexstring <i>substring bool</i>	Hex-String in <i>file</i> schreiben
<i>file string</i>	readstring <i>substring bool</i>	<i>string</i> aus <i>file</i> lesen
<i>file string</i>	writestring	<i>string</i> in <i>file</i> schreiben
<i>file string</i>	readline <i>substring bool</i>	Zeile aus <i>file</i> lesen
<i>file</i>	token <i>any true</i> oder <i>false</i>	Token aus <i>file</i> lesen

<i>file</i>	bytesavailable <i>int</i>	Anzahl der verfügbaren Zeichen
	flush	Gepufferte Daten ausgeben
<i>file</i>	flushfile	Gepufferte Daten ausgeben oder bis EOF lesen
<i>file</i>	resetfile	Löscht gepufferte Ein-/Ausgabedaten
<i>file</i>	status <i>bool</i>	File-Status
<i>string</i>	run	Inhalt von <i>file</i> ausführen, das mit <i>string</i> spezifiziert ist
	currentfile <i>file</i>	Aktuelles File
<i>string</i>	print	<i>string</i> auf Standard-Ausgabe-File schreiben
<i>any</i>	=	Textdarstellung von <i>any</i> auf Standard-Ausgabe-File ausgeben
‡ <i>any</i> ₁ ... <i>any</i> _n	stack ‡ <i>any</i> ₁ ... <i>any</i> _n	Stack mittels '=' ausgeben
<i>any</i>	==	Syntaktische Darstellung von <i>any</i> auf Standard-Ausgabe-File ausgeben
‡ <i>any</i> ₁ ... <i>any</i> ₂	pstack ‡ <i>any</i> ₁ ... <i>any</i> _n	Stack mittels '==' ausgeben
	prompt	Prompt-Zeichen ausgeben
<i>bool</i>	echo	'Echo' an-/ausschalten

Die VM-Operatoren (VM = virtual memory)

	save <i>save</i>	VM-Status speichern
<i>save</i>	restore	VM-Status wiederherstellen
	vmstatus <i>level used max</i>	VM-Status beschreiben

Verschiedene Operatoren

<i>proc</i>	bind <i>proc</i>	Ersetzt Operator-Namen in <i>proc</i> durch Operatoren
	null <i>null</i>	Null auf Stack legen
	usertime <i>int</i>	Benutzerzeit in Millisekunden
	version <i>string</i>	Versionsnr. (String) des PS-Interpretes

Die Graphics-State-Operatoren

	gsave	Graphics-State sichern
	grestore	Graphics-State wiederherstellen
	grestoreall	Untersten Graphics-State wiederherstellen
	initgraphics	Graphics-State initialisieren
<i>num</i>	setlinewidth	Linienstärke setzen
	currentlinewidth <i>num</i>	Linienstärke lesen
<i>int</i>	setlinecap	Form der Linienenden setzen
	currentlinecap <i>int</i>	Form der Linienenden lesen
<i>int</i>	setlinejoin	Form der Linienübergänge setzen
	currentlinejoin <i>int</i>	Form der Linienübergänge lesen
<i>num</i>	setmiterlimit	Form der Linienspitzen setzen

	currentmiterlimit <i>num</i>	Form der Linienspitzen lesen
<i>array offset</i>	setdash	Strichmuster setzen
	currentdash <i>array offset</i>	Strichmuster lesen
<i>num</i>	setflat	Weichheit (Flatness) der Kurvenübergänge setzen
	currentflat <i>num</i>	Weichheit der Kurvenübergänge lesen
<i>num</i>	setgray	Grauwert setzen, 0: schwarz, 1: weiß
	currentgray <i>num</i>	Grauwert lesen
<i>hue saturation brightness</i>	sethsbcolor	HSB-Farben setzen
	currenthsbcolor <i>hue saturation brightness</i>	HSB-Farben lesen
<i>red green blue</i>	setrgbcolor	RGB-Farben setzen
	currentrgbcolor <i>red green blue</i>	RGB-Farben lesen
<i>frequency angle proc</i>	setscreen	Rasterparameter setzen, <i>proc</i> = Spotfkt.
	currentscreen <i>frequency angle proc</i>	Rasterparameter lesen
<i>proc</i>	settransfer	Transferfunktion setzen
	currenttransfer <i>proc</i>	Transferfunktion lesen

Die Matrix-Operatoren

	matrix <i>matrix</i>	Erzeugt Einheitsmatrix
	initmatrix	Setzt CTM (Current Transformation Matrix) auf Initialwert
<i>matrix</i>	identmatrix <i>matrix</i>	Füllt <i>matrix</i> mit den Werten der Einheitsmatrix
<i>matrix</i>	defaultmatrix <i>matrix</i>	Füllt <i>matrix</i> mit den Werten der Initial-Gerätematrix
<i>matrix</i>	currentmatrix <i>matrix</i>	Füllt <i>matrix</i> mit den Werten der CTM
<i>matrix</i>	setmatrix	Ersetzt CTM durch <i>matrix</i>
t_x t_y	translate	Verschiebt Ursprung um (t_x, t_y)
s_x s_y	scale	Skaliert Koordinatensystem mit s_x, s_y
<i>angle</i>	rotate	Rotiert Koordinatensystem um <i>angle</i> (in Grad, entgegen Uhrzeigersinn)
<i>matrix</i>	concat	Ersetzt CTM durch <i>matrix</i> * CTM
<i>matrix</i> ₁ <i>matrix</i> ₂ <i>matrix</i> ₃	concatmatrix <i>matrix</i> ₃	Füllt <i>matrix</i> ₃ mit <i>matrix</i> ₁ * <i>matrix</i> ₂
x y	transform x' y'	Transformiert (x, y) mittels CTM
x y <i>matrix</i>	transform x' y'	Transformiert (x, y) mittels <i>matrix</i>
dx dy	dtransform dx' dy'	Transformiert die Distanz (dx, dy) mittels CTM
x y <i>matrix</i>	dtransform dx' dy'	Transformiert die Distanz (dx, dy) mittels <i>matrix</i>
x' y'	itransform x y	Transformiert (x', y') mittels der invertierten CTM
x' y' <i>matrix</i>	itransform x y	Transformiert (x', y') mittels der inversen <i>matrix</i>
dx' dy'	idtransform dx dy	Transformiert die Distanz (dx', dy') mittels der inversen CTM

dx' dy' <i>matrix</i>	idtransform dx dy	Transformiert die Distanz (dx' , dy')
<i>matrix</i> ₁ <i>matrix</i> ₂	invertmatrix <i>matrix</i> ₂	mittels der inversen <i>matrix</i>
		Füllt <i>matrix</i> ₂ mit der inversen <i>matrix</i> ₁

Die Pfad-Operatoren

	newpath	Eröffnet einen neuen (leeren) Pfad
	currentpoint x y	Aktueller Punkt
x y	moveto	Aktuellen Punkt nach (x , y) verschieben
x y	lineto	Gerades Pfadsegment nach (x , y) anfügen
dx dy	rmoveto	Relatives moveto
dx dy	rlineto	Relatives lineto
x y r $angle_1$ $angle_2$	arc	Kreisbogensegment entgegen Uhrzeigersinn anfügen
x y $angle_1$ $angle_2$	arcn	Kreisbogensegment im Uhrzeigersinn anfügen
x_1 y_1 x_2 y_2 x_3 y_3	curveto	Bezier-Kurvensegment anfügen
x_1 y_1 x_2 y_2	arcto xt_2 yt_1 xt_2 yt_2	Kreisbogensegment mit Tangente anfügen
dx_1 dy_1 dx_2 dy_2 dx_3 dy_3	rcurveto	Relatives curveto
	closepath	Pfad schließen
	flattenpath	Pfad glätten
	reversepath	Pfad umdrehen
	strokepath	Umriss des Pfades
<i>string</i> <i>bool</i>	charpath	Zeichenumriss an Pfad anfügen
	clippath	Macht Clipping-Pfad zum aktuellen Pfad
	pathbox ll_y ll_x ur_x ur_y	Bounding-Box des aktuellen Pfades
<i>move</i> <i>line</i> <i>curve</i> <i>close</i>	pathforall	Enumeriert den aktuellen Pfad
	initclip	Setzt Clipping-Pfad auf Initialwert
	clip	Setzt neuen Clipping-Pfad
	eofclip	Benutzt Even/Odd-Regel für Clipping

Die Zeichen-Operatoren

	erasepage	Löscht aktuelle Seite
	fill	Füllt Pfad mit aktueller Farbe
	eofill	Benutzt Even/Odd-Regel für das Füllen
	stroke	Zeichnet Linie entlang des Pfades
<i>width</i> <i>height</i> <i>bits/sample</i>		
<i>matrix</i> <i>proc</i>	image	Bildet Halbtonbild ab
<i>width</i> <i>height</i> <i>invert</i>		
<i>matrix</i> <i>proc</i>	imagemask	Bildet Halbtonbild als Maske ab

Die Setup- und Ausgabe-Operatoren

	showpage	Gibt aktuelle Seite aus mit Reset
	copypage	Gibt aktuelle Seite aus
<i>matrix</i> <i>width</i> <i>height</i> <i>proc</i>	banddevice	Etabliert Band-Device

<i>matrix width height proc</i>	framedevice	Etabliert Buffer-Device
	nulldevice	Etabliert Null-Device
<i>proc</i>	renderbands	Berechnet Ausgabe für Band-Device

Die Zeichen- und Font-Operatoren

<i>key font</i>	definefont <i>font</i>	Definiert <i>font</i> als Font-Dictionary
<i>key</i>	findfont <i>font</i>	Sucht Font zu <i>key</i>
<i>font scale</i>	scalefont <i>font</i>	Skaliert Font
<i>font matrix</i>	makefont <i>font</i>	Transformiert Font mittels <i>matrix</i>
<i>font</i>	setfont	Setzt aktuellen Font
	currentfont <i>font</i>	Liefert aktuellen Font
<i>string</i>	show	Gibt <i>string</i> aus
<i>a_x a_y string</i>	ashow	Addiert bei der Ausgabe von <i>string</i> zu jedem Zeichen (<i>a_x</i> , <i>a_y</i>)
<i>c_x c_y char string</i>	widthshow	Addiert bei der Ausgabe von <i>string</i> (<i>c_x</i> , <i>c_y</i>) zum Zeichen <i>char</i>
<i>c_x c_y char a_x a_y string</i>	awidthshow	Kombiniert ashow und widthshow
<i>proc string</i>	kshow	Führt <i>proc</i> zwischen der Ausgabe von jeweils zwei Zeichen aus
<i>string</i>	stringwidth <i>w_x w_y</i>	Dicke von <i>string</i> im aktuellen Font
	FontDirectory <i>dict</i>	Dictionary von Font-Dictionaries
	StandardEncoding <i>array</i>	Standard-Decodierung

Die Cache-Operatoren

	cachestatus <i>bsize bmax msize mmax csize cmax blimit</i>	Cache-Status
<i>w_x w_y ll_x ll_y ur_x ur_y</i>	setcachedevice	Definiert Dickten für Zeichen im Cache
<i>w_x w_y</i>	setcharwidth	Definiert Dickten für Zeichen nicht im Cache
<i>num</i>	setcachelimit	Setzt Cache-Obergrenze für einzelnes Zeichen

Die Fehlermeldungen

dictfull	Dictionary voll
dictstackoverflow	Zu viele begin
dictstackunderflow	Zu viele end
execstackoverflow	Schachtelung zu tief
handleerror	Wird im Fehlerfall aufgerufen
interrupt	Externe Unterbrechung
invalidaccess	Unzulässiger File-Zugriff
invalidexit	exit außerhalb der Schleife
invalidfileaccess	Unzulässiger File-Zugriff
invalidfont	Falscher Font-Name oder -Dictionary
invalidrestore	Falsches restore
ioerror	Ein-/Ausgabefehler
limitcheck	Grenzwert überschritten

nocurrentpoint	Aktueller Pfad undefiniert
rangecheck	Operand verletzt Grenzen
stackoverflow	Stack-Überlauf
stackunderflow	Stack-Unterlauf
syntaxerror	Syntaxfehler
timeout	Zeitspanne überschritten
typecheck	Operand hat falschen Typ
undefined	Name nicht bekannt
undefinedfilename	Datei nicht gefunden
undefinedresult	Überlauf/Unterlauf oder unsinniges Ergebnis
unmatchedmark	Auf Stack fehlt <i>mark</i>
unregistered	Interner Fehler
VMerror	VM (virtual memory) erschöpft

04.4 Die PostScript-Standardfonts

Nr	Name	Bemerkungen
1	Courier	Standard
2	Courier-Oblique	kursiv
3	Courier-Bold	fett
4	Courier-BoldOblique	fett/kursiv
5	Times-Roman	Standard
6	Times-Italic	kursiv
7	Times-Bold	fett
8	Times-BoldItalic	fett/kursiv
9	Helvetica	Standard
10	Helvetica-Oblique	kursiv
11	Helvetica-Bold	fett
12	Helvetica-BoldOblique	fett/kursiv
13	Helvetica-Narrow	eng
14	Helvetica-NarrowOblique	
15	Helvetica-Narrow-Bold	
16	Helvetica-NarrowBoldOblique	
17	AvantGard	
18	AvantGard-Demi	
19	AvantGard-BookOblique	
20	AvantGard-DemiOblique	
21	Bookmann-Light	
22	Bookman-Demi	
23	Bookman-LightItalic	
24	Bookman-DemiItalic	
25	NewCenturySchlbk-Roman	
26	NewCenturySchlbk-Bold	
27	NewCenturySchlbk-Italic	
28	NewCenturySchlbk-BoldItalic	

29	Palatino-Roman	
30	Palatino-Bold	
31	Palatino-Italic	
32	Palatino-BoldItalic	
33	ZapfChancery-MediumItalic	
34	ZapfDingbats	
35	Symbol	Der Symbol-Font enthält das griechische Alphabet (groß/klein, a = alpha, A = Alpha, b = beta, B = Beta usw.) und Sonderzeichen

04.5 Implementierungsbeschränkungen

Die folgenden Beschränkungen gelten streng nur für den Apple LaserWriter. Auf anderen Ausgabegeräten können geringfügige Abweichungen vorliegen.

Objekt	Grenzen	Bemerkungen
Integer	+2147483647 -2147483648	= $+2^{31} - 1$. In den meisten Fällen wird bei Bereichsüberschreitung in Real konvertiert. = -2^{31}
Real	$\pm 10^{38}$	Genauigkeit: ca. 8 signifikante Stellen
Array	65535	Maximale Array-Länge. Ein Array (oder eine Prozedur) belegt 8 Byte pro Element.
String	65535	Maximale Stringlänge. 1 Byte pro Zeichen.
file	6	Maximale Anzahl offener Dateien.
userdict	200	Maximale Länge von userdict .
Operanden-Stack	500	Maximale Anzahl der Elemente, die auf den Operanden-Stack gelegt und noch nicht entfernt worden sind.
Dictionary-Stack	20	Maximale Tiefe des Dictionary-Stack.
Ausführungs-Stack	250	Maximale Tiefe des Ausführungs-Stacks. Alle Prozeduren, Files und Strings, deren Ausführung unterbrochen wurde, bilden ein Element des Stacks.
Interpreter-Niveau	10	Maximale Anzahl rekursiver Aufrufe.
save-Niveau	15	Maximale Anzahl aktiver saves .
gsave-Niveau	31	Maximale Anzahl aktiver gsaves .

