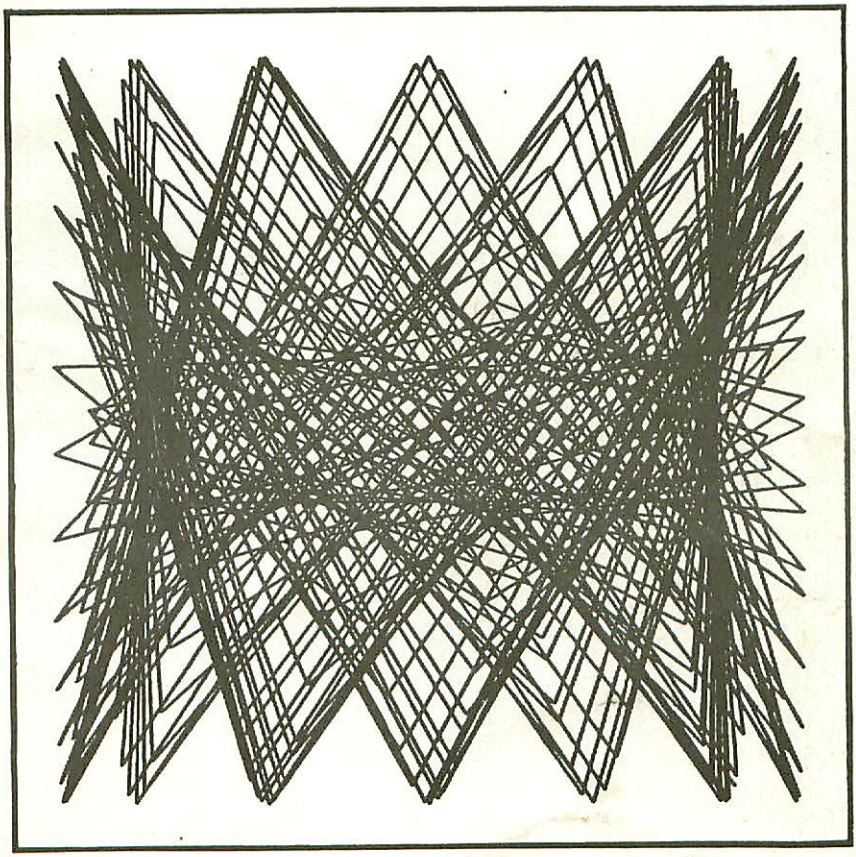
Der NDR-Klein-Computer

GOSI für Z 80



Franzis'

SoftwareService

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
```

Inbetriebnahme von GOSI

GOSI wird in zwei EPROMs geliefert, das sind zwei Bausteine in denen die Sprache programmiert ist. Diese beiden Bausteine werden auf die SBC-II-Karte gesteckt, und zwar das EPROM mit der Beschriftung 0 in den Sockel 0 und das EPROM mit der Beschriftung 1 in den Sockel 1. Das Grundprogramm wird nicht zum Betrieb von GOSI benötigt.
In den Sockeln 2 und 3 verbleiben die RAM-Bausteine, die GOSI zum arbeiten benötigt.

Arbeitsspeicher:

GOSI 0	0 bis FFF
GOSI 1	1000 bis 1FFF
Arbeitsspeicher GOSI	8000 bis ca. 8350
Programmspeicher	8350 bis 8FFF

Einführung in die Programmierung mit GOSI

Nach dem Einschalten der Versorgungsspannung meldet sich GOSI wie im Bild 1 gezeigt.
Dabei flimmert ein helles Feld links darunter. Dieses Feld nennen die Fachleute CURSOR, oder Blinker.
Der CURSOR sagt einem wo man gerade schreiben kann. Wenn man nun eine Taste drückt, zum Beispiel A, so erscheint an dieser Stelle ein A und der CURSOR rückt um eins nach rechts.
Dem Computer muß man stets sagen, wann er den eingegebenen Text verarbeiten soll. Dazu dient die Taste mit der Aufschrift CR. Bei manchen Tastaturen steht auch RETURN oder einfach ein gewinkelter Pfeil darauf. Die Abkürzung CR steht für Carriage Return zu deutsch Wagenrücklauf. Die Bezeichnung erinnert an eine elektrische Schreibmaschine, bei der durch diese Taste der Wagen zurückläuft und eine neue Zeile begonnen werden kann.
Wir geben einmal CR an. Ist in der Zeile mit dem Cursor irgend ein Zeichen gewesen, z.B. das A, so gibt der Computer eine Fehlermeldung aus. Bild 2 zeigt das Beispiel.
?A bedeutet, daß der Computer nicht verstanden hat was er mit dem A tun soll. "Fehler." bedeutet, das wir einen Fehler

gemacht haben. Der Computer ist so freundlich, uns den Fehler mitzuteilen.

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

1

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

?A
Fehler.
```

2

Wir starten am besten noch mal von vorne und drücken die RESET-Taste an der SBC-II-Karte oder schalten die Spannung mal kurz aus.
Nun wollen wir mal einen richtigen Befehl eingeben.
Dazu tippen wir VORWAERTS 100 ein.
Wichtig ist, daß wir das Wort mit AE schreiben und nicht das Ä verwenden, denn sonst versteht uns der Computer nicht.
Noch tut sich gar nichts. Na klar, denn wir müssen noch die Taste CR drücken. Nun malt der Computer eine senkrechte Linie. Tut er das nicht, so haben wir einen Eingabefehler gemacht. Bild 3 zeigt das Bild, wie es sein soll.
Wichtig ist, das man auch den Freiraum zwischen VORWAERTS und der Zahl 100 eingibt. Dies geschieht mit der langen Taste auf der Tastatur, die meist keine Beschriftung trägt.
Der Rechner versteht die Befehle nur, wenn sie eindeutig voneinander getrennt sind.

3

```
GOSI - Graphisch orientierte Sprache I
(C) Muenchen 1984 Rolf-Dieter Klein Vers 1.1
"Ein Hauch von LOGO ..."
vorwaerts 100
```

Nun einmal ein paar weitere Befehle. Wir tippen ein RECHTS 45 und die Taste CR nicht vergessen, der Schreibzeiger zeigt nun 45 Grad nach Rechts. Im Bild sieht man ihn nicht, da er nur auf dem Bildschirm eingeblendet wird. Dann geben wir den Befehl VORWAERTS 50 und die Taste CR anschliessend und es ergibt sich ein Bild wie in Bild 4.

