GRAF® computer

EPROM GOSI

für den

NDR Computer (Z80)

Graf Elektronik Systeme GmbH
8960 Kempten · Tel.: 08 31 - 62 11

Vorwort

GOSI ist eine leicht zu erlernende graphische Sprache, die ähnlich wie LOGO Sprachelemente für einen Schreibzeiger besitzt, den man mit Befehlen bewegen kann.
GOSI besitzt Befehlselemente, die über die Befehle z.B. von BASIC weit hinausgehen, so ist es möglich der Sprache neue Wörter beizubringen und sie von da an zu verwenden. Dadurch wird die Bedienung sehr erleichtert und Programme werden anschaulich und lesbar.
In GOSI ist nicht nur eine Einsteigersprache, die leicht zu erlernen ist, sondern man kann mit ihr auch Steuerungsaufgaben verwirklichen. So kann man sich z.B. Wörter für Ventilan, Ventilaus etc. selbst definieren und dann damit arbeiten. Die dazu nötigen Grund-Befehle, die man braucht um mit der Aussenwelt zu arbeiten, sind in GOSI natürlich vorhanden.
Ferner gibt es Befehle um einen Drucker anzusteuern und man kann seine Programme und Daten auf einem Kassettenrekorder speichern und laden.

München, den 28.3.1984

Rolf-Dieter Klein

```
GOSI - Graphisch orientierte Sprache I
(C) Muenchen 1984 Rolf-Dieter Klein Vers 1.1
"Ein Hauch von LOGO ..."

vw 40 re 90 vw 40 re 90 vw 40 re 90 vw 40 re 90
```

Befehlsliste

Im folgenden werden alle Befehle von GOSI beschrieben.
Manche Befehle besitzen Abkürzungen, die dann rechts neben dem ausgeschriebenen Befehl stehen. Umkehrt gibt es aber auch Befehle, die nur als Abkürzung vorhanden sind, da die ausgeschrieben Namen sehr lang sind.
GOSI akzeptiert Groß- und Kleinschreibung.
Aus drucktechnischen Gründen sind Umlaute an manchen Stellen, an denen Sonderzeichen stehen sollten, z.B. eckige Klammer auf ist Ä.

Eingaben

GOSI ist Bildschirmorientiert. Mit den Cursortasten Pfeil links (CTRL H) Pfeil rechts (CTRL I) Pfeil nach oben (CTRL K) Pfeil nach unten (LF oder CTRL J) kann man eine beliebige Zeile anfahren.
Mit DEL kann man Zeichen löschen, die links neben dem Cursor stehen, mit CTRL-G Zeichen die rechts davon sind.
Mit CTRL-V kann man ein Zeichen einfügen.
Nach Eingabe von CR (Wagenrücklauf) wird alles in der Cursorzeile übernommen und ausgewertet. Der Cursor rückt dabei auf die nächste Zeile vor.
Will man eine Eingabe wiederholen, so positioniert man den Cursor einfach auf die alte Zeile und betätig CR. Davor kann man natürlich Korrekturen an dieser Zeile vornehmen.

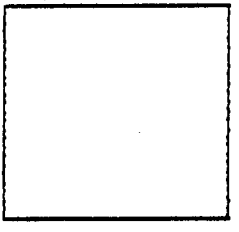
Arithmetik

GOSI besitzt eine Integer-Arithmetik, daß heißt Rechenoperationen werden nur mit 16-Bit dargestellt, also einem Zahlenbereich von maximal +32767 und minimal -32768.
Dadurch ist aber die Arithmetik aber sehr schnell.

Rechenoperationen können aber so geschrieben werden, wie man es von der Mathematik her kennt. Es dürfen jedoch zwischen den Operationen keine Leerzeichen stehen. Dies unterscheidet unser GOSI z.B. von LOGO. Damit ist es aber für den Benutzer einfacher Parameter an Unterprogramme richtig zu gruppieren.
Folgende Operationen sind verfügbar:

+	Addition, z.B. 3+4
-	Subtraktion, z.B. :alpha-5 oder Vorzeichen, z.B. -745

*		Multiplikation, z.B. 45*:betha
/		Ganzzahlige Division, also z.B. 10/3 ergibt 3. Will man z.B. eine Variable mit einer gebrochenen Zahl multiplizieren, z.B. :alpha*1.41, so schreibt man :alpha*141/100 und erhält eine Näherung
\	(Backslash)	Modulo oder auch REST genannt. Damit ergibt sich der Rest bei einer Division, also 20\6 ergibt 2
( )		Mit Klammern können Operationen verschachtelt werden und Uneindeutigkeiten bei der Reihenfolge aufglöst werden. z.B. :mem 1024*3 ist was anderes als (:mem 1024)*3



```
lerne quadrat :seite
wh 4[vw :seite re 90]
ende

Ok gelernt, Platz zum lernen: 2263 Bytes und fuer Namen: 637 Bytes.
quadrat 120
```

logische Verknüpfungen

!	Oder-Verknüpfung, z.B. :alpha!45 oder 3!5 ergibt 7
& (Ampersand)	Und-Verknüpfung, z.B. :betha&:alpha 3&5 ergibt 1
~ (Welle)	Nicht-Verknüpfung, z.B. ß:alpha ~0 ergibt -1, da bei -1 alle Bits auf 1 gesetzt sind.