Pfad	1500	Maximale Anzahl der in allen Pfaden definierten Punkte.
Linienmuster	11	Maximale Anzahl von Elementen in dem Array-Operanden für setdash .
VM	240000	Maximalgröße des virtuellen Speichers in Byte.

04.6 Einleitende Beispiele

04.6.1 Beispiel: PostScript als Grafik-Sprache

Datei **PS-01.PS**

Linien, Kasten, Füllen, Grauton, Überlappung, Kreis und Kreisbogen, Rundungen, Bézierkurven, Linienenden, Linienschnittpunkte, Schriften, Koordinatentransformation, Skalierung

```
% PostScript Datei "PS-01.PS", K. Haller, 7.1.92
% Zeilenrest ab Prozentzeichen ist Kommentar

% Das Demo-Programm #01 zeigt einfache Anwendungen von
% PostScript als Grafik-Sprache. Auf die Fähigkeiten von
% PostScript als Programmiersprache wurde in dieser Demo
% bewusst verzichtet.

%----- 01: Linien zeichnen -----
newpath          % Neuen Pfad anlegen
 90 50 moveto     % "Stift" zum Koordinatenpunkt x = 90 pt
                  % und y = 50 pt bewegen. Kein Zeichnen.
                  % Standard-Einheit 1 pt = 1/72 inch =
                  % 0.352777... mm
 90 800 lineto    % Linie bis zum neuen Punkt markieren
575 800 lineto
575 50 lineto
 90 50 lineto

100 700 moveto
 72 72 rlineto    % "rlineto" = relatives lineto
                  % zum aktuellen Punkt
stroke           % Pfad mit "Farbe" nachziehen

%----- 02: Quadrat zeichnen -----
newpath
200 700 moveto
0 72 rlineto      % Relatives lineto vom letzten Punkt aus
72 0 rlineto
0 -72 rlineto
-72 0 rlineto
10 setlinewidth  % Liniendicke 10 pt
```

```
stroke

%----- 03: Wie 02, jedoch mit Schließen der Lücke -----
newpath
300 700 moveto    % Figur weiter nach rechts
0 72  rlineto
72 0   rlineto
0 -72 rlineto
-72 0  rlineto
closepath          % Pfad schließen um Lücke zu schließen
10 setlinewidth
stroke

%----- 04: Ausgefüllte Figur -----
newpath
400 700 moveto    % Figur weiter nach rechts
0 72  rlineto
72 0   rlineto
0 -72 rlineto
-72 0  rlineto
closepath
fill            % Füllt Figur aus. Standard: 100% schwarz
stroke

%----- 05: Ausfüllen mit Grauton -----
newpath
100 600 moveto
0 72  rlineto
72 0   rlineto
0 -72 rlineto
-72 0  rlineto
closepath
0.3 setgray      % Setzt Grauton. 0 = schwarz, 1 = weiß
fill            % Füllt Figur mit Grauton aus
stroke

%----- 06: Überlappende Flächen (PostScript ist opak) ----
newpath
200 600 moveto
0 72  rlineto
72 0   rlineto
0 -72 rlineto
-72 0  rlineto
closepath
0.5 setgray      % Figur #06a mit Grauton 0.5
fill
stroke

215 615 moveto    % Überlappung mit leichtem Versatz
0 72  rlineto
72 0   rlineto
0 -72 rlineto
-72 0  rlineto
closepath
```

```
0.9 setgray      % Figur #06b mit Grauton 0.9
fill
stroke

%----- 07: Kreis und Kreisbogen, fehlerhaft -----
%           Es sollte Figur wie in #08 erzeugt werden,
%           was aber zuerst nicht ganz gelingt.
newpath
330 630 25 0 360 arc  % arc = Kreisbogen bzw. Kreis
                      % Syntax: xm ym Radius Winkell1 Winkel2 arc
                      % xm = 330, ym = 630, Radius = 25
                      % Winkell1 = 0°, Winkel2 = 360° (Gradmaß)
                      % Drehung gegen Uhrzeigersinn
                      % Mit noch nicht erklärter Skalierung
                      % sind auch Ellipsen bzw. Ellipsen-
                      % bogen mittels "arc" darstellbar.
330 630 50 0 135 arc % Kreisbogen 0° bis 135°
stroke

%----- 08: Ähnlich 07, aber Fehler korrigiert -----
1 setlinewidth    % Liniendicke auf 1 setzen, sonst
                  % wirkt noch alte Liniendicke
0 setgray          % Grauton auf 0 (= schwarz) setzen,
                  % sonst wirkt noch alter Grauton.
                  % Diese Werte gelten bis zu einer
                  % erneuten Änderung
newpath
430 630 25 0 360 arc % arc = Kreisbogen bzw. Kreis
                    % Syntax: xm ym Radius Winkell1 Winkel2
                    % Winkel im Gradmaß
                    % Gegen Uhrzeigersinn

480 630 moveto      % arc zeichnet vom letzten aktuellen
                    % Punkt eine gerade Linie zum Beginn
                    % des Kreisbogens. Hier nur eine sehr
                    % primitive "händische" Korrektur mit
                    % "moveto". 480 = xm + R = 430 + 50
430 630 50 0 135 arc % Kreisbogen 0° bis 135°
stroke

%----- 09: Abgerundete Ecken -----
newpath
100 500 moveto      % Für Demo aktuellen Punkt erzeugen
110 560 170 570 20 arcto
                    % Abgerundete Ecken mit "arcto"
                    % Syntax: x1 y1 x2 y2 r arcto
                    % Zeichnet ab aktuellem Punkt
                    % eine Gerade in Richtung zum
                    % Punkt (x1, y1), aber nur bis
                    % zum Beginn des Bogens. Es wird
                    % dann nur der Bogen gezeichnet.
170 570 lineto      % Linie vom Bogenende bis zum
                    % Punkt (x2, y2), falls gewünscht
stroke
```

```

%----- 10: Bezier-Kurve -----
newpath
200 500 moveto          % Der aktuelle Punkt wird als erster
                        % Bézier-Fixpunkt (x0, y0) angenom-
                        % men. Wenn noch keiner existiert,
                        % dann mit moveto erzeugen
220 650 240 450 270 570 curveto
                        % Bezier-Kurve vom ersten Fixpunkt
                        % über zwei Steuerpunkte (x1, y1)
                        % und (x2, y2) zum zweiten Fix-
                        % punkt (x3, y3)
                        % Syntax: x1 y1 x2 y2 x3 y3 curveto
stroke

%----- 11: Form der Linienenden -----
newpath
15 setlinewidth          % Neue Liniendicke 15 pt

0 setlinecap             % 0 = abgeschnittene Enden = Standard
                        % Werte für setlinecap: 0, 1, 2
                        % Gilt für den gesamten Pfad
300 560 moveto
370 560 lineto
stroke

newpath                  % Neuer Pfad notwendig
1 setlinecap             % 1 = abgerundete Enden
                        % Durchmesser gleich Liniendicke

300 535 moveto
370 535 lineto
stroke

newpath
2 setlinecap             % 2 = projektive Enden. Reichen um
                        % halbe Liniendicke über Endpunkte
                        % hinaus.

300 510 moveto
370 510 lineto
stroke

%----- 12: Form der Linienschnittpunkte -----
newpath
8 setlinewidth           % Liniendicke 8 pt

0 setlinejoin            % 0 = spitze Ecken = Standard
                        % Werte für setlinejoin: 0, 1, 2
                        % Gilt für den gesamten Pfad.

% Hinweis: Bei sehr kleinen Schnittwinkeln kann die Spitze
%          weit über den theoretischen Endwert hinausragen.
%          Die Spitze wird deshalb bei sehr kleinen Winkeln
%          unterdrückt. Mit dem Operator "setmiterlimit"
%          kann Begrenzung selbst vorgenommen werden.
%          Details dazu siehe PostScript-Handbuch.

400 560 moveto

```

```
470 560 lineto
-50 13 rlineto
stroke

newpath                                % Neuer Pfad notwendig
1 setlinejoin                          % 1 = abgerundete Ecken
                                        % Durchmesser gleich Liniendicke

400 535 moveto
470 535 lineto
-50 13 rlineto
stroke

newpath
2 setlinejoin                          % 2 = abgeschnittene Ecken

400 510 moveto
470 510 lineto
-50 13 rlineto
stroke

1 setlinewidth                        % Liniendicke 1 pt
0 setlinejoin                          % 0 = spitze Ecken

% Hinweis: Mit dem Operator "setdash" kann der Linientyp
%          (durchgezogen, unterbrochen mit bestimmtem
%          Rhythmus) bestimmt werden. Details siehe Post-
%          Script-Handbuch

%----- 13: Schriften -----
/Times-Roman findfont % Prozedur für Schrift Times-Roman

30 scalefont                          % Die Schrift kann - wie jedes
                                        % grafisches Objekt in PostScript -
                                        % skaliert werden, hier auf 30 pt

setfont                                % Der Font wird gesetzt

100 450 moveto

(Demonstration Adobe PostScript, #01) show
                                        % Der in runden Klammern enthaltene
                                        % String wird mit "show" ausgedruckt

/Courier findfont 14 scalefont setfont % PostScript
100 420 moveto (Courier, 14 pt) show   % ist formatfrei!

/Courier-Bold findfont 14 scalefont setfont
100 405 moveto (Courier-Bold, 14 pt) show

/Courier-Oblique findfont 14 scalefont setfont
100 390 moveto (Courier-Oblique, 14 pt) show

/Courier-BoldOblique findfont 14 scalefont setfont
100 375 moveto (Courier-BoldOblique, 14 pt) show
% -----
/Times-Roman findfont 14 scalefont setfont
```

```

100 355 moveto (Times-Roman, 14 pt) show

/Times-Bold findfont 14 scalefont setfont
100 340 moveto (Times-Bold, 14 pt) show

/Times-Italic findfont 14 scalefont setfont
100 325 moveto (Times-Italic, 14 pt) show

/Times-BoldItalic findfont 14 scalefont setfont
100 310 moveto (Times-BoldItalic, 14 pt) show
% -----
/Helvetica findfont 14 scalefont setfont
340 420 moveto (Helvetica, 14 pt) show

/Helvetica-Bold findfont 14 scalefont setfont
340 405 moveto (Helvetica-Bold, 14 pt) show

/Helvetica-Oblique findfont 14 scalefont setfont
340 390 moveto (Helvetica-Oblique, 14 pt) show

/Helvetica-BoldOblique findfont 14 scalefont setfont
340 375 moveto (Helvetica-BoldOblique, 14 pt) show
% -----
/Courier findfont 10 scalefont setfont
340 355 moveto (Es folgt Font "Symbol", 14 pt:) show
% -----
/Symbol findfont 14 scalefont setfont
340 340 moveto (abcdefghijklmnopqrstuvwxyz) show
340 325 moveto (ABCDEFGHIJKLMNOPQRSTUVWXYZ) show
% Der Font "Symbol" enthält u.a. den vollständigen
% griechischen Zeichensatz, wobei gilt: a = alpha,
% b = beta, A = Alpha, B = Beta usw.
340 310 moveto (\251 \345 \305 \362 \345 \245) show
% Die über Tastatur nicht erreichbaren Zeichen,
% z.B. die mit Dez-Code >= 128, müssen mit dem
% Backslash "\" dreistellig im Oktal-Code eingegeben
% werden.

/Times-Roman findfont 20 scalefont setfont
100 280 moveto (Fachhochschule Muenchen, DR, kha) show
100 260 moveto (Fachhochschule München, DR  ☐™š „"☐ á) show
% PostScript arbeitet nicht mit dem ASCII-, sondern mit
% dem ANSI-Zeichensatz. Probleme mit ASCII-Zeichen >= 128
% da nicht sie nicht in den PostScript-Fonts codiert
% sind. Es ist in PostScript aber möglich, eigene
% Fonts zu erstellen. Details siehe PostScript-
% Handbuch Kap. 5.7 "Fonts definieren".

%----- 14: Outline-Schrift mit "charpath" -----
/Bookman-Demi findfont 30 scalefont setfont
100 220 moveto (Bookman-) show % Normale Schriftdarstellung
0.5 setgray (Demi ) show % Schrift mit Rasterton 50%
0 setgray (30 pt) show % 0 setgray = Rasterton 100%

100 190 moveto (Bookman-Demi 30 pt) false charpath stroke
% "false charpath" liefert den Pfad für "stroke".
% Damit Outline-Schrift mit "stroke" und nicht mit "show".