4

GOSI - Graphisch orientierte Sprache I
(C) Muenchen 1984 Rolf-Dieter Klein Vers 1.1
"Ein Hauch von LOGO ..."
vorwaerts 100
rechts 45
vorwaerts 50

FD=VW 20
RT=rechts 90
LT=Li
BK=RW
CS=Loesche

Nun kann man durch Kombination der Befehle VORWAERTS und RECHTS Figuren zeichnen.
Wir wollen aber einmal den Bildschirm löschen, das heißt alles was auf dem Bildschirm sichbar war ausradieren. Dazu gibt es auch einen Befehl. Wir tippen BILD ein und drücken anschließend die Taste CR. Nun wird alles gelöscht und in der Bildmitte erscheint der Zeiger für die Zeichnungen, der manchmal auch Schildkröte oder Igel genannt wird. Der Cursor ist ebenfalls sichtbar.
Nun wollen wir mal die Ziffer 1 auf den Bildschirm malen. Dazu noch folgender Hinweis. Befehle wie VORWAERTS, RECHTS kann man abkürzen. Anstelle VORWAERTS auszuschreiben kann man auch VW und anstelle von RECHTS RE schreiben.
Entsprechend zu VORWAERTS gibt es auch RUECKWAERTS oder abgekürzt RW und LINKS oder abgekürzt LI.
Im übrigen kann man auch mehr als nur einen Befehle auf eine Zeile schreiben, die Befehle müssen nur durch mindestens einen Freiraum getrennt sein.

Wir tippen also mal ein:
RE 90 VW 20 RW 10 LI 90 VW 100 LI 135 VW 20
und anschließend die Taste CR.
übrigends ist es egalt ob man Groß- oder Kleinbuchstaben beim tippen der Befehle verwendet, der Computer versteht beide Schreibweisen. Großbuchstaben erhält man wenn man zuerst die Taste SHIFT und dann den dazugehörigen Buchstaben eingibt.
Bild 5 zeigt das Ergebnis unserer Arbeit.

re 90 vw 20 rw 10 li 90 vw 100 li 135 vw 20

Wir geben wieder den Befehl BILD ein (und CR nicht vergessen) um einen neuen, gelöschten Bildschirm zu erhalten.
Nun kann man schon eine ganze Menge mit den bisherigen Befehlen anfangen, jedoch fehlt eine Kleinigkeit. Will man zum Beispiel eine unterbrochene Linie zeichnen, so geht das bisher noch nicht. Dazu gibt es ein paar neue Befehle:
STIFTHOCH bewirkt, das nachfolgende Befehle wie VORWAERTS oder RUECKWAERTS keine Spur mehr hinterlassen. STIFTAB hebt diesen Befehl wieder auf und danach kann wieder gezeichnet werden. Beide Befehle kann man auch abkürzen, SH für STIFTHOCH und SA für STIFTAB.
Bild 6 zeigt ein Beispiel.

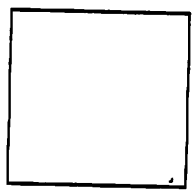
```
vw 50 stifthoch vw 50 stiftab vw 50
```

6

Will man eine Reihe von Befehlen mehrfach wiederholen, so gibt es bei Computern eine raffinierte Vorrichtung dazu, die Wiederholschleife.
Beispiel: Wir wollen ein Quadrat zeichnen. Dazu kann man folgende Befehle geben.
VW 100 RE 90 VW 100 RE 90 VW 100 RE 90 VW 100 RE 90
Dann die Taste CR eingeben und die Leertaste (für den Freiraum) nicht vergessen.
Ein Rechteck erscheint auf dem Bildschirm.
Nun aber die Wiederholschleife. Der Befehle lautet WIEDERHOLE oder abgekürzt WH.

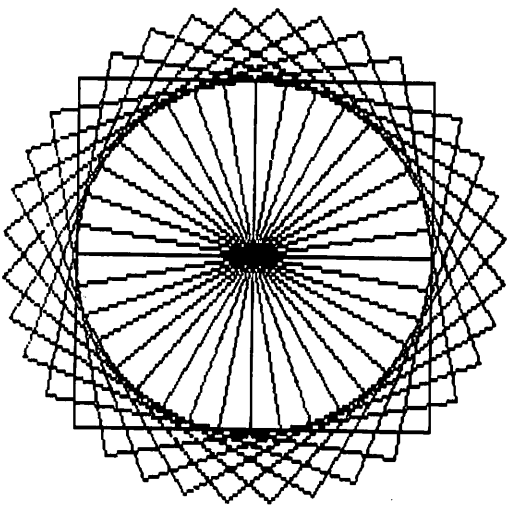
WH 4 Ä VW 100 RE 90 Ü

Anstelle der eckigen Klammern ist hier ein Ä und Ü gedruckt, genauso wie man es auf einer deutschen Tastatur eingeben muß. Auf dem Bildschirmausdruck in Bild 7 sieht man die eckigen Klammern.
Eckige Klammer auf ist die SHIFT-Taste und die Taste ä.
Eckige Klammer zu ist die SHIFT-Taste und die Taste ü.
Die Anzahl der Wiederholungen ist hier mit 4 angegeben, da die Schleife vier Mal durchlaufen werden soll, also alles was in den eckigen Klammern steht wird viermal wiederholt.



wh 4 [vw 100 re 90]

Die Wiederholschleife kann man auch ineinander schachtelt. Bild 8 zeigt ein Beispiel.
Die Befehle VW 100 RE 90 werden viermal wiederholt, man erhält also ein Quadrat. Und das Quadrat wird 36 mal ausgegeben und mit dem Befehl RE 10 jeweils und 10 Grad gedreht. 36 mal 10 Grad gibt aber 360 Grad weshalb sich eine geschlossene Figur ergibt. Man kann hier einmal selbst mit unterschiedlichen Zahlen experimentieren und wird die unterschiedlichsten Figuren erhalten.
Hier gleich ein Hinweis. Auf der Tastatur befinden sich vier Pfeiltasten. Damit kann man den Cursor an eine beliebige Stelle im Bildschirmfenster hinpositionieren. Pfeil nach oben bewegt den Cursor nach oben, Pfeil nach unten, entsprechend nach unten. Nun kann man zur alten Zeile zurückfahren und CR eingeben, die Zeile mit den Befehlen wird erneut ausgeführt. Man kann auch Veränderungen an der Zeile vornehmen, bevor man CR drückt. Mit der Taste DEL lassen sich Zeichen löschen, mit CTRL-V (Die Tasten CONTROL und V gleichzeitig drücken) wird ein Zeichen eingefügt und mit CTRL-G (CONTROL und G gleichzeitig) wird ein Zeichen rechts neben dem Cursor gelöscht. Damit lassen sich auch einfach Eingabefehler korrigieren und man muß nicht die ganze Zeile neu tippen.