Vergleiche

<	Kleiner, 2<3 ist Wahr
>	Größer, -1>-4 ist Wahr
<=	Kleiner Gleich
>=	Grö er Gleich
=	Gleich
<>	Ungleich

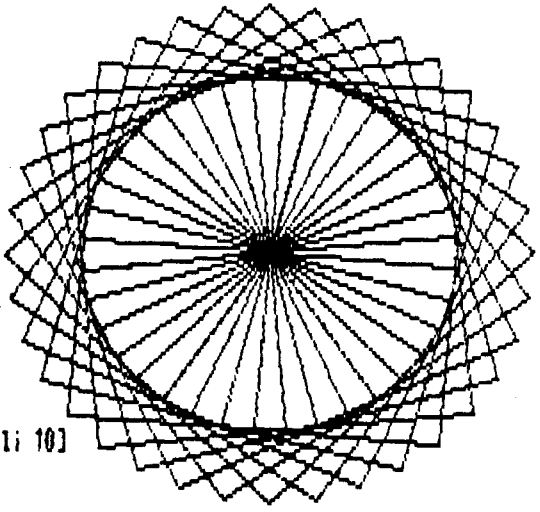
Ist der Vergleich Wahr, so wird der Wert -1 geliefert, sonst der Wert 0. Dadurch sind auch logische Verknüpfungen möglich, z.B. (:alpha<6)!(:betha>=:gamma)!~(:a=:b)
Im Klartext:
Alpha Kleiner 6 oder betha größer gleich Gamma oder Nicht Klammerauf a gleich b Klammerzu.

Funktionen

"	"A liefert den ASCII-Wert des Zeichens A, also 65.
Taste	liefert ein Zeichen von der Eingabe-tastatur im ASCII-Code, Beispiel: WENN :TASTE=12 [VW 10] Wenn die taste CR (Carriage return, Code 12) eingegeben wird, so wird der Befehl VW 10 ausgeführt
Taste?	Damit kann man prüfen ob eine Taste betätigt wurde, es ergibt sich der Wert wahr, wenn dies der Fall ist. Mit :TASTE läßt sich dann der eingegebene Wert abfragen. Beispiel:

```
SOLANGE ~:taste? [VW 30 RE 90]
Es wird solange ein Quadrat gezeichnet
bis man auf der Tastatur ein Zeichen
eingibt.
```

```
lerne quadrat :seite
wh 4[vw :seite re 90]
ende

Ok gelernt, Platz zum lernen: 2263 Bytes und fuer Namen: 637 Bytes.
```



```
wh 36 [quadrat 100 li 10]
```

Zahl

Der Computer wartet dann auf die Eingabe einer Zahl von der Tastatur, alle Cursortasten, ausser denen, die die Zeile verlassen sind gültig, bei CR (carriage return, Wagenrücklauf) wird die aktuelle Cursorzeile als Eingabe verwendet.
Dabei darf vor dem Cursorstart auch schon ein Text stehen, Programmbeispiel:

```
LERNE LIES
DRUCKE "HALLO
SETZE "A :ZAHL
DZ Ä Ü
DZ :A
ENDE
```

Mit LIES wird eine Zahl eingelesen und anschließende ausgegeben, dabei kann bei der Eingabe jeder beliebige Arithmetische Ausdruck eingeben werden, also z.B. auch 234+8*(9-1) und das Resultat wird ausgeben.

ZZ

Zufallszahl. Als Parameter wird noch ein Wert eingeben der die Grenze der Zahl darstellt. Max kann 256 als Grenze gegeben werden.

DZ :ZZ 10 druckt eine Zufallszahl zwischen 0 und 9, die Zahl 10 erscheint nicht.

XKO

:XKO liefert die aktuelle X-Koordinate des Zeigers. Er beträgt im sichtbaren Bereich 0..511

YKO

:YKO liefert die aktuelle Y-Koordinate des Zeigers. Er beträgt im sichtbaren Bereich 0..511 (im Gegensatz zum realen Bereich von 0..255, z.B. beim LINIE-Befehl).

KURS

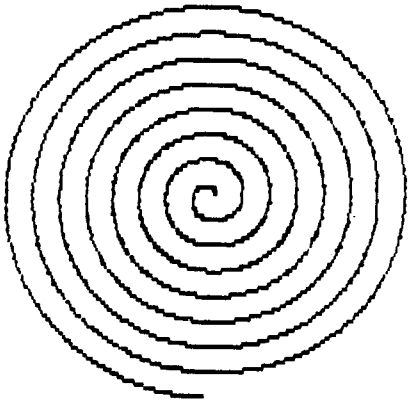
:KURS liefert den aktuellen Winkel des Zeigers. 0 Grad ist er, wenn er nach rechts zeigt, dann mathematisch positiv, nach oben z.B. 90 Grad. (ACHTUNG, dies weicht von anderen LOGO-Implementierungen u.U. ab).

SIN

:SIN 10 liefert einen Wert 256*sin(10), dabei ist dieser Wert ganzzahlig. Die Winkelangabe erfolgt in Grad.