```

```
%----- 15: Koordinatentransformation u. -rotation -----
% Alle grafischen Objekte können in PostScript beliebig
% transformiert, rotiert (und skaliert) werden.

gsave      % gsave = graphic status save. Graphikstatus sichern
           % für später. Wird auf den Operanden-Stack gelegt

520 150 translate    % Koordinatentransformation
75      rotate       % 75° entgegen Uhrzeigersinn drehen

0 0 moveto 100 0 lineto
0 0 moveto 0 50 lineto stroke

/Palatino-Bold findfont 14 scalefont setfont
5 7 moveto (Transformation/Drehung) show
5 30 moveto (Palatino-Bold) show

grestore    % grestore = graphic status restore. Wird vom
           % Stack geholt. Damit wieder alter Graphikstatus
           % mit altem Koordinatensystem und alter Schrift
100 150 moveto (Alter Graphikstatus) show

%----- 16: Schrift skalieren mit "scale" -----
/Times-Roman findfont 30 scalefont setfont
100 120 moveto (FHM) show

gsave      % Graphikstatus sichern

240 120 moveto
2.1 0.5 scale
(FHM) show

grestore    % Alten Graphikstatus wiederherstellen
gsave      % und sichern

425 120 moveto
0.5 2.7 scale
(FHM) show

%----- 16: Schrift skalieren und rotieren -----
grestore    % Alten Graphikstatus wiederherstellen
gsave      % und sichern

550 120 moveto
1.2 2.7 scale
180 rotate
(FHM) show

%----- 17: Achsenteilungen -----
grestore    % Alten Graphikstatus wiederherstellen

% Für die Achsenteilungen wäre vorteilhaft der Programmierbefehl
% (Operator) "for" einzusetzen. In dieser Demo sollen bewusst
```

```
% aber nur die Graphikeigenschaften von PostScript gezeigt werden.

100  50 moveto 0 -5 rlineto
200  50 moveto 0 -5 rlineto
300  50 moveto 0 -5 rlineto
400  50 moveto 0 -5 rlineto
500  50 moveto 0 -5 rlineto

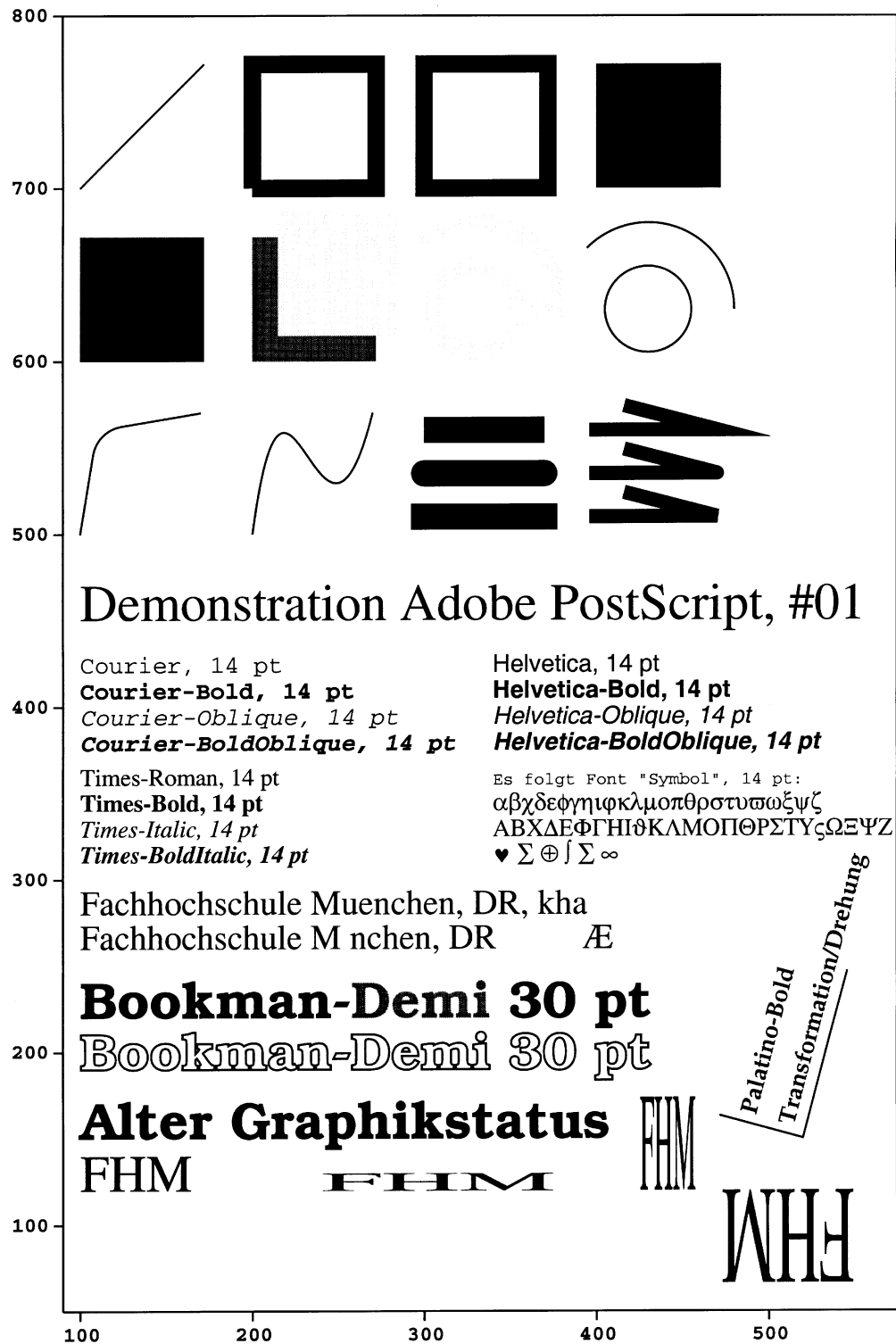
  90 100 moveto -5 0 rlineto
  90 200 moveto -5 0 rlineto
  90 300 moveto -5 0 rlineto
  90 400 moveto -5 0 rlineto
  90 500 moveto -5 0 rlineto
  90 600 moveto -5 0 rlineto
  90 700 moveto -5 0 rlineto
  90 800 moveto -5 0 rlineto
stroke

/Courier-Bold findfont 12 scalefont setfont
  90  33 moveto (100) show
190  33 moveto (200) show
290  33 moveto (300) show
390  33 moveto (400) show
490  33 moveto (500) show

  60  97 moveto (100) show
  60 197 moveto (200) show
  60 297 moveto (300) show
  60 397 moveto (400) show
  60 497 moveto (500) show
  60 597 moveto (600) show
  60 697 moveto (700) show
  60 797 moveto (800) show

%-----
showpage           % Seite drucken
%----- Ende -----
```

Achtung: Alle nachfolgenden PostScript-Ausdrucke sind gegenüber dem Original auf ca. 71% verkleinert dargestellt.



04.6.2 Beispiel: PostScript als Programmiersprache

Bedingungen, in Pascal: **if/then** und **if then/else**
 Allgemeine Schleifen, in Pascal: **repeat/until**
 Zähl-Schleifen, in Pascal: **for**

Beispiel: Iterative Nullstellensuche nach Regula falsi

- Pascal-Programm **Regula.PAS**
- PostScript-Programm **Regula.PS**

Zuerst die Pascal-Version:

```

program Regula; { "Regula.PAS"
                  { Pascal-Äquivalent zum PostScript-Programm "Regula.PS"
                  { Nullstellensuche mit Regula falsi. 7.1.92   K. Haller }
uses
  PRINTER;
const
  Epsilon = 1.0e-7; { Iterationsgenauigkeit }
  iMax    = 20;     { Für Abbruch der Schleife, wenn i > iMax }
  x1Start = 1.7;    { Unterer Startwert }
  x2Start = 2.5;    { Oberer Startwert }
var
  i:          Byte;
  x, y, x1, y1,
  x2, y2, x2neu: Real;

function yBerechnen(x: Real): Real;
begin
  yBerechnen := x*x - 4;
end;

procedure Ausgabe;
begin
  WriteLn(Lst, 'Die berechnete L"sung:      x = ', x);
  WriteLn(Lst, 'Die Schleifenzahl:        i = ', i);
  WriteLn(Lst, 'Die Iterationsgenauigkeit: Epsilon = ',
            Epsilon); WriteLn(Lst);
end;

procedure Abbruch;
begin
  WriteLn('Abbruch wegen Schleifenueberschreitung:  iMax = ', iMax);
  ReadLn; Halt(1);
end;

begin
  WriteLn(Lst, 'Demo:  Iterative Nullstellenberechnung nach Regula falsi');
  WriteLn(Lst, 'Pascal-Äquivalent zum PostScript-Programm "Regula.PS"');
  WriteLn(Lst, 'Funktion:  y = x^2 - 4,  Nullstellen bei:  ±2'); WriteLn(Lst);

  x1 := x1Start;
  x2 := x2Start;
  i  := 0;
  repeat
    Inc(i);

```

```

    if i > iMax then Abbruch;
    y1 := yBerechnen(x1);
    y2 := yBerechnen(x2);
    x2neu := x2 - (x2 - x1)/(y2 - y1)*y2;
    x1 := x2;
    x2 := x2neu;
until Abs(x2 - x1) < Epsilon;
x := x2;
Ausgabe;

for i := Round(x1Start*10) to Round(x2Start*10) do
begin
    x := i/10;
    y := yBerechnen(x);
    WriteLn(Lst, 'x = ', x, '    y = ', y);
end;
end.

```

Und jetzt die PostScript-Version:

```

% PostScript-Programm "Regula.PS". Nullstellensuche mit Regula falsi
% Loesungsvariante mit "Variablen", 7.1.92, K. Haller
% 28.7.92

/String30 30 string def % Stringvariable, um Zahlen für die Ausgabe in String
% zu konvertieren. Mit "cvs" (= convert to string).
/Epsilon 1.0e-7 def % Iterationsgenauigkeit
/i 0 def % Schleifenzaehler
/iMax 20 def % Fuer Abbruch der Schleife, wenn i > iMax
/x1Start 1.7 def % Unterer Startwert
/x2Start 2.5 def % Oberer Startwert

/yBerechnen % Die Funktion "x*x - 4"
{ dup mul 4 sub } bind def

/Courier findfont 12 scalefont setfont

/Ausgabe
{ 50 730 moveto (Die berechnete Loesung: x = ) show
  x String30 cvs show
  50 710 moveto (Die Schleifenzahl: i = ) show
  i String30 cvs show
  50 690 moveto (Die Iterationsgenauigkeit: Epsilon = ) show
  Epsilon String30 cvs show
} bind def

/Abbruch
{ 50 730 moveto (Abbruch wegen Schleifenueberschreitung: iMax = ) show
  iMax String30 cvs show
} bind def

50 780 moveto
(Demo: Iterative Nullstellenberechnung nach Regula falsi in PostScript) show
50 760 moveto (Funktion: y = x^2 - 4, Nullstellen: +2 und -2) show

/x1 x1Start def
/x2 x2Start def

{ /i i 1 add def % Loesungsvariante mit "Variablen"
  i iMax gt { Abbruch exit } if % Zur Uebung empfohlen: Umstellung
  /y1 x1 yBerechnen def % auf Stack-Operationen

```

```
/y2 x2 yBerechnen def
/dx x2 x1 sub      def          % dx := x2 - x1
/dy y2 y1 sub      def          % dy := y2 - y1
/x2neu dx dy div y2 mul neg x2 add def % x2neu := x2 - dx/dy*y2
/x1 x2             def
/x2 x2neu          def
x2 x1 sub abs Epsilon lt { /x x2 def Ausgabe exit } if
} loop

/yPosition 650 def

x1Start 0.1 x2Start % Anfangswert, Schrittweite, Endwert der for-Schleife
{ /x exch def      % Die "Laufvariable" der for-Schleife vom Stack nehmen
  % und auf Variable "x" legen
  /y x yBerechnen def
  50 yPosition moveto (x = ) show x String30 cvs show
  150 yPosition moveto (y = ) show y String30 cvs show
  /yPosition yPosition 17 sub def
} for
showpage
```

Demo: Iterative Nullstellenberechnung nach Regula falsi in PostScript

Funktion: $y = x^2 - 4$, Nullstellen: +2 und -2

Die berechnete Loesung: $x = 2.0$

Die Schleifenzahl: $i = 5$

Die Iterationsgenauigkeit: $\text{Epsilon} = 1\text{e-}07$

$x = 1.7$	$y = -1.11$
$x = 1.8$	$y = -0.76$
$x = 1.9$	$y = -0.39$
$x = 2.0$	$y = 0.0$
$x = 2.1$	$y = 0.409999$
$x = 2.2$	$y = 0.839999$
$x = 2.3$	$y = 1.29$
$x = 2.4$	$y = 1.76$
$x = 2.5$	$y = 2.25$

04.7 Beispiel FHM-Logo "FHM-Lo2.PS"

```
% PostScript-Programm "Transfor.PS": Demo PostScript-Transformationen
% Als Basis dient das folgende und modifizierte ...
% PostScript-Programm "FHM-Lo2.PS": Logo der Fachhochschule Muenchen
% Ueberarbeitung: Dr. K. Haller, 5211099, FHM, DR
% Das von einem Grafiker geschaffene FHM-Logo wurde durch Ausmessen
% rekonstruiert; es duerfte den nachstehend gezeigten Aufbau haben.

/FHM-Logo
{ /Breite      exch def      % Die drei
  /UntererRand exch def      % Parameter des
  /LinkerRand  exch def      % Prozeduraufrufes
  %-----
  /rSp 0.12 def % rSp fuer einfaches "Hinting" anpassen: -----+
                  % rSp = relative Spaltenbreite (= Spaltenbreite
                  %           geteilt durch Balkenbreite Modul)
                  % Standard bei Filmausgabe:           rSp = 0.09
                  % Bei Laserdruckern und bei
                  % Breite < 10 mm wegen Hinting:       rSp = 0.13
                  %-----+
  /Modul      Breite 4 rSp 3 mul add div def % Modul = Balkenbreite
  /Sp         Modul rSp mul def             % Sp      = Spaltenbreite
  /ModulSp    Modul Sp add def              % ModulSp = Balkenbreite +
                                          %           + Spaltenbreite

  gsave
  LinkerRand UntererRand translate

  newpath % ----- 1. Zeichen "f" -----
  0 ModulSp moveto
  0 ModulSp 2 mul rlineto
  Modul ModulSp 3 mul Modul 180 90 arcn
  ModulSp 0 rlineto
  0 Modul neg rlineto
  ModulSp neg 0 rlineto
  0 Sp neg rlineto
  ModulSp 0 rlineto
  0 Modul neg rlineto
  ModulSp neg 0 rlineto
  0 ModulSp neg rlineto
  closepath 0.0 setgray fill          % Fuer "setgray": 0.0 = schwarz,
                                     %                               1.0 = weiss

  newpath % ----- 2. Zeichen "h" -----
  ModulSp 2 mul ModulSp 2 mul moveto
  0 ModulSp 2 mul Modul add rlineto
  Modul 0 rlineto
  0 ModulSp neg rlineto
  Sp 0 rlineto
  ModulSp 3 mul ModulSp 3 mul Modul 90 0 arcn
  0 ModulSp neg rlineto
  Modul neg 0 rlineto
  0 ModulSp rlineto
  Sp neg 0 rlineto
  0 ModulSp neg rlineto
  closepath 0.0 setgray fill
```

```

newpath % ----- 3. Zeichen "m" -----
ModulSp 0 moveto
0 ModulSp Modul add rlineto
ModulSp 2 mul 0 rlineto
ModulSp 3 mul ModulSp Modul 90 0 arcn
0 ModulSp neg rlineto
Modul neg 0 rlineto
0 ModulSp rlineto
Sp neg 0 rlineto
0 ModulSp neg rlineto
Modul neg 0 rlineto
0 ModulSp rlineto
Sp neg 0 rlineto
0 ModulSp neg rlineto
closepath 0.0 setgray fill

grestore
} bind def %---- Ende Prozedur "FHM-Logo" -----

/FHM-Logo-Aufbau %-----
{ /Breite      exch def      % Die drei
  /UntererRand exch def      % Parameter des
  /LinkerRand  exch def      % Prozeduraufrufes
  gsave

  LinkerRand UntererRand Breite FHM-Logo

  newpath
  LinkerRand UntererRand 3 0 360 arc fill
  LinkerRand UntererRand translate
  0.5 setgray
  0.3 setlinewidth
  0 1 4 { /ix exch def
    0 1 5 { /iy exch def
      ix ModulSp mul iy ModulSp mul moveto
      0 Modul      rlineto
      Modul 0      rlineto
      0 Modul neg  rlineto
      Modul neg 0  rlineto
      Modul Modul  rlineto
      Modul neg 0  rmoveto
      Modul Modul neg rlineto
    } for
  } for
  stroke
  grestore
} bind def %-----

/mm { 72 25.4 div mul } def % Umrechnung PostScript-point in mm

/MitGitter true def % Optionales Gitternetz beim FHM-Logo

% Die Parameter beim Aufruf der Prozedur "FHM-Logo", von rechts:
% 3: "Breite"          (Stackspitze) Gesamtbreite des Logos in mm
%                     Breite = 4 Balken + 3 Spalten
%                     Hoehe  = 5 Balken + 4 Spalten
%                     Hoehe  = ca. 1.25 * Breite