8

```
wh 36 [ wh 4 [ vw 100 re 90 ] re 10 ]
```

Nun kommt etwas ganz Besonderes. Wir können dem Computer neue Wörter beibringen. Dazu gibt es den Befehl LERNE.
Wir haben schon öfters das Quadrat gezeichnet. Günstig wäre es wenn wir dem Computer das Wort QUADRAT beibringen könnten. Dazu tippen wir folgendes ein

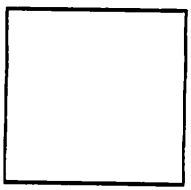
LERNE QUADRAT

und dann natürlich CR nicht vergessen. Nichts passiert und das so in Ordnung. Dann geben wir ein

WH 4 Ä VW 100 RE 90Ü

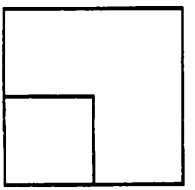
auch hier passiert nach CR-Eingabe nichts weiter.
Dann geben wir den Befehl
ENDE
ein (auch mit CR). Nun antwortet der Computer, wie in Bild 9 sichtbar. Er sagt es sind noch 2524 Bytes zum lernen frei und 679 Bytes für Namen. Diese Angabe kann unterschiedlich ausfallen, sie gibt aber immer an wieviel Platz noch im Speicher des Computers bleibt.
Nun um das Quadrat jetzt auf den Bildschirm zu bringen tippen wir QUADRAT ein (CR nicht vergessen) und es sieht dann aus wie in Bild 10.

```
lerne quadrat
wh 4 [ vw 100 re 90 ]
ende

Ok gelernt, Platz zum lernen: 2524 Bytes und fuer Namen: 679 Bytes.
```



```
lerne quadrat
wh 4 [ vw 100 re 90 ]
ende

Ok gelernt, Platz zum lernen: 2524 Bytes und fuer Namen: 679 Bytes.
quadrat
```

Nun ist es langweilig für jede Quadrat-Größe ein neues Wort zu lehren. Das geht in GOSI einfacher. Wir tippen ein:
LERNE QUADRAT :LAENGE
und die Taste CR. :LAENGE ist ein sogenannter Parameter. Später können wir eine Zahl dafür eingeben.
Mit
WH 4 Ä VW :LAENGE RE 90 Ü
und
ENDE
wird der Rest eingegeben.
Hier wurde anstelle VW 100 der Ausdruck VW :LAENGE verwendet. :LAENGE ist ein Platzhalter (Parameter) der erst später durch eine Zahl ersetzt wird.
Wir können nun QUADRAT 100 oder QUADRAT 50 etc. eingeben und jedesmal wird ein Quadrat der gewünschten Größe auf dem Bildschirm erscheinen.
Bild 11 zeigt das Beispiel.
Übrigends spielt der gewählte Name keine Rolle, er darf beliebig lang sein, aber nur Buchstaben und Zahlen enthalten, keine Sonderzeichen wie Komma etc.



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

Ok gelernt, Platz zum lernen: 2512 Bytes und fuer Namen: 679 Bytes.
quadrat 100 quadrat 50
```

Wir wollen einmal ein kleines Experiment machen. Dazu wird das folgende Programm eingetippt:

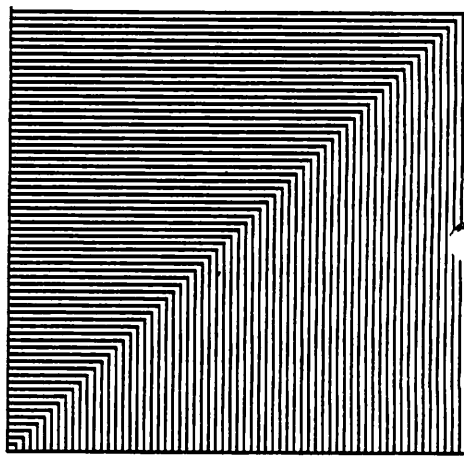
```
LERNE Q1 :N
QUADRAT :N
Q1 :N+4
ENDE
```

und CR nicht vergessen. Das QUADRAT muß vorher schon eingegeben worden sein.
Nun geben wir folgenden Befehl:

```
Q1 1
```

Es werden sehr schnell viele immer größer werdende Quadrate auf dem Bildschirm erscheinen.
Zum Anhalten können wir gleichzeitig die Tasten CONTROL und S drücken. Will man weiter machen, so drückt man CONTROL und Q. Will man das Ganze beenden, so drückt man zuerst CONTROL und S und dann CONTROL und C. Es erscheint dann die Meldung "Ende." auf dem Bildschirm.

Diese Art der Programmierung nennt man REKURSION (was man mit dem Begriff "Selbstaufrufend" übersetzen könnte), denn in der Definition von Q1 wurde Q1 wieder selbst verwendet.
Das Programm hört nie auf zu laufen, da wir nicht programmiert haben, wie lange es laufen soll; wie man eine Rekursion abbricht, erfahren wir später.



```
lerne q1 :n
quadrat :n
q1 :n+4
ende
q1 1
```

Ende.

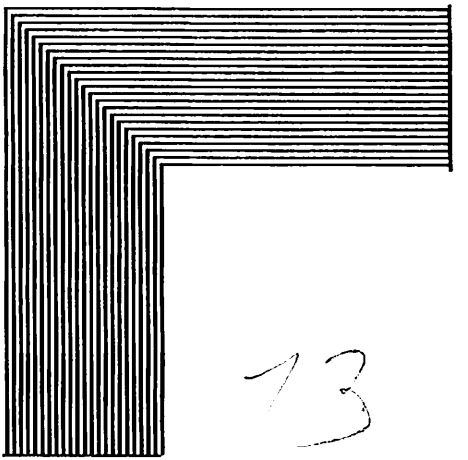
12

Will man Teile eines Bildes löschen, so geht das mit dem Befehl STIFT 1 alle Befehle wie VW, RW wirkend dann löschend. Mit STIFT 0 kommt man wieder in den Normalzustand zurück.

Bild 13 zeigt ein Beispiel, das durch CTRL-S (CONTROL und Taste S) angehalten und durch CTRL-C abgebrochen wurde.

Der Befehl MITTE setzt den Schreibzeiger wieder in die Bildmitte zurück.

Das Bild flimmert normalerweise immer etwas: Der eigentliche Bildteil und der Schreibzeiger werden quasi gleichzeitig dargestellt, indem sehr schnell zwei unterschiedliche Bildseiten abwechselnd angezeigt werden. Dies kann man durch Eingabe des Befehls
VI
unterbinden. Mit ZI kann man den Vorgang wieder rückgängig machen. VI entstammt der LOGO-Sprache un bedeutet Verstecke Igel.

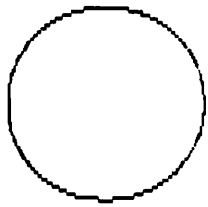


13