COS :COS 90 liefert den Wert 256*cos(90).

```
zeige rarc
rarc :n
wh 90[schritt :n li 1]
ende
rarc 40
```



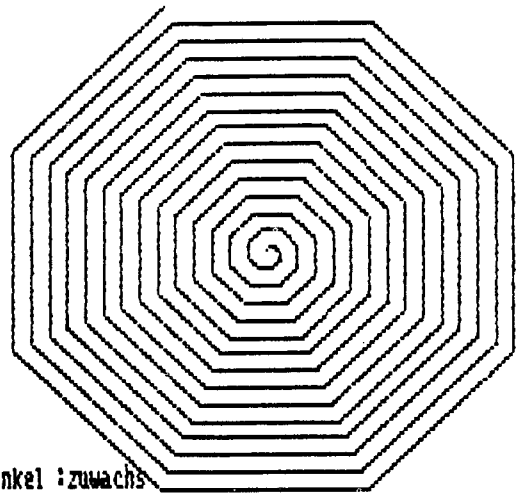
```
lerne spiro :n
rarc :n
spiro :n+1
ende
Ok gelernt, Platz zum lernen: 2192 Bytes und fuer Namen: 614 Bytes.
spiro 1
```

Port

Damit kann man einen IO-Port des Z80 einlesen. Dazu gibt man noch eine Adresse an. Beispiel:
DZ :PORT 48
Der Inhalt des Portes 30h oder 48 dezimal, wird ausgegeben.
Will man weiter mit dem Wert rechnen, so muß man auf Klammerung achten,
:PORT 48*2 ließt den Inhalt des Ports 96 ein,
(:PORT 48)*2 ließt den Inhalt des Port 48 ein und multipliziert das Ergebnis mit 2.

Mem

Speicherinhalt einlesen. Damit kann man den Inhalt jeder Speicherzelle bekommen.
Beispiel: DZ :MEM 5 ,gibt den Inhalt der Speicherzelle 5 aus.



```
zeige vspirale
vspirale :seite :winkel :zuwachs
vw :seite re :winkel
vspirale (:seite+:zuwachs) :winkel :zuwachs
ende
vspirale 1 45 1
```

GOSI-BEFEHLE
---

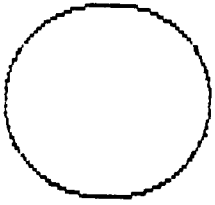
hier nun die Befehlsliste.
Nochmals der Hinweis, das eckige Klammern u.U. als Umlaute gedruckt sind, jedoch später auf dem Bildschirm als eckige Klammern erscheinen.

[ist eckige Klammer auf
] ist eckige Klammer zu
~ ist das Welle-Zeichen

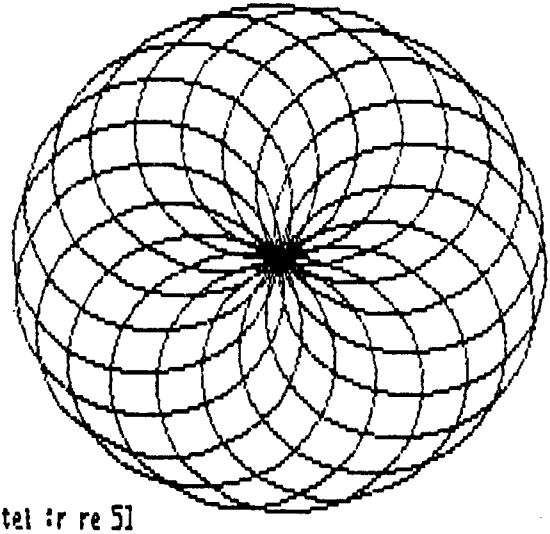
1. Befehle für den Bidlschirm

Befehl	Abkürzung	Erklärung mit Beispielen
Vorwaerts	VW	VW 10 schreitet 10 Vorwärts, also in die Richtung in die der Zeiger schaut. Ist die Zahl negativ, z.B. VW -50 so wird rückwärts geschritten
Rueckwaerts	RW	VW :alpha schreitet je nach Inhalt von alpha, rueckwaerts. Ist der Inhalt negativ, so wird vorwärts geschritten.
Links	LI	LI 90, dreht den Zeiger um 90 Grad gegen den Uhrzeigersinn, also math. positiv.
Rechts	RE	RE 45, dreht den Zeiger um 45 Grad im Uhrzeigersinn.
Schr16tel		wie Vorwärts, jedoch wird um 1/16 Punkt geschritten, damit lassen sich z.B. auch einfach kleine Kreise erzeugen, z.B. WH 360 [SCHR16TEL 7 RE 1] Der Befehl ist praktisch, da ja nur Integer-Zahlen vorhanden sind.
Stiftab	SA	Der Zeiger schreib von nun an sichtbare Linien.
Stifthoch	SH	Der Zeiger bewegt sich nur zu den Endpunkten, ohne eine Linie zu hinterlassen.
	VI	Der Zeiger ist nicht mehr sichtbar und das Bild wird nicht mehr hin und hergeschaltet.