```

```

% 2: "UntererRand"      Nullpunkt: Leere untere Ecke
% 1: "LinkerRand"       des FHM-Logos. Beide in mm

% LinkerRand, UntererRand, Breite, Prozedur-Aufruf
37 mm 175 mm 35 mm FHM-Logo
80 mm 175 mm 35 mm FHM-Logo-Aufbau

132 mm 175 mm 20 mm FHM-Logo
156 mm 175 mm 15 mm FHM-Logo

132 mm 205 mm 10 mm FHM-Logo
143 mm 205 mm 9 mm FHM-Logo
153 mm 205 mm 8 mm FHM-Logo
162 mm 205 mm 7 mm FHM-Logo
170 mm 205 mm 6 mm FHM-Logo
177 mm 205 mm 5 mm FHM-Logo

132 mm 222 mm 4 mm FHM-Logo
137 mm 222 mm 3 mm FHM-Logo
141 mm 222 mm 2 mm FHM-Logo
144 mm 222 mm 1 mm FHM-Logo
146 mm 222 mm 0.5 mm FHM-Logo
147.5 mm 222 mm 0.2 mm FHM-Logo
148.7 mm 222 mm 0.1 mm FHM-Logo

gsave
1.5 1 scale
24.7 mm 112 mm 35 mm MitGitter { FHM-Logo-Aufbau }
                                { FHM-Logo } ifelse
grestore

gsave
1 1.5 scale
112 mm 57 mm 35 mm MitGitter { FHM-Logo-Aufbau }
                              { FHM-Logo } ifelse
grestore

gsave
50 mm 44 mm translate
15 rotate
0 mm 0 mm 35 mm MitGitter { FHM-Logo-Aufbau }
                           { FHM-Logo } ifelse
grestore

gsave
/sX 1.3 def % Neue Skalierung x-Strich
/sY 0.7 def % Neue Skalierung y-Strich
/Alpha 15 def % Winkel ab x-Achse in Grad, mathemat. Drehsinn
/Beta 55 def % Winkel ab y-Achse in Grad, mathemat. Drehsinn
/x0 126 mm def % Neuer Nullpunkt x-Strich-Achse
/y0 44 mm def % Neuer Nullpunkt y-Strich-Achse
%-----
/A Alpha cos sX mul def
/B Alpha sin sX mul def
/C Beta sin sY mul neg def
/D Beta cos sY mul def
%-----
[A B C D x0 y0] concat % Neue Transformations-Matrix CTM

```



```

0 mm      0 mm  35 mm MitGitter { FHM-Logo-Aufbau }
                                { FHM-Logo          } ifelse
grestore
newpath
0.12 mm setlinewidth
x0 y0 moveto
35 mm 0 mm rlineto
x0 y0 moveto
0 mm 35 mm rlineto
x0 32 mm add y0 moveto
x0 y0 32 mm 0 Alpha arc

x0 y0 32 mm add moveto
x0 y0 32 mm 90 Beta 90 add arc
stroke

/Symbol findfont 14 scalefont setfont
x0 32.5 mm add y0 4 mm add moveto (a) show      % Zeichen alpha
x0 16 mm sub y0 30 mm add moveto (b) show      % Zeichen beta

gsave
/Times-Bold findfont 25 scalefont setfont
140 mm 36 mm moveto (FHM) show
/sX 0.7 def      % Neue Skalierung x-Strich
/sY 1.5 def      % Neue Skalierung y-Strich
/Alpha 0 def     % Winkel ab x-Achse in Grad, mathemat. Drehsinn
/Beta 20 def     % Winkel ab y-Achse in Grad, mathemat. Drehsinn
/x0 168 mm def   % Neuer Nullpunkt x-Strich-Achse
/y0 36 mm def   % Neuer Nullpunkt y-Strich-Achse
%-----
/A Alpha cos sX mul      def
/B Alpha sin sX mul      def
/C Beta sin sY mul neg   def
/D Beta cos sY mul       def
%-----
[A B C D x0 y0] concat  % Neue Transformations-Matrix CTM
0 mm 0 mm moveto (FHM) show
grestore

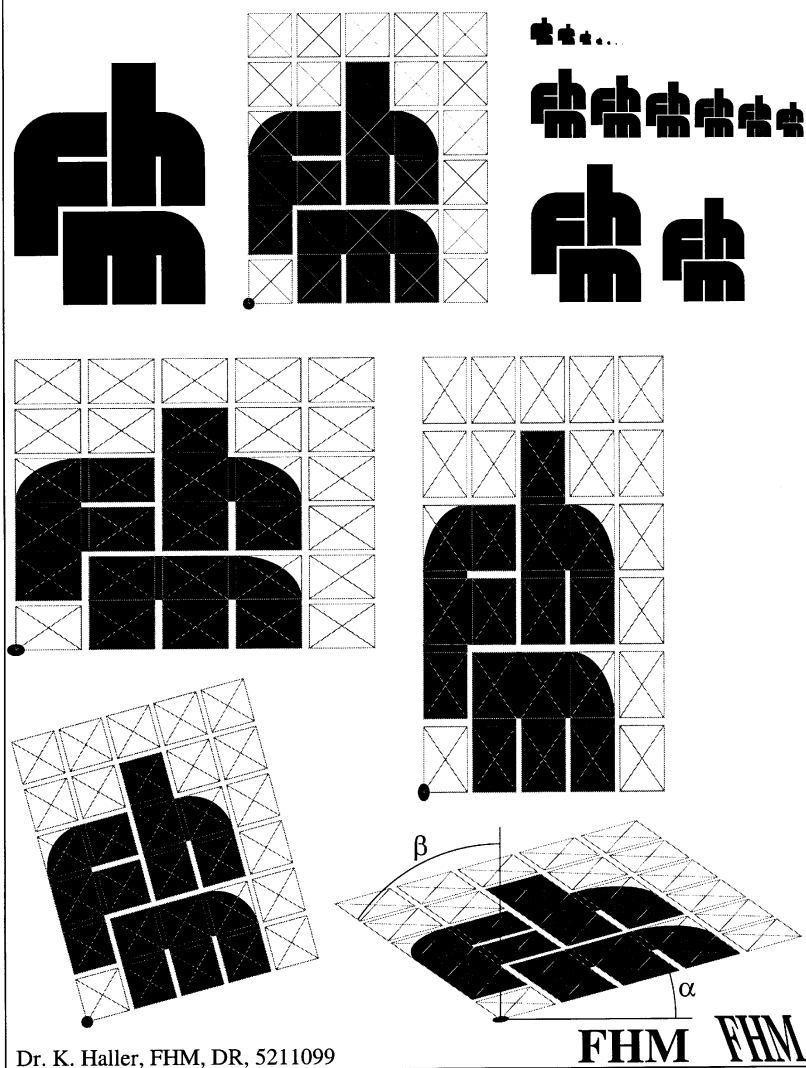
/Times-Bold findfont 16 scalefont setfont
36 mm 245 mm moveto
(Skalierung und Transformation in PostScript) show
36 mm 239 mm moveto (Beispiel FHM-Logo, "FHM-Lo2.PS") show
/Times-Roman findfont 12 scalefont setfont
37 mm 36 mm moveto
(Dr. K. Haller, FHM, DR, 5211099) show

newpath 0.2 mm setlinewidth
35 mm 35 mm moveto
0 216.8 mm      rlineto
149.8 mm 0      rlineto
0 -216.8 mm     rlineto
closepath stroke

showpage %-----

```

Skalierung und Transformation in PostScript
Beispiel FHM-Logo, "FHM-Lo2.PS"



04.8 Beispiel FHM-Logo "FHM-Lo3.PS"

```
% PostScript-Programm "FHM-Lo3.PS": Logo der Fachhochschule Muenchen
% Ueberarbeitung: Dr. K. Haller, 7216099, FHM, Druckereitechnik
% Das von einem Grafiker geschaffene FHM-Logo wurde durch Ausmessen
% rekonstruiert; es duerfte den nachstehend gezeigten Aufbau haben.

/FHM-Logo %-----
{ /Breite      exch def      % Die drei
  /UntererRand exch def      % Parameter des
  /LinkerRand  exch def      % Prozedur-Aufrufes

  % Die Parameter der Prozedur "FHM-Logo", von rechts: -----+
  % 3: "Breite"      (Stackspitze) Gesamtbreite des Logos in mm |
  %                  Breite = 4 Balken + 3 Spalten             |
  %                  Hoehe  = 5 Balken + 4 Spalten             |
  %                  Hoehe  = ca. 1.25 * Breite                 |
  % 2: "UntererRand" Nullpunkt: Leere linke untere Ecke        |
  % 1: "LinkerRand"  des FHM Logos. Beide in mm                |
  % Aufrufbeispiel: "50 mm 200 mm 35 mm FHM-Logo"              |
  % Linker Rand 50mm, Unterer Rand 200mm, Breite 35 mm -----+

  /rSp 0.12 def % rSp fuer einfaches "Hinting" anpassen: -----+
                  % rSp = relative Spaltenbreite (= Spaltenbreite
                  %         geteilt durch Balkenbreite Modul)
                  % Standard bei Filmausgabe:          rSp = 0.10
                  % Bei Laserdruckern und bei
                  % Breite < 10 mm wegen Hinting:      rSp = 0.13
                  %-----+

  /Modul  Breite 4 rSp 3 mul add div def % Modul = Balkenbreite
  /Sp     Modul rSp mul def             % Sp      = Spaltenbreite
  /ModulSp Modul Sp add def             % ModulSp = Balkenbreite +
                                      %         + Spaltenbreite

/Zeichen-f
{ newpath %----- 1. Zeichen "f" -----
  0 ModulSp moveto
  0 ModulSp 2 mul rlineto
  Modul ModulSp 3 mul Modul 180 90 arcn
  ModulSp 0 rlineto
  0 Modul neg rlineto
  ModulSp neg 0 rlineto
  0 Sp neg rlineto
  ModulSp 0 rlineto
  0 Modul neg rlineto
  ModulSp neg 0 rlineto
  0 ModulSp neg rlineto
  closepath
} bind def

/Zeichen-h
{ newpath %----- 2. Zeichen "h" -----
  ModulSp 2 mul ModulSp 2 mul moveto
```

```

    0 ModulSp 2 mul Modul add rlineto
    Modul 0 rlineto
    0 ModulSp neg rlineto
    Sp 0 rlineto
    ModulSp 3 mul ModulSp 3 mul Modul 90 0 arcn
    0 ModulSp neg rlineto
    Modul neg 0 rlineto
    0 ModulSp rlineto
    Sp neg 0 rlineto
    0 ModulSp neg rlineto
    closepath
} bind def

/Zeichen-m
{ newpath %----- 3. Zeichen "m" -----
  ModulSp 0 moveto
  0 ModulSp Modul add rlineto
  ModulSp 2 mul 0 rlineto
  ModulSp 3 mul ModulSp Modul 90 0 arcn
  0 ModulSp neg rlineto
  Modul neg 0 rlineto
  0 ModulSp rlineto
  Sp neg 0 rlineto
  0 ModulSp neg rlineto
  Modul neg 0 rlineto
  0 ModulSp rlineto
  Sp neg 0 rlineto
  0 ModulSp neg rlineto
  closepath
} bind def

gsave
LinkerRand UntererRand translate

Zeichen-f 1 Tonwert sub setgray fill
Zeichen-f 0 setgray stroke
Zeichen-h 1 Tonwert sub setgray fill
Zeichen-h 0 setgray stroke
Zeichen-m 1 Tonwert sub setgray fill
Zeichen-m 0 setgray stroke

grestore
} bind def %-----

/mm { 72 25.4 div mul } def % Umrechnung PostScript-point in mm

/Satzspiegel-Rahmen
{ newpath
  0.25 mm setlinewidth
  35 mm 35 mm translate
  0 mm 0 mm moveto
  0 mm 217 mm rlineto
  150 mm 0 mm rlineto
  0 mm -217 mm rlineto
  closepath
  stroke

```

```

} bind def

/Satzspiegel-mm-Netz
{ /Times-Roman findfont 10 scalefont setfont
/dXY 2 mm def      % bei 1 mm Pixelinterferenzen bei 300 dpi
/DXY dXY 72 div 25.4 mul round cvi def
/MM-Setgray { /MM XY 72 div 25.4 mul round cvi def
              MM DXY mod 0 eq
              { 0.10 mm setlinewidth 0.5 setgray } if
              MM 10 mod 0 eq
              { 0.15 mm setlinewidth 0.0 setgray } if
            } bind def
35 mm 35 mm translate
0 mm dXY 150.01 mm { /XY exch def
                    MM-Setgray
                    newpath
                    XY      0 mm moveto
                    0 mm 217 mm rlineto
                    stroke
                    XY      2 mm sub -3 mm moveto
                    0 setgray
                    MM 10 mod 0 eq { MM =string cvs show } if
                } for
0 mm dXY 217.01 mm { /XY exch def
                    MM-Setgray
                    newpath
                    0 mm XY      moveto
                    150 mm 0 mm rlineto
                    stroke
                    -6 mm XY 0.5 mm sub moveto
                    0 setgray
                    MM 10 mod 0 eq { MM =string cvs show } if
                } for
    } bind def

Satzspiegel-Rahmen
% Satzspiegel-mm-Netz

/Tonwert 1 def
% LinkerRand, UntererRand, Breite, Prozeduraufruf
    2 mm      160 mm      35 mm      FHM-Logo

%----- Es folgt Kreisel -----
gsave
/xx0 80 mm def
/yy0 165 mm def
/DeltaAlpha 5 def
xx0 yy0 translate
0 DeltaAlpha 360 { /Alpha exch def
                  Alpha rotate
                  /Tonwert 1 Alpha 360 div sub def
                  5 mm 5 mm 20 mm FHM-Logo
                  Alpha neg rotate
                } bind for
grestore

%----- Es folgt Dynamik I links -----
gsave