```
sh mitte sa stift 1 ql 1

Ende.
```

Wir wollen einmal einen Kreis auf dem Bildschirm darstellen. Da wir nur einzelne Punkte darstellen können müssen wir den Kreis annähern, einen mathematisch genaueren Kreis können wir nicht bilden.
Wir schreiten einen Schritt vorwärts und drehen dann um ein Grad. Das Ganze wird 360 mal wiederholt. Auf dem Bildschirm erscheint eine Kreisform, Bild 14 zeigt das Ergebnis.

Will man kleinere Kreise darstellen, so kann man schneller drehen, z.B. um 2 Grad und dafür nur 180 mal die Schleife durchlaufen lassen. Damit man aber alle Stufungen erreichen kann gibt es noch einen weiteren Befehl:
SCHR16TEL
damit kann man um 1/16 Bildpunkt schreiten und damit auch sehr fein positionieren.
Beispiel:
WH 360 A SCHR16TEL 2 RE 1Ü zeichnet einen sehr kleinen Kreis. Bild 15 zeigt das Beispiel.

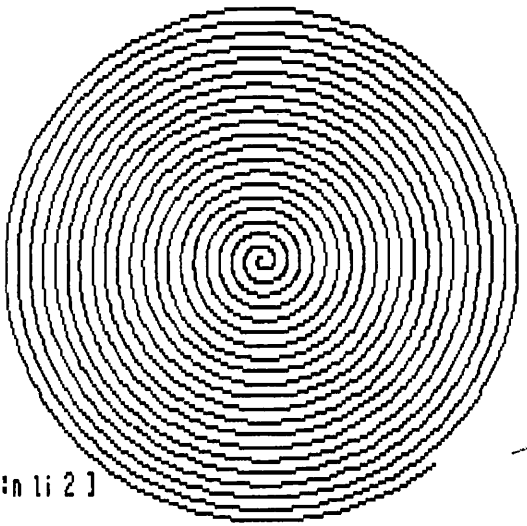


```
wh 360 [ vw 1 re 1 ]
```

```
wh 360 [ schr16tel 2 re 1 ]
```

Nun wäre es einmal interessant eine Spirale zu zeichnen, dazu zeigt Bild 16 das Beispiel.

Ähnlich interessante Ergebnisse liefert das Programm aus Bild 17, Bild 18 und Bild 19 zeigen Varianten davon. Hier wurden nun mehr als ein Parameter verwendet. Die Anzahl ist nicht begrenzt, nur durch die Anzahl der Zeichen pro Zeile.

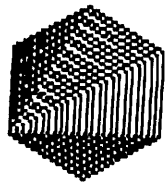


16

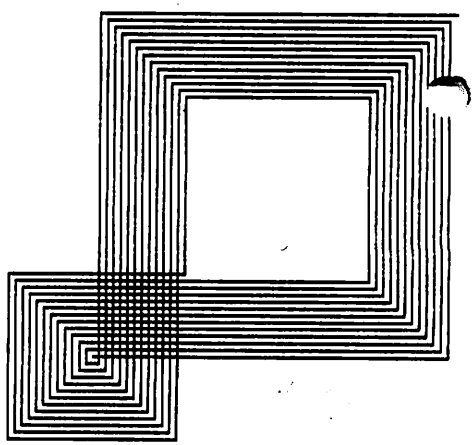
```
lerne spiro :n
wh 45 [ schr16tel :n li 2 ]
spiro :n+1
ende

Ok gelernt, Platz zum lernen: 2438 Bytes und fuer Namen: 662 Bytes.
spiro 1
```

Nun aber zum Versprochenen Abbruchkriterium. Bisher mußten wir alle Programme mit CTRL-S CTRL-C abbrechen, das soll nun anders werden.
Der Befehl WENN erlaubt es Bedingungen abzufragen und mit dem Befehl RUECKKEHR oder abgekürzt RK kann man einen automatischen Abbruch erzwingen.
Bild 20 zeigt ein Beispiel.
Mit den Operationen, wie > < <= >= = oder <> kann man Bedingungen angeben, wie in der Mathematik gebräuchlich.

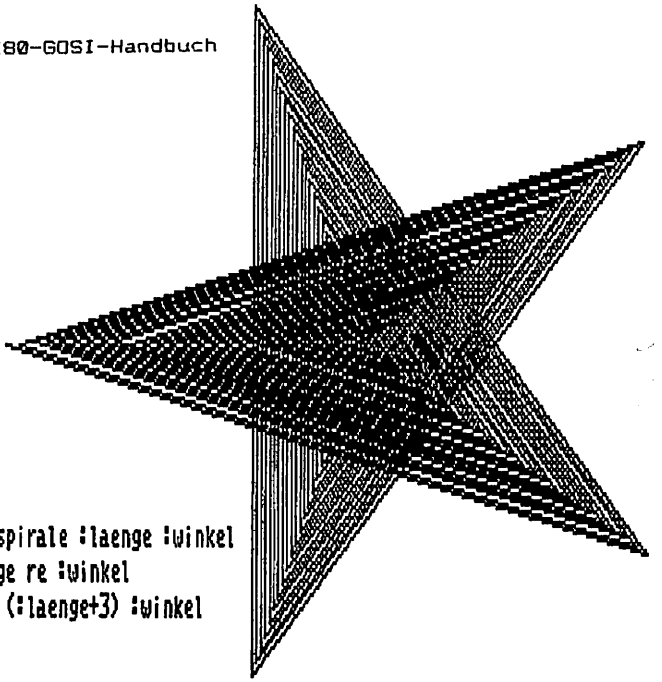
Bild 21 zeigt ein weiteres Beispiel. Es dürfen beliebig viele solcher Bedingungen in einem Programm vorkommen.
Wird eine zweite Klammer bei WENN angeben, so ist das der sogenannte SONST-Teil, der nur dann ausgeführt wird, wenn die Bedingung nicht erfüllt ist.



```
lerne mitabbruch :n
wenn :n>50 [ rueckkehr ]
wh 6 [ vw :n re 60 ]
mitabbruch :n+2
ende