ZI Der Zeiger wird wieder sichtbar.



```
wh 360 [vw 1 li 1]
```



```
zeige drehkreise
drehkreise :r
wh 36 [re 5 schriftet :r re 5]
rechts 20
drehkreise :r
ende
drehkreise 200
```

Mitte		Der Zeiger bewegt sich zur Bildmitte, dabei wird, wenn der Zeiger schreibend ist, eine Linie gezeichnet. Will man das nicht, so schreib man, z.B.: SH MITTE SA
Aufx		AUFX 10, bewegt den Zeiger entlang der X-Achse bis die Koordinate X=10 erreicht ist, dabei wird ggf. eine Linie gezeichnet. (0..511 ist der sichtbare Bereich).
Aufy		AUFY 300, bewegt den Zeiger entlang der Y-Achse zur neuen Position mit Y-300. (Bereich 0..511)
Aufxy		AUFX 100 200 bewegt den Zeiger zur Position x=100 und y=200 (ggf. schreiben).
Aufkurs	AK	AUFKURS 90 setzt den Zeiger in Schreitrichtung nach oben. Damit kann der Winkel eingestellt werden (passend zu :KURS) 0 Grad ist Blinkrichtung nach rechts.
Loeschebild	LB	Loescht den Bildschirm ohne die Position des Zeigers zu verändern
Loescheschirm	LS	Loescht des Bildschirm und setzt den Zeiger in die Bildmitte, dabei ist dieser aber nach diesem Löschvorgang solange nicht sichtbar bis ein entsprechender Befehl, wie VW 20 etc. ausgeführt wird.

```
turm :seite
wenn :seite<3 [rueckkehr]
wh 4[vw :seite re 90]
vorwaerts :seite
turm :seite#6/10
ende
turm 50
```

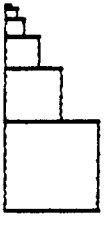
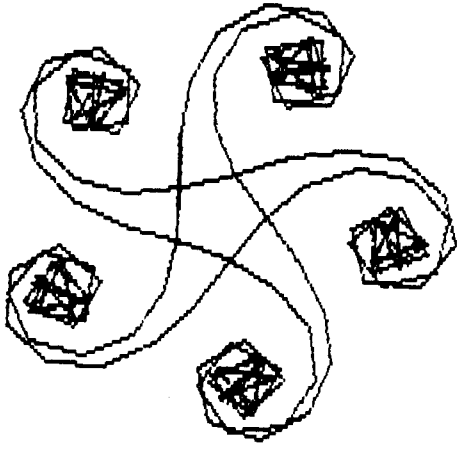


Bild	Der Bildschirm wird gesetzt, der Zeiger in die Mitte zurück und er wird in jedem Fall sichtbar, auch wenn er vorher nicht sichbar war.
Clrinv	Löscht die gerade aktuelle Schreibseite, und nur diese.
Blinker	BLINKER 0 4 setzt den Cursor in Spalte 0 Zeile 4. Als Spalte ist der Bereich 0..79 und als Zeile der Bereich 0..7 zugelassen.
Linie	Linie 0 1 100 200 zeichnet eine Linie von den Koordinaten x=0 y=1 nach x=100 y=200. Wichtig ist, das hierbei die physikalischen Koordinaten verwendet werden, also x geht von 0 bis 511 und y von 0 bis 255. Es wird auf die aktuelle Schreibseite gezeichnet.
Stift	STIFT 0 setzt den Zeiger schreibend, STIFT 1 setzt den Zeiger löschend. Beispiel: STIFT 0 VW 20 STIFT 1 RW 10 STIFT 0 zeichnet eine Linie mit der Länge 20 und löscht dann 10 Elemente wieder weg.
Seite	SEITE 3 0 setzt die Schreibseite auf 3 und die Leseseite auf 0. Wichtig ist, das der Cursor normalerweise auf Seite 0 liegt, und Texte auf 0 und 1 geschrieben werden, dabei wird automatisch auf diese Seiten zurückgeschaltet wenn eine Textausgabe erfolgt. Für den Zeiger gilt das nicht, eine eingestellte Schreibseite wird bis zum nächsten Seite-Befehl beibehaltet. Der Zeiger selbst wird jedoch immer auf Seite 1 dargestellt. Will man ihn nicht haben, so muß man ihn mit VI abstellen.
Flip	Flip 3 0, stellt die Bildaustauschrate zwischen Seite 0 und Seite 1 in 20ms-Schritten dar. Oder falls Seite 2 und 3 zuletzt

aktiv waren, zwischen denen.
Flip 1 0 ist der Normalzustand
wenn mit dem Zeiger gearbeitet
wird.
Flip 0 1, stellt die rate zwischen
allen vier Seiten dar, die dann
alle 20ms (bei 1) dargestellt
werden. damit lassen sich
interessante Bewegungseffekte
erzeugen.
Flip 0 0, läßt nur eine Seite
stehen, sie kann mit Seite 0 :x
angewählt werden.
Wird eine Textausgabe vorgenommen
so stellt sich die Lese-Seite
jedoch wieder auf 0.

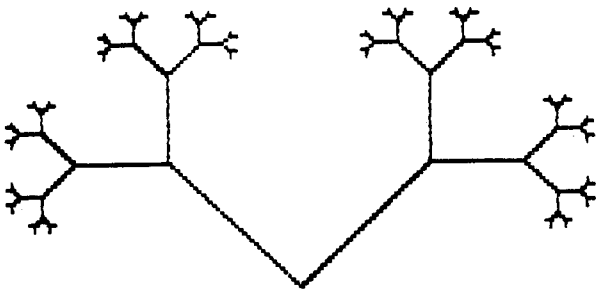
```
lerne vschritt :seite :winkel
vw :seite
rechts :winkel
ende

Ok gelernt, Platz zum lernen: 2250 Bytes und fuer Namen: 614 Bytes.
```



```
zeige knaeul
knaeul :seite :winkel
vschritt :seite :winkel
knaeul :seite (:winkel+10)
ende
knaeul 30 1
```