```

```

/xx0      18 mm def
/yy0      100 mm def
/DeltaXY   1   def
/MaxXY     40   def
xx0 yy0 translate
0 DeltaXY MaxXY { /XY exch def
                  /X  XY DeltaXY mul      def
                  /Y  XY DeltaXY mul neg 0.5 mul def
                  /Tonwert 1 XY DeltaXY mul MaxXY div sub def
                  X mm Y mm 20 mm FHM-Logo
                } bind for
grestore

%----- Es folgt Dynamik I rechts -----
gsave
/xx0      122 mm def
/yy0      100 mm def
/DeltaXY   1   def
/MaxXY     40   def
xx0 yy0 translate
0 DeltaXY MaxXY { /XY exch def
                  /X  XY DeltaXY mul      neg def
                  /Y  XY DeltaXY mul neg 0.5 mul def
                  /Tonwert 1 XY DeltaXY mul MaxXY div sub def
                  X mm Y mm 20 mm FHM-Logo
                } bind for
grestore

%----- Es folgt Dynamik II -----
gsave
/xx0 10 mm def
/yy0 10 mm def
xx0 yy0 translate
/di 1 def
/iMax 50 def
0 1 iMax { /i exch def
           /X i di mul 2.1 mul def
           /Y i 10 mul sin 3 mul def
           /Tonwert 1 i di mul iMax div sub def
           i 0.5 mul rotate
           X mm Y mm i mm 0.1 mul 10 mm add FHM-Logo
           i 0.5 mul neg rotate
         } bind for
grestore

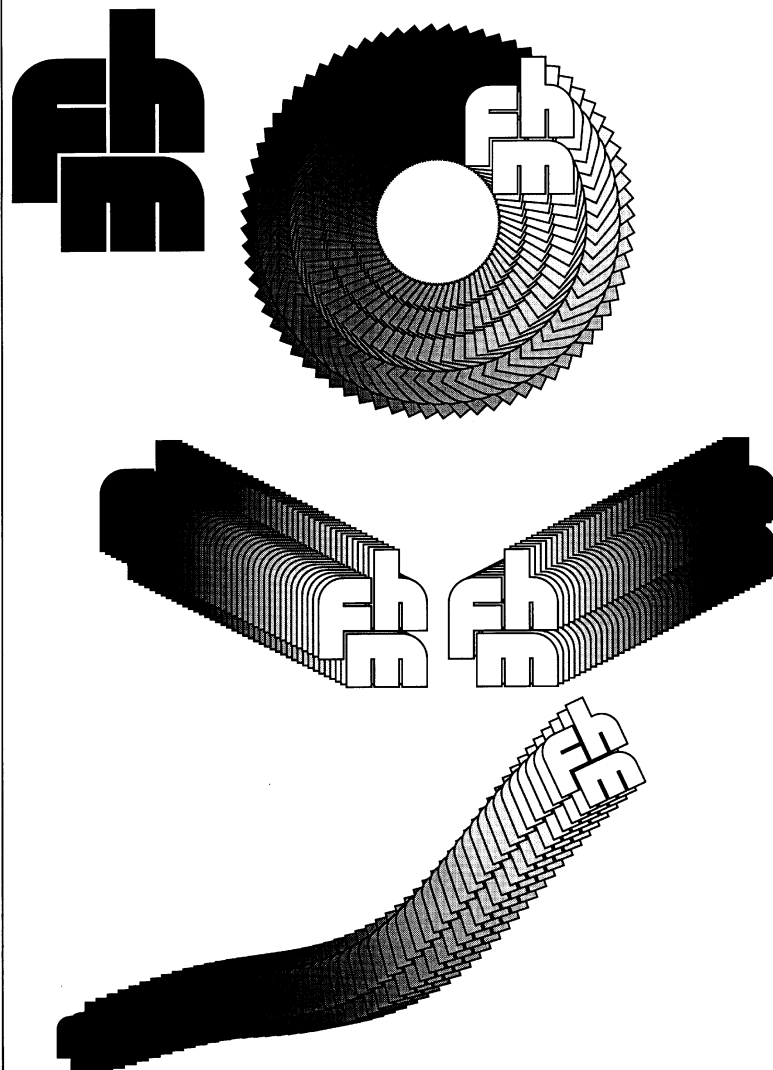
/Times-Roman findfont 12 scalefont setfont
2 mm 1 mm moveto
(Dr. K. Haller, Fachhochschule Muenchen, ) show
(Studiengang Druckereitechnik, 1221099) show

/Times-Bold findfont 16 scalefont setfont
2 mm 210 mm moveto
(.... weitere PostScript-Spielereien. "FHM-Lo3.PS") show

showpage %-----

```

.... weitere PostScript-Spielereien. "FHM-Lo3.PS"



Dr. K. Haller, Fachhochschule Muenchen, Studiengang Druckereitechnik, 1221099

04.9 PostScript-Spotfunktion für die klassische Rasterpunktform "Rpm90-kl.PS"

Die Beschreibung und die mathematische Ableitung der klassischen Rasterpunktform ist im Internet-Auftritt des Studienganges Druck- und Medientechnik unter "Beiträge zur digitalen Rastertechnologie" enthalten.

Die URL:

www.fh-muenchen.de/home/fb/fb05/druck/Prof_Haller/Rastertechnologie/HTML-Rpm-Start.htm

Hier das Programm "Rpm90-kl.PS":

```

%!PS-Adobe 2.0  =====
% PostScript-Datei "Rpm90-kl.PS"
% 100 Rasterfelder mit Tonwerten 1 bis 100% drucken. Dr. Karl Haller, 30.6.1999

/RF-Lcm 3 def % Rasterfrequenz (Rasterlinienzahl, Rasterfeinheit, Rasterweite)
               % in L/cm. Fuer Demo z.B. 10 L/cm, ggf. aendern
/RW 0 def % Rasterwinkel in Grad, ggf. aendern
%----- Ab hier nichts mehr aendern -----
/RF RF-Lcm 2.54 mul def % Umrechnung Rasterfrequenz in lpi (lines per inch)

RF RW { /zRpm-Klassisch % PostScript-Spotfunktion fuer
    { dup mul exch dup mul dup 4 mul % die klassische Rasterpunktform
      3 1 roll sub 1 sub dup mul exch sub % Nicht optimal, da vierfacher
    } def % Punktschluss mit grossem
    abs exch abs exch 2 copy add 1 le % Tonwertsprung.
    { zRpm-Klassisch } % Besser: Optimale Rasterpunkt-
    { 1 sub abs exch 1 sub abs exch % form nach "Rpm90" mit einem
      zRpm-Klassisch neg } ifelse % Kettenbereich von 15%.
    } setscreen % Dr. K. Haller

%-----
/mm { 72 25.4 div mul } def % Umrechnung mm in Punkte (1 Punkt = 1 Pt = 1/72")
/x0 30 mm def % Nullpunkt x0 von links unten
/y0 220 mm def % Nullpunkt y0 von links unten

/k 10 mm def % Kantenlaenge der Rasterfelder
/dx 3 mm def % Abstand zwischen den Rasterfeldern in x-Richtung
/dy 5 mm def
/DX k dx add def % Abstand der Rasterfelder in x-Richtung
/DY k dy add def

/Str 10 string def % Allgemeine String-Variable

/Helvetica findfont 20 scalefont setfont
x0 y0 20 mm add moveto (Klassische Rasterpunktform in PostScript) show
/Helvetica findfont 10 scalefont setfont
x0 y0 16 mm add moveto (Besser: Optimale Rasterpunktform nach ) show
                    ("Rpm90" mit 15% Kettenbereich. kha) show
x0 y0 10 mm add moveto (Rasterfrequenz: ) show
RF-Lcm Str cvs show ( L/cm) show ( = ) show
RF Str cvs show ( lpi) show
x0 y0 6.5 mm add moveto (Rasterwinkel: ) show
RW Str cvs show (\312) show
x0 y0 45 mm add moveto (Fachhochschule Muenchen, ) show
(Studiengang Druck- und Medientechnik, Prof. Dr. K. Haller) show
x0 y0 40 mm add moveto (PostScript-Datei "Rpm90-kl.PS") show
/Helvetica findfont 6 scalefont setfont
%-----

```



```
/iMax 10 def
1 1 iMax { /iy exch def    % "Laufvariable" von Stackspitze auf Variable "iy"
            /y y0 iy 1 sub DY mul sub def
            1 1 iMax { /ix exch def    % "Laufvariable" auf "ix"

                        /x x0 ix 1 sub DX mul add def

                        /Phi iy 1 sub iMax mul ix add iMax iMax mul div def
                        x y 0.5 mm add moveto
                        Phi 100 mul cvi Str cvs show (%) show

                        x y moveto          % Rahmen zeichnen
                        0.1 mm setlinewidth
                        k 0 rlineto 0 k neg rlineto k neg 0 rlineto
                        closepath stroke

                        x y moveto          % Rasterfeld zeichnen
                        k 0 rlineto 0 k neg rlineto k neg 0 rlineto
                        closepath 1 Phi sub setgray fill

                        0 setgray
                    } for    % Ende ix-Schleife
            } for          % Ende iy-Schleife

x0 y0 iMax DY mul sub moveto % Volltonbalken zeichnen
iMax DX mul dx sub 0        rlineto
0 k neg                      rlineto
iMax DX mul dx sub neg 0 rlineto
closepath 0 setgray fill

showpage %===== EOF =====
```

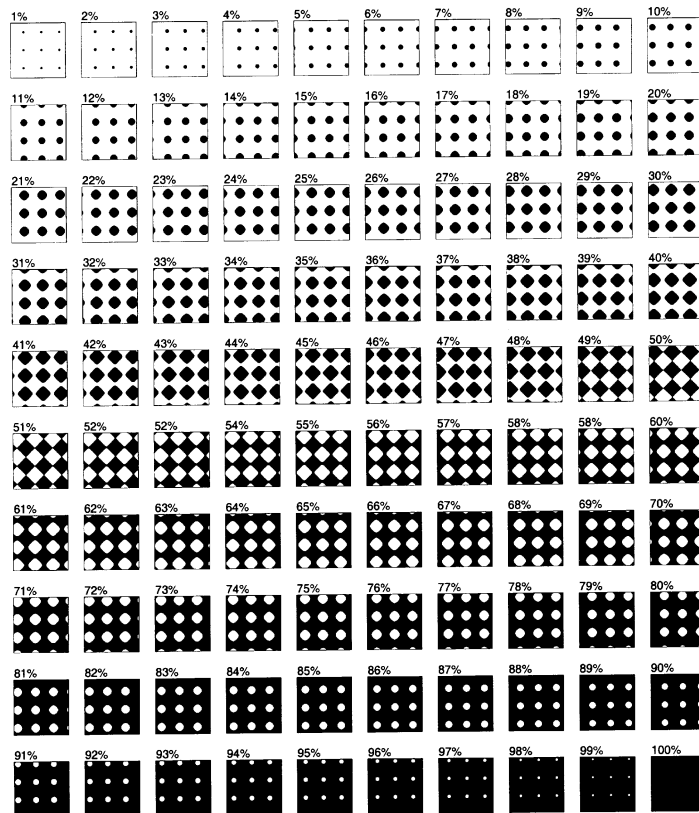
Fachhochschule Muenchen, Studiengang Druck- und Medientechnik, Prof. Dr. K. Haller
PostScript-Datei "Rpm90-kl.PS"

Klassische Rasterpunktform in PostScript

Besser: Optimale Rasterpunktform nach "Rpm90" mit 15% Kettenbereich. kha

Rasterfrequenz: 3 L/cm = 7.62 lpi

Rasterwinkel: 0°



04.10 PostScript-Spotfunktionsammlung für andere Rasterpunktformen "Spot-Fkt.PS"

Dieses Datei enthält eine Sammlung von einfachen Spotfunktionen.

Am Ende des Quelltextes ist die Nummer gewünschten Spotfunktion einzugeben, z.B. "Nr_01". Rasterfrequenz und Rasterwinkel können ebenfalls gewählt werden. Das Programm erzeugt Rasterfelder mit den Flächendeckungsgraden 1%, 2%; ... 100%.

```

%!PS-Adobe 2.0
% ===== PostScript-Programm "Spot-Fkt.PS" Spotfunktionen =====
% Sammlung von einfachen Spotfunktionen. Zusammengestellt zum FOGRA-
% Symposium am 17./18.04.91: "Aktuelle Entwicklungen im Computer Publishing"
% Dr. K. Haller, Fachhochschule Muenchen, Studiengang Druckereitechnik
% Modifiziert und erweitert am: 27.04.91, 15.06.91, 03.11.1999 K. Haller
%
% +-----+
% | Die gewuenschte Spotfunktion mittels Nummer entsprechend dem |
% | Hinweis am Programmende einsetzen. In aehnlicher Weise auch |
% | die Rasterlinienzahl (Rasterfrequenz) in L/cm |
% | und den Rasterwinkel am Programmende eintragen. |
% +-----+
% ---- Beginn der Sammlung -----
/Nr_01 { /Text1 (Nr_01: Positiv-Kreispunkte durchgehend) def
        /Text2 (Ab F = 78.5% = 100%*Pi/4 ueberlappen sich die Kreise,) def
        /Text3 (was zu unguenstigen Rasterpunktformen fuehrt) def
        /Text4 (z = x^2 + y^2 - 1 Siehe auch Nr_30, 31, 32) def
        /Text5 ({ dup mul exch dup mul add 1 exch sub }) def
        dup mul exch dup mul add 1 exch sub
      } def

/Nr_02 { /Text1 (Nr_02: Linienraaster) def
        /Text2 (Die einfachste und schnellste Spotfunktion) def
        /Text3 (Fuer Testzwecke durchaus sinnvoll) def
        /Text4 (z = x) def
        /Text5 ({ pop }) def
        pop
      } def

/Nr_03 { /Text1 (Nr_03: Klassischer Rasterpunkt I, algebraischer Term) def
        % Entwicklung K. Haller
        /Text2 (Spezielles FOGRA-Rasterpunktmodell, Variante I ) def
        /Text3 (FOGRA-Forschungsbericht 6.012, Seite 38, Glch. 63) def
        % anfüegen: Koordinatendrehung 45°
        % oder Seite 89, Gleichung 93, mit b = 1
        /Text4 (Nach Rotation 45 Grad: z = ..... ) def
        /Text5 ({ .....}) def
        /Rpm { dup mul exch dup mul dup 4 mul 3 1 roll
              sub 1 sub dup mul exch sub } def
        abs exch abs exch 2 copy add 1 gt
        { 1 sub abs exch 1 sub abs exch Rpm neg }
        { Rpm } ifelse
      } def

/Nr_04 { /Text1 (Nr_04: Klassische Rasterpunktform II, Cos mal Cos) def
        % Entwicklung K. Haller
        /Text2 (Spezielles FOGRA-Rasterpunktmodell, Variante II ) def
        /Text3 (FOGRA-Forschungsbericht 6.012, Seite 33, Glch. 56 ) def
        % anfüegen: Koordinatendrehung 45°
        /Text4 (Nach Rotation 45 Grad: Cos[90*(x - y)] * Cos[90*(x + y)]) def