mitabbruch 1
```

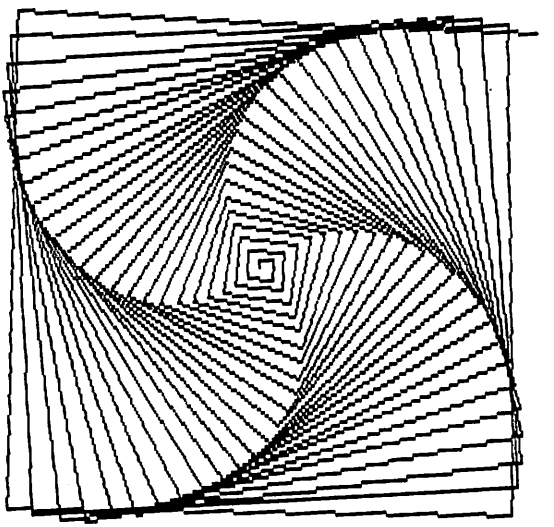


```
lerne spez :n
wenn :n>200 [ rueckkehr ]
wenn :n<100 [vw :n] [rw :n]
re 90
spez :n+2
ende
spez 1
```



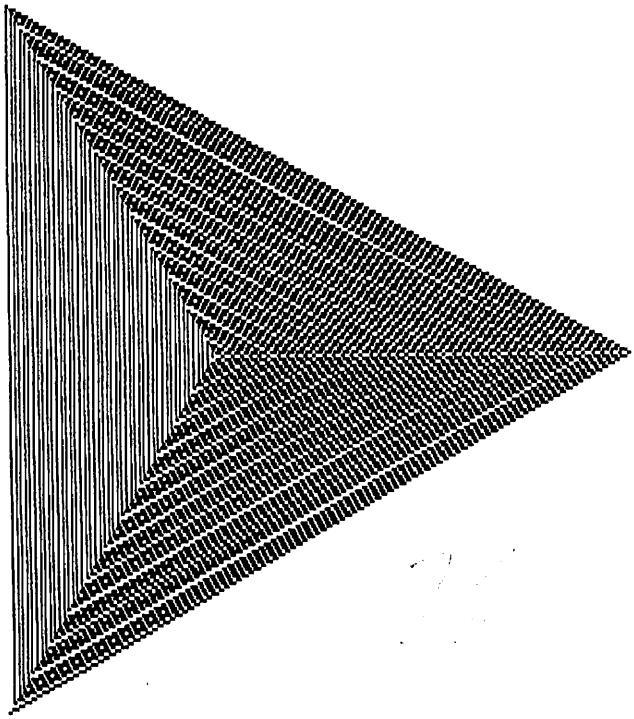
```
lerne  vspirale :laenge :winkel
vw :laenge re :winkel
vspirale (:laenge+3) :winkel
ende
```

Ok gelernt, Platz zum lernen: 2356 Bytes und fuer Namen: 633 Bytes.
vspirale 5 144



vspirale 5 91

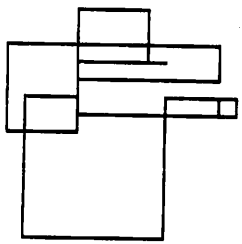
Ende.



,vspirale 5 120

Ende.

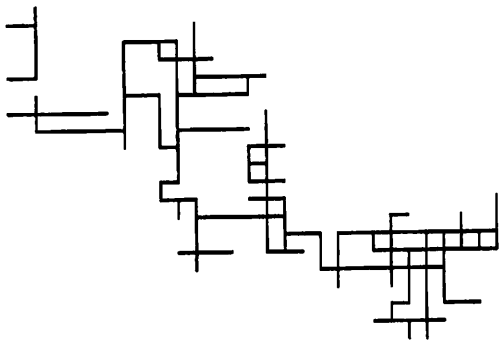
Mit WENN lassen sich ganz interessante Dinge programmieren. Bild 22 zeigt ein weiteres Beispiel. :TASTE ist ein fest eingebauter Befehl. :TASTE wartet auf die Eingabe einer Taste von der Tastatur. Man erhält dann einen Wert der dem ASCII-Code (siehe Buch Mikrocomputer selbstgebaut und programmiert) entspricht.
Mit SETZE "name wert kann man einem Namen, vor dem ein Anführungszeichen stehen muß einen Wert zuordnen.
SETZE "A 1 belegt A mit dem Wert 1
:A liefert dann den Wert für weitere Operationen. Den Doppelpunkt darf man nicht vergessen.
Das Programm aus Bild 22 wartet auf die Eingabe von Tasten. Mit SHIFT V (Die Tasten SHIFT und V gleichzeitig drücken) bewegt man den Schreibzeiger um 10 Schritte vorwärts. Mit SHIFT R um 10 Schritte rückwärts und mit SHIFT L und SHIFT E kann man ihn drehen. Damit haben wir eine Art Mini-Sprache realisiert, mit der man auch Figuren zeichnen kann.



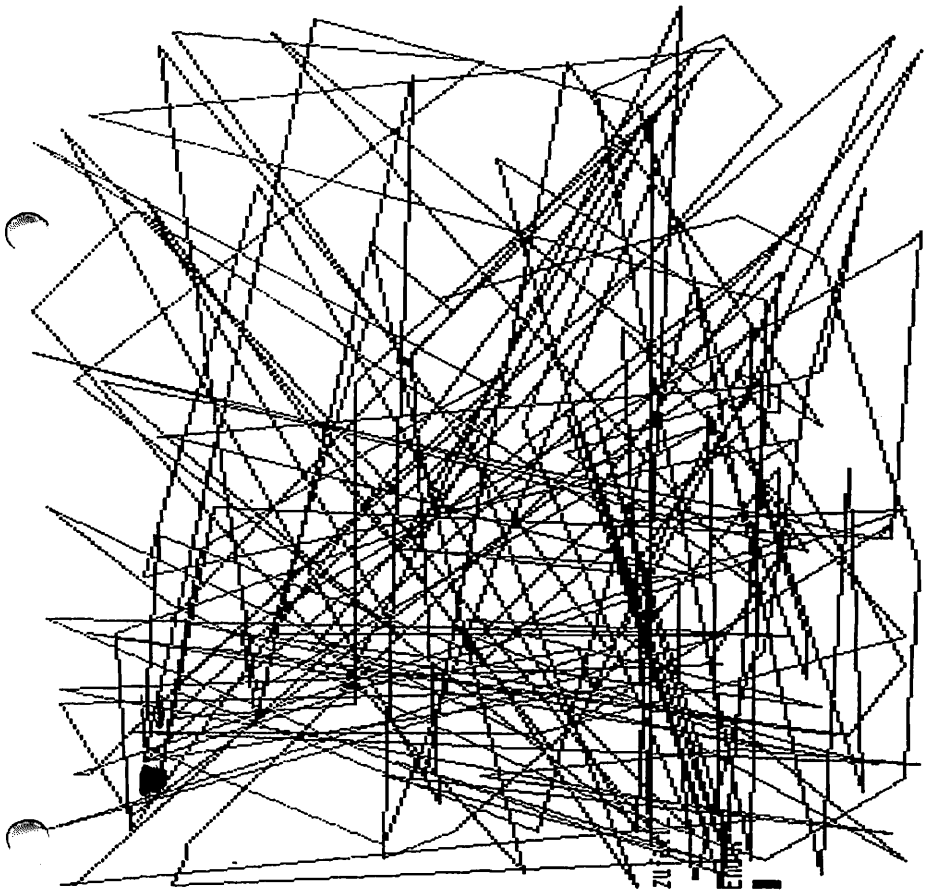
22

```
manu
setze "a :taste
wenn :a="V [vw 10] wenn :a="R [rw 10]
wenn :a="L [li 90] wenn :a="E [re 90]
manu
ende
manu
```

Nun kommt noch was Tolles. Der Zufall (besser: die Zufallszahl). Es gibt in GOSI eine Variable :ZZ, die immer unterschiedliche,zufällige Werte liefert. Dazu wird noch eine Zahl hinten angestellt, die festlegt, in welchen Grenzen sich die Zufallszahlen bewegen sollen. so liefert: SETZ "A :ZZ 10 eine Zufallszahl zwischen 0 und 9.
Bild 23 zeigt ein Programm damit und das graphische Ergebnis, das aber bei jedem anders ausfallen muß.
Bild 24 zeigt ein anderes Programm und Bild 25 ,das Ergebnis. Der Befehl AUFXY n1 n2 positioniert den Schreibzeiger direkt auf die absoluten Koordinaten, n1 steht für die x-Koordinate und n2 für die y-Koordinate. Der Wertebereich liegt zwischen 0 und 511.
Bild 25 zeigt ein weiteres Zufalls-Programm und Bild 26 das Bild dazu.