2. Befehle für Variablenverarbeitung

Befehl	Abkürzung	Erklärung
Setze		Damit ist eine Zuweisung an eine Variable möglich. SETZE "ALPHA 5 setzt den Wert von ALPHA auf 5 und definiert gleichzeitig die Variable. Listen können nicht an Variable zugewiesen werden. Jedoch beliebige Ausdrücke, z.B.: SETZE "ALPHA :ALPHA+5 SETZE "ALPHA :TASTE SETZE "ALPHA (:A<6)!(:B>="Z)
:		:ALPHA liefert den Wert der Variablen ALPHA, ALPHA muß entweder mit ", bei der Definition oder mit : , bei der Anwendung verwendet werden. Als Variablennamen dürfen beliebige Buchstabenkombinationen verwendet werden, die auch Zahlen enthalten können, jedoch immer mit einem Buchstaben anfangen müssen. Namen können beliebig lang sein. Groß- und Kleinschreibung wird dabei nicht unterschieden, klein gschriebene Namen sind identisch mit groß geschriebenen.



3. Ein/Ausgabe-Befehle

Befehl	Abkürzung	Erklärung
Drucke	DR	DR "ALPHA gibt den Text ALPHA auf dem Bildschirm aus. DR "ALPHA DR "BETHA gibt den Text ALPHABETHA ohne Trennung aus. Der Cursor bleibt in der Zeile stehen. DR :ALPHA gibt den Inhalt der Variablen ALPHA aus. Also nach unserem Beispiel in SETZE den Wert 5 DR 1 DR 2 gibt 12 ohne Leerzeichen aus. DR [ ] gibt ein Leerzeichen aus, DR 1 DR [ ] DR 2 gibt 1 2 mit Leerzeichen aus. DR [ALPHA [BETHA]] gibt ALPHA [BETHA] aus, dies ist ein Fragment aus der Listenverarbeitung, die hier noch übernommen wurde.
Druckezeile	DZ	wie DR, jedoch wird nachfolgend der Cursor in die nächste Zeile gesetzt.
Zeichen		ZEICHEN 65 gibt den Buchstaben A auf dem Bildschirm aus ZEICHEN 65 ZEICHEN 66 gibt die Buchstaben AB aus. ZEICHEN 26 löscht den Bildschirm, ZEICHEN 12 setzt den Cursor links an den Zeilenanfang, ZEICHEN 10 bewegt den Cursor eine Zeile nach unten. Mit ZEICHEN lassen sich alle ASCII-Zeichen direkt ausgeben, dabei sind eben auch Sonderzeichen möglich.
Port		PORT 48 0 gibt den Wert 0 an den IO-Port 48 (Sedezimal 30) aus. Damit lassen sich Ein-/Ausgaben auf Port des Z80 durchführen. Der erste Wert ist die Adresse, von 0 bis 255 und der zweite Wert

ist der Datenwert (auch von 0 bis 255). Mit diesem Befehl lassen Steuerungsaufgaben lösen.

Mem

MEM 34048 25 setzt den Inhalt der Speicherzelle 34048 (8500H) auf den Wert 25. Damit kann man einfache Felder aufbauen und verarbeiten, da der Sprache GOSI die Listen-verarbeitung fehlt ist dies eine einfache Möglichkeit.
Man muß dabei darauf achten nicht in Daten- oder Programmbereiche von GOSI zu geraten.
Bereiche:
von 8000h bis 834Fh ist der fest reservierte Bereich, dann folgt die Symboltabelle, die alle Namen im Klartext enthält.
Ab Adresse 8600h steht das Anwenderprogramm, das durch LERNE definiert wurde.
Ab 8FFFh rückwaerts liegt der Stack, in dem alle Lokalen Variable gespeichert werden.
Im Prinzip muß man seine Felder dazwischen legen. Doch dies sollte man nur dann tun, wenn man genug Erfahrung im Umgang mit GOSI gewonnen hat.

Druckan

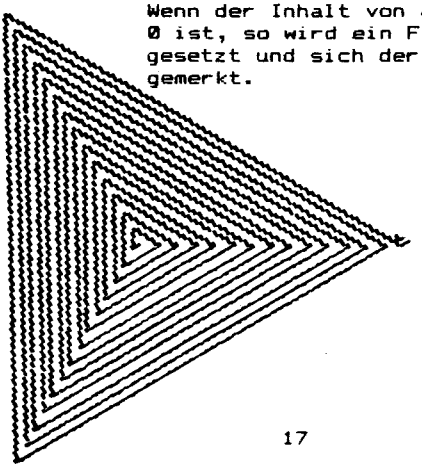
Dadurch werden alle Ausgaben einem Drucker parallel geschaltet, so kann man Programme mit ZEIGE auf dem Drucker listen.
Der Drucker wird über eine Centronix-Schnittstelle mit dem Computer verbunden. Die IO-Adresse ist 48H.

Druckaus

Der Drucker wird wieder ausge-schaltet.