```

```

    /Text5 ({ 2 copy sub 90 mul cos 3 1 roll add 90 mul cos mul }) def
    2 copy sub 90 mul cos 3 1 roll add 90 mul cos mul
  } def

/Nr_05 { /Text1 (Nr_05: Klassische Rasterpunktform III, Cos plus Cos) def
  /Text2 (PostScript Language Reference Manual, 2nd. Edition) def
  /Text3 (Addison-Wesley, USA, Dez. 1990) def
  /Text4 ([Cos(180*x) + Cos(180*y)] / 2) def
  /Text5 ({ 180 mul cos exch 180 mul cos add 2 div }) def
  180 mul cos exch 180 mul cos add 2 div
} def

/Nr_06 { /Text1 (Nr_06: Super Circle, Kreis/klassische Punktform) def
  /Text2 (St. F. Roth: "Real World PostScript", Addison-Wesley) def
  /Text3 (USA, 1988. Siehe auch IGT-Rasterpunktmodell 1959) def
  /Text4 (Bis 39.3% (100%*Pi/8) Kreisform, dann bis 50% abgerundete) def
  /Text5 (Quadrate. Bei 50% exakte Quadratform ) def
  abs exch abs 2 copy add 1 gt
  { 1 sub dup mul exch 1 sub dup mul add 1 sub }
  { dup mul exch dup mul add 1 exch sub } ifelse
} def

/Nr_07 { /Text1 (Nr_07: Quadrate) def
  /Text2 (Wenn es unbedingt Quadrate sein muessen, ) def
  /Text3 (dann bitte sehr.) def
  /Text4 (z = [Abs(x) + Abs(y)] / 2) def
  /Text5 ({ abs exch abs add 2 div }) def
  abs exch abs add 2 div
} def

/Nr_08 { /B 0.85 def
  /Text1 (Nr_08: Ellipse-Kette-Ellipse) def
  /Text2 (B = 0.85, B = 0: Linienraster, B = 1: Kreispunkte) def
  /Text3 (Probleme wie beim durchgehenden Kreispunkt) def
  /Text4 (z = y^2 + (B*x)^2 - 1) def
  /Text5 ({ dup mul exch B mul dup mul add 1 exch sub }) def
  dup mul exch B mul dup mul add 1 exch sub
} def

/Nr_09 { /Text1 (Nr_09: Herzchen) def
  % Entwicklung K. Haller
  /Text2 (Hier ohne Laufzeitoptimierung) def
  /Text3 (Mit Optimierung ca. 2.5 * schneller) def
  /Text4 (z = ..... ) def
  /Text5 ({ ..... }) def
  /A 0.9 def /B 0.4 def /C 1.4 def
  % Herzchen-Parameter A, B und C
  neg 2 copy exch abs B C add dup mul A dup mul add 1 sub sqrt
  B C add A mul sub 1 A dup mul sub div dup dup mul 1 add
  C mul B C add B C add dup mul A dup mul add 1 sub sqrt B
  C add A mul sub 1 A dup mul sub div mul A add div sub mul
  le { exch abs B C add dup mul A dup mul add 1 sub sqrt B
  C add A mul sub 1 A dup mul sub div mul exch sub C div 1
  B C add dup mul A dup mul add 1 sub sqrt B C add A mul
  sub 1 A dup mul sub div add C div div 2 mul 1 sub neg }
  { A dup mul B dup mul add 1 sub abs 1e-6 gt { exch abs 2
  copy A mul exch B mul add dup 4 1 roll dup mul 3 1 roll
  dup mul exch dup mul add A dup mul B dup mul add 1 sub
  mul sub sqrt sub A dup mul B dup mul add 1 sub div } { 2
  copy B mul exch abs A mul add 2 mul 3 1 roll dup mul exch
  dup mul add exch div } ifelse 1 B C add dup mul A dup mul
  add 1 sub sqrt B C add A mul sub 1 A dup mul sub div add
  C div div 2 mul 1 sub neg } ifelse
} def

```

```

/Nr_10 { /A 1.0 def
% Entwicklung K. Haller
/Text1 (Nr_10: Kleeblatt) def
/Text2 (Parameter A = 1) def
/Text3 (A = 0...1, bei A = 0: Kreispunkte) def
/Text4 (Bringt aber kein Druckerglueck. z = ...) def
/Text5 ({ .....}) def
abs exch abs 2 copy add A mul dup 4 1 roll dup mul 3 1
roll dup mul exch dup mul add A dup mul 2 mul 1 sub mul
sub sqrt sub A dup mul 2 mul 1 sub div A 2 mul 2 sqrt sub
A dup mul 2 mul 1 sub div A dup dup mul 1 exch sub sqrt
sub A dup mul 2 mul 1 sub div gt { A 2 mul 2 sqrt sub A
dup mul 2 mul 1 sub div } { A dup dup mul 1 exch sub sqrt
sub A dup mul 2 mul 1 sub div } ifelse div 2 mul 1 exch
sub
} def

/Nr_11 { /Text1 (Nr_11: Dreieck) def
/Text2 ( ..... ) def
/Text3 ( ..... ) def
/Text4 (z = (y + x)/2) def
/Text5 ({ add 2 div }) def
add 2 div
} def

/Nr_12 { /Text1 (Nr_12: Propeller) def
/Text2 (Fliegen ist aber schoener) def
/Text3 ( ) def
/Text4 (z = y * x) def
/Text5 ({ mul }) def
mul
} def

/Nr_13 { /Text1 (Nr_13: Und so weiter ...) def
/Text2 ( ..... ) def
/Text3 ( ..... ) def
/Text4 (z = Abs(y*x) ) def
/Text5 ({ mul abs }) def
mul abs
} def

/Nr_14 { /Text1 (Nr_14: Auch ganz schoen ...) def
/Text2 ( ..... ) def
/Text3 ( ..... ) def
/Text4 (z = (y * x)^2 ) def
/Text5 ({ mul dup mul }) def
mul dup mul
} def

/Nr_15 { /Text1 (Nr_15: Mal was anderes ...) def
/Text2 ( ..... ) def
/Text3 ( ..... ) def
/Text4 (z = x * y^2 ) def
/Text5 ({ dup mul dup }) def
dup mul mul
} def

/Nr_16 { /Text1 (Nr_16: Doch nicht so neu ...) def
/Text2 ( ..... ) def
/Text3 ( ..... ) def
/Text4 (z = y * x^2 ) def
/Text5 ({ exch dup mul mul }) def
exch dup mul mul

```

```

    } def

/Nr_17 { /Text1 (Nr_17: Auch diese nicht ...) def
  /Text2 ( ..... ) def
  /Text3 ( ..... ) def
  /Text4 (z = x * (y^2 - 1)) def
  /Text5 ({ dup mul 1 sub mul }) def
  dup mul 1 sub mul
} def

/Nr_18 { /Text1 (Nr_18: Irgendwie bekannt ...) def
  /Text2 ( ..... ) def
  /Text3 ( ..... ) def
  /Text4 (z = (x*y)^2 ) def
  /Text5 ({ mul dup mul } ) def
  mul dup mul
} def

/Nr_19 { /Text1 (Nr_19: Auch diese aehnelt einem Vorgaenger ...) def
  /Text2 ( ..... ) def
  /Text3 ( ..... ) def
  /Text4 (z = 1 - (x*y)^2 ) def
  /Text5 ({ mul dup mul 1 exch sub }) def
  mul dup mul 1 exch sub
} def

/Nr_20 { /Text1 (Nr_20: Auch das noch ... !) def
  /Text2 (Mit der rechten Hand ) def
  /Text3 (hintern linken Ohr gekratzt. ) def
  /Text4 (z = Sin(x) * Cos(x)) def
  /Text5 ({ sin exch cos mul }) def
  sin exch cos mul
} def

/Nr_21 { /Text1 (Nr_21: Mit aller Gewalt ...) def
  /Text2 (x := x + 0.1 ..... ) def
  /Text3 (y := y + 0.1 ..... ) def
  /Text4 (z = Frac(|x|^|y|) ) def
  /Text5 ({abs 0.1 add exch abs 0.1 add exch exp dup truncate sub}) def
      abs 0.1 add exch abs 0.1 add exch exp dup truncate sub
} def

/Nr_22 { /Text1 (Nr_22: Oder auch so ...) def
  /Text2 ( x := x + 0.1 ) def
  /Text3 ( y := y ) def
  /Text4 ( z = Frac(|x|^y) ) def
  /Text5 ({ exch abs 0.1 add exch exp dup truncate sub }) def
  exch abs 0.1 add exch exp dup truncate sub
} def

/Nr_23 { /Text1 (Nr_23: Allmaehlich reicht es) def
  /Text2 ( ..... ) def
  /Text3 ( ..... ) def
  /Text4 (z = Sqrt(Abs(x*y)) ) def
  /Text5 ({ mul abs sqrt }) def
  mul abs sqrt
} def

/Nr_24 { /Text1 (Nr_24: Zum Schluss ein "Zufallsraster") def
  /Text2 (Eine optimale Standard-Rasterpunktform waere doch besser.) def
  /Text3 (Zum Beispiel nach "Rasterpunktmodell 90".) def
  /Text4 (z = Zufallszahl / MaxInt ) def
  /Text5 ({ clear rand MaxInt div }) def
  /MaxInt 2147483647 def % 2^31 - 1

```

```

    clear rand MaxInt div
  } def

/Nr_25 { /Text1 (25: Gitter_negativ ) def
  /Text2 (Diese Punktform wird mitunter auch als quadratisch) def
  /Text3 (bezeichnet, was ziemlich irrefuehrend ist. Siehe 07.) def
  /Text4 (Pyramide_1: if Abs(x) < Abs(y) ) def
  /Text5 (then z := +Abs(x) else z := +Abs(y) ) def
  abs exch abs 2 copy exch lt
  { exch pop }
  { pop } ifelse
} def

/Nr_26 { /Text1 (26: Raute durchgehend) def
  /Text2 (Die einfachste und schnellste Moeglichkeit fuer ) def
  /Text3 (die Darstellung von "Kettenpunkten".) def
  /B 0.85 def % Punktschlußfaktor b = 0 ... 0.85 ... 1.0
  /Str20 20 string def
  /Text4 (Punktschlußfaktor b zwischen 0 und 1) def
  /Text5 B Str20 cvs def
  abs exch abs 1 exch sub B mul exch sub
} def

/Nr_27 { /Text1 (27: Gitter_negativ) def
  /Text2 (Diese Punktform wird mitunter auch als quadratisch) def
  /Text3 (bezeichnet, was ziemlich irrefuehrend ist. Siehe 07.) def
  /Text4 (Pyramide_1: if Abs(x) < Abs(y) ) def
  /Text5 (then z := +Abs(x) else z := +Abs(y)) def
  abs exch abs 2 copy exch lt
  { exch pop }
  { pop } ifelse
} def

/Nr_28 { /Text1 (28: Gitter_positiv) def
  /Text2 (Auch diese Punktform wird mitunter als quadratisch) def
  /Text3 (bezeichnet, was ziemlich irrefuehrend ist. Siehe 07.) def
  /Text4 (Pyramide_2: if Abs(x) < Abs(y) ) def
  /Text5 (then z := -Abs(x) else z := -Abs(y)) def
  abs exch abs 2 copy exch lt
  { exch pop neg }
  { pop neg } ifelse
} def

/Nr_29 { /Text1 (29: Gitter_positiv_negativ) def
  /Text2 (Durch einen bewussten Vorzeichen-Fehler im else-Zweig) def
  /Text3 (wird gegenueber Nr_27/28 eine "Kick-Pyramide" erzeugt.) def
  /Text4 (Knick-Pyramide: if Abs(x) < Abs(y) ) def
  /Text5 (then z := -Abs(x) else z := +Abs(y)) def
  abs exch abs 2 copy exch lt
  { exch pop neg }
  { pop } ifelse
} def

/Nr_30 { /Text1 (Nr_30: Negativ-Kreispunkte durchgehend) def
  /Text2 (Bis F = 21.5% = 100*(1-Pi/4) ueberlappen sich die Kreise,) def
  /Text3 (was zu unguenstigen Rasterpunktformen fuehrt) def
  /Text4 (z = x^2 + y^2 - 1 Siehe auch Nr_01, und 31) def
  /Text5 ({ dup mul exch dup mul add 1 exch sub neg }) def
  dup mul exch dup mul add 1 exch sub neg
} def

/Nr_31 { /Text1 (Nr_31: Kreispunkte mit Punktformwechsel) def
  /Text2 (Wechsel von Positivform zu Negativform bei 50% ) def
  /Text3 (Wechsel aber beliebig zwischen 21.5% und 78.5% moeglich) def

```

```

/Text4 (78.5% =  $\pi/4 \cdot 100\%$   $z = x^2 + y^2 - 1$ ) def
/Text5 () def

/Phi_Wechsel 0.50 def % Phi = F[%]/100
/Pi 3.141593 def
/R_Quadrat Phi_Wechsel 4 mul Pi div def

% dup mul exch dup mul add dup R_Quadrat lt {1 sub neg} {1 sub} ifelse
2 copy dup mul exch dup mul add R_Quadrat lt
{ dup mul exch dup mul add 1 sub neg }
{ abs 1 sub dup mul exch abs 1 sub dup mul add 1 sub } ifelse

} def

% ---- Ende der Sammlung -----

/RasterfelderDrucken
{ /RF-Lcm Rasterfrequenz def
  /RF RF-Lcm 2.54 mul def

  /RW Rasterwinkel def

  /mm { 72 25.4 div mul } def % Umrechnung mm in Punkt
                                % 1 Punkt = 1 pt = 1/72" = 0.3528 mm
  /x0 40 mm def % Nullpunkt x0 von links unten
  /y0 240 mm def % Nullpunkt y0 von links unten

  /k 10 mm def % Kantenlaenge der Rasterfelder
  /dx 3 mm def % Abstand zwischen den Rasterfeldern in x-Richtung
  /dy 5 mm def
  /DX k dx add def % Abstand der Rasterfelder in x-Richtung
  /DY k dy add def

  /Str 10 string def % Allgemeine String-Variable

  /Helvetica findfont 13 scalefont setfont

  x0 y0 25 mm add moveto (Fachhochschule Muenchen, ) show
  (Studiengang Druck- und Medientechnik) show

  /Helvetica findfont 10 scalefont setfont

  x0 y0 15 mm add moveto (Prof. Dr. Karl Haller) show

  x0 y0 10 mm add moveto
  (PostScript-Datei "Spot-Fkt.PS", Sammlung von einfachen Spotfunktionen) show

  /y00 y0 170 mm sub def
  x0 y00 moveto (Rasterfrequenz: ) show
    RF-Lcm Str cvs show ( L/cm) show ( = ) show
    RF Str cvs show ( lpi) show
  x0 y00 5 mm sub moveto (Rasterwinkel: ) show
    RW Str cvs show (\312) show

  x0 y00 10 mm sub moveto (Spotfunktion ) show Text1 show
  x0 y00 15 mm sub moveto Text2 show
  x0 y00 20 mm sub moveto Text3 show
  x0 y00 25 mm sub moveto Text4 show
  x0 y00 30 mm sub moveto Text5 show

  /Helvetica findfont 6 scalefont setfont
%-----