```
lerne zufall
setze "a :zz 3
wenn :a=0 [vw 10] wenn :a=1 [rw 10]
wenn :a=2 [re 90] wenn :a=3 [li 90]
zufall
ende
zufall
```



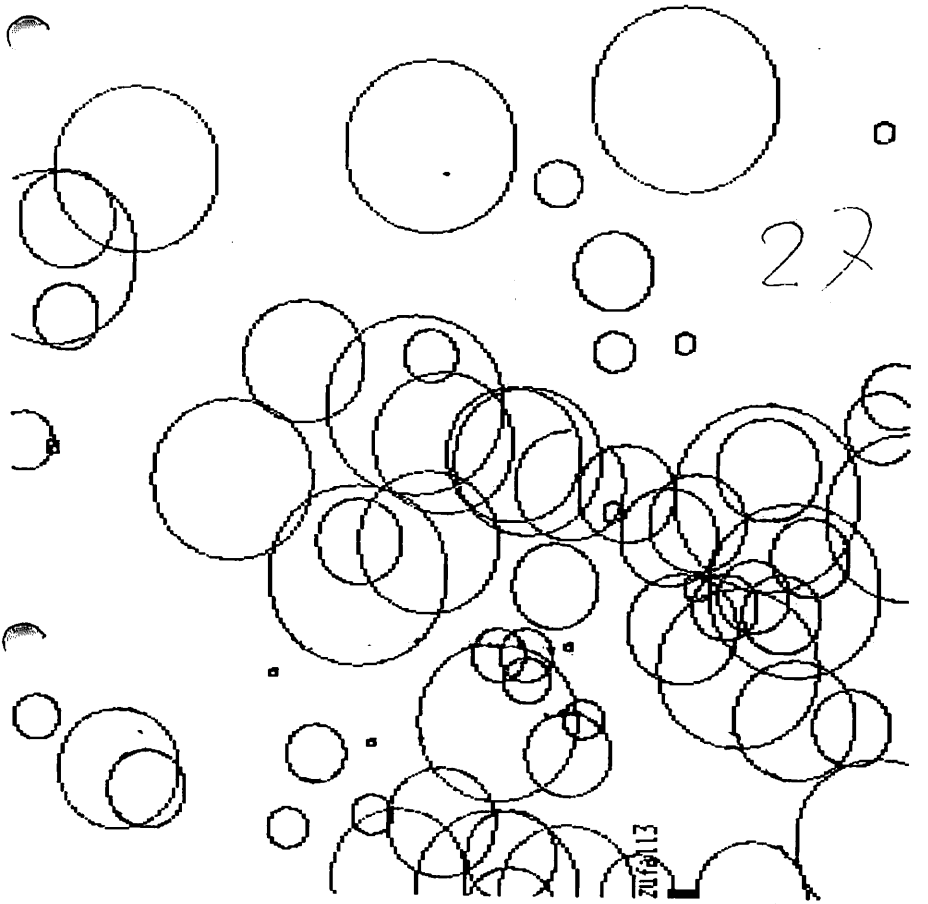
```
lerne zufall2
aufxy (:zz 64)*8 (:zz 64)*8
zufall2
ende

Ok gelernt, Platz zum lernen: 2404 Bytes und fuer Namen: 666 Bytes.
```

Nun ein anderes wichtiges Kapitel. Das Darstellen von Funktionen. Bild 27 zeigt das Programm und Bild 28 das Ergebnis. Mit dem Befehl LINIE n1 n2 n3 n4 lassen sich Linie von x1=n1 y1=n2 nach x2=n3 y2=n4 zeichnen. Dabei gilt für x der Wertebereich 0 bis 511 und für y der Wertebereich 0 bis 255. (Achtung dies ist anders wie bei AUFXY etc.).

Das Programm läßt sich leicht auf andere Funktionen übertragen, jedoch muß man aufpassen, da GOSI ja keine Kommastellen besitzt und nur mit ganzen Zahlen rechnen kann.

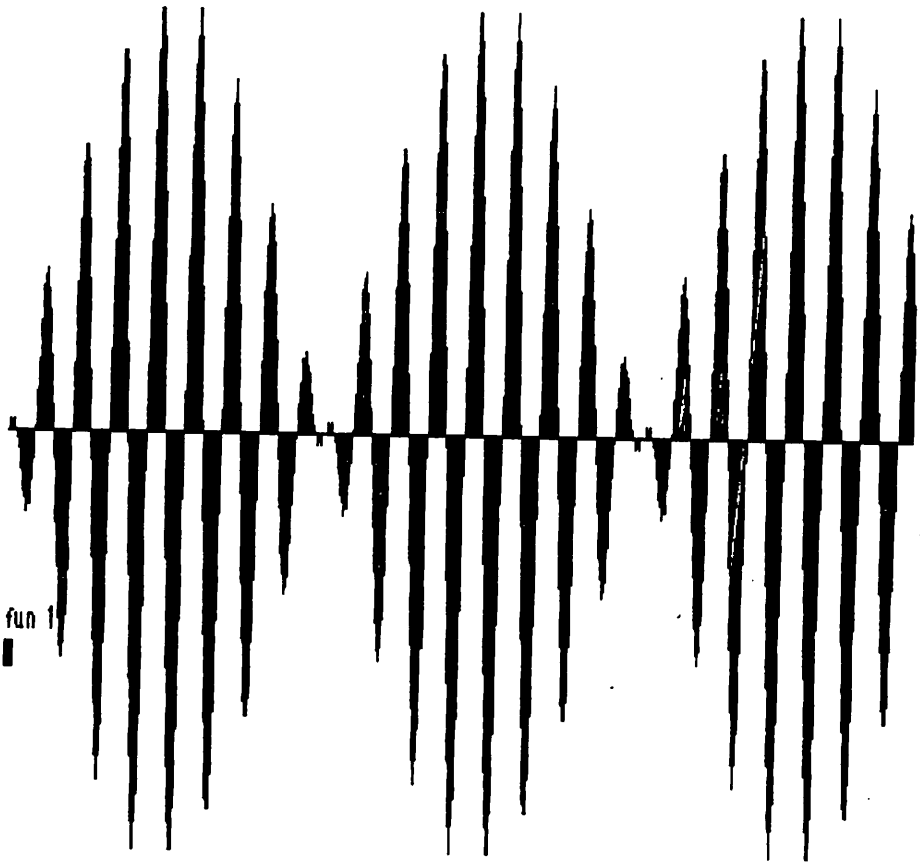
```
lerne zufall3
sh aufxy (:zz 64)*8 (:zz 64)*8 sa
setze "a :zz 30 wh 180 [schr16tel :a+1 li 2]
zufall3
ende