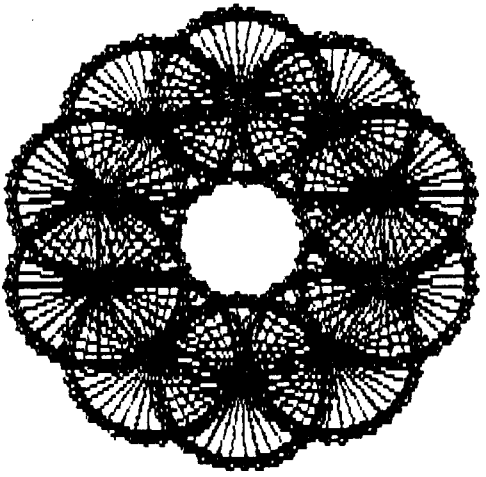
4. Bedingungen

Befehl	Abkürzung	Erklärung
Wenn		davon gibt es zwei Formen WENN bedingung [ ] WENN bedingung [ ] [ ] Beispiel: WENN :TASTE="A [VW 30] WENN :ALPHA<=:BETHA [VW 20] [RW 20] Beim ersten Fall wird der Befehl VW 30 ausgeführt, wenn die Taste A betätigt wurde, beim zweiten Fall wird VW 20 ausgeführt, wenn der Inhalt von ALPHA kleiner oder gleich dem Inhalt von BETHA war. Beispiel: WENN :A=0 [VW 20 RE 90 VW 20 RE 90] Wenn der Inhalt von a gleich 0 war, so wird die Befehlsfolge VW 20 RE 90 VW 20 RE 90 ausgeführt. In den eckigen Klammern dürfen also auch mehrere Befehle stehen.
Wiederhole	WH	WH anzahl [ ] WH 4 [VW 40 RE 90] zeichnet ein Quadrat, die Befehle, die in Klammern stehen werden vier Mal ausgeführt.
Pruefe		Pruefe bedingung PRUEFE :A=0 Wenn der Inhalt von a gleich 0 ist, so wird ein Flag gesetzt und sich der Zustand gemerkt.



VSpirale 5 120 6

Wennwahr	WW	Wenn die vorherige pruefe Anweisung wahr ergab, so werden die nachfolgenden Befehle ausgeführt, WW VW 20 RE 90 wenn wahr, dann wird VW 20 und RE 90 ausgeführt. Diesmal sind keine eckigen Klammern nötig.
Wennfalsch	WF	Die nachfolgenden Befehle werden ausgeführt, wenn die Pruefe-Anweisung falsch ergab. Dieses Flag ist immer global, das heißt es behält seinen Zustand, auch wenn zwischendurch in anderen aufgerufen Prozeduren eine Pruefe-Anweisung vorkam.
Solange	SO	SOLANGE bedingung[ ] SOLANGE ~:TASTE? [VW 50 RE 90] der Zeiger zeichnet immerfort ein Quadrat bis eine Taste gedrückt wird. Die in eckigen Klammern stehenden Befehle werden solange ausgeführt, wie die Bedingung erfüllt ist.



```
zeige grafik
grafik
wh 10 [ wh 36[wh 3[vw 50 re 120]re 10] vw 50 re 36]
ende
vi
```

5. Speicherverwaltung

Befehl	Abkürzung	Erklärung
Lerne		Lerne name parameter parameter ... Damit wird dem Computer eine eigene Prozedur beigebracht. Beispiel: LERNE QUADRAT :GROESSE WH 4 [VW :GROESSE RE 90] ENDE QUADRAT 50 zeichnet ein Quadrat, QUADRAT 100 zeichnet ein größeres Quadrat.
Ende		nach Eingabe von ENDE ist der Lernvorgang beendet. Da eine Prozedur aus mehreren Zeilen bestehen darf, muß der Computer wissen wann der Lernvorgang beendet sein soll. Hier noch etwas zu den Parametern. Eine Prozedur darf beliebig viele Parameter besitzten, diese Parameter sind Platzhalter, denen beim Aufruf Wert zugewiesen werden, die dann innerhalb der Prozedur verwendet werden. Diese Platzhalter sind lokal, das heißt nur innerhalb der Prozedur mit diesem Werten verbunden. Beispiel: SETZE "ALPHA 5 LERNE NEU :ALPHA DZ :ALPHA ENDE NEU 20 ergibt den Wert 20 DZ :ALPHA ergibt aber immer noch 5. Ein weiterer Aspekt sind die Rekursionen. Beispiel: LERNE REC1 :ZAHLEIN DZ :ZAHLEIN REC1 :ZAHLEIN+1 ENDE bei Aufruf von REC1 1 beginnt der Computer aufsteigend