```



```

/iMax 10 def
1 1 iMax { /iy exch def % "Laufvariable" von Stackspitze auf "iy"
           /y y0 iy 1 sub DY mul sub def
           1 1 iMax { /ix exch def % "Laufvariable" auf "ix"

                       /x x0 ix 1 sub DX mul add def

                       /Phi iy 1 sub iMax mul ix add iMax iMax mul div def
                       x y 0.5 mm add moveto
                       Phi 100 mul cvi Str cvs show (%) show

                       x y moveto % Rahmen zeichnen
                       0.1 mm setlinewidth
                       k 0 rlineto 0 k neg rlineto k neg 0 rlineto
                       closepath stroke

                       x y moveto % Rasterfeld zeichnen
                       k 0 rlineto 0 k neg rlineto k neg 0 rlineto
                       closepath 1 Phi sub setgray fill

                       0 setgray
           } for % Ende ix-Schleife
       } for % Ende iy-Schleife

       x0 y0 iMax DY mul sub moveto % Volltonbalken zeichnen
       iMax DX mul dx sub 0 rlineto
       0 k neg rlineto
       iMax DX mul dx sub neg 0 rlineto
       closepath 0 setgray fill

   } def
% -----

/Rasterfrequenz 5 def % L/cm, geraeteabhaengig. Fuer Demo moeglichst klein
/Rasterwinkel 45 def % In Grad.

Rasterfrequenz 2.54 mul Rasterwinkel { Nr_01 } setscreen RasterfelderDrucken
% ^^^^^ Hier Nr der gewuenschten Spot-
% ^^^^^ funktion einsetzen, z.B. "Nr_03"

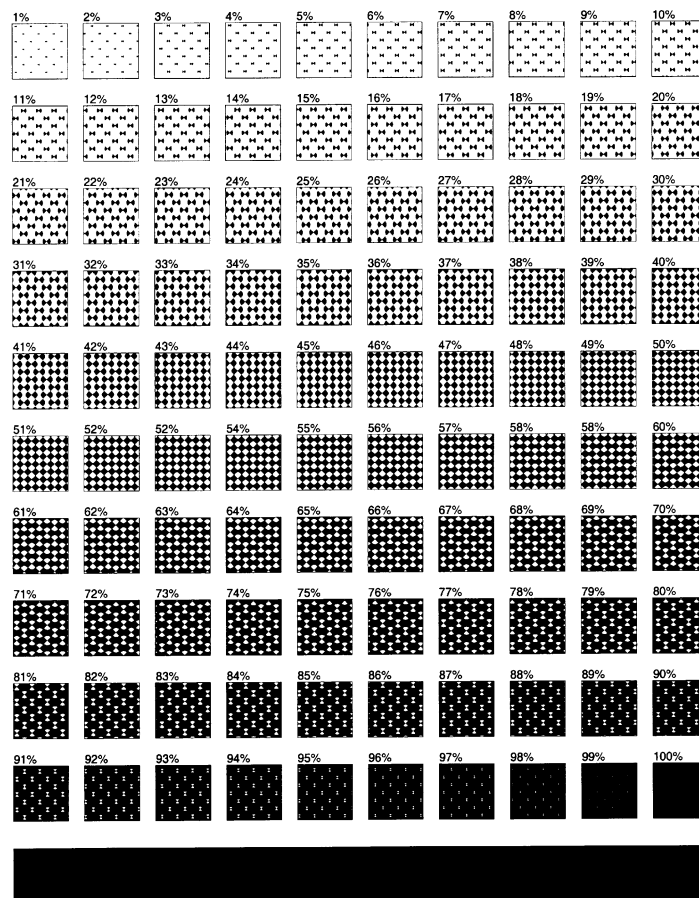
showpage % -----

```

Fachhochschule Muenchen, Studiengang Druck- und Medientechnik

Prof. Dr. Karl Haller

PostScript-Datei "Spot-Fkt.PS", Sammlung von einfachen Spotfunktionen



Rasterfrequenz: 5 L/cm = 12.7 lpi

Rasterwinkel: 45°

Spotfunktion Nr_12: Propeller

Fliegen ist aber schoener

$z = y * x$

{ mul }

04.11 Die Moduldatei "Modul-0.PS"

In fast allen PostScript-Dateien werden bestimmte Prozeduren immer wieder benötigt. Statt sie jedesmal neu zu programmieren, kann man sie in einer Modul-Datei zusammenfassen, ähnlich den Units oder Include-Dateien in Pascal. Wenn man die Moduldatei zuerst auf das PostScript-Ausgabegerät schickt, kann man in der darauffolgenden PostScript-Datei (Arbeitsdatei) die Deklarationen der Moduldatei benutzen.

Zur Veranschaulichung ein Beispiel: Die Moduldatei habe den Namen "Modul-0.PS" und die aktuelle Arbeitsdatei den Namen "XYZ.PS". Mit

COPY MODUL-0.PS + XYZ.PS PRN

werden beide Dateien an den Drucker geschickt. Man kann auch beide Dateien zu einer Datei zusammenkopieren.

Die folgende Moduldatei "Modul-01.PS" enthält eine Sammlung von Prozeduren, die der Autor für sehr nützlich hält, so z.B. Umcodierung des Zeichensatzes, damit Umlaute "bequem" eingefügt werden können. Die Moduldatei kann auch alleine ausgedruckt werden, wenn man in der Zeile 15 den Boolean-Wert von "Demo_Modul-0" ... auf "true" setzt. Für konkrete Anwendung ist "false" einzusetzen, damit die Moduldatei selbst keine Ausgabe erzeugt.

```
%!PS-Adobe-2.0
% PostScript-Modul "Modul-0.PS", 10.12.1998, K. Haller, FH Muenchen, DR
% Enthael: 1. "Umlaute.PS", Einbau der Umlaute
%          2. "System.PS", Systemspezifische Parameter abfragen
%          3. "Arcus.PS", ArcSin und ArcCos mit Fallunterscheidungen
%          4. "FHM-Logo.PS", Logo der FH Muenchen
% Zweck: Als "Include-Datei" v o r anderen PS-Programmen in Stapeldatei

/Genau { transform round exch round exch itransform } bind def
/Moveto { Genau moveto } bind def % Diese fuenf Prozeduren
/Lineto { Genau lineto } bind def % fuer genaue Positionierung
/Rmoveto { Genau rmoveto } bind def % ab Dez. 1993
/Rlineto { Genau rlineto } bind def % kha

/Demo_Modul-0 true def % Für Demo auf "true", danach auf "false"

/STOP { showpage systemdict begin quit } def % Nur fuer Test-Unterbrechung
/SHOWPAGE { showpage systemdict begin quit } def

/cm { 72 2.54 div mul } def
/mm { 72 25.4 div mul } def
/Zoll { 72 mul } def
/inch { 72 mul } def
/Punkt_in_cm { 72 div 2.54 mul } def
/Punkt_in_mm { 72 div 25.4 mul } def
/Punkt_in_Zoll { 72 div } def
/Punkt_in_inch { 72 div } def
/Genau { transform round exch round exch itransform } bind def

/LR 3 cm def % Linker Rand links, auch 0 cm
/Schriftgroesse 10 def % Schriftgroesse in Punkt
/relZeilenvorschub 1.20 def % 1 = Schriftgroesse, 1.20 = 20% mehr
```

```

/showLF { show LR currentpoint exch pop Schriftgroesse relZeilenvorschub mul
        sub moveto } def

/Tab    { LR add currentpoint exch pop moveto } def

/Pi      3.141592654 def % Kreiszahl
/Tan     { dup sin exch cos div } def % Tangens
/ArcSin  { dup +1.0 eq % Auf Bereich (-90 Grad bis +90 Grad) abgestimmt
          { pop +90.0 }
          { dup -1.0 eq { pop -90.0 }
            { dup dup mul 1 exch sub sqrt atan
              dup 90 gt { -360 add } if } ifelse
          } ifelse
        } bind def
/ArcCos  { dup -1.0 eq % Auf Bereich (0 Grad bis +180 Grad) abgestimmt
          { pop +180.0 }
          { dup neg 1 add exch 1 add div sqrt 1 atan 2 mul } ifelse
        } bind def

/Pfeil { /Pfeilfuellung exch def % true = mit Fuellung, false = nur Rand
        /Pfeilstrichdicke exch def
        /Pfeilstrich exch def % optionaler Zusatzstrich, wenn nicht 0
        /Pfeilrichtung exch def % 0, 90, 180, 270, 360 Grad oder beliebig
        /Pfeilwinkel exch def
        /Pfeillaenge exch def
        /yPfeil exch def
        /xPfeil exch def
        /AlteLiniendicke currentlinewidth def
        gsave
        Pfeilstrichdicke setlinewidth
        xPfeil yPfeil translate
        Pfeilrichtung rotate %xPfeil =string cvs show
        0 0 moveto
        Pfeillaenge Pfeilwinkel 2 div sin Pfeillaenge mul rlineto
        0 Pfeilwinkel 2 div sin Pfeillaenge mul 2 mul neg rlineto
        closepath
        Pfeilfuellung true eq
        { 0 setgray fill }
        { stroke } ifelse
        Pfeillaenge 0 moveto
        Pfeilstrich 0 rlineto
        Demo_Modul-0 true eq
        { Pfeilrichtung 0 eq Pfeilrichtung 45 eq or
          { /xTemp currentpoint pop def
            /yTemp currentpoint exch pop def
            xTemp 1 mm add yTemp 1 mm sub moveto
            Pfeilrichtung =string cvs show (\312) show
          } if
        } if
        stroke
        grestore
        AlteLiniendicke setlinewidth
        } bind def

/FHM-Logo %-----
{ /Breite exch def % Die drei
  /UntererRand exch def % Parameter des
  /LinkerRand exch def % Prozedur-Aufrufes
  %-----
  /MitSpaltenkorrektur true def % nur fuer Test auf "false"
  /rSp 0.10 def % rSp fuer einfaches "Hinting" anpassen
  /B0 12 mm def % Bezugsbreite
  /KF 0.0005 def % Korrekturfaktor
  MitSpaltenkorrektur Breite B0 lt DPI 2400 lt and and

```

```

{/rSp rSp0 B0 Breite sub dup mul 1 DPI 2400 div sub mul KF mul add def }
{/rSp rSp0 def } ifelse
/Modul Breite 4 rSp 3 mul add div def % Modul = Balkenbreite
/Sp Modul rSp mul def % Sp = Spaltenbreite
/ModulSp Modul Sp add def % ModulSp = Balkenbreite +
                               % + Spaltenbreite

%-----
gsave
LinkerRand UntererRand translate
newpath %----- 1. Zeichen "f" -----
0 ModulSp moveto
0 ModulSp 2 mul rlineto
Modul ModulSp 3 mul Modul 180 90 arcn
ModulSp 0 rlineto
0 Modul neg rlineto
ModulSp neg 0 rlineto
0 Sp neg rlineto
ModulSp 0 rlineto
0 Modul neg rlineto
ModulSp neg 0 rlineto
0 ModulSp neg rlineto
closepath 0.0 setgray fill
% Zu "setgray": 0.0 = schwarz
% 1.0 = weiss
newpath %----- 2. Zeichen "h" -----
ModulSp 2 mul ModulSp 2 mul moveto
0 ModulSp 2 mul Modul add rlineto
Modul 0 rlineto
0 ModulSp neg rlineto
Sp 0 rlineto
ModulSp 3 mul ModulSp 3 mul Modul 90 0 arcn
0 ModulSp neg rlineto
Modul neg 0 rlineto
0 ModulSp rlineto
Sp neg 0 rlineto
0 ModulSp neg rlineto
closepath 0.0 setgray fill
newpath %----- 3. Zeichen "m" -----
ModulSp 0 moveto
0 ModulSp Modul add rlineto
ModulSp 2 mul 0 rlineto
ModulSp 3 mul ModulSp Modul 90 0 arcn
0 ModulSp neg rlineto
Modul neg 0 rlineto
0 ModulSp rlineto
Sp neg 0 rlineto

0 ModulSp neg rlineto
Modul neg 0 rlineto
0 ModulSp rlineto
Sp neg 0 rlineto
0 ModulSp neg rlineto
closepath 0.0 setgray fill
grestore
} bind def %-----
% Aufrufbeispiel: LinkerRand, UntererRand, Breite, Prozeduraufruf
% 110 mm 120 mm 35 mm FHM-Logo

%=====
% 1. Umlaute nach PostScript-Einfuehrungshandbuch (blauer Band) Seite 249
% K. Haller, 12.07.92. Bezeichnungen hier modifiziert.
% Unveraenderte Originalversion siehe "Umlaute0.PS"

```

```

%-----+
/UmcodierungsDict 12 dict def % Lokaler Speicher fuer Prozedur "Umcodierung"
/Umcodierung % 12: 7 Sonderzeichen + 5 "def"
{ UmcodierungsDict
  begin
    /NeueCodesUndNamen exch def
    /NeuerFontName      exch def
    /BasisFontName      exch def
    /BasisFontDict      BasisFontName findfont      def
    /NeuerFont          BasisFontDict maxlength dict def
    BasisFontDict
    { exch dup /FID ne
      { dup /Encoding eq
        { exch dup length array copy NeuerFont 3 1 roll put }
        { exch
          NeuerFont 3 1 roll put } ifelse
      }
      { pop pop } ifelse
    } forall
    NeuerFont
    /FontName NeuerFontName put
    NeueCodesUndNamen aload pop
    NeueCodesUndNamen
    length 2 idiv { NeuerFont /Encoding get
                    3 1 roll put
                    } repeat
    NeuerFontName NeuerFont
    definefont pop
  end % Dictionary wieder loeschen
} def %----- Ende der Prozedur "Umcodierung" -----

/DeutscheSonderzeichen
[ 8#204 /adieresis 8#224 /odieresis 8#201 /udieresis 8#341 /germandbls
  8#216 /Adieresis 8#231 /Odieresis 8#232 /Udieresis
] def

% ----- Anwendungsbeispiel -----
/Helvetica /Helvetica-Deutsch DeutscheSonderzeichen Umcodierung
/Helvetica-Deutsch findfont Schriftgroesse scalefont setfont
% /Times-Roman /Times-Roman-Deutsch DeutscheSonderzeichen Umcodierung
% /Times-Roman-Deutsch findfont Schriftgroesse scalefont setfont

Demo_Modul-0
{ /T1 { 1.5 cm Tab } def % Temporaerer Tabulator
  LR 28.5 cm moveto
  (PostScript-Modul "Modul-0.PS", Umlaute und Systemparameter. ) show
  (Als Vorspann (Include-Datei) v o r ) showLF
  (andere PS-Programme. ) show
  (Test-Ausgabe des Moduls alleine nur, wenn ) show
  ("Demo_Modul-0" auf "true". ) showLF
  (Beispiel:) show T1 (Es soll die Datei "Test.PS" mit dem Modul ) show
  ("Modul-0.PS" auf einem MS-DOS-System aus-) showLF
  T1 (gegeben werden:      copy Modul-0.PS + Test.PS prn ) showLF
  /xAlt currentpoint pop      def
  /yAlt currentpoint exch pop def
  16 cm 0 cm rlineto
  xAlt yAlt moveto
  () showLF
  (Deutsche und andere Sonderzeichen:) showLF
  % ä ö ü Ä Ö ü ß ° •
  %204 224 201 216 231 232 341 312 267
  (\204 = \204) show 2.5 cm Tab (\224 = \224) show 5.0 cm Tab
  (\201 = \201) show 7.5 cm Tab (\341 = \341) show 10.0 cm Tab
  (\267 = \267) show ( (im Standard-Zeichensatz)) showLF
  (\216 = \216) show 2.5 cm Tab (\231 = \231) show 5.0 cm Tab
  (\232 = \232) show 7.5 cm Tab (\312 = \312) show
}