Ok gelernt, Platz zum lernen: 2304 Bytes und fuer Namen: 656 Bytes.
```

```
lerne fun :t
setze "erg ((:sin :t)/8)*((:cos :t*17)/8)/8
linie :t 128 :t :erg+128
fun :t+1
ende

Ok gelernt, Platz zum lernen: 2214 Bytes und fuer Namen: 650 Bytes.
```


27
Zufall3

Die weiteren Befehle kann man der nachfolgenden Übersicht entnehmen und sich nach und nach aneignen. Am besten ist immer das Experiment, wenn man eine Computersprache erlernen will.

Weitere Anregungen kann man den zahlreichen LOGO-Büchern entnehmen. Man muß nur ggf. die Schreibweise der GOSI-Schreibweise etwas anpassen.



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,eckige Klammer zu ist Ü.

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 Stelle im Text 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.

Ablauf

Ein Programm kann durch Eingabe der Tasten CTRL-S angehalten werden, CTRL-Q läßt es dann wieder weiterlaufen. Wird nach CTRL-S die Sequenz CTRL-C betätigt, so wird das gerade laufende Programm abgebrochen.

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

logische Verknüpfungen

!	Oder-Verknüpfung, z.B. :alpha!45 oder 3!5 ergibt 7 weil: 3 = 0011 5 = 0101 --------- 3!5 = 0110 = 7
&	Und-Verknüpfung, z.B. :beta&:alpha 3&5 ergibt 1 0011 & 0101 = 0001
~ (Welle)	Nicht-Verknüpfung, z.B. ~: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

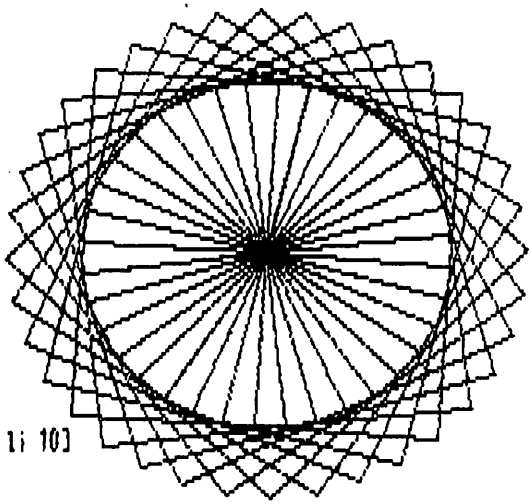
Ist der Vergleich Wahr, so wird der Wert -1 geliefert, sonst der Wert 0. Dadurch sind auch logische Verknüpfungen möglich, z.B. (:alpha<6)!(:betha>=:gamma)!ß(:a=:b)
Im Klartext:
Alpha Kleiner 6 oder betha größer gleich Gamma oder Nicht Klammerauf a gleich b Klammerzu.

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=13 ÄVW 10Ü Wenn die Taste CR (Carriage return, ASCII 13) 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 eine Taste gedrückt ist.

Mit :TASTE läßt sich dann der
eingegebene Wert abfragen.
Beispiel:
SOLANGE ~:taste? AVW 30 RE 90Ü
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

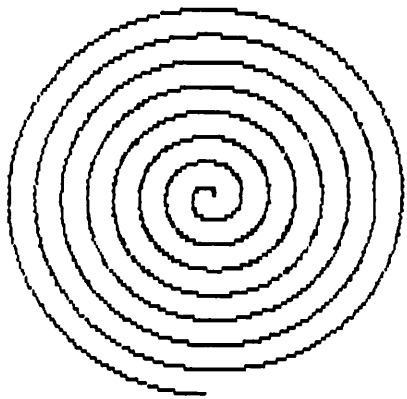
:KURS liefert den aktuellen Winkel des Zeigers. 0 Grad ist er, wenn er nach rechts zeigt, dann mathematisch positiv, nach oben z.B. 90 Grad. (ACHTUNG, dies weicht von anderen LOGO-Implementierungen u.U. ab).

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 6tel :n li 1]
ende
rarc 40
```



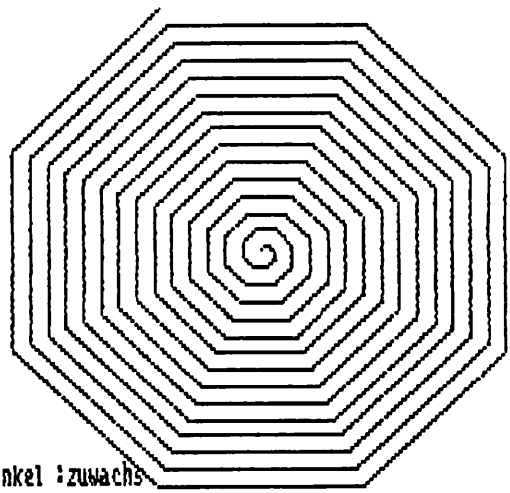
```
lerne spiro :n
rarc :n
spiro :n+1
ende

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

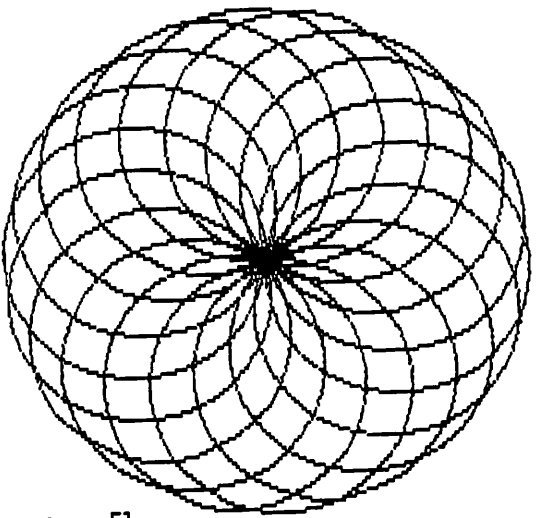
Hier nun die Befehlsliste.
Nochmals der Hinweis, daß eckige Klammern u.U. als Umlaute gedruckt sind, jedoch später auf dem Bildschirm als eckige Klammern erscheinen.

Ä ist eckige Klammer auf
Ü ist eckige Klammer zu
ß ist das Welle-Zeichen

1. Befehle für den Bildschirm

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 360ÄSCHR16TEL 7 RE 1Ü 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.