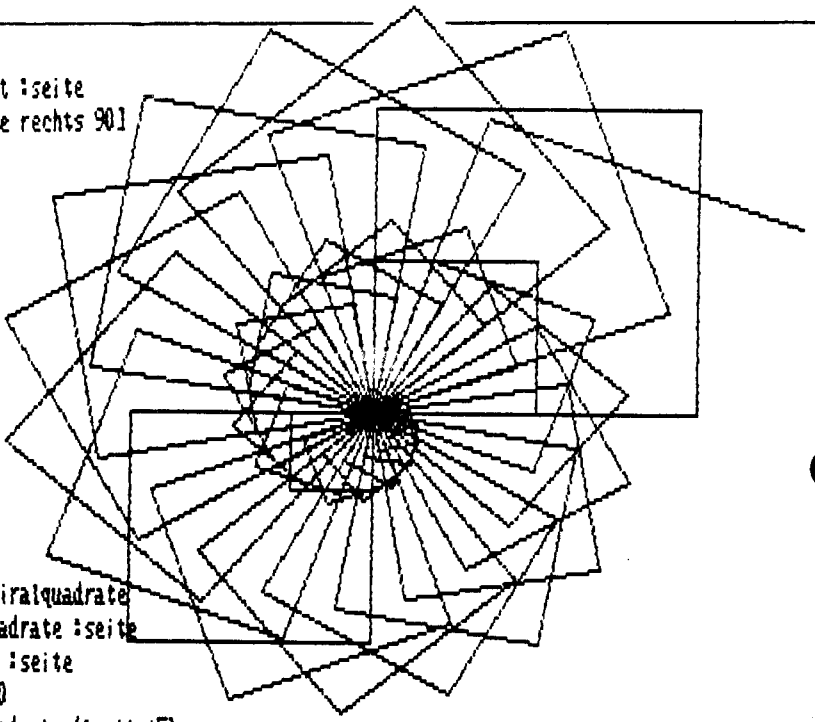
Zeige	ZEIGE gibt alle definierten Namen von Variablen und Prozeduren aus. ZEIGE ALPHA gibt den Wert der Variablen ALPHA aus, wenn es eine Variable war, sonst die Definition der Prozedur. Dies ist übrigends eine Möglichkeit Tippfehler in Prozeduren zu korrigieren, man listet sie auf den Bildschirm, und gibt dann neu LERNE ... ein. Dann fährt man mit dem Cursor auf die entsprechenden Zeilen. Dies geht aber nur, wenn die Prozeduren nicht zu lang sind, was man ohnehin jedoch vermeiden sollte um sie besser testen zu können.
Vergiss	VERGISS ALPHA löscht die Variable ALPHA aus dem Speicher, und VERGISS REC2 löscht die Procedur REC2 (je nach Definition). Dadurch gewinnt man wieder neuen Speicherplatz. VERGISS ohne Parameter löscht alles im Speicher, jedoch muß man auf die Frage ALLES ? J =JA mit J antworten.
Bewahre	BEWAHRE SAMMLUNG1 damit werden alle Prozeduren und Variable auf die Kassette gespeichert (CAS). Der Name den man angibt ist beliebig und dient später der leichten Auffindbarkeit von Programmen.
Lade	LADE lädt das nächste Programm von der Kassette in den Speicher. Es werden alle Variable und Prozeduren neu definiert. LADE gibt zuerst den gefundenen Namen aus und dann OK, oder eine Fehlermeldung.
Test	damit kann man unmittelbar nach dem Abspeichern durch Bewahre prüfen, ob alle richtig abgespeichert ist. Durch TEST wird der Speicher nicht verändert.

Call

Call adresse ruft ein Maschinenunterprogramm auf. Das Programm ist mit einem RET (C9) abzuschliessen. Diesen Befehl sollte man nicht verwenden, da er zu nicht übertragbaren Programmen führt. Er ist dennoch vorhanden um eine gewisse Vollständigkeit zu bieten und auch als Möglichkeit diesen Befehl didaktisch zu verwerten und um auf Gefahren damit hinzuweisen.

Die Ausgabe kann jederzeit mit CONTROL/S angehalten werden, mit CONTROL/Q geht es dann weiter.
Durch Eingabe von CONTROL/S und danach CONTROL/C wir der Programmablauf abgebrochen.

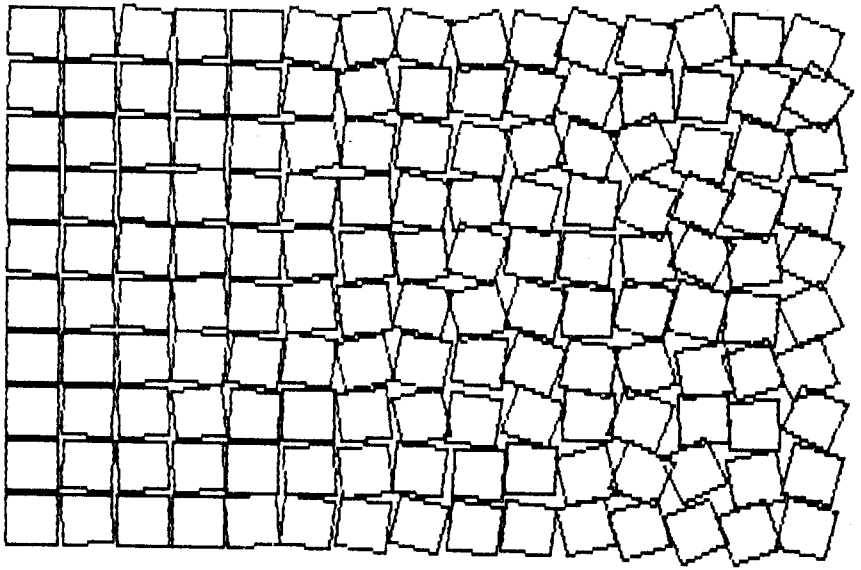
```
lerne rquadrat :seite
wh 4[vw :seite rechts 90]
ende
```