```

```

    ( (im Standard-Zeichensatz))      showLF
  } if

%=====+
% 2. Systemspezifische PostScript-Parameter abfragen, K. Haller, 03.03.92 |
% Siehe auch: N. Kollack, Programm 30: setscreen |
%-----+
/Druckername  31 string  def
/Produktname  31 string  def
/PS-Version   10 string  def
/PS-Revision  10 string  def

statusdict  % Dieses Dictionary auf den Stack
begin
  Druckername  printername pop
  /Produktname product      def
  /PS-Version  version      def
  /PS-Revision revision     =string cvs def
  Demo_Modul-0
  { () showLF
    (----- Systemparameter (21.09.92) -----) showLF
    (Verwendbarkeit der Bezeichner (global/lokal) beachten!) showLF
    (Global: Drucker "Druckername": ) show 7 cm Tab Druckername showLF
    (Lokal: Produkt "Produktname": ) show 7 cm Tab Produktname showLF
    (Lokal: PostScript-Version "PS-Version": ) show
    7 cm Tab PS-Version showLF
    (Lokal: PostScript-Revision "PS-Revision": ) show
    7 cm Tab PS-Revision showLF
  } if
end % Status-Dictionary vom Stack entfernen

/DPI 72 0 dtransform dup mul exch dup mul add sqrt def
% Zu dtransform = delta transform
% DX DY dtransform ==> DX' DY'
% Alle Real-Typen. DPI = Sqrt( DX^2 + DY^2 )

/Pixel 10000 DPI div 2.54 div def % Pixelgroesse in Mymeter

Demo_Modul-0
{ () showLF
  (Die Ger\204teaflu\224sung "DPI": ) show
  5 cm Tab DPI =string cvs show ( dpi = ) show
  DPI 2.54 mul =string cvs show ( dots per cm ) showLF
  (Pixelgr\224\341e "Pixel": ) show
  5 cm Tab Pixel =string cvs show
  /Symbol findfont Schriftgroesse scalefont setfont ( \155) show
  /Helvetica-Deutsch findfont Schriftgroesse scalefont setfont (m) showLF

  currentscreen pop /Winkel exch def /Frequenz exch def
  () showLF
  (Default-Werte von 'setscreen':) showLF
  (Rasterfrequenz "Frequenz": ) show
  5 cm Tab Frequenz =string cvs show ( lpi = ) show
  Frequenz 2.54 div =string cvs show ( L/cm) showLF
  (Rasterwinkel "Winkel": ) show
  5 cm Tab Winkel =string cvs show ( \312) showLF

  /xAlt currentpoint pop def
  /yAlt currentpoint exch pop def
  16 cm 0 cm rlineto stroke
  xAlt yAlt moveto
  () showLF

```

```

/T1 { 1.5 cm Tab } def % Temporaerer Kurz-Tabulator fuer interne Zwecke
/T2 { 5.5 cm Tab } def % dto.

(Im Hauptprogramm kann auf folgende Prozedur- und Variablen) show
(deklarationen von "Modul-0.PS" ) showLF
(zugegriffen werden (zudem auch auf "Frequenz" und "Winkel" ): ) showLF
() showLF

(/STOP) show T1 ({ showpage systemdict begin quit } def ) show
      8.0 cm Tab (% Setzen von tempor\204ren Haltepunkten) showLF
(/cm)  show T1 ({ 72 2.54 div mul } def ) show
      T2 (% Umrechnung in cm, nicht ver\204ndern) showLF
(/mm)  show T1 ({ 72 25.4 div mul } def ) show
      T2 (% Umrechnung in mm, nicht ver\204ndern) showLF
(/Zoll) show T1 ({ 72 mul } def ) show
      T2 (% Umrechnung in Zoll (inch), nicht ver\204ndern) showLF
(/LR)  show T1 (3 cm def) show
      T2 (% Linker Rand, ver\204nderbar, auch 0 cm ) showLF
(/Tab) show T1 ({ LR add currentpoint exch pop moveto } def ) show
      (% Tabulator ab LR, Beispiel: 1.0 cm Tab) showLF
(/Schriftgroesse 10 def) show
      T2 (% Schriftgr\224\341e, ver\204nderbar) showLF
(/relZeilenvorschub 1.20 def) show
      T2 (% relZeilenvorschub 1.20 * Schriftgr\224\341e, ) show
      (ver\204nderbar) showLF
(/showLF) show T1 ({ show LR currentpoint exch pop ) show
      (Schriftgroesse relZeilenvorschub mul sub moveto } def ) showLF
T1 (% Zeigt Text mit anschlie\341endem LineFeed LF. ) show
      (Leerzeile wie folgt: ) show ( ) showLF) showLF
(Druckername ) show T2 (% Bezeichnung des Ausgabeger\204tes, ) show
      (max. 31 Zeichen) showLF

/T2 { 7.5 cm Tab } def
(/DPI)  show T1 (72 0 dtransform dup mul exch dup mul ) show
      (add sqrt def % Ger\204teaupl\224sung in dpi) showLF
(/Pixel) show T1 (10000 DPI div 2.54 div def) show
      T2 (% Pixelgr\224\341e in Mymeter ) showLF
(/Pi)  show T1 (3.141592654 def) show T2 (% Kreiszahl Pi) showLF
(/Tan)  show T1 ({ dup sin exch cos div } def ) show
      T2 (% Tanges, Argument im Gradma\341 ) showLF
(/ArcSin) show T1 ({... mit Fallunterscheidungen ...} def ) show
T2 (% Arcussinus, Ergebnis im Gradma\341, -90\312 ... +90\312) showLF
(/ArcCos) show T1 ({... mit Fallunterscheidungen ...} def ) show
T2 (% Arcuscossinus, Ergebnis im Gradma\341, 0\312 ... +180\312) showLF
(/FHM-Logo) show ( {.....} def ) show
T2 (% Syntax und Beispiel siehe unten) showLF
%-----
(/Pfeil ) show T1 ({ .....} def ) show
T2 (% Syntax und Beispiel siehe unten) showLF
() showLF
(Beispiele f\201r Schriftenumkodierung:) showLF
( /Helvetica /Helvetica-Deutsch DeutscheSonderzeichen Umcodierung) showLF
( /Times-Roman /Times-Roman-Deutsch DeutscheSonderzeichen ) show
(Umcodierung) showLF
(Voreingestellt ist die Schrift "/Helvetica-Deutsch" mit 10 Punkt ) showLF
() showLF

/T1 { 2.7 cm Tab } def
(Kreiszahl Pi = ) show T1 Pi =string cvs showLF
(Tan von 30\312: ) show T1 30 Tan =string cvs showLF
(ArcSin von 0.5: ) show T1 0.5 ArcSin =string cvs show (\312) showLF
(ArcCos von 0.5: ) show T1 0.5 ArcCos =string cvs show (\312) showLF
() showLF

```



```

(Syntax f\201r Aufruf FHM-Logo: "LinkerRand UntererRand ) show
(Breite FHM-Logo") showLF
(Beispiel: 16 cm 5.5 cm 2.5 cm FHM-Logo) showLF
() showLF

/xAltPfeil currentpoint pop      def
/yAltPfeil currentpoint exch pop def

(Syntax f\201r Aufruf Pfeil:) showLF
("xPfeil yPfeil Pfeillaenge Pfeilwinkel Pfeilrichtung ) show
(LaengeZusatzstrich Strichdicke boolean Pfeil") showLF
(Die Koodinaten der Pfeilspitze sind "xPfeil" und "yPfeil". ) show
(Pfeilrichtung 0, 90, 180, 270 Grad oder) showLF
(beliebiger Winkel entgegen Uhrzeigersinn. Bei boolean ) show
("true" wird Pfeil gef\201llt, bei "false" nicht. ) showLF
() showLF

/xAlt currentpoint pop      def
/yAlt currentpoint exch pop def

16.0 cm 5.5 cm 2.5 cm FHM-Logo

/BreiteLogo 0.4 cm def
8 cm 6.1 cm moveto
7 { currentpoint BreiteLogo FHM-Logo
  currentpoint exch BreiteLogo add 0.3 cm add exch moveto
  /BreiteLogo BreiteLogo 0.1 cm add def
} repeat

() showLF

/xPfeil xAltPfeil 12.3 cm add def
/yPfeil yAltPfeil 0.5 cm add def

xPfeil yPfeil 1 cm 15 0 0.5 cm 0.1 mm true Pfeil
% | | | | | | | | | |
% | | | | | | | | | | +--- true/false = mit/ohne Fuellung
% | | | | | | | | | | +--- Strichdicke von Pfeil/Pfeilstrich
% | | | | | | | | | | +--- Laenge des Zusatz-Pfeilstrichs, auch 0
% | | | | | | | | | | +--- Pfeilrichtung in Grad, entgegen Uhrzeiger
% | | | | | | | | | | +--- Oeffnungswinkel des Pfeils
% | | | | | | | | | | +--- Laenge des Pfeils
% | | | | | | | | | | +--- y-Position der Pfeilspitze
% +---- x-Position der Pfeilspitze

xPfeil yPfeil 0.5 cm 30 90 0.5 cm 0.1 mm false Pfeil
xPfeil yPfeil 0.4 cm 15 180 1.0 cm 0.1 mm true Pfeil
xPfeil yPfeil 0.5 cm 60 270 0.0 cm 0.3 mm false Pfeil
xPfeil yPfeil 0.7 cm 15 45 0.3 cm 0.1 mm true Pfeil

xAlt yAlt moveto
(02.05.1993, Dr. K. Haller, Fachhochschule M\201nchen, ) show
(Studiengang Druckereitechnik) showLF
currentpoint 0.8 cm add moveto
16 cm 0 cm rlineto stroke
showpage quit %-----
} if
%----- EOF -----

```

PostScript-Modul "Modul-0.PS", Umlaute und Systemparameter. Als Vorspann (Include-Datei) v o r andere PS-Programme. Test-Ausgabe des Moduls alleine nur, wenn "Demo_Modul-0" auf "true".
Beispiel: Es soll die Datei "Test.PS" mit dem Modul "Modul-0.PS" auf einem MS-DOS-System ausgegeben werden: copy Modul-0.PS + Test.PS prn

Deutsche und andere Sonderzeichen:
 ä = \204 ö = \224 ü = \201 ß = \341 • = \267 (im Standard-Zeichensatz)
 Å = \216 Ö = \231 Ü = \232 ' = \312 (im Standard-Zeichensatz)

----- Systemparameter (21.09.92) -----

Verwendbarkeit der Bezeichner (global/lokal) beachten!

Global: Drucker "Druckername": HP LaserJet 4000 Series
 Lokal: Produkt "Produktname": HP LaserJet 4000 Series
 Lokal: PostScript-Version "PS-Version": 2014.108
 Lokal: PostScript-Revision "PS-Revision": 0

Die Geräteauflösung "DPI": 600.0 dpi = 1524.0 dots per cm
 Pixelgröße "Pixel": 6.56168 µm

Default-Werte von 'setscreen':
 Rasterfrequenz "Frequenz": 60.0 lpi = 23.622 L/cm
 Rasterwinkel "Winkel": 0.0°

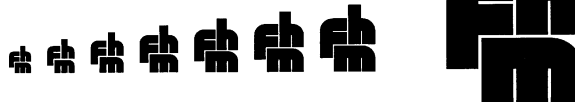
Im Hauptprogramm kann auf folgende Prozedur- und Variablendeklarationen von "Modul-0.PS" zugegriffen werden (zudem auch auf "Frequenz" und "Winkel"):

```
/STOP { showpage systemdict begin quit } def % Setzen von temporären Haltepunkten
/cm { 72 2.54 div mul } def % Umrechnung in cm, nicht verändern
/mm { 72 25.4 div mul } def % Umrechnung in mm, nicht verändern
/Zoll { 72 mul } def % Umrechnung in Zoll (inch), nicht verändern
/LR 3 cm def % Linker Rand, veränderbar, auch 0 cm
/Tab { LR add currentpoint exch pop moveto } def % Tabulator ab LR, Beispiel: 1.0 cm Tab
/Schriftgroesse 10 def % Schriftgröße, veränderbar
/relZeilenvorschub 1.20 def % relZeilenvorschub 1.20 * Schriftgröße, veränderbar
/showLF { show LR currentpoint exch pop moveto } def % Zeigt Text mit anschließendem LineFeed LF. Leerzeile wie folgt: () showLF
Druckername % Bezeichnung des Ausgabegerätes, max. 31 Zeichen
/DPI 72 0 dtransform dup mul exch dup mul add sqrt def % Geräteauflösung in dpi
/Pixel 10000 DPI div 2.54 div def % Pixelgröße in Mymeter
/Pi 3.141592654 def % Kreiszahl Pi
/Tan { dup sin exch cos div } def % Tanges, Argument im Gradmaß
/ArcSin { ... mit Fallunterscheidungen ... } def % Arcussinus, Ergebnis im Gradmaß, -90° ... +90°
/ArcCos { ... mit Fallunterscheidungen ... } def % Arcuscossinus, Ergebnis im Gradmaß, 0° ... +180°
/FHM-Logo { ..... } def % Syntax und Beispiel siehe unten
/Pfeil { ..... } def % Syntax und Beispiel siehe unten
```

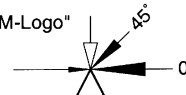
Beispiele für Schriftenumkodierung:

/Helvetica /Helvetica-Deutsch DeutscheSonderzeichen Umcodierung
 /Times-Roman /Times-Roman-Deutsch DeutscheSonderzeichen Umcodierung
 Voreingestellt ist die Schrift "/Helvetica-Deutsch" mit 10 Punkt

Kreiszahl Pi = 3.14159
 Tan von 30°: 0.57735
 ArcSin von 0.5: 30.0°
 ArcCos von 0.5: 60.0°



Syntax für Aufruf FHM-Logo: "LinkerRand UntererRand Breite FHM-Logo"
 Beispiel: 16 cm 5.5 cm 2.5 cm FHM-Logo



Syntax für Aufruf Pfeil:

"xPfeil yPfeil Pfeillaenge Pfeilwinkel Pfeilrichtung LaengeZusatzstrich Strichdicke boolean Pfeil"
 Die Koordinaten der Pfeilspitze sind "xPfeil" und "yPfeil". Pfeilrichtung 0, 90, 180, 270 Grad oder beliebiger Winkel entgegen Uhrzeigersinn. Bei boolean "true" wird Pfeil gefüllt, bei "false" nicht.

02.05.1993, Dr. K. Haller, Fachhochschule München, Studiengang Druckereitechnik