```
zeige drehkreise
drehkreise :r
wh 36 [re 5 schrlötel :r re 51
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.
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.

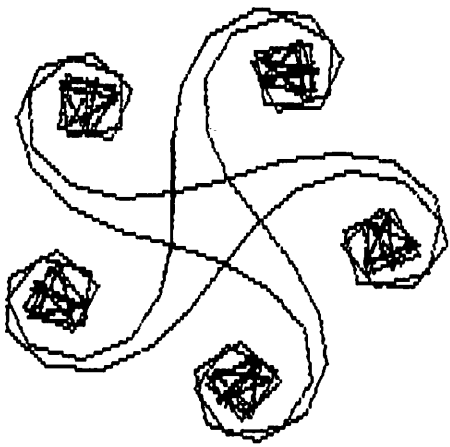
Seite

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.
```

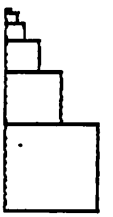


```
zeige knaeul
knaeul :seite :winkel
vschritt :seite :winkel
knaeul :seite (:winkel+10)
ende
knaeul 30 1
```

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.

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



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 Ä Ü gibt ein Leerzeichen aus, DR 1 DR Ä Ü DR 2 gibt 1 2 mit Leerzeichen aus. DR ÄALPHA ÄBETHAÜÜ gibt ALPHA ÄBETHAÜ 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 Listenverarbeitung 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

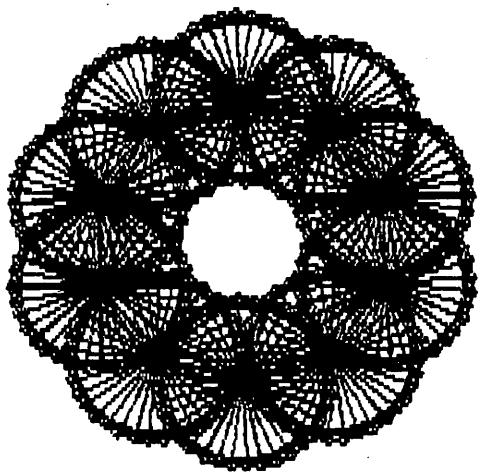
Der Drucker wird wieder ausgeschaltet.

4. Bedingungen

Befehl	Abkürzung	Erklärung
Wenn		Davon gibt es zwei Formen WENN Bedingung Ä Ü WENN Bedingung Ä Ü Ä Ü Beispiel: WENN :TASTE="A ÄVW 30Ü WENN :ALPHA<=:BETHA ÄVW 20Ü ÄRW 20Ü 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 ÄVW 20 RE 90 VW 20 RE 90Ü 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 Ä Ü WH 4 ÄVW 40 RE 90Ü zeichnet ein Quadrat. Die Befehle, die in Klammern stehen werden viermal ausgeführt.
Pruefe		Pruefe Bedingung PRUEFE :A=0 Wenn der Inhalt von a gleich 0 ist, wird ein Flag gesetzt und der Computer merkt sich den Zustand.
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 aufgerufenen Prozeduren
eine Pruefe-Anweisung vorkam.

Solange	SO	SOLANGE Bedingung Ä Ü SOLANGE ß:TASTE? ÄVW 50 RE 90Ü 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
```

5. Speicherverwaltung

Befehl	Abkürzung	Erklärung
Lerne		Lerne name parameter parameter ... Damit wird dem Computer eine eigene Prozedur beigebracht. Beispiel: LERNE QUADRAT :GROESSE WH 4AVW :GROESSE RE 90Ü 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 Werte zugewiesen werden, die dann innerhalb der Prozedur verwendet werden. Diese Platzhalter sind lokal, das heißt nur innerhalb der Prozedur mit diesen 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

Zahlen auszugeben. Das Programm endet nicht.
Diese Form der Rekursion nennt man TAIL-Recursion (endständige Rekursion), da der Aufruf von sich selbst am Schluß der Prozedur erfolgt. GOSI erkennt dies und definiert keine neuen lokalen Variable, sondert führt nur eine neue Zuweisung durch. Ferner wird auch nur ein Sprung ausgeführt und kein Unterprogramm-aufruf.

Rueckkehr RK

```
Beispiel 2 zur Rekursion.
LERNE REC2 :ZAHLEIN
 WENN :ZAHLEIN=0 ÄRUECKKEHRÜ
 REC2 :ZAHLEIN-1
 DZ :ZAHLEIN
ENDE
```

Hier ist eine andere Form der Rekursion. Mit RUECKKEHR wird die Prozedur beendet, so als ob sie an das ENDE gekommen wäre. REC2 5 gibt die Zahlen in aufsteigender Reihenfolge, also 1 2 3 4 5, auf dem Bildschirm in jeweils neuen Zeilen aus. Ist die angegebene Zahl zu groß, (ca 200 bis 300), dann erfolgt die Meldung SPEICHERENDE. Denn diesmal muß jeder Wert der lokalen Variable auf dem Stack gemerkt werden, und daher ergibt sich eine Vervielfachung des Speicherbedarfs.

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. Das ist übrigens 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 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

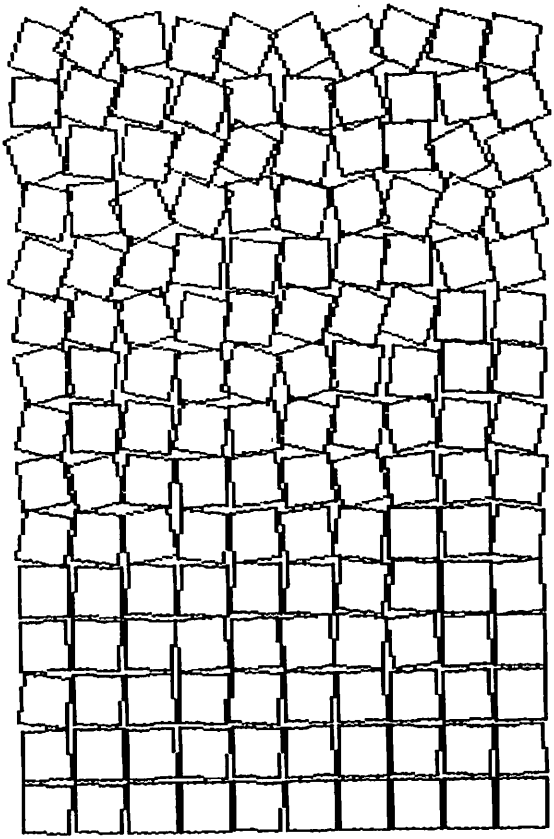
damit kann man unmittelbar nach dem Abspeichern durch Bewahre prüfen, ob alle richtig abgespeichert ist. Durch TEST wird der Speicher nicht verändert.

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.

```
************************************************************
*    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

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