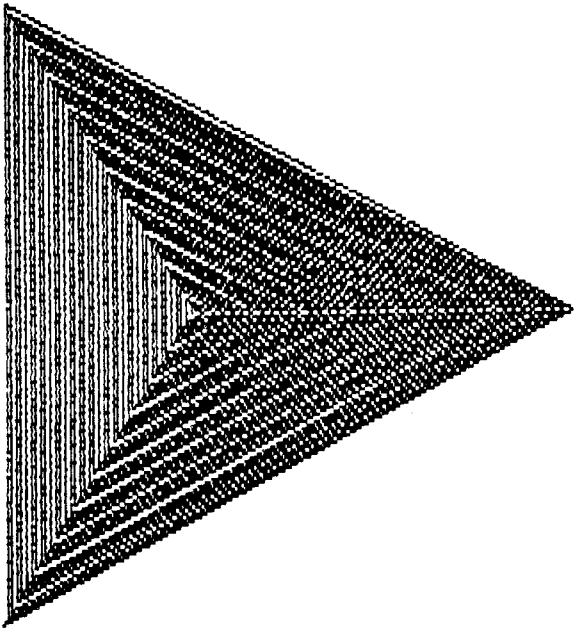
```
zeige spiralquadrate
spiralquadrate :seite
rquadrat :seite
rechts 20
spiralquadrate (:seite+5)
ende
spiralquadrate 1
```

```
************************************************************
*    G O S I   - Kurzinformation     28.3.1984 R.D.Klein    *
************************************************************

     - Graphisch orientierte Sprache I, für den Z80
     - Läuft auf dem NDR-Klein-Computer
     - benötigt :   SBCII (Z80,8K EPROM,4K RAM), GDP64, KEY
     - unterstützt :  CAS (Kassettenaufzeichnung)
                      CENT ( Druckeranschluß)
                      IOE etc. (Peripherie und Steuerung)
     - geeignet für:
          Anfänger und Fortgeschrittene im Unterricht,
          Anwendung und bei Experimenten
     - blockstrukturierte Sprache, prozedural, graphisch,
          lokale und globale Variable
------------------------------------------------------------

arbeitet mit Integer-Arithmetik (+32767 -32768):
  + - * / \ ( )          Grundrechenarten
  ! & ~                  logische Verknüpfungen
  < > <= >= = <>         Vergleiche

  Taste Taste? Port Zahl          Funktionen
  Sin Cos ZZ XKO YKO Kurs MEM     Sin ist 256*sin()

besitzt einen deutschen Befehlssatz (und Abkürzungen):
    Rueckwaerts RW Vorwaerts VW Schr16tel
    Links LI Rechts RE Stiftab SA Stifthoch SH
    VI ZI Wenn Wiederhole WH Rueckkehr RK Druckezeile
    DZ Mitte Aufx Aufkurs AK Ende Loeschebild LB
    Setze Blinker Drucke DR Zeichen Port Linie
    Seite Pruefe Wennwahr WW Wennfalsch WF
    Solange SO Druckan Druckaus Bild Loescheschirm
    LS Aufx Aufy Lerne Flip MEM Stift Test Vergiss
    Lade Zeige Bewahre Call CLRINV

Variable:
    :name          Inhalt einer Variablen (nur Integer, Zeichen)
    :MEM adr       Speicherzugriff (damit Felder möglich)
    :PORT adr      IO-Zugriff
    Beispiel:  Setze "alpha :PORT 112       Zuweisung
------------------------------------------------------------
Bezugsquellen:

                FRANZIS SOFTWARE SERVICE, München
                GES Graf Elektronik Systeme, Kempten

Kosten:         ca 60.- DM in zwei 2732 EPROMs
                incl. Unterlagen.

Literatur:
                Harald Abelson. Einführung in LOGO.
                IWT-Verlag,1983. ISBN 3-88322-023-X
```

```
zeige
QUADRAT ist gelernt
POSITION ist gelernt
SERIE ist gelernt
REST ist gelernt
I ist Variable
QUER ist Variable
HOCH ist Variable
XMAL ist Variable
YMAL ist Variable
XKOR ist Variable
YKOR ist Variable
YANF ist Variable
P ist Variable
Q ist Variable
OBEN ist Variable
UNTEN ist Variable
P1 ist Variable
Q1 ist Variable
WMAX ist Variable
WMIN ist Variable
WINKEL ist Variable
K ist Variable
R ist Variable
zeige quadrat
quadrat :p :q
setze "oben 5*:i/200
setze "unten -:oben
setze "p1 :p+5+:unten+:zz :oben-:unten
setze "q1 :q+5+:unten+:zz :oben-:unten
setze "wmax 45+45*:i/200
setze "wmax 45+45*:i/200
setze "wmin 45-45*:i/200
setze "winkel :wmin+:zz :wmax-:wmin
setze "k 7
sh position sa
wh 4 [setze "winkel :winkel+90 position]
setze "i :i+1
ende

zeige position
position
aufxy (:p1+:r*(:cos :winkel)/256) (:q1+:r*(:sin :winkel)/256)
ende

zeige rest
rest
setze "xkor :xkor+:quer
setze "ykor :yanf
ende

zeige serie
serie :quer :hoch :xmal :ymal
setze "i 0
setze "r 20
setze "xkor 20
setze "ykor 180
setze "yanf :ykor
wh :xmal [wh :ymal [quadrat :xkor :ykor setze "ykor :ykor+:hoch] rest]
ende
```



serie 30 30 15 10
vi



```
zeige vspirale
vspirale :seite :winkel
vorwaerts :seite
rechts :winkel
vspirale (:seite+3) :winkel
ende
vspirale 5 120
```


Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany. Imprimé en Allemagne.

Telefonservice
08 31- 62 11
jeden Mittwochabend
bis 20.00 Uhr

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr