

**ATARI®**

**130 XE™**



**HANDBUCH**





# **DAS 130 XE HANDBUCH**

ATARI Corp. (Deutschland) GmbH





## VORWORT

Mit Ihrer Entscheidung für das neue ATARI XE-Computer-System haben Sie eine wohlüberlegte Wahl getroffen. Leistungsfähig und zukunftssicher bietet es neben einer Palette sinnvoll aufeinander abgestimmter Peripherie-Geräte auch ein umfangreiches Angebot fertiger Computer-Programme vom Bereich Unterhaltung bis zur kaufmännisch-technischen Tabellenkalkulation.

Sollten Sie bereits mit einem Computer gearbeitet haben, wird Ihnen vieles in dieser Dokumentation bekannt sein und wir empfehlen Ihrer Aufmerksamkeit die im Anhang aufgeführte Spezial-Literatur. Als Computer-Neuling sollten Sie dieses Handbuch aber aufmerksam lesen. Erfahrungs-gemäß sind auftauchende Probleme meistens auf fehlerhafte Bedienung oder Nichtbeachtung von Programm-Anleitungen zurückzuführen.

Aber keine Angst: durch Eingabebefehle wird Ihr ATARI XE-Computer nicht beschädigt! Im schlimmsten Fall ist das zuvor eingegebene Programm nur gelöscht und muß nochmals eingetippt werden.

Dieses Handbuch soll Ihnen helfen, den Computer richtig zu bedienen und einen Einblick in BASIC als am weitesten verbreitete Programmier-Sprache zu gewinnen, um selbst kleinere Programme schreiben zu können.

Kritische Hinweise zur verbesserten Darbietung des Stoffes, sind uns stets willkommen.

Wir wünschen Ihnen mit Ihrem neuen ATARI XE-Computer viel Spaß — sei es beim Selbstprogrammieren oder bei Benutzung der vielen fertigen ATARI-Computer-Programme.

ATARI  
Corp. (Deutschland) GmbH



**INHALT****Seite**

Vorwort	5
Das ATARI XE Computersystem	9
Systemübersicht	10
Tastatur	11
Peripheriegeräte	15
Inbetriebnahme	16
Systemprüfung	17
BASIC	20
ATARI BASIC	21
BASIC Kurs	22
Übungsbeispiele	58
Fehlermeldungen	69
ATARI Computer-Programme	73
ATARI Club	75
Literaturhinweis	77
Service	80
Anhang	81
Technische Daten	82
Pinbelegung	83
Blockschaltbild	85
Verbindungskabel	86
Interner Zeichensatz	87
Speicheraufteilung	88
Speicheradressen	89
BASIC Befehls Worte	92
RAM-Konfigurationsmöglichkeiten	95
C. ATASCII Zeichensatz	96





## Das ATARI XE-Computer-System

Das ATARI XE-Computersystem wurde aus der weltweit erfolgreichen Modellreihe ATARI 400/800 weiterentwickelt unter besonderer Beachtung einer durchgehenden Systemkompatibilität, das hat für den Anwender den Vorteil, bereits vorhandene ATARI Peripherie-Geräte oder Computer-Programme auch mit dem neuen ATARI XE-Computer nutzen zu können.

Die Einsatzmöglichkeiten des ATARI XE-Computers sind vielfältig und eigentlich nur begrenzt durch die eigene Phantasie und die vorhandenen finanziellen Möglichkeiten. Neben den Spiel-Programmen zur Unterhaltung wird eine Reihe weiterer Programme aus dem Bereich Lernen, Weiterbildung, Heim und Beruf sowie System-Programme zur Programmierunterstützung angeboten. Darüberhinaus stehen neben der bereits in den Computer eingebauten Programmiersprache ATARI BASIC weitere Sprachen auf verschiedenen Datenträgern zur Verfügung. Das ATARI Programm-Angebot wird ständig erweitert und aktualisiert.

Der ATARI XE-Computer hilft Jugendlichen z.B. beim Mathematik-, Deutsch- und Fremdsprachenunterricht, unterstützt bei der Datenverwaltung und erleichtert die Arbeit bei der sonst so mühseligen Tabellenkalkulation und -planung. Nicht zuletzt unterstützt er den in vielen Schulen angebotenen Informatik-Unterricht. Die technologische Entwicklung der letzten Jahre hat bei gleicher Leistungsfähigkeit Computer in Schreibmaschinengröße ermöglicht, und genauso einfach sind sie auch zu bedienen. Die Programme können über die Tastatur eingegeben oder als fertige Programme in Form von Steckmodulen oder anderen Datenträgern wie Cassette oder Diskette benutzt werden.



## System-Übersicht

### Serielle I/O Buchse

Anschluß für ATARI I/O-Kabel zu Peripheriegeräten, wie Diskettenlaufwerken, Programmrecorder, Drucker und Modems.

### Steckmodul Einschub

Programm-Steckmodule erlauben schnelles Wechseln und leichtes Starten von Programmen.

### Erweiterung Erweiterter Modul Anschluß

Zum Anschluß schneller Peripheriegeräte (Festplatten) und anderen Ein-/Ausgabegeräten.

### Monitor Anschlußbuchse

Zum Anschluß eines Monochromen- oder Farbmonitors über ein Monitorkabel.

### VHF-Kanal Wahlschalter

Wählt VHF-Kanal 2 oder 3 für beste Bildarstellung.

### TV Anschlußbuchse

In diese Buchse wird das TV-Anschlußkabel eingesteckt.

### Anschluß für Stromversorgung

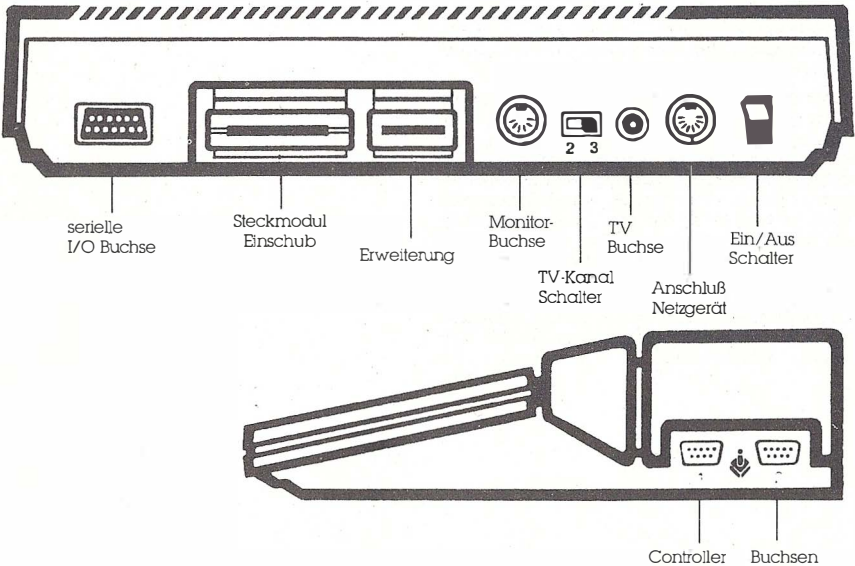
Zur Verbindung mit dem Netzgerät.

### Ein/Aus Schalter

Schaltet die Stromversorgung des Computers Ein und Aus.

### Controller Buchsen

Zum Anschluß von Touch Tablet, Numerischer Tastatur, Joysticks und Paddle-Controllern. Wird nur ein Controller benutzt, diesen in Port 1 einstecken.





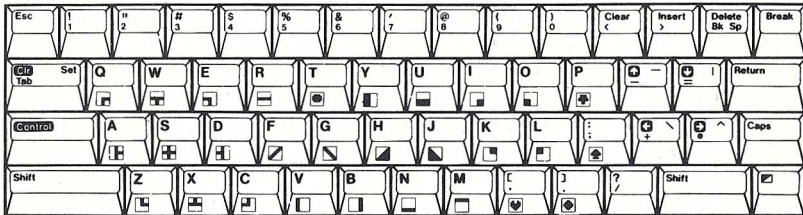
Neben dem Arbeitsspeicher zur Verarbeitung von Daten und Programmen gibt es im Computer noch einen weiteren Speicherbereich, das Betriebssystem (ROM). Dieser Bereich steuert, rechnet und vergleicht. Um auf beide Bereiche zugreifen zu können und so mit dem Computer zu "sprechen" bedarf es einer Programmiersprache.

Einfache und leicht zu erlernende Programmiersprachen wie z.B. das bereits in die ATARI Computer eingebaute ATARI BASIC bringen es häufig mit sich, daß der Computer Daten nicht schnellstmöglich bearbeiten kann, da er sie noch in seine eigene (Maschinen-) Sprache übersetzen muß und daher noch etwas mehr Zeit benötigt. Schnelle Programmiersprachen wie z.B. ATARI Assembler erfordern einen etwas höheren Lernaufwand, ermöglichen aber durch die computer-gerechte Darstellung auch schnelle Programm-Abläufe.

## TASTATUR

Die Tastatur entspricht einer Schreibmaschinentastatur mit Groß- und Kleinbuchstaben und zusätzlich einigen Graphik-Tasten. Genau genommen enthält die Tastatur drei verschiedene Tasten-Sätze: Normal-, Graphik und Umlautastatur.

### Normaltastatur



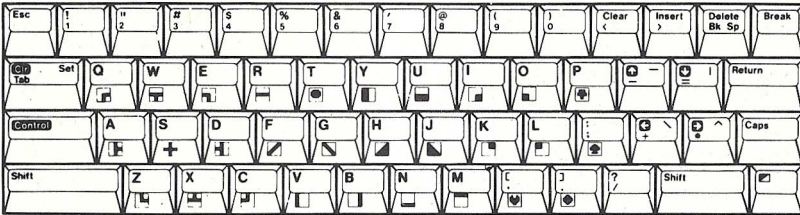
Keyboard

Mit der CAPS-Taste kann zwischen Groß- und Kleinbuchstaben umgeschaltet werden.



## TASTATUR

### Graphiktastatur



Durch Drücken der CONTROL-Taste zusammen mit einer weiteren Taste wird das jeweilige Graphik-Zeichen dargestellt.

### Tastatur mit Umlauten



Auf diesen intern. Zeichensatz kann umgeschaltet werden durch Eingabe von POKE 756, 204 (RETURN). Das gewünschte Zeichen (Umlaut) wird dann bei zusätzlichem Drücken der CONTROL-Taste auf dem Bildschirm dargestellt. Durch Eingabe von POKE 756, 224 kehrt man in den Normalmodus zurück.



## Funktionstasten

Die separat angeordneten Tasten RESET, OPTION, START, SELECT und HELP haben besondere Funktionen:



läßt den Computer die gerade durchgeführte Aufgabe abbrechen und zum Ausgangszustand zurückkehren. Diese Taste sollte deshalb nicht zum gewollten Unterbrechen eines Programms gedrückt werden, da in bestimmten Fällen der gesamte Speicherbereich gelöscht wird.



rufft in einigen Programmen Erläuterungen und Bedienungshinweise ab.

Die folgenden drei Sonderfunktionstasten sind in der Anwendung grundsätzlich gleichwertig und frei programmierbar. Zum leichteren Arbeiten werden sie aber meist wie folgt genutzt:



wählt Variationen innerhalb eines Programmes.



erlaubt die Auswahl der Anwendungen innerhalb eines Programmes.



beginnt ein Spiel oder einen Programm-Teil.

# ATARI® COMPUTER



**BREAK** dient zum Abbruch selbstgeschriebener Programme.



**ESC (ESCAPE)** ist in der Funktion vom jeweiligen Programm anhängig.



Mit **CONTROL INSERT** können Leerzeichen bzw. Zwischenräume eingetüft werden.



**DELETE BACK SPACE** bewegt den Cursor nach links und löscht vorhandene Zeichen.



**CONTROL** mit **DELETE BACK SPACE** löscht das Zeichen unter dem Cursor. Die Zeichen rechts rücken nach.



**SHIFT** führt zur zweiten Belegung der Tasten, wie z.B. Großbuchstaben und Sonderzeichen.



**CAPS** dient zum Umschalten zwischen Groß- und Kleinschreibung. Gibt den **CONTROL LOCK**-Modus frei.



**CONTROL** mit **CAPS** bewirkt den **CONTROL LOCK**-Modus, so daß **CONTROL** "verriegelt" ist.



**SHIFT** mit **CAPS** verriegelt den Computer im Großbuchstaben-Modus. Für die Sonderzeichen der numerischen Tasten muß weiterhin **SHIFT** gedrückt werden.



**CONTROL** mit **1** unterbricht die Ausgabe auf dem Bildschirm. Ein nochmaliges Drücken setzt die Ausgabe fort.



**CONTROL** mit **2** läßt einen Summiererton ertönen.



**CONTROL** mit **3** markiert das Datenende (EOF) nach Eingabe.



**SHIFT** mit **INSERT** fügt eine Leerzeile oberhalb des Cursors ein.



**SHIFT** mit **DELETE BACK SPACE** löscht die Zeile, in der sich der Cursor befindet.



**CONTROL** mit Pfeiltaste bewegt den Cursor, ohne das Programm oder die Anzeige zu ändern.



**RETURN** bewegt den Cursor an den linken Rand. Informiert den Computer über das Ende eines Schreibvorganges oder Editieren einer Programmzeile.



**INVERSE VIDEO** schaltet die Negativ-Darstellung ein und aus. In manchen Programmen ist diese Taste (ATARI-Taste) benannt.

**AUTO REPEAT** bewirkt durch längeres Festhalten einer Taste die Wiederholung des entsprechenden Zeichens.



### Datenspeicher

Ihr ATARI 130XE Computer stellt den Grundstein eines leistungsfähigen und vielseitig verwendbaren Systems dar. Der 130XE unterstützt jede Anwendung und jedes Spiel durch passende Peripheriegeräte. Die bekanntesten Peripheriegeräte sind in der Zeichnung dargestellt.

### TV-Geräte oder Monitor

Zur Bildschirmdarstellung können Sie sowohl ein TV-Gerät oder einen Computermonitor anschließen. Ein Monitor wird empfohlen, da er eine schärfere Darstellung bietet. Es kann sowohl ein Farbmonitor oder ein monochromer Monitor verwendet werden. Viele Programme verwenden jedoch die umfangreichen Möglichkeiten der Farbdarstellung des 130XL Computers.

**Programm-Recorder** wie z.B. das Modell ATARI 1010 bieten bei geringen Anschaffungskosten umfassende Möglichkeiten zum Speichern und Laden von Programmen und Daten. Als Datenträger werden Tonband-Cassetten (Compact-Cassetten) in Ferro-Qualität empfohlen. Chromdioxid-Cassetten sind aus technischen Gründen weniger geeignet. Die Speicherkapazität beträgt je nach Bandlänge bis zu 100K Bytes, das entspricht ungefähr 60 Schreibmaschinenseiten. Programm-Recorder sind weniger geeignet, wenn es auf schnellen, wahlfreien Datenzugriff ankommt.

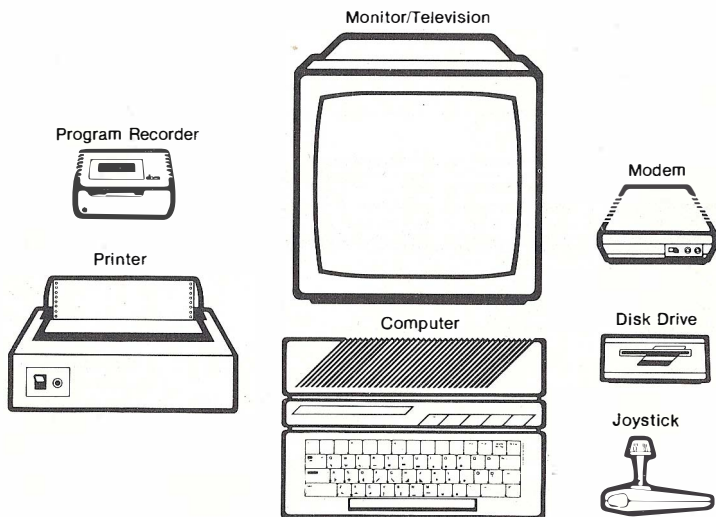
**Disketten-Stationen** wie z.B. das Modell ATARI 1050 bieten bei einer Speicherkapazität von bis zu 127K Bytes schnellen Datenzugriff. Speichermedium sind 5 1/4" Disketten. Durch Zusammenschalten mehrerer Stationen läßt sich der gesamte geschlossen verwaltete Datenbestand um ein vielfaches Erhöhen.

### DRUCKER

Dem Anwender eines ATARI Computers steht eine umfangreiche Palette an Druckern zur Verfügung. Die Drucker X1M201 (1M) und X1C 201 (1M) sind grafikfähige Matrixdrucker zum Anschluß an Ihren 130XE – mit dem XTC201 drucken Sie sogar in Farbe. Briefe, Berichte, Graphiken und Ihre Computerprogramme drucken Sie am günstigsten auf dem XMM801 (1M), einem hochwertigen und dabei preiswerten Matrixdrucker.

Professionell wirkende Ausdrücke Ihrer Briefe erhalten Sie mit dem XDM121 (1M), ein Typendrucker in Korrespondenzqualität.

**Kontroller** dienen zur zusätzlichen individuellen Dateneingabe. Als Drehregler oder Steuerknüppel werden sie vor allem für schnelle Spiele benötigt. Eine Weiterentwicklung dieser Regler ist der Trak-Ball. Er hat anstelle des Steuerhebels eine daisseitig bewegliche Kugel und erlaubt deshalb besonders schnelle Reaktion.

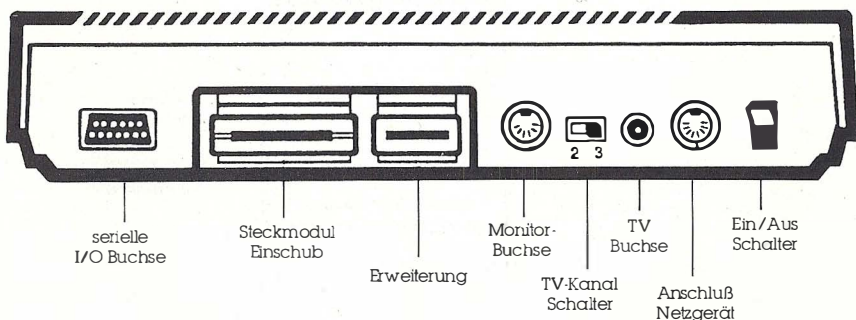




## Inbetriebnahme

1. Schieben Sie den runden Stecker des Netzgerätes in die POWER beschriftete Buchse an der Rückseite des Computers. Schließen Sie dann das Netzgerät an die häusliche Stromversorgung an.
2. Das Antennenkabel ist mit der TV-Buchse des Computers und dem Antenneneingang des Fernsehgerätes zu verbinden. Sollte Ihr Fernsehgerät keinen Normanschluß besitzen, fragen Sie bitte Ihren Fachhändler nach passenden Adaptern. Bei Verwendung eines Monitors benutzen Sie bitte die Monitor-Buchse des Computers und ein geeignetes Verbindungskabel.
3. Das Fernsehgerät auf Kanal 4 (VHF) oder 36 (UHF) abstimmen. Den Computer einschalten. Mit einem surrenden Ton erscheint die Meldung READY auf dem Bildschirm. Bei Bedarf ist am Fernsehgerät die Bildschärfe nachzuregulieren.

Hinweis: Wenn eine Anzeige längere Zeit unverändert auf dem Bildschirm erscheint, ändern sich deren Farben in regelmäßigen Abständen. Das ist normal und dient zum Schutz der Bildröhre. Nach Abschalten des Computers sollte bis zum Wiedereinschalten 3 bis 4 Sekunden gewartet werden, um die Elektronik nicht zu überlasten.



TV-Anschlußkabel und Netzgerät sind im Lieferumfang enthalten.



## SYSTEMPRÜFUNG

Nach jedem Einschalten prüft sich der Computer automatisch selbst und zeigt einen Moment später das Start-Bild des Programms auf dem Bildschirm an. Sofern im Computer etwas nicht fehlerfrei funktioniert, so erscheint die Meldung MEMORY TEST auf dem Bildschirm.

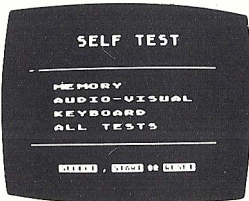
Sie können diesen Test auch selbst starten.

- indem bei gedrückter OPTION-Taste der Computer eingeschaltet wird,

oder

- nach der Meldung READY den Befehl BYE eingetippt und die RETURN-Taste gedrückt wird.

Es erscheint das SELF-TEST-Menü. Als Menü wird eine vorgegebene Programm- oder Funktionswahl bezeichnet.



SELBSTTESTMENÜ

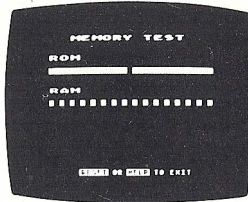
Wählen Sie mit der SELECT-Taste die gewünschte Prüfung und drücken Sie dann auf die START-Taste. Ein Druck auf die HELP-Taste führt zum SELF-TEST-Menü zurück. Mit der RESET-Taste kann der gesamte Test unterbrochen und zum ATARI BASIC zurückgekehrt werden.

### Speichertest

Der Computer hat zwei Arten von Speichern: Den Festwertspeicher (ROM) und den frei verfügbaren Arbeitsspeicher (RAM), in dem Programme und Daten abgelegt und verarbeitet werden.

ROM-Test: Es erscheinen zwei Balken, die sich von weiß in hellgrün (bzw. blau) verändern müssen. Bei Verfärbungen in Rot oder Rosa liegt ein Defekt vor.

Der Speichertest ist mit einer Verkehrsampel zu vergleichen: Am oberen Bildschirmrand erscheinen zwei Balken, die den Zustand des Festwertspeichers (ROM) anzeigen. Grün bedeutet "alles in Ord-



SPEICHERTEST  
(MEMORY-TEST)

nung"; Rot bedeutet "Halt!". Leider kann nur Ihr Service Center wieder auf Grün schalten.

Während des RAM-Tests werden 112 KByte des Schreib-/Lesespeichers geprüft. Die verbleibenden 16 KByte sind nur mit spezieller Software zugänglich. 48 kleine Quadrate, von denen jedes 1K RAM repräsentiert und 4 größere Balken (als Zeichen für je eine 16K Speicherbank), signalisieren die Funktion des Speichers. Auch hier bedeutet Rot, daß etwas nicht stimmt und Sie Ihr Service Center aufsuchen sollen.



### Ton/Graphik Test

Der ATARI XE-Computer besitzt vier unabhängig von einander arbeitende Tonkanäle und hochauflösende Graphik. Über der Nummer des gerade geprüften Kanals erscheinen fünf Notenlinien mit Violschlüssel. Für jeden der vier Kanäle werden sechs Noten dargestellt und gespielt. Bitte prüfen Sie bei fehlendem Ton die eingestellte Lautstärke Ihres Fernsehgerätes. Die Farben innerhalb eines Testlauts müssen unverändert bleiben.



TON/GRAPHIK-PRÜFUNG  
(AUDIO-VISUAL TEST)

### Tastatur Test

Es erscheint die Tastatur auf dem Bildschirm. Gedrückte Tasten blinken auf. Bei SHIFT und CONTROL ist gleichzeitig eine weitere Taste zu drücken. BREAK ist ohne Funktion, HELP oder RESET beenden den Test.



TASTATUR-TEST  
(KEYBOARD-TEST)

Hinweis: Aus technischen Gründen entspricht die obere Reihe nicht der Tastaturanordnung Ihres Computers. Die dargestellten Zahlen 1 – 4 sind für spätere Funktionserweiterungen vorgesehen.



**NOTIZEN**

**BASIC**

Die Programmiersprache BASIC wurde bereits in den 60er Jahren in USA entwickelt und ist heute die am weitesten verbreitete Programmiersprache der Welt.

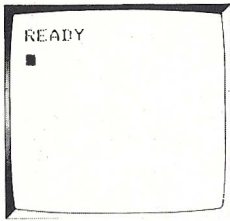
Die ständige Weiterentwicklung für weitere Anwendungsmöglichkeiten führte auch zu hersteller-spezifischen Dialekten. Das hier zu vermittelnde ATARI BASIC ist eine auf die Leistungsfähigkeit Ihres ATARI Computers ausgelegte Sprache. Die einfach zu lernende Struktur erleichtert das Verständnis für weitere höhere Programmiersprachen.

Der folgende Kurs ATARI BASIC behandelt die wichtigen Begriffe und geht besonders auf die faszinierenden Farb-, Grafik- und Tonmöglichkeiten ein. Er ist in erster Linie für Anfänger gedacht. Für weitergehende Informationen wird auf die Literaturvorschläge im Anhang dieses Buches hingewiesen.

Die diesem Kurs nachfolgenden Übungsprogramme sollen das vermittelte Wissen festigen und Sie ermutigen, selbst einfache Programme zu erstellen.

### ATARI BASIC

Sie haben das ATARI Computer-System wie beschrieben mit einem TV-Gerät/Monitor verbunden und eingeschaltet. Das eingebaute ATARI BASIC ist aktiviert und meldet sich mit:



Das kleine (weiße) Quadrat unterhalb von READY heißt 'Cursor'. Es gibt die Bildschirmposition an, an der das nächste Zeichen stehen wird.

Wurde beim Einschalten des Computers auch die OPTION-Taste gedrückt, so erscheint das SELF-TEST-Menü. Sofern eine ATARI Diskettenstation mit eingeschaltet ist, kann bei Verwendung der falschen Diskette oder unsachgemäßer Bedienung die Meldung BOOT-ERROR erscheinen.

Noch einige wichtige Hinweise:

- Alle Eingaben müssen im Normal-Modus, d.h. Großbuchstaben für Befehle und Anweisungen erfolgen. Kleinbuchstaben werden nur innerhalb von Anführungszeichen akzeptiert.
- Eine Zeile kann bis zu 114 Zeichen lang sein. Da der Bildschirm aber nur 38 Zeichen pro Zeile anzeigt, geht der Computer automatisch eine Zeile tiefer. Das Ende dieser "Logischen Zeile" von 118 Zeichen wird vom Computer mit einem akustischen Signal angezeigt.
- Alle Eingaben sind am Zeilenende durch Drücken der RETURN-Taste abzuschließen.
- Achten Sie auf den Unterschied von '0' (Null) und 'O'. Der Wert 0 ist auf der oberen Tastaturreihe angeordnet. Er hat zur besseren Darstellung bei Ausgabe über einen Drucker häufig einen Querstrich.



Der Buchstabe O steht eine Reihe tiefer.



- Nach jeder Zeile prüft der Computer automatisch, ob die Eingabe logisch richtig erfolgte und meldet sich bei Fehlern mit ERROR....
- Bereits mehrfach vermittelte Informationen wie Drücken der RETURN-Taste am Zeilenende werden im Verlauf dieses BASIC-Kurses bei den verschiedenen Programmbeispielen der besseren Verständlichkeit wegen nicht mehr beschrieben.

<b>Befehl/Anweisung</b>	<b>Seite</b>
PRINT	23
Direktmodus und Programmiermodus	24
LIST	25
NEW	26
END	27
Fehlermeldung	28
Aneinanderreihen von Anweisungen	29
Erweiterung der PRINT Anweisung	30
Tips zum Umgang mit Zahlen	31
Variablen	32
INPUT	33
REM	36
FOR/NEXT	35
GOTO	36
IF/THEN	37
READ/DATA/RESTORE	38
Indizierte Variablen	40
GOSUB/RETURN	41
ON/GOTO	42
ON/GOSUB	43
GRAPHICS 0-2	44
GRAPHICS 3-8	46
PEEK/POKE	50
GRAPHICS 9-11	51
Tabelle der Graphikstufen	53
SOUND	54
Abfrage der Steuerungen	55







## PRINT

Mit der Anweisung PRINT und einer beliebigen Ergänzung wird vom Computer etwas auf dem Bildschirm ausgedruckt. Mit dem PRINT-Befehl lassen sich u.a. Rechenoperationen durchführen.

Statt des Befehls PRINT kann auch das Fragezeichen eingegeben werden. Bei dem ? handelt es sich um eine Abkürzung für den PRINT Befehl.

Grundrechenarten:

Addition	:	+ mit der		Taste
Subtraktion	:	- mit der		Taste
Multiplikation:	:	* mit der		Taste
Division	:	/ mit der		Taste

Um das Ergebnis zu bekommen wird kein '=' Zeichen benötigt. Nach der Eingabe des PRINT-Befehls wird nur die RETURN-Taste gedrückt.

Einige Beispiele:

```
PRINT 5+7
PRINT 13-11
PRINT 4*5
PRINT 36/6
```

```
(RETURN)
(RETURN)
(RETURN)
(RETURN)
```

Auf dem Bildschirm sieht das wie folgt aus:



```
READY
PRINT 5+7
12

READY
PRINT 13-11
2

READY
PRINT 4*5
20

READY
PRINT 36/6
6

READY
```

Zur Potenzrechnung sind die Tasten  und  zu drücken.

Die Anweisung PRINT eignet sich aber auch dazu. Zeichen oder Texte auf dem Bildschirm ausdrucken. Sie müssen dazu nur den Text, den der Computer schreiben soll, in Anführungszeichen setzen.

Einige Beispiele:

```
PRINT "HALLO"
PRINT "DAS IST EIN SATZ."
PRINT "12 * 7"
```

```
(RETURN)
(RETURN)
(RETURN)
```

Der Bildschirm zeigt folgendes:

```
READY
PRINT "HALLO"
HALLO

READY
PRINT "DIES IST EIN SATZ."
DIES IST EIN SATZ.
```

```
READY
PRINT "12 * 7"
12 * 7
```

READY

Beachten Sie dabei, daß der Computer die Anführungszeichen nicht mit auf den Bildschirm ausdruckt.

## LPRINT

Der BASIC-Befehl LPRINT (Abkürzung für Line Print) hat die gleiche Funktion wie die Anweisung PRINT, nur gibt der Computer die Berechnungen und Texte nicht mehr auf dem Bildschirm sondern auf einem angeschlossenen Drucker aus.

Hinweis: Sollte der Computer nach der Benutzung des LPRINT Befehls einen ERROR-138 melden, so bedeutet dies meistens, daß kein Drucker angeschlossen ist, oder die Anschlußleitungen zu überprüfen sind.

## Direktmodus und Programmiermodus

### Direktmodus:

Bisher hat der Computer die Befehle sofort nach Betätigen der RETURN Taste ausgeführt. Die Eingabe im sog. Direktmodus ist nützlich, um z.B. eine Rechenoperation sofort auszuführen und so schnell ein Ergebnis zu bekommen.

### Programmiermodus:

Um komplexere Aufgaben zu lösen ist es notwendig, dem Computer eine Reihe von Anweisungen einzugeben, die er erst nach Eingabe eines bestimmten Befehls ausführt. Eine Reihe von Anweisungen, die der Computer nicht sofort ausführt, wird Programm genannt. Ein Programm wird mit dem BASIC-Befehl RUN gestartet. Nach jeder Eingabe von RUN läuft das Programm erneut ab.

Ein Programm unterscheidet sich von den Befehlen im Direktmodus dadurch, daß vor jedem Befehl noch eine sog. Zeilennummer steht. Als Zeilennummer eignet sich beim ATARI-Computer jede Zahl zwischen 0 und 32767.

Ein Beispiel:

```
10 PRINT "GUTEN TAG"      (RETURN)
20 PRINT "WALTER"        (RETURN)
30 PRINT "WIE GEHT ES DIR ?" (RETURN)
RUN                       (RETURN)
```

Wenn Sie obige Befehle eingeben, zeigt der Bildschirm:

```
READY
10 PRINT "GUTEN TAG"
20 PRINT "WALTER"
30 PRINT "WIE GEHT ES DIR ?"
RUN
GUTEN TAG
WALTER
WIE GEHT ES DIR ?
READY
```

Nachdem ein Programm mit RUN gestartet wurde, führt der Computer die Programmzeilen in aufsteigender Reihenfolge aus.

Beim Programmieren benutzt man oft Zeilennummern in 10-er Abständen, so daß noch genügend Platz bleibt, um eventuell noch Zeilen einzufügen.

Um eine Zeile zu verändern ist es am einfachsten, eine neue Zeile mit der gleichen Zeilennummer zu schreiben und dann RETURN zu drucken.

```
alte Zeile: 10 PRINT "GUTEN TAG"
neue Zeile: 10 PRINT "HALLO"      (RETURN)
```

Die alte Zeile wird durch die neue ersetzt, auch wenn diese kürzer ist.

Eine Zeile läßt sich ganz löschen, indem man die Nummer der zu löschenden Zeile eingibt und danach RETURN drückt.

```
30 (RETURN)
```

Nach dieser Eingabe ist die Zeile 30 aus dem Programm gelöscht.

**LIST**

Der Befehl LIST dient dazu, die einmal eingegebenen Programmzeilen auf dem Bildschirm darzustellen. Wenn die o.g. Beispiele ausgeführt wurden, sieht der Bildschirm nach der Eingabe von LIST wie folgt aus:

```
LIST
10 PRINT "HALLO"
20 PRINT "HALTER"
```

Bei längeren Programmen ist es zur besseren Übersicht sinnvoll, nur einen Teil des Programms aufzulisten. Um z.B. nur eine einzige Zeile aufzulisten, genügt es, LIST mit anschließender Zeilennummer einzugeben.

So ergibt die Eingabe von

```
LIST 20
```

folgende Zeile auf dem Bildschirm:

```
20 PRINT "HALTER"
```

Um einen bestimmten Programmteil aufzulisten, muß zusätzlich zum LIST Befehl noch die erste und die letzte Zeilennummer des Programmblocks, der aufgelistet werden soll, eingegeben werden.

Die Eingabe

```
LIST 10,20
```

listet alle Zeilen von 10 bis 20, auch diejenigen, die sich evtl. zwischen diesen beiden Zeilen befinden. Dabei darf das Komma zwischen den beiden Zahlen nicht vergessen werden.

Um ein Programmlisting auf einem angeschlossenen Drucker auszugeben, dient der Befehl

```
LIST "P:"
```

Mit folgender Anweisung läßt sich ein bestimmter Programmteil auf dem Drucker ausgeben.

```
LIST "P:",10,20 (RETURN)
```

Wichtig hierbei ist das Komma zwischen der LIST "P," Anweisung und der ersten Zahl. Das zweite Komma und die zweite Zahl kann auch weggelassen werden, wenn z.B. nur der Ausdruck der Zeile 10 gewünscht wird.

**NEW**

Durch die Eingabe von **NEW** wird das gesamte im Computer gespeicherte Programm gelöscht. Eine Eingabe von **LIST** nach dem **NEW** Befehl läßt den Computer nur **READY** auf den Bildschirm schreiben, da er kein Programm zum Listen mehr im Speicher hat.

**READY**  
**NEW**

**READY**  
**LIST**

**READY**

**END**

Manchmal wird ein Befehl benötigt, der das Programm an einer bestimmten Stelle zum Halten bringt. Der dazu notwendige Basic-Befehl heißt END.

Die Zeile

**100 END**

läßt den Computer ein Programm in der Zeile 100 beenden.

### Fehlermeldung

Beim ATARI-Computer gibt es zwei verschiedene Arten von Fehlermeldungen.

Die erste Fehlerart tritt dann auf, wenn man dem Computer Befehle eingibt, die er nicht ausführen kann, da sie keine BASIC Befehle sind. Hierbei kann es sich um einen falsch geschriebenen Basic Befehl handeln, oder man hat ein Semikolon statt eines Kommas eingegeben, usw.

Der Computer antwortet auf o.g. Fehler wie folgt:

```
10 PRINT "GUTEN TAG" (RETURN)
10 ERROR- PRINT "GUTEN TAG"
```

```
LIST 20:30
ERROR- LIST 20:30
```

(RETURN)

Dabei kennzeichnet der Computer die Stelle, an der der Fehler aufgetreten ist dadurch, daß er das betreffende Zeichen in inverser Schrift druckt. Die Stelle, an der der Computer den Fehler entdeckt, stimmt nicht immer mit der wahren Fehlerposition überein.

Die zweite Fehlerart tritt meistens bei Ausführung eines Programmes auf. Der Computer bricht dann das Programm ab und schreibt das Wort ERROR, gefolgt von einer Zahl, die den Fehler kennzeichnet und der Zeilennummer, an der der Fehler auftrat.

In diesem Beispielprogramm meldet der Computer bei nicht angeschlossenem Drucker folgenden Fehler.

```
10 LPRINT "DRUCKERTEST"
RUN
ERROR- 138 AT LINE 10
```

Aus dem Abschnitt über den LPRINT Befehl wissen Sie, daß ERROR 138 bedeutet, daß der Drucker nicht angeschlossen ist. Die Bedeutung weiterer Fehlerzahlen ist im Anschluß an die Übungsbeispiele aufgeführt.

**Aneinanderreihen von Anweisungen**

In Basic ist es möglich, mehrere Befehle statt in verschiedenen Zeilen in einer zu schreiben, die der Computer dann auch nacheinander ausführt. Die einzelnen Befehle müssen dabei mit einem Doppelpunkt getrennt werden. Zwei PRINT-Befehle lassen sich dann folgendermaßen zusammenfassen.

```
10 PRINT "HALLO":PRINT "HALTER"
```

Der Computer führt die beiden PRINT-Befehle so aus, als ob sie in zwei Zeilen stehen würden.

```
RUN  
HALLO  
HALTER  
READY
```

Damit lassen sich Befehle, die in einem Programm logisch zusammengehören, auch in eine Zeile schreiben. Durch diese Technik erhöht sich die Übersichtlichkeit der Programme. Als zweites läßt sich dadurch auch Speicherplatz einsparen, der bei sehr großen Programmen schon mal knapp werden kann.

### Erweiterung der PRINT Anweisung

Mit zwei Zeichen und einem zusätzlichen Basic-Befehl lassen sich die Anwendungsmöglichkeiten des PRINT Befehls einfach erweitern.

#### Das Komma ,

Mit Hilfe des Kommas kann man Texte in jeweils zehn Zeichen breiten Spalten untereinander ausdrucken.

```
10 PRINT "EINS", "ZWEI", "DREI"
20 PRINT "1", "2", "3"
RUN
```

```
EINS      ZWEI      DREI
1         2         3
```

READY

#### Das Semikolon ;

Ein Semikolon hinter einem PRINT Befehl bewirkt, daß der Text des nächsten PRINT Befehls genau hinter dessen Ende gedruckt wird.

Das Programm

```
10 PRINT "GUTEN TAG ";
20 PRINT "WALTER"
```

ergibt folgenden Bildschirmausdruck:

```
GUTEN TAG WALTER
```

### POSITION

Die Anweisung POSITION dient dazu, einen Text an einer bestimmten Stelle des Bildschirms zu positionieren. Der Befehl muß dabei vor dem entsprechenden PRINT Befehl gegeben werden. Ein typischer POSITION Befehl sieht z.B. wie folgt aus.

```
POSITION 20,11
```

Die erste Zahl gibt an, um wieviel Leerzeichen der Text vom linken Rand entfernt stehen soll. Die zweite Zahl gibt die Entfernung vom oberen Bildschirmrand an. Die beiden Zahlen müssen durch ein Komma getrennt werden.

Der Bildschirm hat beim ATARI-Computer waagrecht 40 Zeichen und senkrecht 24 Zeichen. Die für den POSITION Befehl gültigen Zahlen liegen damit im Bereich von 0-39 bzw. 0-23.

Da verschiedene TV-Geräte die linke Bildseite ganz oder teilweise verschieben, werden normalerweise nur 38 Zeichen pro Zeile dargestellt. Erfahrene Programmierer können bei Bedarf mit Hilfe einer POKE-Anweisung aber auch 40 Zeichen pro Bildschirmzeile nutzen.

## Tips zum Umgang mit Zahlen

### Punkt und Komma

Der Computer kennt bei Dezimalzahlen kein Komma. Die Zahl Zehn-Komma-Drei wird bei Computern mit einem Punkt geschrieben.

**Falsch:** 10,3

**Richtig:** 10.3

Bei einem Computer dürfen auch die einzelnen Tausendergruppen nicht durch ein Komma getrennt werden.

**Falsch:** 9,999,999

**Richtig:** 9999999

### Reihenfolge der Rechenoperationen

Der Computer rechnet einen mathematischen Ausdruck von links nach rechts aus. Dabei gilt aber folgende Reihenfolge der mathematischen Operationen:

1. Operationen in Klammern werden zuerst ausgeführt
2. Potenzierung, d.h. Erhebung einer Zahl in eine Potenz.  $2^3$  z.B. ist dabei gleich  $2 \cdot 2 \cdot 2$
3. Multiplikation bzw. Division.
4. Addition bzw. Subtraktion

Beispiele:

$$4 \cdot 5 + 2 = 22$$

$$4 \cdot (5 + 2) = 28$$

$$2^3 + 1 = 9$$

$$2^3 + 1 = 16$$

### Darstellung von sehr großen oder sehr kleinen Zahlen

Der Computer benutzt die sog. Gleitkommadarstellung um Zahlen darzustellen, die sehr groß oder sehr klein sind.

Bei dem Befehl

PRINT 50000000000 (Fünfhundertmilliarden)  
druckt der Computer 5.OE+11  
Die Darstellung bedeutet 5 mal 10 hoch 11, oder  $5 \cdot 10^{11}$

Bei dieser Schreibweise wird die 5 als Mantisse und die +11 als Exponent bezeichnet. Beide werden durch den Buchstaben E getrennt.

Bei der Umformung von Gleitkommazahlen in die übliche Schreibweise müssen zwei Fälle unterschieden werden.

### • bei positivem Exponenten

Die Mantisse wird separat geschrieben. Dann muß der Dezimalpunkt um so viele Stellen nach rechts verschoben werden, wie es der Exponent angibt. Eventuell entstehende leere Stellen werden mit Nullen aufgefüllt.

Beispiel : 3,141592E+10

Mantisse : 3,141592

Verschiebung des Dezimalpunktes um zehn Stellen nach rechts:

Dezimalzahl: 31415920000

### • bei negativem Exponenten

Die Mantisse wird ebenfalls separat geschrieben. Der Dezimalpunkt muß dann um so viele Stellen nach links verschoben werden, wie es der Exponent angibt. Leere Stellen werden auch hier mit Nullen aufgefüllt.

Beispiel : 4674E-8

Mantisse : 4674

Verschiebung des Dezimalpunktes um acht Stellen nach links.

Dezimalzahl: 0.0000004674

## Variablen

Eine Variable ist ein Speicher für eine Zahl oder eine Zeichenfolge. Der Variablenname besteht aus einem oder mehreren Zeichen (beim ATARI-Computer aus bis zu 128 Zeichen).

Bei der Benennung von Variablen müssen folgende Punkte beachtet werden.

1. Innerhalb eines Variablennamens sind nur die Buchstaben A-Z, sowie die Zahlen 0-9 erlaubt.
2. Innerhalb eines Variablennamens darf kein Leerzeichen stehen.
3. Das erste Zeichen des Variablennamens muß ein Buchstabe sein.
4. Es sind nur Großbuchstaben erlaubt.

Mit dem Basic Befehl

### LET

kann man einer Variablen eine Zahl oder eine Zeichenfolge zuweisen.

Mit der Zeile:

```
LET ZAHL=27
```

wird der Variablen ZAHL der Wert 27 zugewiesen.

Mit Hilfe des PRINT Befehls kann der Wert einer Variablen ausgedruckt werden.

```
PRINT ZAHL
27
```

Um den Wert der Variablen ZAHL zu bekommen, darf das Wort ZAHL hinter dem PRINT Befehl nicht in Anführungszeichen geschrieben werden.

Variablen, in den Zahlen gespeichert werden, heißen numerische Variablen. Außer den numerischen Variablen gibt es noch die Stringvariablen. In ihnen werden Zeichenfolgen gespeichert. Stringvariablen werden dadurch gekennzeichnet, daß am Ende des Variablennamens noch das Dollarzeichen \$ angehängt wird.

Der Stringvariablen TEXTS wird wie folgt der Text STRINGVARIABLE zugewiesen:

```
LET TEXTS="STRINGVARIABLE"
```

Der Text, der der Variablen zugewiesen wird, muß dabei in Anführungszeichen stehen.

Mit folgender Eingabe kann man den Inhalt von TEXTS wieder auf dem Bildschirm darstellen:

```
PRINT TEXTS
STRINGVARIABLE
```

Bevor man in einem Programm eine Stringvariable verwenden kann, muß man dem Computer sagen, wieviele Zeichen maximal in diesem String gespeichert werden dürfen. Dies erfolgt mit dem Befehl

### DIM

Eine Zeile 10, in der die Stringvariable TEXTS auf eine Länge von 20 Zeichen dimensioniert wird, sieht wie folgt aus:

```
10 DIM TEXTS(20)
```

In der Stringvariablen TEXTS können jetzt auch weniger als 20 Zeichen gespeichert werden. Es ist aber nicht möglich, mehr als 20 Zeichen zu speichern. Die restlichen Zeichen gehen dann verloren.

Es ist nicht möglich, einer Stringvariablen einen Text zuzuweisen, wenn diese vorher nicht dimensioniert wurde. Bei dem Versuch würde der Computer eine Fehlermeldung ausgeben, da ihm die Stringvariable nicht bekannt ist.

**INPUT**

Der INPUT-Befehl ermöglicht es, bei jedem Programmdurchlauf den Variablen andere Werte zuzuweisen. Bei einem INPUT Befehl wartet der Computer auf eine Eingabe, und weist diese Eingabe dann einer Variablen zu.

Das folgende Programm ist eine einfache Demonstration des INPUT-Befehls:

```
10 DIM NAMES(20)
20 PRINT "BITTE GEBEN SIE IHREN NAMEN
EIN"
30 INPUT NAMES
40 PRINT "SIE HEISSEN ";NAMES
```

Daß der Computer auf eine Eingabe wartet, erkennt man am Fragezeichen ?, das automatisch bei jeder INPUT-Anweisung, die der Computer ausführt, auf dem Bildschirm ausgegeben wird. Das Programm wird erst dann fortgesetzt, wenn die Eingabe mit (RETURN) abgeschlossen wurde.

**REM**

Bei komplexeren Programmen ist es sinnvoll, Kommentare mit in ein Programm zu schreiben. Kommentare erleichtern es z.B. Dritten, das Programm zu verstehen. In Basicprogramme lassen sich Kommentare mit der REM-Anweisung einfügen. (REMARK = Bemerkung)

Alles was in einer Programmzeile hinter einem REM-Befehl eingegeben wird, ignoriert der Computer beim Programmablauf.

```
10 REM *** BEISPIELPROGRAMM ***  
20 REM ZEILEN 10 UND 20 WERDEN BEIM PR  
PROGRAMMABLAUF IGNORIERT  
30 PRINT "HALLO"
```

Nach dem Start des Programms führt der Computer nur den PRINT-Befehl in Zeile 30 aus.

```
RUN  
HALLO  
READY
```

Hinter einem REM-Befehl lassen sich keine weiteren Basic Befehle ausführen, da der Computer sie als Kommentare interpretieren würde. Für jeden REM-Befehl sollte man also eine eigene Zeile verwenden.

**FOR/NEXT**

Eine sog. FOR-NEXT-Schleife wird in einem Basic Programm dazu benutzt, einen bestimmten Programmteil in vorgegebener Anzahl zu wiederholen. Eine FOR-NEXT-Schleife beginnt mit dem FOR-Befehl und endet mit dem NEXT-Befehl. Dazwischen liegt der Programmteil, der wiederholt werden soll. Einige zusätzliche Informationen hinter dem FOR-Befehl zeigen dem Computer an, wie oft die Schleife durchlaufen wird. Für eine FOR-NEXT-Schleife wird eine Kontrollvariable benötigt, die bei den Schleifendurchläufen hoch- bzw. runtergezählt wird.

Folgendes Programm zählt die Variable ANZAHL von 1 bis 4 hoch. Der Computer druckt bei jedem Schleifendurchlauf den aktuellen Wert der Variable ANZAHL auf dem Bildschirm aus.

```
10 REM *** FOR-NEXT-SCHLEIFE
20 FOR ANZAHL=1 TO 4
30 PRINT ANZAHL
40 NEXT ANZAHL
```

```
RUN
1
2
3
4
```

READY

Zwischen dem Anfangs- und Endwert der Kontrollvariable muß der Befehl TO stehen. Der Befehl TO ist fester Bestandteil einer FOR-NEXT-Schleife.

Man kann mit dem FOR/NEXT-Befehl aber nicht nur in Einerschritten zählen. Durch den Befehl **STEP** hinter der FOR-Anweisung kann eine bestimmte Schrittweite festgelegt werden.

Folgendes Programm druckt alle durch sechs teilbaren Zahlen zwischen 6 und 60 aus.

```
10 FOR ZAHL=6 TO 60 STEP 6
20 PRINT ZAHL;" IST DURCH 6 TEILBAR"
30 NEXT ZAHL
```

Mit Hilfe des STEP-Befehls kann auch rückwärts gezählt werden. Dazu muß nur eine negative Schrittweite angegeben werden.

```
10 FOR ZAHL=100 TO 1 STEP -1
20 PRINT ZAHL
30 NEXT ZAHL
```

Innerhalb einer FOR-NEXT-Schleife darf man eine zweite verwenden. Es können also mehrere Schleifen ineinander verschachtelt werden.

```
10 REM *** VERSCHACHELTE FOR-NEXT-SCH
LEIFEN
20 FOR AUSSEN=1 TO 4
30 PRINT "AEUSSERE SCHLEIFE"
40 FOR INNEN=1 TO 2
50 PRINT "INNERE SCHLEIFE"
60 NEXT INNEN
70 NEXT AUSSEN
```

In diesem Beispiel wird die innere Schleife viermal ausgeführt. Der Computer schreibt also viermal den Text

```
AEUSSERE SCHLEIFE
INNERE SCHLEIFE
INNERE SCHLEIFE
AEUSSERE SCHLEIFE.....
```

**GOTO**

Bis jetzt wurde ein Programm vom Computer immer stur vom Anfang bis zum Ende abgearbeitet. Für einen zweiten Durchlauf mußte das Programm dann erneut mit RUN gestartet werden. Mit dem GOTO-Befehl ist es möglich, ein Programm an einer beliebigen Stelle zu unterbrechen, und den Programmablauf an einer anderen Stelle fortzusetzen. Bei einem GOTO-Befehl springt der Computer immer zu der Zeilennummer, die die Zahl hinter dem GOTO festlegt.

Mit dem folgenden Beispielprogramm lassen sich Kilometer in Meilen umrechnen. Durch den GOTO-Befehl kann eine unbegrenzte Anzahl von Berechnungen durchgeführt werden, ohne daß das Programm immer wieder mit RUN gestartet werden muß.

```
10 PRINT "KILOMETER"  
20 INPUT KM  
30 LET M=KM/1.609344  
40 PRINT KM;" KILOMETER ENTSPRECHEN ";  
50 PRINT M;" MEILEN"  
60 GOTO 10
```

**IF/THEN**

Mit Hilfe des IF/THEN-(Wenn-Dann)-Befehls kann der Computer Vergleiche ausführen. Der weitere Programmablauf hängt dann von dem Ausgang dieses Vergleichs ab. Das Ergebnis des Vergleichs kann entweder 'wahr' oder 'falsch' sein. Ist das Ergebnis 'wahr', so führt der Computer die Befehle hinter dem THEN-Befehl aus. Ist die Bedingung nicht erfüllt, so überspringt der Computer die Anweisungen hinter dem THEN-Befehl, und setzt das Programm in der nächsten Zeile fort.

Es existieren folgende logische Vergleiche:

```
IF A = B THEN (Wenn A gleich B dann...)
IF A > B THEN (Wenn A größer B dann...)
IF A < B THEN (Wenn A kleiner B dann...)
IF A >= B THEN (Wenn A größer oder gleich B dann...)
IF A <= B THEN (Wenn A kleiner oder gleich B dann...)
IF A <> B THEN (Wenn A ungleich B dann...)
```

Mit einem kurzen Programm kann der Computer zwei Zahlen vergleichen.

```
10 PRINT "GEBEN SIE DIE ZAHLEN EIN"
20 PRINT "A= " : INPUT A
30 PRINT "B= " : INPUT B
40 IF A=B THEN PRINT "A IST GLEICH B"
50 IF A>B THEN PRINT "A IST GROESSER A
L5 B"
60 IF A<B THEN PRINT "A IST KLEINER AL
S B"
70 GOTO 10
```

Hinter dem THEN-Befehl kann jeder andere BASIC-Befehl stehen. Durch ein GOTO kann das Programm, abhängig vom Ergebnis des Vergleiches, an einer anderen Stelle fortgesetzt werden. Genau wie in einer 'normalen' Zeile können auch hinter dem THEN mehrere Befehle stehen, die durch den Doppelpunkt getrennt werden müssen.

```
40 IF A=B THEN PRINT "A IST GLEICH B" :
GOTO 10
```

Natürlich lassen sich mit dem IF/THEN-Befehl auch Zeichenfolgen vergleichen.

```
10 DIM C$(5)
20 PRINT "GEBEN SIE DAS CODEMORT EIN"
30 INPUT C$
40 IF C$<>"ATARI" THEN PRINT "FALSCHES
CODEMORT" : GOTO 20
50 PRINT "CODEMORT AKZEPTIERT"
```

### READ/DATA/RESTORE

Der **READ**-Befehl ist eine dritte Möglichkeit, neben **LET** und **INPUT**, einer Variablen einen Wert zu zuweisen. Die Daten, die den Variablen zugewiesen werden, müssen dabei schon in der Form von **DATA**-Zeilen im Programm vorhanden sein.

Eine **DATA**-Zeile sieht z.B. wie folgt aus.

```
70 DATA 23,34,545,3,323,53,9
```

Die Werte hinter dem **DATA**-Befehl müssen jeweils durch ein Komma getrennt werden. Vor dem ersten und hinter dem letzten Wert einer **DATA**-Zeile darf dabei kein Komma stehen. Für den Fall, daß mehr Werte vorhanden sind, als in einer **DATA**-Zeile Platz finden, kann einfach eine neue **DATA**-Zeile angefangen werden.

Mit dem **READ**-Befehl können einer Variablen nacheinander die Werte der **DATA**-Zeile zugewiesen werden. Ein **READ**-Befehl, der der Variablen **KM** Werte zuweist, sieht wie folgt aus:

```
20 READ KM
```

Im Computer läuft bei jedem **READ**-Befehl ein interner Zähler mit der angibt, welcher Wert als nächster gelesen werden soll. Beim Start eines Programms zeigt er auf den ersten Wert der ersten **DATA**-Zeile.

Das Programm zur Umrechnung von Kilometern in Meilen läßt sich wie folgt umschreiben.

```
10 PRINT "UMRECHNUNG VON KILOMETERN IN
MEILEN"
20 READ KM
30 LET M=KM/1.609344
40 PRINT KM;" KILOMETER ENTSPRECHEN";
50 PRINT M;" MEILEN"
60 GOTO 20
70 DATA 23,34,545,3,323,53,9
```

Das Programm hat allerdings einen Nachteil. Nachdem der letzte vorhandene Wert gelesen wurde, versucht das Programm nochmal einen Wert zu lesen. Da kein weiterer Wert vorhanden ist, bricht das Programm mit einer Fehlermeldung ab. Dieses läßt sich vermeiden, indem als letzter **DATA**-Wert eine bestimmte Zahl genommen wird. Mit einem **IF-THEN**-Befehl läßt sich das Programm beim Auftreten dieses Wertes anhalten.

```
10 PRINT "UMRECHNUNG VON KILOMETERN IN
MEILEN"
20 READ KM
30 IF KM=999999 THEN END
40 LET M=KM/1.609344
50 PRINT KM;" KILOMETER ENTSPRECHEN";
60 PRINT M;" MEILEN"
70 GOTO 20
80 DATA 23,34,545,3,323,53,999999
```

Wenn alle Werte gelesen wurden, gibt es eine Möglichkeit, den **DATA**-Zeiger wieder auf den ersten Wert der ersten **DATA**-Zeile zu setzen. Der Befehl dafür heißt

### RESTORE

Folgendes Programm gibt endlos die Zahlen in den **DATA**-Zeilen auf dem Bildschirm aus

```
10 READ Z
20 IF Z=999 THEN RESTORE :GOTO 10
30 PRINT Z
40 GOTO 10
50 DATA 6,12,18,24,30,36,42,48,54,60,9
99
```

Zusätzlich zum **RESTORE**-Befehl kann noch eine bestimmte Zeilennummer angegeben werden. Der Computer setzt dann den **DATA**-Zeiger auf den Anfang der **DATA**-Zeile, die die Zahl hinter dem **RESTORE**-Befehl angibt.

z.B. **RESTORE 120**

Obige Zeile setzt den **DATA**-Zeiger auf den ersten Wert der Zeile 120.

Mit dem **READ**-Befehl lassen sich aber auch Zeichenfolgen in Stringvariable einlesen.

```
10 DIM OBST5(20)
20 READ OBST5
30 IF OBST5="END" THEN END
40 PRINT OBST5
50 GOTO 20
60 DATA APFEL,APFELSINE,BANANE,BIRNE,END
```

In folgendem Programm werden mit einer **READ**-Anweisung gleichzeitig zwei Werte aus den **DATA**-Zeilen gelesen. Dies wird dadurch erreicht, das hinter dem **READ**-Befehl zwei Variablen angege-



ben werden. Es können auch mehr als zwei Variablen verwendet werden. Jede Variable muß dabei durch ein Komma von der nächsten getrennt werden.

```
10 DIM OS(20),OBST$(20)
20 PRINT "GEBEN SIE EINE OBSTSORTE EIN"
30 INPUT OS
40 RESTORE
50 READ OBST$,PREIS
```

```
60 IF OBST$="END" THEN GOTO 100
70 IF OS=OBST$ THEN PRINT "DER PREIS BETRÄGT ";PREIS;" DM":GOTO 20
80 GOTO 50
100 PRINT "FALSCHER SORTE":GOTO 20
120 DATA APFEL,1.34,APFELSIME,1.99
130 DATA BANANE,0.78,BIRNE,2.13
140 DATA END,0
```

Die Null am Ende der Zeile 140 ist sehr wichtig. Der Computer liest mit dem READ-Befehl immer zwei Werte aus den DATA-Zeilen ein. Bei fehlender Null könnte der Computer der Variablen PREIS keinen Wert zuweisen, und das Programm würde mit einer Fehlermeldung abbrechen.

## Indizierte Variablen

Eine indizierte Variable ist eine numerische Variable, der aber mehr als ein Wert zugewiesen werden kann. Durch einen sog. Index kann auf jeden Wert eindeutig zugegriffen werden. Als Index kann jede ganze Zahl dienen.

Der Name einer indizierten Variablen besteht aus zwei Teilen. Der erste Teil des Namens setzt sich genauso zusammen, wie der Name einer numerischen Variablen. An diesen Namen wird dann noch der in Klammer eingeschlossene Index angehängt. Die Variable

A(5)

ist z.B. eine indizierte Variable. Jeder der einzelnen Speicherstellen

A(●)  
A(1)  
A(2)  
A(3)  
A(4)  
A(5)

kann ein Wert zugewiesen werden.

Bevor in einem Programm eine indizierte Variable verwendet werden darf, muß diese dimensioniert werden. Die Anweisung DIM ist ja schon von der Dimensionierung der Stringvariablen bekannt.

In einem Programm besagt die Anweisung

```
10 DIM A(5),
```

daß man der Variablen A im folgenden sechs Werte zuweisen kann. Jeder einzelne Speicher wird durch einen Index von 0 bis 5 gekennzeichnet.

Das folgende Programm liest sechs Zahlen in die indizierte Variable FELD ein, und druckt dann die zu jedem Index zugehörige Zahl auf dem Bildschirm aus.

```
10 DIM FELD(5)
20 FOR PLATZ=0 TO 5
30 INPUT X
40 LET FELD(PLATZ)=X
50 NEXT PLATZ
60 FOR PLATZ=0 TO 5
70 PRINT "FELD(" ; PLATZ ; ")=" ; FELD(PLATZ)
80 NEXT PLATZ
```

Der Computer unterscheidet zwischen den Variablen FELD, FELD5 und FELD(5). Alle drei Variablen können in dem Programm nebeneinander verwendet werden.

Indizierte Variable werden im englischen Array genannt. Bis jetzt hatten die Variablen nur einen Index. Sie werden als eindimensionale Arrays oder als Vektor bezeichnet. Daneben gibt es noch die zweidimensionalen Arrays, die Matrizen genannt werden. Eine Matrix ist eine indizierte Variable mit zwei Indizes.

Bei der Dimensionierung und dem Zugriff auf einer Matrix müssen zwei Indizes angegeben werden, die durch ein Komma getrennt werden. Die Dimensionierung einer Matrix sieht z.B. wie folgt aus.

```
DIM B(3,2)
```

Die Variable B besteht jetzt aus 4 mal 3 Speicherstellen (der Index Null muß mitgezählt werden), die einzeln angesprochen werden können.

B(0,0)	B(0,1)	B(0,2)
B(1,0)	B(1,1)	B(1,2)
B(2,0)	B(2,1)	B(2,2)
B(3,0)	B(3,1)	B(3,2)

Mit Hilfe einer Matrix läßt sich z.B. eine einfache Artikelverwaltung programmieren. Das Programm verwaltet sechs Artikel mit den Artikelnummern 0 bis 5. Für jeden Artikel werden Preis und Lagerbestand gespeichert.

```
10 DIM ART(5,1)
20 PRINT "*** ARTIKELEINGABE ***"
30 FOR ART=0 TO 5
40 PRINT :PRINT "ARTIKEL NR. ";ART
50 PRINT "PREIS " ; INPUT PR
60 ART(CART,0)=PR
70 PRINT "LAGERBESTAND " ; INPUT LA
80 ART(CART,1)=LA
90 NEXT ART
100 PRINT :PRINT "*** VERKAUF ***"
110 PRINT :PRINT "ARTIKELNR. "; INPUT ART
120 IF ART<0 OR ART>5 THEN 110
130 PRINT "STUECK " ; INPUT ST
140 IF ART(CART,1)-ST<0 THEN PRINT "NICHT GENUEGEND ARTIKEL IM LAGER!" : GOTO 110
150 LET PR=ART(CART,0)*ST
160 LET ART(CART,1)=ART(CART,1)-ST
170 PRINT "SUMME: ";PR;" DM"
180 PRINT "NOCH " ; ART(CART,1) ; " STUECK IM LAGER"
190 GOTO 110
```

**GOSUB/RETURN**

Die Anweisung GOSUB ist nahe verwandt mit dem GOTO-Befehl. Trifft der Computer in einem Programm auf einen GOSUB-Befehl, so wird der Programmablauf in der Zeile fortgesetzt, die die Zahl hinter dem GOSUB angibt. Im Gegensatz zum GOTO-Befehl merkt sich der Computer, an welcher Stelle das GOSUB erfolgte. Eine nachfolgende RETURN-Anweisung bewirkt, daß das Programm hinter dem GOSUB-Aufruf fortgesetzt wird.

Durch die GOSUB/RETURN Anweisung kann ein bestimmter Programmteil, der öfters benötigt wird, von verschiedenen Zeilen aus aufgerufen werden. Ein Programmteil, der mit der RETURN-Anweisung abgeschlossen wird, heißt Unterprogramm.

Im folgenden Programmbeispiel steht das Unterprogramm ab Zeile 100. Es wird zweimal aufgerufen.

```

10 PRINT "ERSTER AUFRUF"
20 GOSUB 100
30 PRINT "ZWEITER AUFRUF"
40 GOSUB 100
50 PRINT "ENDE DES PROGRAMMS"
60 END
100 FOR T=1 TO 3
110 PRINT "DIES IST DAS UNTERPROGRAMM"
120 NEXT T
130 RETURN

```

Die END Anweisung in Zeile 60 verhindert, daß der Programmablauf nach dem zweiten Unterprogrammaufruf in Zeile 100 fortgesetzt wird. Der Computer würde sonst auf das RETURN in Zeile 130 stoßen, ohne daß vorher das Unterprogramm aufgerufen wurde. Dieses hätte dann eine Fehlermeldung zur Folge.

**ON/GOTO**

Durch die ON/GOTO Anweisung kann man aufgrund des Wertes einer Variablen zu bestimmten Zeilennummern verzweigen. In einem Programm kann z.B. folgende Zeile auftreten.

```
50 ON X GOTO 100,110,120,130
```

Bei diesem Beispiel hängt es von der Variablen X ab, in welcher Zeile das Programm fortgesetzt wird. Ist X gleich eins, so verzweigt das Programm zur Zeile 100. Bei einer zwei springt das Programm zu Zeile 110, usw.

Hinter dem ON/GOTO-Befehl können so viele Zeilennummern stehen, wie in eine Basic-Programmzeile passen. Der maximale Wert der Variablen richtet sich dann nach der Anzahl der Zeilennummern. Bei der Variablen muß es sich um eine numerische Variable handeln. Stringvariablen akzeptiert der Computer hierbei nicht.

Beispielprogramm:

```
50 ON X GOTO 100,110,120,130

10 PRINT :PRINT "1: ZEILE 100"
20 PRINT "2: ZEILE 200"
30 PRINT "3: ZEILE 300"
40 PRINT "4: ZEILE 400"
50 PRINT "5: ZEILE 500"
60 PRINT "GEBEN SIE EINE ZAHL VON 1 BI
5 5 EIN"
70 INPUT X
80 IF X<1 OR X>5 THEN PRINT "UNGUELTIG
E EINGABE":GOTO 10
90 ON X GOTO 100,200,300,400,500
100 PRINT "ZEILE 100":GOTO 10
200 PRINT "ZEILE 200":GOTO 10
300 PRINT "ZEILE 300":GOTO 10
400 PRINT "ZEILE 400":GOTO 10
500 PRINT "ZEILE 500":GOTO 10
```

## ON/GOSUB

Die Anwendung ON/GOSUB wird genauso verwendet wie der ON/GOTO-Befehl. Der Computer merkt sich aber wie bei einer 'normalen' GOSUB Anweisung die Zeilennummer, in der der Unterprogrammaufruf erfolgte. Nach einem RETURN-Befehl wird das Programm in der Zeile nach der ON/GOSUB Anweisung fortgesetzt.

Das Programm aus dem Kapitel über die ON/GOTO Anweisung kann folgendermaßen modifiziert werden. Die Zeilen 100 bis 500 werden so zu Unterprogrammen.

```

10 PRINT :PRINT "1: ZEILE 100"
20 PRINT "2: ZEILE 200"
30 PRINT "3: ZEILE 300"
40 PRINT "4: ZEILE 400"
50 PRINT "5: ZEILE 500"
60 PRINT "GEBEN SIE EINE ZAHL VON 1 BI
5 5 EIN"
70 INPUT X
80 IF X<1 OR X>5 THEN PRINT "UNGUETIG
E EINGABE":GOTO 10
90 ON X GOSUB 100,200,300,400,500
95 GOTO 10
100 PRINT "ZEILE 100":RETURN
200 PRINT "ZEILE 200":RETURN
300 PRINT "ZEILE 300":RETURN
400 PRINT "ZEILE 400":RETURN
500 PRINT "ZEILE 500":RETURN

```

## GRAPHIC

In den folgenden Kapiteln wird Ihnen erläutert, wie Sie Farbe und Aussehen der Objekte auf dem Bildschirm beeinflussen können. Ferner wird erklärt, wie man selbst Bilder malt.

Dabei können Sie unter 16 verschiedenen Betriebsarten (Graphikstufen) wählen. Die ersten drei Graphikstufen dienen zur Darstellung von Texten, die restlichen dreizehn Stufen zum Zeichnen von Bildern in unterschiedlicher fein auflösender Graphik.

Die drei Graphikstufen 0 bis 2 ermöglichen es, Text in verschiedenen Größen und Farben darzustellen. Sie können die erste Stufe durch folgende Anweisung erreichen:

### GRAPHICS 0 oder GR. 0

Der Bildschirm sieht nun immer noch so aus, wie Sie es gewohnt sind. Mit dieser Anweisung können Sie später auch aus anderen Graphikstufen wieder zurückkommen. Texte werden in der Graphikstufe 0 über eine PRINT-Anweisung auf den Bildschirm gebracht.

Beispiel:

```
PRINT "HALLO"
```

Das entstandene Bild ist mit dem aus dem Kapitel PRINT identisch.

Es können aber auch eine Reihe von Graphikzeichen auf dem Bildschirm erzeugt werden. Dies geschieht mit Hilfe der Control-Taste, die zusammen mit einer mit Graphikzeichen belegten Taste gedrückt werden muß.

Beispiel:

```
Print "S"
```

Beim Drücken der Taste S muß die CONTROL-Taste mitgedrückt werden. Es erscheint dann auf dem Bildschirm

```
PRINT "+"
```

Nach Drücken der RETURN-Taste erscheint auf dem Bildschirm nur noch das Graphikzeichen.

```
+  
READY
```

Es ist nun möglich, mit Hilfe der Graphikzeichen einfache graphische Figuren zu zeichnen.

Beispiel:

```
10 PRINT "HJ"  
20 PRINT "VB"  
30 PRINT "MM"
```

```
10 PRINT " " "  
20 PRINT " " "  
30 PRINT " " "
```



Wenn Sie gerne mit den Graphikzeichen experimentieren möchten, drücken Sie die CONTROL-Taste und die CAPS-Taste gleichzeitig. Beenden können Sie die Festeinstellung der CONTROL-Taste durch gleichzeitiges Drücken der SHIFT und CAPS-Taste.

Die Graphikstufen 1 und 2 werden genauso wie die Graphikstufe 0 aufgerufen, wobei der Bildschirm dann in zwei Teile unterteilt wird.

## GRAPHICS 1

Der größte Teil des Bildschirms ist nun schwarz, nur im unteren Bereich befindet sich das sogenannte Textfenster.

Gibt man nun eine PRINT-Anweisung, so erscheint der Text im Textfenster. Um nun aber den Text auf den schwarzen Hintergrund zu bekommen, muß die PRINT-Anweisung um die Zeichen #4; erweitert werden.

Beispiel:

```
PRINT #4;"ICH KANN ES"
```

Der Text muß nun oben auf dem schwarzen Hintergrund in oranger Farbe sichtbar werden.

Versuchen Sie dieses jetzt einmal in der Graphikstufe 2.

## GRAPHICS 2

```
GR. 2
```

```
PRINT #6;"ICH KANN ES"
```

Sie sehen den Text wieder oben auf dem Bildschirm. Wenn Sie nun den Text der PRINT-Anweisung nach einmaligem Drücken der INVERSE-Taste eingeben, erscheint der Text in Blau auf dem Bildschirm. Der Text kann auch in Grün auf den Bildschirm gebracht werden, indem er in Kleinschrift eingegeben wird.

Beispiel:

```
PRINT #6;"BLAUER TEXT"  
PRINT #6;"gruener text"
```

Durch die Kombination der INVERSE-Taste und der Kleinschreibung kann noch eine weitere Farbe auf den Bildschirm gebracht werden. In den Kapiteln COLOR, SETCOLOR, PEEK und POKE wird noch erläutert, wie man längere Texte in beliebiger Farbe schreiben kann.



Die Graphikstufen 1 und 2 können auch ohne das Textfenster aufgerufen werden. Dies ist allerdings nur im Programmiermodus möglich. Sie erreichen dies durch Addieren der Zahl 16 zur jeweiligen Graphikstufe.

Beispiel:

```
10 GRAPHICS 2+16
20 PRINT #6;"DIES IST GRAPHIK 2"
30 PRINT #6;"ohne textfenster"
40 GOTO 40
```

Das Programm wird durch Drücken der BREAK-Taste gestoppt. Es erscheint dann wieder die Graphikstufe 0.

Um nun den Text an eine bestimmte Stelle des Bildschirms zu schreiben, muß die schon bekannte POSITION-Anwendung verwendet werden. Hierzu müssen Sie wissen, wieviele Positionen in den einzelnen Graphikstufen anwählbar sind.

	horizontal	vertikal	vertikal (ohne Textfenster)
GR. 0	0 - 39	-	0 - 23
GR. 1	0 - 19	0 - 19	0 - 23
GR. 2	0 - 19	0 - 9	0 - 11

Beispiel:

```
10 GRAPHICS 2+16
20 FOR Y=0 TO 11
30 POSITION 6,Y
40 PRINT #6;"POS. 6,";Y
50 FOR W=0 TO 100:NEXT W
60 NEXT Y
70 FOR M=0 TO 500:NEXT M
80 END
```

Beachten Sie aber, daß Sie bei der POSITION-Anwendung nicht zu große Zahlen wählen.

Beispiel:

```
10 GRAPHICS 1
20 POSITION 20,12
RUN
ERROR - 141 AT LINE 20
READY
```

In diesem Fall ist die Zahl 20 zu groß, da die Graphikstufe 1 nur Positionen von 0 bis 19 zuläßt.

## GRAPHICS 3—B

Die Graphikstufen 3 bis 8 ermöglichen das Zeichnen und Malen auf dem Bildschirm. Die Unterschiede der einzelnen Stufen bestehen in der unterschiedlichen Farbauswahlmöglichkeit und der Auflösung der Bilder. Erreicht werden die einzelnen Graphikstufen durch die gleiche Anweisung wie bei GRAPHICS 0-2.

Die Graphikstufen 3,5 und 7 erlauben es, vier verschiedene Farben gleichzeitig zu verwenden, die Stufen 4 und 6 dagegen nur zwei Farben und die Stufe 8 nur eine Farbe in zwei Helligkeitsstufen. Dafür hat die Graphikstufe 8 die höchste Auflösung, d.h. es können am meisten einzelne Bildpunkte auf dem Bildschirm angewählt werden. Zusammenfassend kann man sagen, daß jede Graphikstufe ihre Vorteile hat und es bei der Auswahl der Stufe auf das zu lösende Problem ankommt. Wenn Sie z.B. eine statistische Blockübersicht auf den Bildschirm bringen wollen, empfiehlt sich die Stufe 3, da sie nicht so viele einzelne Punkte bestimmen müssen, wie bei der Stufe 8 und außerdem mehr Farben zur Verfügung haben. Wollen Sie aber auf dem Bildschirm malen, so empfiehlt sich eine höhere Graphikstufe, weil das Bild detaillierter wird.

Damit Sie überhaupt die graphischen Möglichkeiten des ATARI-Computers nutzen können, brauchen Sie die nachfolgenden BASIC-Anweisungen.

### PLOT

Die PLOT-Anweisung (Abkürzung: PL) wird bei der Darstellung von Graphik anstelle der POSITION-Anweisung verwendet. Diese Anweisung läßt den Computer an der durch die zwei der Anweisung nachfolgenden Zahlen bestimmten Stelle des Bildschirms einen Punkt setzen.

Beispiel:

```
10 GRAPHICS 3
20 COLOR 1
30 PLOT 19,11
```

In diesem Beispiel wird ein einzelner Punkt auf die Mitte des Bildschirms gemalt. (Die Anweisung 'COLOR' wird auf den folgenden Seiten erklärt)

Genau wie bei der POSITION-Anweisung tritt bei der PLOT-Anweisung ein Fehler häufig dadurch auf, daß die Werte zu groß gewählt wurden. Wie groß der Bereich der anzuwählenden Punkte in welcher Graphikstufe ist, können Sie der an das Kapitel GRAPHICS 9—11 anschließenden Tabelle entnehmen.

## DRAWTO

Mit Hilfe der DRAWTO-Anweisung ist es möglich, Linien auf den Bildschirm zu zeichnen. Die Linie beginnt bei dem durch eine PLOT-Anweisung bestimmten Punkt und endet bei dem durch die DRAWTO-Anweisung bestimmten Punkt. Als Ausgangspunkt für eine weitere DRAWTO-Anweisung kann auch der Endpunkt der vorherigen Linie dienen.

Beispiele:

```
10 GRAPHICS 3
20 COLOR 1
30 PLOT 19,1
40 DRAWTO 34,16
50 DRAWTO 4,16
60 DRAWTO 19,1
```

Durch dieses Programm entsteht ein Dreieck auf dem Bildschirm.

Die DRAWTO-Anweisung kann durch DR abgekürzt werden.

Beispiel:

```
10 REM HAUS
20 GRAPHICS 4+16
30 COLOR 1
40 PL. 39,15
50 DR. 54,25:DR. 24,25:DR. 39,15
60 PL. 54,25
70 DR. 54,45:DR. 24,45:DR. 24,25
80 GOTO 80
```

Durch die Zeile 80 läuft das Programm endlos weiter und das Bild bleibt bestehen. Stoppen kann man das Programm mit der BREAK-Taste. Wenn man die Zeile 80 durch die folgende ersetzt,

```
80 FOR I=0 TO 750:NEXT I
```

wird das Ende des Programms verzögert. Diese beiden Möglichkeiten sind vor allem dann wichtig, wenn man das Textfenster ausschaltet, weil sonst das Bild nach Beendigung des Programms verschwindet.

Das folgende Programm ermöglicht beliebige Formen auf den Bildschirm zu zeichnen. Es macht deutlich, daß man bestimmte Figuren durch geeignete Formeln berechnen kann. (Die Anwendung POKE wird auf den folgenden Seiten erklärt).

```

10 REM KREIS IN GR.8
20 GRAPHICS 8+16
30 POKE 709,14
40 COLOR 1
50 REM KREISBERECHNUNG
60 FOR K5=1 TO 400
70 LET I=I+0.05
80 LET X=SIN(I)*50:LET Y=COS(I)*45
90 PLOT X+160,Y+85
100 NEXT K5
110 GOTO 110

```

Ein weiteres Beispiel ist die Berechnung und Zeichnung einer Ellipse.

```

10 REM ELIPSE IN GR.8
20 GRAPHICS 8+16
30 POKE 709,14
40 COLOR 1
50 REM ELIPSENBERECHNUNG
60 FOR E5=1 TO 400
70 LET I=I+0.05
80 LET X=SIN(I)*80:LET Y=COS(I)*15
90 PLOT X+160,Y+85
100 NEXT E5
110 GOTO 110

```

### SETCOLOR

In allen Graphikstufen haben Sie die Möglichkeit in verschiedenen Farben zu malen oder zu schreiben. Diese Möglichkeit haben Sie in dem Kapitel GRAPHICS 0-2 schon kennengelernt. Man kann die Farben aber noch wesentlich stärker beeinflussen.

In jeder Graphikstufe hat der Computer mehrere Farbregister zur Verfügung, deren Inhalt man verändern kann. Allerdings können nicht in jeder Stufe alle Farbregister verwendet werden.

Beispiel:

GRAPHICS 0

Farbregister	Ausgangsfarbe	Anwendung
0	—	—
1	Hellblau	Schriftfarbe
2	Dunkelblau	Hintergrundfarbe
3	—	—
4	Schwarz	Randfarbe

Will man jetzt die Randfarbe verändern, so ist z.B.

SETCOLOR 4, 2, 10 einzugeben.

Dabei gilt:

1. Zahl bestimmt das zu verändernde Farbregister. In GRO ist die Schrift-Farbe immer gleich der Hintergrundfarbe. Es läßt sich nur deren Kontrast verändern.
2. Zahl bestimmt die Farbe.
3. Zahl bestimmt die Farbhelligkeit.

Die zweite und dritte Zahl dürfen jeweils Werte von 0 bis 15 annehmen.

Farbe (Helligk. 10)	Farbnummer
Grau	0
Gold	1
Orange	2
Rot-Orange	3
Rosa	4
Purpur	5
Purpur-Blau	6
Blau	7
Blau	8
Hellblau	9
Türkis	10
Grün-Blau	11
Grün	12
Gelb-Grün	13
Orange-Grün	14
Hellorange	15

Bei den Helligkeitswerten (dritte Zahl) ist die 0 die dunkelste und die 15 die hellste Stufe.

Es ergeben sich 16 Farben in 16 Helligkeitsstufen, also 256 Farben.

Beispiel:

```
10 GR. 1+16
20 POS. 5,12
30 PRINT #6;"LIGHTSHOW"
40 FOR F=0 TO 15
50 FOR H=2 TO 10 STEP 2
60 SETCOLOR 4,F,H
70 FOR M=1 TO 80:NEXT M
80 NEXT H
90 NEXT F
100 GOTO 40
```

Ein Programm zur gleichzeitigen Darstellung aller 256 Farben ist in den Übungsbeispielen aufgeführt

Wenn keine SETCOLOR-Anweisung gegeben wird, benutzt der Computer in den einzelnen Graphikstufen immer die gleichen Ausgangsfarben.

### COLOR

Diese Anweisung dient dazu ein bestimmtes Farbregister zum Malen auszuwählen. Befindet man sich z.B. in der Graphikstufe 3, stehen vier Farbregister für das Bild zur Auswahl.

### GRAPHICS 3

Farbregister	Ausgangsfarbe	Anwendung
0	Orange	Graphikpunkte
1	Hellgrün	Graphikpunkte
2	Dunkelblau	Graphikpunkte
3	—	—
4	Schwarz	Hintergrund, Rand

Wollen Sie einer PLOT- oder DRAWTO-Anweisung eine bestimmte Farbe zuordnen, so geschieht dies durch die COLOR-Anweisung.

Beispiel:

```
10 GR. 3
20 COLOR 1
30 PLOT 10,4:DRAWTO 27,4
40 COLOR 2
50 PLOT 10,8:DRAWTO 27,8
60 COLOR 3
70 PLOT 10,12:DRAWTO 27,12
```

In diesem Beispiel sieht man, daß die mit der COLOR-Anweisung ausgewählten Farbregister nicht mit den gleichen Zahlen angewählt werden, wie bei der SETCOLOR-Anweisung. Dies ist vielleicht am Anfang etwas verwirrend, aber wenn man weiß, wie die Werte der COLOR-Anweisung zustandekommen, kein Problem mehr.

COLOR 0 wählt in jeder Graphikstufe das Register für die Hintergrundfarbe an. Die anderen Register werden der Reihenfolge nach von 1 bis 3 durchnummeriert.

### GRAPHICS 3

Farbregister	Ausgangsfarbe	COLOR-Wert
0	Orange	COLOR 1
1	Heilgrün	COLOR 2
2	Dunkelblau	COLOR 3
3	—	—
4	Schwarz	COLOR 0

Die COLOR-Anweisung kann nicht in den Graphikstufen 0—2 verwendet werden. Wird sie in den anderen Graphikstufen weggelassen, so setzt der Computer automatisch COLOR 0 (Hintergrundfarbe) als Farbe für jedes Zeichen auf dem Bildschirm.

Beispiel:

```

10 LET X=0:LET Y=0:LET M=20:LET N=12
20 FOR I=3 TO 7 STEP 2
30 GRAPHICS I+16
40 SETCOLOR 0,0,0
50 SETCOLOR 1,15,14
60 REM VIERECK
70 COLOR 1
80 PLOT 8+X,5+Y
90 DRAWTO 30+X,5+Y:DRAWTO 30+X,16+Y
100 DRAWTO 8+X,16+Y:DRAWTO 8+X,5+Y
110 REM DREIECK
120 COLOR 2
130 PLOT 19+X,7+Y
140 DRAWTO 28+X,14+Y:DRAWTO 10+X,14+Y:
DRAWTO 19+X,7+Y
150 LET X=M:LET Y=N:LET M=60:LET N=36
160 REM FARBDURCHLAUF
170 FOR F=0 TO 15
180 LET F1=15-F
190 FOR H=0 TO 14 STEP 2
200 LET H1=14-H
210 SETCOLOR 0,F1,H1
220 SETCOLOR 1,F1,H1
230 FOR WARTEN=1 TO 80:NEXT WARTEN
240 NEXT H
250 NEXT F
260 NEXT I
270 GOTO 10

```

Dieses Programm ist nur durch Drücken der BREAK-Taste zu stoppen. Es zeigt die Auflösung der drei Graphikstufen, in denen jeweils vier Farbregister zur Verfügung stehen.

**PEEK/POKE**

Wenn Sie die Farben in den Farbbregistern ändern wollten, haben Sie bisher immer die SETCOLOR-Anwendung verwendet. Es gibt auch noch eine andere Möglichkeit, die POKE-Anweisung.

Beispiel:

```
POKE 710,22
```

Sie werden feststellen, daß der Hintergrund jetzt goldfarben ist.

Bei der POKE-Anweisung steht die erste Zahl für das Farbbregister. Folgende Übersicht zeigt, welcher POKE-Wert für welches Farbbregister steht.

GRAPHICS	SETCOLOR	Anwendung	POKE-Register
0	2	Hintergrund	710
	4	Rand	712
	1	Zeichen	709
1, 2	2	Textfenster	710
	1	Zeichen im Textfenster	709
	4	Hintergrund	712
	0	Zeichen	708
3, 5, 7	4	Hintergrund	712
	2	Textfenster	710
	1	Zeichen im Textfenster	709
	0	Graphikpunkte	708
4, 6	4	Hintergrund	712
	0	Graphikpunkte	708
	1	Zeichen im Fenster	709
8	2	Textfenster	710
	1	Hintergrund	710
	4	Graphikpunkt	709
		Rand	712

Die zweite Zahl der POKE-Anweisung steht für die Farbe und ihren Helligkeitswert. Es gibt 16 Farben in 16 verschiedenen Helligkeitsstufen. Das ergibt zusammen 256 Farben.

Zur Berechnung des POKE-Wertes für eine bestimmte Farbe multiplizieren Sie die gewünschte Farbnummer mit der Zahl 16 und addieren Sie den Wert der gewünschten Helligkeitsstufe dazu.

Beispiel:

Hintergrundfarbe in Hellblau (Farbnummer 9) in der Helligkeitsstufe 11

$9 \cdot 16 + 11 = 155$

```
POKE 710,155
```

Am Besten schauen Sie sich die einzelnen Farben und ihre Werte einmal mit Hilfe des folgenden Programms an.

```
10 GRAPHICS 1
20 FOR K=0 TO 255
30 PRINT K
40 POKE 712,K
50 FOR M=1 TO 50:NEXT M
60 NEXT K
```

Sie können dieses Programm an jeder Stelle stoppen, indem Sie die CONTROL-Taste gedrückt halten und zusätzlich die Taste L drücken. Wollen Sie das Programm fortsetzen, so drücken Sie beide Tasten nochmals.

Die PEEK-Anweisung ist das Gegenstück zur POKE-Anweisung. Sie verändert kein Register, sondern fragt den aktuellen Wert eines Registers ab.

Im folgenden Beispiel wird nach dem aktuellen Farbwert des Hintergrundregisters in Graphikstufe 0 gefragt (PRINT steht dabei zur Darstellung auf dem Bildschirm)

```
PRINT PEEK(710)
148
```

READY

Es gibt nun auch noch andere Möglichkeiten der POKE/PEEK-Anweisungen. So existieren bestimmte Register, um z.B. die Randbreite zu verändern, oder um den Programmrecorder (falls vorhanden) zu steuern. Außerdem können über bestimmte Register einzelne Tasten abgefragt werden.

POKE 82.X	linker Rand. X gibt die Randbreite an
POKE 83.X	rechter Rand. X gibt die Randbreite an
POKE 755.4	kehrt den Text um und läßt den Cursor verschwinden
POKE 54018.52	startet den Programmrecorder, wenn die PLAY-Taste gedrückt ist

Weitere interessante Speicherstellen sind im Anhang aufgeführt.

**GRAPHICS 9-11**

Diese drei Graphikstufen bieten besonders große Farbmöglichkeiten bei guter Bildauflösung. Alle drei Stufen haben kein Textfenster und sind nur für Graphik gedacht.

**GRAPHICS 9**

Diese Stufe erlaubt es, eine Farbe in 16 Helligkeitsstufen gleichzeitig auf den Bildschirm zu bringen. Die SETCOLOR-Anweisung sieht folgendermaßen aus:

SETCOLOR 4,	(0-15),	0
fester Wert	wahlweise	fester Wert
	von 0-15	

Um die einzelnen Helligkeitswerte zu erreichen, benutzt man die COLOR-Anweisung mit Werten von 0 bis 15.

**GRAPHICS 10**

In dieser Stufe können neun verschiedene Farben in unterschiedlicher Helligkeit auf den Bildschirm gebracht werden. Es ist allerdings etwas schwierig die Farbregister zu füllen und sie später auszuwählen. Dabei soll die folgende Tabelle helfen.

Ausgangs- farbe		POKE- Register
COLOR 0 Schwarz		704
COLOR 1 Schwarz		705
COLOR 2 Schwarz		706
COLOR 3 Schwarz		707
COLOR 4 Orange	SETCOLOR 0,**	708
COLOR 5 Hellgrün	SETCOLOR 1,**	709
COLOR 6 Hellblau	SETCOLOR 2,**	710
COLOR 7 Orange	SETCOLOR 3,**	711
COLOR 8 Schwarz	SETCOLOR 4,**	712

\* = beliebige Werte lt. Tabelle für Farbe und Helligkeit.

**GRAPHICS 11**

In dieser Stufe können Sie 16 verschiedene Farben gleichzeitig auf den Bildschirm bringen, allerdings alle in einer Helligkeitsstufe. Ausgewählt werden die Farben mit der COLOR-Anweisung, wobei kein Farbregister sondern die Farbe selbst angewählt wird.

COLOR<0—15>

Der Helligkeitswert, der für alle Farben gilt, wird mit der SETCOLOR-Anweisung verändert.

SETCOLOR 4.0.<0—15>

**GRAPHICS 12**

In dieser Stufe können Sie Textzeichen in der Größe der Graphics 0 Zeichen mit 5 Farben gleichzeitig auf den Bildschirm bringen.

Diese Zeichen müssen jedoch vorher speziell definiert werden, da der normale Zeichensatz keine Farbinformationen enthält.

**GRAPHICS 13**

Diese Stufe erlaubt die Darstellung von Zeichen ähnlich Graphics 12, jedoch mit doppelter Höhe.

**GRAPHICS 14**

Dieser Bildschirmmodus erlaubt die Darstellung hochauflösender Graphiken (160 x 192 Punkte) in zwei Farben.

**GRAPHICS 15**

Hier zeigt der ATARI so richtig, welche grafischen Möglichkeiten in ihm stecken. In Graphik 15 zeichnen Sie mit einer Auflösung von 160 x 192 Bildpunkten in 4 Farben.

**Tabelle der Graphikstufen**

GR.	Anwen- dung	Hori- zontal	Verti- kal	Verti- kal (ohne Text)	An- zahl der Farben
0	Text	40	—	24	2
1	Text	20	20	24	5
2	Text	20	10	12	5
3	Graphik	40	20	24	4
4	Graphik	80	40	48	2
5	Graphik	80	40	48	4
6	Graphik	160	80	96	2
7	Graphik	160	80	96	4
8	Graphik	320	160	192	1/2
9	Graphik	80	—	192	1/16
10	Graphik	80	—	192	9
11	Graphik	80	—	192	16
12	Graphik	40	20	24	5
13	Graphik	40	10	12	5
14	Graphik	160	160	192	2
15	Graphik	160	160	192	4

Die in der Tabelle verwendeten Ausdrücke 1/2 und 1/16 bedeuten, daß eine Farbe in zwei bzw. 16 Helligkeiten benutzt werden kann.



## Abfrage der Steuerungen

### Joystick / Steuerknüppel

Der Joystick ist ein einfach zu bedienendes Eingabegerät und darum besonders für Videospiele geeignet. Um den Joystick in eigenen Programmen verwenden zu können, sind im ATARI-Basic zwei Anweisungen vorgesehen, um die Werte des Steuerknüppels abzufragen.

### STICK

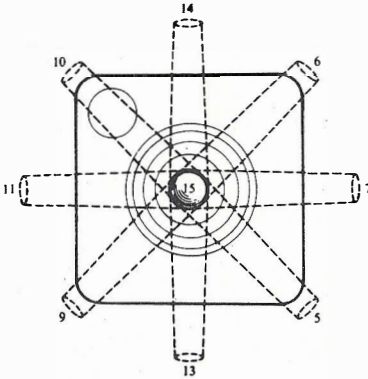
Es können zwei Joysticks an den ATARI-Computer angeschlossen werden. Diese können mit der STICK-Anweisung abgefragt werden.

Der Joystick in der linken Buchse wird mit STICK(O) abgefragt, der Joystick in der rechten Buchse mit STICK(1).

Beispiel:

```
10 PRINT STICK(0)
20 GOTO 10
```

Beim Anwenden dieses Beispiels sehen Sie, daß das Register des Joysticks neun Werte annehmen kann.



Das folgende Programm macht deutlich, welche Werte zu welcher Position gehören.

Beispiel:

```
10 GRAPHICS 2+16
20 ST=STICK(0)
50 IF ST=14 THEN POSITION 9,1:GOTO 140
60 IF ST=13 THEN POSITION 9,9:GOTO 140
70 IF ST=11 THEN POSITION 4,5:GOTO 140
80 IF ST=7 THEN POSITION 14,5:GOTO 140
90 IF ST=10 THEN POSITION 4,1:GOTO 140
100 IF ST=9 THEN POSITION 4,9:GOTO 140
110 IF ST=5 THEN POSITION 14,9:GOTO 140
120 IF ST=6 THEN POSITION 14,1:GOTO 140
130 IF ST=15 THEN POSITION 9,5:GOTO 140
140 PRINT ST;ST
150 FOR M=0 TO 300:NEXT M
160 GOTO 10
```

### STRIG

Mit dieser Anweisung läßt sich der 'Feuerknopf' (Trigger) des Joysticks abfragen. Genau wie bei der STICK-Anweisung werden die Knöpfe der beiden möglichen Joysticks über

STRIG(O)  
und  
STRIG(1)

abgefragt. Bei der Abfrage können die Werte 0 und 1 im Register stehen. Der Wert 0 steht für den gedruckten Zustand, der Wert 1 für den Ruhezustand.

Beispiel:

```
10 PRINT STRIG(0)
20 GOTO 10
```

Das folgende Programm zeigt, wie man mit dem Joystick auf dem Bildschirm zeichnen kann. Bei gedrücktem Knopf läßt sich eine Linie zeichnen. Ansonsten wird die Linie in der Hintergrundfarbe gezeichnet und ist damit also nicht zu sehen. Diese Funktion kann deshalb also auch zum Korrigieren verwendet werden.

```

10 GRAPHICS 7+16: X=80: Y=40
20 ST=STICK(0)
30 IF ST=7 THEN X=X+1
40 IF ST=6 THEN X=X+1: Y=Y-1
50 IF ST=14 THEN Y=Y-1
60 IF ST=5 THEN X=X+1: Y=Y+1
70 IF ST=11 THEN X=X-1
80 IF ST=10 THEN X=X-1: Y=Y-1
90 IF ST=13 THEN Y=Y+1
100 IF ST=9 THEN X=X-1: Y=Y+1
110 IF X>159 THEN X=159
120 IF Y>95 THEN Y=95
130 IF X<0 THEN X=0
140 IF Y<0 THEN Y=0
150 COLOR 1
160 PLOT X, Y
170 FOR W=0 TO 20: NEXT W
180 COLOR 0
190 IF STRIG(0)=0 THEN COLOR 1
200 PLOT X, Y
210 GOTO 20

```

### Paddle Controller / Drehregler

Dieses Gerät ist wie der Joystick ein Eingabegerät. Es hat den Vorteil, daß es mehr als neun Werte an den Computer weitergeben kann. Je nach Stellung des Paddle Controllers ergibt sich eine Zahl zwischen 1 und 228.

Beispiel:

```

10 PRINT PADDLE(0)
20 GOTO 10

```

Daraus ergeben sich viele Programmiermöglichkeiten, z.B. für ein einfaches Musikprogramm.

```

10 LET PA=PADDLE(0)
20 SOUND 0, PA, 10, 10
30 GOTO 10

```

Der "Feuerknopf" des Paddle Controllers kann wie der des Joysticks die Werte 0 und 1 im Register stehen haben. Der Wert wird über die PTRIG-Anweisung abgefragt.

Beispiel:

```

10 SETCOLOR 2, PA1, 2: COLOR 1
20 LET PA=PADDLE(0): LET PA1=PA/10
30 POKE 82, PA1+7: PRINT PA
40 SOUND 0, PA, 10, 10
50 IF PTRIG(0)=0 THEN SOUND 0, PA, 6, PA1
60 GOTO 10

```



**BASIC**

**NOTIZEN**

### Übungsbeispiele

Die folgenden Beispiele (Listings) sollen Ihnen helfen, die im vorangegangenen BASIC-Kurs erworbenen Kenntnisse zu vertiefen. Wir empfehlen, die Programme vollständig abzulippen und dann zu variieren. Bitte berücksichtigen Sie dabei die jeweiligen Vermerke.

Alle Beispiele sind so abgedruckt, wie sie auf dem Bildschirm erscheinen. Nach jeder Programmzeile ist die RETURN-Taste zu drücken. Der Syntax-Check (Meldung: ERROR) zeigt Ihnen sofort, ob eine nicht zulässige Eingabe erfolgte. Bitte ändern Sie die Zeile dann entsprechend.

Programmstart: Durch Eingabe von RUN (RETURN)

Abbruch:       Durch Drücken der BREAK-Taste.  
                  Erneuter Start durch Eingabe von  
                  RUN  
                  Erneutes Listen durch Eingabe von  
                  LIST

Auch bei Drücken der RESET-Taste bleibt das Programm im RAM-Speicher und kann wie bei einem Programm-Abbruch neu gestartet oder gelistet werden.

Mit einem ATARI Programm-Recorder oder einer ATARI Disketten-Station können Sie alle Programme für weitere Übungen speichern und bei Bedarf neu laden. Stehen Ihnen diese Datenspeicher nicht zur Verfügung, so wird das gespeicherte Programm durch die Neueingabe überschrieben. Geben Sie dann zur Sicherheit zuvor NEW (RETURN) ein, damit je nach Programmlänge nicht alte Programmzeilen mit ins neue Programm übernommen werden und den Ablauf stören.



### Beispiel 1

```

10 REM *** SIRENE 1 ***
20 GRAPHICS 18
30 POSITION 4,6:PRINT #6;"SIRENE "
"
40 FOR I=80 TO 180
50 SOUND 0,I,10,10:SETCOLOR 0,INT(I/10)
,8
60 NEXT I:SETCOLOR 2,0,0
70 FOR I=180 TO 80 STEP -1
80 SOUND 0,I,10,10:SETCOLOR 0,INT(I/10)
,8
90 NEXT I:SETCOLOR 2,10,10
100 GOTO 40

```

Diese Werte (0 bis 255) bestimmen die aufsteigende Tonhöhe.

Diese Werte (0 bis 255) bestimmen die absteigende Tonhöhe.

### Beispiel 2

```

10 REM *** SIRENE 2 ***
20 GRAPHICS 0:? "LAENGE":INPUT AA:GRA
PHICS 18
30 POSITION 4,6:PRINT #6;"SIRENE "
"
40 FOR I=1 TO AA:NEXT I
50 SOUND 0,80,10,10:SETCOLOR 0,8,8
60 SOUND 1,60,10,10:SETCOLOR 0,8,8
70 SETCOLOR 2,0,0
80 FOR I=1 TO AA:NEXT I
90 SOUND 0,120,10,10:SETCOLOR 0,2,8
100 SOUND 1,100,10,10:SETCOLOR 0,2,8
110 SETCOLOR 2,10,10
120 GOTO 40

```

Vor Ausführung werden Sie zur Eingabe eines Wertes (0 bis 255) aufgefordert.

### Beispiel 3

```

10 REM *** TUERKLINGEL ***
20 GRAPHICS 18
30 POSITION 1,6:PRINT #6;"TUERKLING
EL "
"
40 FOR I=1000 TO 200 STEP -4
50 IF I<=1000 AND I>700 THEN SOUND 0,1
50,10,INT((I-700)/20)
60 IF I<=800 AND I>500 THEN SOUND 1,11
0,10,10,INT((I-500)/20)
70 IF I<=600 AND I>300 THEN SOUND 2,90
,10,INT((I-300)/20)
80 IF I<=1000 AND I>800 THEN SETCOLOR
2,4,8
90 IF I<=800 AND I>600 THEN SETCOLOR 2
,8,8
100 IF I<=600 AND I>400 THEN SETCOLOR
2,10,8
110 IF I<=400 AND I>200 THEN SETCOLOR
2,0,0
120 SETCOLOR 0,INT(I/35),8
130 NEXT I
140 GOTO 40

```

Diese Werte (-20 bis -1) bestimmen die Schnelligkeit des Tonwechsels.

Die Werte (1 bis 15) bestimmen die Farbe der Sternchen.

### Beispiel 4

```

10 REM *** SPIELEN EINER NOTENFOLGE ***
*
20 DIM A(10)
30 GRAPHICS 0:PRINT "BITTE GEBEN SIE Z
EHN ZAHLEN EIN":? I:
40 FOR I=1 TO 10:PRINT "ZAHL ";I;" ";
:INPUT B:A(CI)=B:NEXT I
50 GRAPHICS 0:POSITION 15,12:? "ICH SP
IELE"
60 FOR I=1 TO 10:SOUND 0,A(CI),10,15
70 FOR J=1 TO 200:NEXT J
80 NEXT I
90 SOUND 0,0,0,0:GOTO 30

```

Hier können Sie die Anzahl der Töne (Werte) ver-  
ändern.

Vor Ausführung werden Sie zur Eingabe von zehn  
Werten gefragt.

### Beispiel 5

```

10 REM *** MUSIK PER TASTATUR ***
20 GRAPHICS 18
30 POSITION 7,S:? H6:"MUSIK"
40 OPEN H1,4,0,"K:"
50 FOR I=1 TO 10000
60 GET H1,A:A=A-65:A=A*10+5
70 SOUND 1,A,10,15:POKE 708,A:NEXT I
80 CLOSE H1

```

Ton und Hintergrundfarbe ändern sich je nach ge-  
drückter Buchstaben-Taste (A bis Z). Andere Tasten  
führen zum Programm-Abbruch.

### Beispiel 6

```

10 REM *** GRAPHIK-FAECHER ***
20 GRAPHICS 11:MM=0
30 FOR Z=1 TO 2:FOR ZE=0 TO 15
40 COLOR Z:MM=MM+5
50 PLOT 0,0:DRAWTO 79,MM
60 NEXT ZE
70 NEXT Z:MM=0
80 FOR Z=1 TO 2:FOR ZE=0 TO 15
90 COLOR Z:MM=MM+5
100 PLOT 79,0:DRAWTO 0,MM
110 NEXT ZE
120 NEXT Z
130 GOTO 170
140 FOR H=0 TO 14:SETCOLOR 4,0,H
150 FOR M=1 TO 100:NEXT M:NEXT H
160 GOTO 140
170 MM=191:FOR Z=1 TO 2:FOR ZE=0 TO 15
180 COLOR Z:MM=MM-5
190 PLOT 79,191:DRAWTO 0,MM
200 NEXT ZE
210 NEXT Z
220 MM=191:FOR Z=1 TO 2:FOR ZE=0 TO 15
230 COLOR Z:MM=MM-5
240 PLOT 0,191:DRAWTO 79,MM
250 NEXT ZE
260 NEXT Z
270 FOR H=0 TO 14 STEP 2:SETCOLOR 4,0,
H:FOR S5=1 TO 10:NEXT S5:NEXT H
280 FOR H=14 TO 0 STEP -2:SETCOLOR 4,0
,H:FOR S5=1 TO 10:NEXT S5:NEXT H
290 GOTO 270

```

Graphikstufe 10 oder 11.

► Paarweise Veränderung von +1 bis -5.

► Paarweise Veränderung von +1 bis -5.

## Beispiel 7

```

10 REM *** ADRESSEN-MASKE ***
20 OPEN H1,4,0,"K":OPEN H2,12,0,"S:"
30 DIM AS(24)
40 GRAPHICS 0:POKE 710,0
50 POSITION 2,2:?"NAME : _____
   "
60 POSITION 2,6:?"STRASSE : _____
   "
70 POSITION 2,10:?"WOHNORT : _____
   "
80 POSITION 2,14:?"ALTER : ____"
90 POSITION 15,14:?"BERUF : _____
   "
100 POSITION 10,2:?" ":"
110 AS(1)=" ":"AS(24)=" ":"AS(2)=AS
120 B=2
130 FOR I=1 TO 3
140 FOR K=1 TO 24
150 GET H1,A:?" H2;CHRS(A);
160 IF A=155 THEN POSITION 11,B:?" AS:
POP :GOTO 190
170 AS(K,K)=CHRS(A)
180 NEXT K
190 AS(1)=" ":"AS(24)=" ":"AS(2)=AS
200 IF I=1 THEN B=6:POSITION 10,B:?" ":"
   "
210 IF I=2 THEN B=10:POSITION 10,B:?" ":"
   "
220 NEXT I
230 POSITION 10,14:?" ":"
240 AS="":AS=" "
250 FOR K=1 TO 2
260 GET H1,A:?" H2;CHRS(A);
270 IF A=155 THEN POSITION 11,14:?" AS:
:POP :GOTO 300
280 AS(K,K)=CHRS(A)
290 NEXT K
300 POSITION 21,14:?" ":"
310 AS="":AS(1)=" ":"AS(13)=" ":"AS(2)=A
S
320 FOR K=1 TO 13
330 GET H1,A:?" H2;CHRS(A);
340 IF A=155 THEN POSITION 22,14:?" AS:
:POP :GOTO 370
350 AS(K,K)=CHRS(A)
360 NEXT K
370 POKE 752,1
380 ? :? :? :?
400 OPEN H3,8,0,"P":GOTO 420
410 FOR I=1 TO 40:GET H2,A:?" H3;CHRS(A
):NEXT I:?" H3," ":RETURN
420 POSITION 0,2:GOSUB 410
430 POSITION 0,6:GOSUB 410
440 POSITION 0,10:GOSUB 410
450 POSITION 0,14:GOSUB 410
600 CLOSE H1:CLOSE H2:CLOSE H3
610 POKE 752,0

```

Für den Aufbau einer Adressenliste ist dieses Programm nicht vorgesehen. Dazu wären umfangreiche Veränderungen und Programmier Erfahrung erforderlich.

Falls kein Drucker vorhanden ist, kann das Programm bereits in Zeile 360 beendet werden. Dann ist einzugeben: 360 NEXT K:End

Die Befehle OPEN in Zeile 20 und 400 sowie CLOSE in Zeile 600 werden zum Datenaustausch mit einem Peripheriegerät genutzt. Ihre ausführliche Besprechung würde den Rahmen dieses BASIC-Kurses überschreiten. Wir verweisen deshalb auf die Literaturempfehlungen.

**Beispiel B**

```

10 REM ** HOCHAUFLOESENDE GRAPHIK **
20 GRAPHICS 24:POKE 710,0:COLOR 1
30 POSITION 40,40
40 FOR I=0 TO 1 STEP 0.01
50 ZEI=6.283*I
60 PLOT 160+45*SIN(ZEI),96+45*COS(ZEI)
:DRAWTO 160+90*SIN(ZEI+0.01*6.283),96+
90*COS(ZEI+0.01*6.283)
70 NEXT I
80 FOR I=0 TO 1 STEP 0.01
90 ZEI=6.283*I
100 PLOT 160+90*SIN(ZEI),96+90*COS(ZEI)
:DRAWTO 160+90*SIN(ZEI+0.01*6.283),96
+90*COS(ZEI+0.01*6.283)
110 NEXT I
120 FOR I=0 TO 1 STEP 0.01
130 ZEI=6.283*I
140 PLOT 160,96:DRAWTO 160+90*SIN(ZEI),
96+90*COS(ZEI)
150 NEXT I
160 GOTO 80

```

Der Wert kann 0 bis 255 betragen. Angleichung an Bildschirmfarbe ist möglich.

Die Werte können von 0.001 bis 0.25 ansteigen, müssen aber innerhalb jeder Gruppe (Zeile 40/60 und Zeile 80/100 und Zeile 120) gleich sein.

### Beispiel 9

```

10 REM ** BALLSPIEL/PING-PONG **
20 GRAPHICS 5:SETCOLOR 2,0,0
30 COLOR 1:PLOT 0,0:DRANTO 79,0:DRANTO
79,39:DRANTO 0,39:DRANTO 0,0
40 POKE 752,1:? "RSCORE ";P;" Z
EIT ";
50 POKE 559,46
60 POKE 704,188
70 I=PEEK(106)-8
80 POKE 54279,I
90 POKE 53277,3
100 POKE 53256,1
110 PMBASE=IX*256
120 FOR I=PMBASE+512 TO PMBASE+640:POK
E I,0:NEXT I
130 POKE PMBASE+512+85,60:POKE PMBASE+
513+85,60
140 FOR I=PMBASE+384 TO PMBASE+512:POK
E I,0:NEXT I
150 POKE PMBASE+384,3:5=999
160 YV=INT(CRND(0)+2):XV=INT(-4*MRND(0
)+4)
170 YM=19:XM=120
180 ST=STICK(0):X=X+4*(ST=7)+2*(X<195
):X=X-4*(ST=11)+2*(X>45)
190 POKE 53248,X
200 POKE 53278,0
210 POKE PMBASE+384+YM,0:A=A+1:5=5-I
220 IF 5<10 THEN ? "P";
230 IF 5<100 THEN ? "P";
240 POKE 752,1:? " 4444";5;"4444";
250 XM=XM+XV:YM=YM+YV:YM=YM+YV:IF XM<50
OR XM>205 THEN XV=-XV:5OUND 0,20,10,15
:5OUND 0,0,0,0:GOTO 210
260 IF 5=0 THEN 340
270 IF YM<19 OR YM>92 THEN YV=-YV:5OUN
D 0,20,10,15:5OUND 0,0,0,0:GOTO 210
280 POKE PMBASE+384+YM,3:POKE 53252,XM
290 DF=0
300 IF PEEK(53256)<>0 AND A>3 THEN POK
E 752,2:P=P+1:? "RSCORE ";P;" Z
EIT ";:YV=-YV:XV=-XV:DF=1
310 IF DF=1 THEN POKE 53278,0:A=0:GOTO
330
320 GOTO 180
330 XV=INT(-4*MRND(0)+2):GOTO 210
340 POKE 53277,0:POKE 53265,0:POKE 532
61,0
350 GRAPHICS 18
360 POSITION 2,3:? B6;"punkte : ";P
370 POSITION 2,9:? B6;"STICKE"=neues sp
iel"
380 POSITION 2,11:? B6;"SELEKTI"=ende"
390 IF PEEK(53279)=6 THEN CLR :GOTO 20
400 IF PEEK(53279)=5 THEN GRAPHICS 0:E
ND
410 GOTO 390

```

Ein Programm in PLAYER/MISSILE-Technik. Zur Bewegung des Schlägers ist ein Steuerknüppel erforderlich.

Zum Programm-Abbruch ist nicht die BREAK-, sondern die RESET-Taste zu drücken, da sonst bei weiteren Eingaben am linken Bildschirmrand ein durchlaufender grüner Streifen sichtbar bleibt.

Für das Zeichen "↵" bitte die Taste ESC drücken, dann gleichzeitig CONTROL und CLEAR.



### Beispiel 10

```

10 REM *** GRAPHIK-KREISE I ***
20 GRAPHICS 31:COLOR 1
30 POKE 708,91:POKE 709,181:POKE 710,1
31
40 A=20:B=33
50 FOR K=1 TO 5
60 FOR I=0 TO 1 STEP 0.03
70 ZEI=6.283*I
80 PLOT A,B:DRAWTO A+15*SIN(ZEI+0.01*6
.283),B+30*ICOS(ZEI+0.01*6.283)
90 NEXT I
100 IF K=1 THEN A=50:B=33:COLOR 2
110 IF K=2 THEN A=80:B=33:COLOR 3
120 IF K=3 THEN A=110:B=33:COLOR 1
130 IF K=4 THEN A=140:B=33:COLOR 2
140 NEXT K
150 A=20:B=129:COLOR 3
160 FOR K=1 TO 5
170 FOR I=0 TO 1 STEP 0.03
180 ZEI=6.283*I
190 PLOT A,B:DRAWTO A+15*SIN(ZEI+0.01*
6.283),B+30*ICOS(ZEI+0.01*6.283)
200 NEXT I
210 IF K=1 THEN A=50:B=129:COLOR 1
220 IF K=2 THEN A=80:B=129:COLOR 2
230 IF K=3 THEN A=110:B=129:COLOR 3
240 IF K=4 THEN A=140:B=129:COLOR 1
250 NEXT K
260 A=35:B=81:COLOR 2
270 FOR K=1 TO 4
280 FOR I=0 TO 1 STEP 0.03
290 ZEI=6.283*I
300 PLOT A,B:DRAWTO A+15*SIN(ZEI+0.01*
6.283),B+30*ICOS(ZEI+0.01*6.283)
310 NEXT I
320 IF K=1 THEN A=65:B=81:COLOR 3
330 IF K=2 THEN A=95:B=81:COLOR 1
340 IF K=3 THEN A=125:B=81:COLOR 2
350 NEXT K
360 IF A=1 THEN POKE 708,INT(RND(0)*25
5)
370 IF A=2 THEN POKE 709,INT(RND(0)*25
5)
380 IF A=2 THEN POKE 710,INT(RND(0)*25
5)
390 FOR U=1 TO 20:NEXT U
400 A=INT(RND(0)*3+1.)
410 GOTO 360

```

Hier wird der Farbwechsel durch POKE-Befehle erreicht.

Mit Werten von 1 bis 1000 wird die Schnelligkeit des Farbwechsels beeinflusst.



## Beispiel 11

```

10 REM *** GRAPHIK-KREISE II ***
20 GRAPHICS 31:COLOR 1
30 SETCOLOR 0,3,10:SETCOLOR 1,8,10:SET
COLOR 2,11,10
40 A=20:B=33
50 FOR K=1 TO 5
60 FOR I=0 TO 1 STEP 0.03
70 ZEI=6.283*I
80 PLOT A,B:DRANTO A+15*SIN(ZEI+0.01*6
.283),B+30*COS(ZEI+0.01*6.283)
90 NEXT I
100 IF K=1 THEN A=50:B=33:COLOR 2
110 IF K=2 THEN A=80:B=33:COLOR 3
120 IF K=3 THEN A=110:B=33:COLOR 1
130 IF K=4 THEN A=140:B=33:COLOR 2
140 NEXT K
150 A=20:B=129:COLOR 3
160 FOR K=1 TO 5
170 FOR I=0 TO 1 STEP 0.03
180 ZEI=6.283*I
190 PLOT A,B:DRANTO A+15*SIN(ZEI+0.01*
6.283),B+30*COS(ZEI+0.01*6.283)
200 NEXT I
210 IF K=1 THEN A=50:B=129:COLOR 1
220 IF K=2 THEN A=80:B=129:COLOR 2
230 IF K=3 THEN A=110:B=129:COLOR 3
240 IF K=4 THEN A=140:B=129:COLOR 1
250 NEXT K
260 A=35:B=81:COLOR 2
270 FOR K=1 TO 4
280 FOR I=0 TO 1 STEP 0.03
290 ZEI=6.283*I
300 PLOT A,B:DRANTO A+15*SIN(ZEI+0.01*
6.283),B+30*COS(ZEI+0.01*6.283)
310 NEXT I
320 IF K=1 THEN A=65:B=81:COLOR 3
330 IF K=2 THEN A=95:B=81:COLOR 1
340 IF K=3 THEN A=125:B=81:COLOR 2
350 NEXT K
360 IF A=1 THEN SETCOLOR 0,3,INT(RND(0
)*15)
370 IF A=2 THEN SETCOLOR 1,3,INT(RND(0
)*15)
380 IF A=2 THEN SETCOLOR 2,3,INT(RND(0
)*15)
390 FOR U=1 TO 20:NEXT U
400 A=INT(RND(0)*3+1)
410 GOTO 360

```

Ähnlich Beispiel 10, nur wird hier der Farbwechsel nicht durch POKE sondern SETCOLOR erzielt.

Beliebige Werte von 1 bis 3 möglich.

Beliebige Werte von 1 bis 15 möglich.

**Beispiel 12**

```

10 REM ** STREICHHOLZ-SPIEL **
20 DIM A$(30):GRAPHICS 7
30 SETCOLOR 0,1,5:SETCOLOR 1,3,1:SETCO
LOR 2,0,6:SETCOLOR 4,0,6
40 A=1$7:B=10:C=30:D=33:RESTORE
50 K=1:E=A:F=B:G=C:ANDE=21:ANFANG=1:GO
SUB 360
60 K=2:E=A:F=C:G=D:ANDE=21:ANFANG=1:GO
SUB 360
70 FOR I=1 TO 6
80 A$="":READ A$:LANG=LEN(A$)
90 FOR J=1 TO LANG:FOR K=1 TO 50:NEXT
K
100 ? A$(J,J):NEXT J: ? : ?
110 NEXT I:N=21
120 ? : ? "WIEVIELE STREICHHOELZER " ;
130 INPUT STREICH
140 IF STREICH<0 THEN STREICH=0:IF STREICH>4 THEN ? : ?
? "OHNE MICH, SO GEHT ES JA NICHT." : ? :
GOTO 120
160 M=N-STREICH
170 K=3:E=A:F=B:G=D:ANDE=21-M:ANFANG=1
:GOSUB 360
180 STREICH=5-STREICH
190 FOR I=1 TO 300:NEXT I
200 ? : ? "ICH NEHME " ;STREICH
210 M=N-STREICH
220 K=3:E=A:F=B:G=D:ANDE=21-M:ANFANG=1
:GOSUB 360
230 IF M>1 THEN 120
240 ? "K *** ICH HABE GEWONNEN ***"
250 ? : ? "NOCH EIN SPIEL (J,N) " ;
260 A$="":INPUT A$
270 IF A$="J" THEN ? "K":GOTO 30
280 GRAPHICS 0:END
290 ? : ? " HERZLICHEN GLUECKWUNSCH"
300 DATA DAS SPIEL BEGINNT !!!!!
310 DATA WIR HABEN 21 STREICHHOELZER
320 DATA WER DAS LETZTE HOLZ NIMMT
330 DATA HAT VERLOHREN
340 DATA WAELLEN SIE ZWISCHEN 1 UND 4
HOELZERN
350 DATA SIE DUERFEN BEGINNEN
360 COLOR X
370 FOR I=ANFANG TO ANDE
380 PLOT E-7* I,F:DRANTO E-7* I,G
390 NEXT I
400 RETURN

```

Diese Werte bestimmen die Länge des Streichholzes.



### Beispiel 13

```

10 REM *** ATARI MISCHFARBEN ***
20 GRAPHICS 11
30 FOR X=0 TO 79 STEP 5
40 FOR Y=0 TO 191 STEP 12
50 FOR I=0 TO 9 STEP 2
60 COLOR X/5
70 PLOT X,Y+I:DRAWTO X+3,Y+I
80 COLOR Y/12
90 PLOT X,Y+1+I:DRAWTO X+3,Y+1+I
100 NEXT I
110 NEXT Y:NEXT X
120 GOTO 120

```

### Beispiel 14

```

10 REM ** ATARI BRINGT 256 FARBEN **
20 GRAPHICS 9
30 FOR A=0 TO 79:COLOR INT(A/5)
40 PLOT A,4:DRAWTO A,191:NEXT A
50 FOR A=1536 TO 1562:READ B:POKE A,B:
NEXT A:D=PEEK(560)+256*PEEK(561)
60 FOR A=0 TO 14:READ B:POKE D+B,143:N
EXT A
70 POKE 161,0:POKE 512,0:POKE 513,6:P
OKE 54286,192
80 GOTO 80
90 DATA 72,173,80,6,24,105,16,141,80,6
,141,10,212,141,26,208,201,240,208,5,1
69,0,141,80,6,104,64
100 DATA 17,29,41,53,65,77,89,104,116,
128,140,152,164,176,188
110 GOTO 110

```

Hier zeigt Ihr ATARI Computer alle möglichen 256 Farben.

### Beispiel 15

```

10 REM ** ATARI REGENBOGEN-FARBEN **
20 DIM C$(24)
30 SETCOLOR 2,0,0:POKE 752,1:? "K"
40 FOR I=1 TO 24
50 READ D
60 C$(I,1)=CHR$(D)
70 NEXT I
80 D=USR(CADR(C$))
90 END
100 DATA 162,0,173,11,212,201,32,208,2
49,141
110 DATA 10,212,142,24,208,232,232,208
,246,142
120 DATA 24,208,240,232

```

Das Programm kann nur mit der RESET-Taste unterbrochen werden.



---

**NOTIZEN**



## FEHLERMELDUNGEN UND ERKLÄRUNGEN

Beim ATARI-Computer wird, wenn ein Fehler auftritt, eine Kodezahl auf dem Bildschirm ausgegeben. Die Bedeutung dieser Kodezahlen ist im folgenden Abschnitt erläutert.

### 2 NICHT GENUG SPEICHERPLATZ VORHANDEN

Es ist nicht genug Speicherplatz zur Speicherung des Programmes oder der Variablen vorhanden, oder es sind zu viele FOR-NEXT-Schleifen oder Unteiprogramme ineinander verschachtelt worden.

### 3 FALSCHER WERT

Ein numerischer Wert ist zu groß, zu klein oder an einer Stelle, wo er positiv sein sollte, negativ.

### 4 ZU VIELE VARIABLEN

In einem Programm dürfen nicht mehr als 128 verschiedene Variablen verwendet werden. Diese Grenze wird unter Umständen auch deshalb überschritten, weil zu der Gesamtzahl der Variablen auch einige früher benutzte Variablen zählen, die inzwischen jedoch nicht mehr gebraucht werden.

### 5 LÄNGE EINER ALPHANUMERISCHEN VARIABLEN ÜBERSCHRITTEN

Es ist ein Teil einer alphanumerischen Variablen angesprochen worden, für den diese Variable nicht dimensioniert wurde.

### 6 ZU WENIG DATA-WERTE

Es wurde versucht, mit Hilfe von READ-Befehlen mehr Werte zu lesen, als bei DATA-Befehlen angegeben waren.

### 7 ZAHLENWERT GRÖßER ALS 32767

Ein numerischer Wert ist an einer Stelle, wo er es nicht sein dürfte, größer als 32767 oder kleiner als 0.

### 8 INPUT-FEHLER

Bei der Ausführung eines INPUT-Befehls wurde festgestellt, daß der angegebene Variablentyp und der Wert, der dieser Variablen zugewiesen werden soll, nicht zueinander passen. Eine numerische Variable kann zum Beispiel keine Buchstaben, Satzzeichen oder Grafikzeichen enthalten.

### 9 DIMENSIONIERUNGSFEHLER

Ein DIM-Befehl enthält ein Feld oder eine alphanumerische Variable, die bereits dimensioniert wurde oder ein Feld, das mehr als 32767 Bytes belegt. Oder es wurde versucht, auf ein noch nicht dimensioniertes Feld oder auf eine noch nicht di-

mensionierte alphanumerische Variable zuzugreifen.

### 11 ZAHL ZU GROSS

Es wurde versucht, durch O zu teilen, oder es ist eine Zahl aufgetreten, deren Absolutwert größer ist als 9,99999999x10<sup>97</sup>.

### 12 ZEILE NICHT GEFUNDEN

Bei einem GOSUB-, GOTO-, IF-THEN-, ON-GOSUB- oder ON-GOTO-Befehl steht eine Zeilennummer, die im Programm nicht vorkommt.

### 13 FOR-BEFEHL FEHLT

Es wurde ein NEXT-Befehl gefunden, zu dem der entsprechende FOR-Befehl fehlt. Möglicherweise überschneiden sich zwei FOR-NEXT-Schleifen.

### 14 ZEILE ZU LANG

Ein Befehl ist zu komplex oder geht über das Ende einer logischen Zeile hinaus.

### 15 ZEILE MIT GOSUB- ODER FOR-BEFEHL WURDE GELOESCHT

Bei der Ausführung eines RETURN- oder NEXT-Befehls konnte die Zeile, in der sich der entsprechende GOSUB- oder FOR-Befehl befand, nicht mehr gefunden werden.

### 16 GOSUB-BEFEHL FEHLT

Es ist ein RETURN-Befehl gefunden worden, obwohl noch kein GOSUB-Befehl ausgeführt wurde.

### 17 BEFEHL NICHT AUSFÜHRBAR

Ein Befehl wurde durch einen Fehler im Speicher, einen POKE-Befehl oder ein Maschinenprogramm so verändert, daß er nicht mehr erkannt und somit nicht ausführbar ist.

### 18 FALSCHES ZEICHEN

Es wurde versucht, eine alphanumerische Variable mit Hilfe der VAL-Funktion in einen numerischen Wert umzuwandeln, obwohl in dieser Variablen Zeichen enthalten sind, die nicht umgewandelt werden können.

Anmerkung: Die folgenden Fehlermeldungen betreffen Ein-/Ausgabefehler (Input/Output = I/O), die bei der Benutzung von Diskettenstationen, Druckern oder anderen Peripheriegeräten auftreten können. Weitergehende Informationen enthalten die Betriebsanleitungen der Zusatzgeräte.

### 19 PROGRAMM IST ZU LANG

Es wurde versucht, ein Programm zu laden, das mehr Speicherplatz belegt als im Computer vorhanden ist.



## 20 FALSCHER KANALNUMMER

Es wurde versucht, Kanal 0 zu benutzen, oder es wurde eine Kanalnummer angegeben, die größer als 7 ist.

## 21 LADEN MIT LOAD NICHT MÖGLICH

Es wurde versucht, Daten oder ein unter Verwendung des `CSAVE`- oder `LIST`-Befehls gespeichertes Programm mit Hilfe des `LOAD`-Befehls zu laden.

## 128 ABRUCH DURCH BREAK-TASTE

Während einer Ein/Ausgabeoperation wurde die `BREAK`-Taste gedrückt.

## 129 KANAL BEREITS OFFEN

Es wurde versucht, einen Kanal zu öffnen, der bereits offen war. Zur Ausführung der Grafikbefehle wird zum Beispiel Kanal 6 und zur Ausführung einiger anderer Befehle Kanal 7 verwendet. Beim Auftreten dieses Fehlers wird der Kanal, der den Fehler verursacht hat, unter Umständen automatisch geschlossen.

## 130 UNBEKANNTES GERÄT

Es wurde versucht, auf ein unbekanntes Gerät zuzugreifen.

## 131 KANAL NUR FÜR AUSGABE GEÖFFNET

Es wurde ein `GET`- oder `INPUT`-Befehl im Zusammenhang mit einem Ein/Ausgabekanal verwendet, der nur für Ausgabe geöffnet ist.

## 132 XIO-BEFEHL FEHLERHAFT

Bei der Ausführung eines `XIO`-Befehls ist ein Fehler aufgetreten.

## 133 KANAL IST NICHT GEÖFFNET

Es wurde versucht, einen Ein/Ausgabekanal zu benutzen, der noch nicht geöffnet war.

## 134 FALSCHER IOCB NUMMER:

Die Datei-Nummer ist falsch.

## 135 KANAL NUR FÜR EINGABE GEÖFFNET

Es wurde ein `PUT`- oder `PRINT`-Befehl im Zusammenhang mit einem Ein/Ausgabekanal verwendet, der nur für Eingabe geöffnet ist.

## 136 FILENDE ERREICHT

Es wurde ein `End-of-file-Record` gelesen, oder es wurde versucht, einen Sektor der Diskette zu lesen, der nicht zu dem geöffneten File gehört.

## 137 DATENSATZ UNVOLLSTÄNDIG:

Diese Fehlermeldung tritt auf, wenn der (von dem Peripheriegerät) zu lesende Datensatz länger ist, als beim Aufruf der `CIO`-Routine angegeben wurde.

(Die größte Länge eines Datensatzes (einer Input-Zeile) in `BASIC` beträgt 119 Bytes.)

## 138 GERÄT NICHT ANSPRECHBAR

Das angegebene Gerät konnte nicht angesprochen werden. Vergewissern Sie sich, ob das Gerät eingeschaltet ist, ob alle notwendigen Verbindungen ordnungsgemäß hergestellt sind und ob alle Schalter mit einer "ONLINE"- und einer "LOCAL"-Stellung in der "ONLINE"-Stellung stehen.

## 139 GERÄT ARBEITET NICHT EINWANDFREI

Der Programmrecorder oder die Diskettenstation arbeitet nicht einwandfrei oder kann einen bestimmten Befehl nicht ausführen.

## 140 FEHLER AUF DEM SERIELLEN BUS

Möglicherweise ist eine Kassette oder eine Diskette defekt.

## 141 CURSOR AUSSERHALB DER BILDSCHIRMBEGRENZUNG

## 142 FORMATFEHLER BEI DER DATENÜBERTRAGUNG ÜBER DEN SERIELLEN BUS

Bei der Datenübertragung über den seriellen Bus ist ein Fehler aufgetreten. Möglicherweise ist eine Kassette oder eine Diskette defekt.

## 143 PRÜFSUMMENFEHLER BEI DER DATENÜBERTRAGUNG ÜBER DEN SERIELLEN BUS

Bei der Datenübertragung über den seriellen Bus ist ein Fehler aufgetreten. Eine Kassette oder Diskette konnte nicht beschrieben oder gelesen werden. Möglicherweise ist die Kassette oder Diskette defekt.

## 144 DISKETTENFEHLER

Eine Diskette ist schreibgeschützt oder in der Diskette befindet sich ein Fehler.

## 145 DISKETTENSCHREIBFEHLER ODER FEHLER BEI DER BILDSCHIRMDARSTELLUNG

Es wurde ein Unterschied festgestellt zwischen dem, was auf eine Diskette geschrieben werden sollte und dem, was tatsächlich geschrieben wurde, oder es wurde ein Fehler im Zusammenhang mit der Bildschirmdarstellung festgestellt.

## 146 FUNKTION NICHT AUSFÜHRBAR

Es wurde versucht, eine nicht ausführbare Funktion auszuführen, zum Beispiel Daten an die Tastatur auszugeben oder Daten vom Drucker zu lesen.



### **147 SPEICHERPLATZ REICHT FÜR DIE GEWÄHLTE GRAFIKBETRIEBSART NICHT AUS**

Für die verschiedenen Grafikbetriebsarten wird unterschiedlich viel Speicherplatz benötigt.

### **160 UNBEKANNTE DISKETTENSTATION**

Wenn die Diskettenstation angesprochen wird, können nur die Gerätenamen D, D1, D2, D3, oder D4, verwendet werden.

### **161 ZU VIELE DATEIEN OFFEN**

### **162 KEIN SPEICHERPLATZ MEHR AUF DER DISKETTE**

Auf der Diskette ist kein Speicherplatz mehr vorhanden; alle Sektoren sind belegt.

### **163 SYSTEMFEHLER**

Während der Ein- oder Ausgabe von Daten wurde ein Fehler festgestellt, der nicht behoben und dessen Ursache nicht festgestellt werden konnte.

### **164 DATEI/SEKTOR-FEHLER**

Der Zeiger einer Datei wurde mit Hilfe eines POINT-Befehls auf einen Sektor gesetzt, der nicht zu der geöffneten Datei gehört, oder die Verbindungen zwischen den einzelnen Sektoren sind zerstört.

### **165 FALSCHER DATEINAME**

Ein Dateiname beginnt mit einem Kleinbuchstaben oder enthält unzulässige Zeichen. Es ist auch möglich, daß die Wild-Card-Zeichen nicht richtig benutzt wurden.

### **166 POINT-BEFEHL FEHLERHAFT**

Das bei einem POINT-Befehl angegebene Byte existiert nicht.

### **167 DATEI IST SCHREIBGESCHÜTZT**

Schreibgeschützte Dateien können weder beschrieben oder gelöscht, noch kann der Name dieser Datei geändert werden.

### **168 UNBEKANNTER XIO-BEFEHL**

Es wurde versucht, einen XIO-Befehl auszuführen, den es nicht gibt.

### **169 KEIN PLATZ MEHR IN DER DIRECTORY**

In der Directory können 64 Dateinamen eingetragen werden, unabhängig davon, wieviel Speicherplatz auf der Diskette noch vorhanden ist.

### **170 DATEI NICHT GEFUNDEN**

Auf der in der angegebenen Diskettenstation eingelegten Diskette kann keine Datei mit dem angegebenen Dateinamen gefunden werden.

### **171 POINT-BEFEHL NICHT AUSFÜHRBAR**

Es wurde versucht, auf einen Sektor zuzugreifen, der nicht zu der geöffneten Datei gehört.

### **172 VERBOTENE DATEIVERKETTUNG:**

Der Anwender versuchte eine DOS II Datei an eine unter DOS I erstellte Datei anzuhängen. Kopieren Sie die DOS I Datei mit DOS II auf eine DOS II Diskette.

### **173 DEFEKTE SEKTOREN BEIM FORMATIEREN:**

Die Diskette ist defekt, verwenden Sie eine andere. Tritt der Fehler oft auf, kann auch die Diskettenstation defekt sein.

 **ATARI**<sup>®</sup>  
COMPUTER

---

**NOTIZEN**



## ATARI COMPUTER-PROGRAMME

Zur Durchführung einer Aufgabe benötigt der Computer bestimmte Informationen in Form von Programmen (Software), die entweder selbstgeschrieben oder fertig gekauft werden können. Grundlage dieser Programme sind die unterschiedlichen Programmiersprachen. Das ATARI BASIC ist bereits in den Computer eingebaut; weitere Sprachen stehen je nach Aufgabenstellung auf verschiedenen Datenträgern zur Verfügung.

Die ersten erfolgreich selbstgeschriebenen Programme ermutigen zu anspruchsvolleren schwierigen Problemlösungen, die vielleicht sogar zu verkaufsfähiger Software führt. Deshalb wird niemand erfreut sein, wenn sein Programm von Dritten ohne Genehmigung kopiert und weiterverkauft wird ohne irgendeine Entschädigung für den Urheber. Verletzungen dieses Urheberrechts sind ungesetzlich und führen zu strafrechtlichen Konsequenzen für den Kopierenden und auch den Käufer solcher Computer-Programme.

Die ATARI Computer-Programme zeichnen sich durch anwenderfreundliche Bedienung und Nutzung aller Systemvorteile aus. Lassen Sie sich von Ihrem ATARI-Fachhändler das aktuelle Angebot einmal vorführen.

Die folgende Aufstellung soll Ihnen einen ersten Überblick ermöglichen.

### Programmiersprachen:

- ATARI BASIC      Bereits im ATARI XL-Computer eingebaut.
- Microsoft BASIC II      Erweitertes BASIC der Firma MICROSOFT.
- Assembler/Editor      Zum Programmieren in Maschinensprache.
- Macro Assembler      Besonders komfortabler Assembler.
- Pascal      Leicht lern- und anwendbare Sprache für Anfänger geeignet. Weitverbreitet im Schulunterricht.
- Pilot      Durch "Turtle"-Grafik besonders für Schulanfänger geeignet.
- ATARI LOGO      Einfache, und dennoch leistungsfähige Programmiersprache.

- Forth      Eine schnelle Sprache, welche strukturiertes Programmieren ermöglicht.

### LERNEN:

- Programmieren leicht gemacht      Ein Fortsetzungskurs auf Cassette, der zusätzliche gesprochene Informationen bietet.
- und viele weitere Programme vom Rechtschreibtrainer bis zum Kinderprogramm.

### WEITERBILDUNG:

- Music-Composer      Komponieren mit Hilfe des Computers, Jetzt spricht Ihr Computer zu Ihnen.
- Sprach-Box
- und vieles mehr.

### HEIM UND BERUF:

- Karteikarten      Zum Sortieren, Ändern, Ausdrucken usw.
- VisiCalc      Die umfassende Tabellenkalkulation.
- ATARI Schreiber      Das erste deutsche Textverarbeitungsprogramm von ATARI

### SYSTEM:

- Viele Programme, die das Programmieren wesentlich erleichtern.

### UNTERHALTUNG:

- Hier finden Sie die Computerversionen der sensationellen Arcade-Hits, vom Klassiker PAC-MAN über POLE POSITION bis zum Superprogramm LOS ANGELES 1984 GAMES.



---

**NOTIZEN**



## ATARI CLUB

### Noch Fragen, bitte...?

Die faszinierende Beschäftigung mit einem Computer wirft gerade für Anfänger häufig Fragen auf, die auch ein gutes Handbuch nicht immer lösen kann. Ideal ist dann der Freund in der Nachbarschaft, der mit Tips und Tricks beim Programmieren helfen kann, der zum neuen Computer-Programm auch die sinnvollen Peripheriegeräte empfiehlt, etc.

Einige der häufigsten Fragen sind hier kurz erklärt.

### PROBLEME

Einige bereits vorhandene Disketten-Programme vom ATARI 800 laufen nicht mehr auf dem neuen XE Computer.

Wie werden die Umlaute (ä, ö, ü) dargestellt?

Kann man mit dem ATARI Programmrecorder auch vom BASIC steuern?

Wie kann man die Sonderfunktionstasten im eigenen Programm nutzen?

### INFORMATION

Bei Nicht-ATARI-Programmen kann es zu Ladeproblemen kommen. Abhilfe bringt die ATARI Translator-Diskette, mit der das ältere Betriebssystem 400/800 in den neuen 130 XE Computer geladen wird.

Sie sind im internen Zeichensatz enthalten. Einschalten durch POKE 756204 und ausschalten durch POKE 756224. Darstellung durch Drücken der CTRL-Taste mit weiterer Taste.

Cassetten-Motor starten durch POKE 5401852 und ausschalten durch POKE 5401860

Ein kleines Programm als Beispiel:  
 10 A = PEEK (5327)  
 20 IF A = N THEN GOTO 40  
 30 GOTO 10  
 40 PRINT A : END

N = 3 bei OPTION-Taste  
 N = 5 bei SELECT-Taste  
 N = 6 bei START-Taste

Die HELP-Taste wird mit folgender Zeile ausgelesen:  
 N = PEEK (732) : POKE 7320  
 Wenn N nicht NULL ist, wurde die HELP-Taste gedrückt. Der POKE-Befehl löscht die Speicherstelle.



## PROBLEME

Wie werden eigene BASIC Programme ausgedruckt?

Wie kann man BASIC-Programme speichern und laden?

Programm-Recorder

Diskettenstation

## INFORMATION

Bei betriebsbereitem Drucker einfach LIST"P;" eingeben und RETURN drücken. Einzelne Zeilen durch LIST"P;" IO.ZO. etc.

Es gibt verschiedene Möglichkeiten

Speichern	Laden
LOAD"C;"	SAVE"C;"
CLOAD	CSAVE
ENTER"C;"	LIST"C;"

Speichern  
SAVE"D. Name  
LIST"D. Name  
LOAD"D. Name  
ENTER"D. Name  
Dateiname

Laden  
Das Laden und Starten kann zusammengefaßt werden mit RUN"C;" bzw. RUN"D. Dateiname



### Deutschsprachige Literatur zum ATARI Computer-System

VERLAG	Verfasser/Titel	ISBN	Inhalt
te-wi Verlag GmbH München	L. Poole, M. McNiff, S. Cook  Mein ATARI Computer	3-921803-18-7	System-/Gerätebeschreibung der ATARI Computer mit Peripheriegeräten, BASIC-Befehle mit Programmbeispielen. Ein umfassendes Systembuch.
Markt & Technik Verlag AG München	H. Kohl, T. Kahn, L. Lindsay, P. Cleland  Spiel + Spaß mit dem ATARI	3-89090-002-X	Für Anfänger und Fortgeschrittene. Wie man mit dem Computer "spricht". Wie man Spiele entwickelt. Mit über 100 Programm-Bausteinen und vielen fertigen Programmen.
Markt & Technik Verlag AG München	John White  Strategische Computer- Spiele für Ihren ATARI	3-89090-004-6	Alle Entwicklungsstufen intelligenter Spiele wie zB Schach werden sorgfältig anhand von Beispielen erläutert.
Idea Verlag München	D. Inman, K. Inman  Der ATARI Assembler	3-88793-025-8	Mit etwas BASIC-Grundwissen wird ohne weitere Assembler-Kenntnisse in die hochentwickelte Programmiersprache eingeführt.
Frech-Verlag Stuttgart	ATARI BASIC spielend lernen	3-7724-0603-3	Kurz-Einführung in ATARI BASIC mit vielen Anwendungsbeispielen.
W. Hotacker GmbH München	Programmieren in Maschinensprache mit dem 6502	3-921682-61-4	Allgemeine Informationen über das Programmieren in Maschinensprache (mit Prozessor 6502, wie auch im ATARI verwendet).
Sybex-Verlag Düsseldorf	Einführung in PASCAL und UCSD/PASCAL	3-88745-004-3	Alles über PASCAL. Geht nicht auf spezielle Systeme (ATARI etc.) ein.
W. Hotacker GmbH Holzkirchen	E. Flogel  Forth Handbuch	3-911682-88-6	Einführung in die Programmiersprache Forth. Forth ist kompakt, schnell und strukturiert. Forth hat sein eigenes Betriebssystem.

Weitere Titel sind in Vorbereitung.



### Englischsprachige Literatur zum ATARI Computer-System

VERLAG	Verfasser/Titel	ISBN	Inhalt
Reston Publishing Company Inc. Reston, Virginia USA	H. Kohl, T. Kahn, L. Lindsay, P. Cleland  ATARI Games und Recreations	0-8359-0242-0	Kurze Systemeinführung mit Programmiertips für Spiele, Ton und Graphik.
W. Hofacker GmbH Holzkirchen	S. Roberts  Games for the ATARI (400/800)	3-911682-84-3	Listings für Spielprogramme in BASIC und Maschinensprache.
Reston Publishing Company, Inc. Reston, Virginia USA	D. Inman, K. Inman	0-8359-0236-6	Für BASIC-erfahrene Anwender ohne Vorkennntnis in ATARI ASSEMBLER/Editor.
HAYDEN BOOK Company Rochette Park New Jersey USA	E.W. Engel  Stimulating Simulations	0-8104-5197-2	Zwölf BASIC Spielprogramme mit Listings für ATARI Computer.
W. Hofacker GmbH Holzkirchen	S.D. Roberts How to programm your ATARI Computer in 6502 Machine language	3-921682-97-5	Für BASIC-erfahrene Anwender, die ATARI ASSEMBLER/Editor anhand von Beispielen lernen wollen.
Osborne/Mc Graw-Hill Berkeley, California USA	L.A. Leventhal, W. Saville  6502 Assembly Language Subroutines	0-931988-59-4	Spezielle Information über das Programmieren in Maschinensprache (6502).
Compute! Publications, Inc. Greensboro, NC USA	B. Wilkinson, K. O'Brien, P. Laughton  The ATARI BASIC Source Book	0-942386-15-9	Allgem. und spezielle Informationen über ATARI BASIC und Source Code.
John Wiley + Sons 605 Third Av., NY USA	H. Moore, I. Lower, B. Albrecht ATARI Sound and Graphics	0-471-09593-1	Alles über Ton und Graphik.
W. Hofacker GmbH Holzkirchen	T.E. Rowley  ATARI BASIC Learning by using	3-92-1682-86-X	Programmbeispiele für Ton und Farbe und Kontroller.
W. Hofacker GmbH Holzkirchen	E. Floegel  Forth on the ATARI Learning by using	3-88963-170-3	Einführung in die Programmiersprache FORTH anhand von Beispielen.
Compute! Publications, Inc. Greensboro, N.C. USA	Compute!'s First book of ATARI	0-942386-00-0	Artikel, die bereits in dem Magazin 'Compute!' veröffentlicht wurden. System- und Programmiertips mit Listings.



### Englischsprachige Literatur zum ATARI Computer-System

VERLAG	Verfasser/Titel	ISBN	Inhalt
Reston Publishing Company, Inc. Reston, Virginia USA	I. Conlau, T. Deliman W. Dymax  ATARI Pilot for Beginners	0-8359-0301-X	Einführung in die einfache Programmiersprache PILOT die speziell für Kinder und Jugendliche entwickelt wurde. (Pilot mit Turtle-Graphik)
Sybex Verlag Düsseldorf	P. Lamoitier  BASIC Exercises for the ATARI	0-89588-102-2	BASIC-Programmbeispiele mit listings aus den Bereichen Mathematik, Planung und Unterhaltung
Creative Computing Morris Plains, N.J. USA	D. Small, S. Small, G. Blank  The creative ATARI	0-916688-34-8	Umfangreiche Systemdarstellung mit Listings und Software-Besprechung
CompuTel Publications, Inc. Greensboro, NC USA	CompuTel's Second book of ATARI	0-942386-06-X	Neue, bislang nicht veröffentlichte Artikel über das ATARI System, Programmier-techniken, Grafik, Spiele und andere Anwendungen, Maschinensprache
CompuTel Publications, Inc. Greensboro, NC USA	CompuTel's First book of ATARI Graphik	0-942386-08-6	Alles über Graphik, Nutzung der 256 Farben, Player-Missile etc.
CompuTel Publications, Inc. Greensboro, NC USA	CompuTel's First book of ATARI Games	0-942386-14-0	15 professionelle Spiel-Programme zum Selbststudium, um die Möglichkeiten des Systems kennenzulernen und die eigene Technik zu verbessern

**SERVICE**

Alle ATARI Computer werden nach modernsten Gesichtspunkten gefertigt und vor Auslieferung mehrfach geprüft. Sofern es doch einmal zu Problemen kommen sollte, verfahren Sie bitte wie folgt:

**Im Garantiefall**

1. Wenden Sie sich bitte an das nächstgelegene autorisierte ATARI-Service-Center.
2. Reichen Sie bitte alle Teile des Computer-Systems zurück, die einen Defekt aufweisen, oder damit im Zusammenhang stehen.
3. Achten Sie bitte auf sorgfältige Verpackung (insbesondere bei Postversand). Verwenden Sie nach Möglichkeit die Originalverpackung.
4. Legen Sie bitte eine kurze, genaue Fehlerbeschreibung bei.
5. Bei Garantieanspruch legen Sie bitte unbedingt die vollständig ausgefüllte und abgestempelte Garantiekarte und den Kaufbeleg bei.

**Nicht- arantie-Service**

Bei abgelaufener Garantie verfahren Sie bitte wie unter Punkt 1--4. Die Reparatur ist dann kostenpflichtig und wird entsprechend dem Aufwand berechnet.

 **ATARI**<sup>®</sup>  
COMPUTER

---

# ANHANG

**Die folgenden Informationen sind nur für Anwender von Bedeutung, die mehr über technische Einzelheiten der ATARI XL-Computer, wie zum Beispiel Belegung der Buchsen, Nutzung besonderer Speicherstellen etc. wissen möchten.**



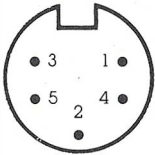
### Technische Daten des ATARI 130XE

<b>Prozessor:</b>	6502C Mikroprozessor, Taktfrequenz 1,79 MHz
<b>Spezial-Bausteine:</b>	GTIA Chip — Graphik-Darstellung POKEY CHIP — Tonerzeugung und serieller Bus I/O AN11C CHIP — Bildschirmdarstellung FREDDY CHIP — Speicherverwaltung
<b>Speicher:</b>	131.072 Byte RAM 24.576 Byte ROM (Betriebssystem + ATARI BASIC Interpreter)
<b>Bildschirm:</b>	11 Graphik-Modi 256 Farben 320 x 192 Bildpunkte 40 Spalten x 24 Zeilen Textdarstellung 5 Text-Modi
<b>Tonerzeugung:</b>	4 unabhängige Tonkanäle (Stimmen) 3 1/2 Oktaven Tonumfang
<b>Tastatur:</b>	Schreibmaschinentastatur 62 Tasten mit HELP-Taste und 4 Spezialfunktionstasten Internationaler Zeichensatz 29 Graphik-Tasten
<b>Programmierung:</b>	ATARI BASIC Interpreter eingebaut Softwarekompatibel zu allen ATARI Computern
<b>Anschlüsse:</b>	Steckmodul-Anschluß TV-Anschluß Monitor-Anschluß 2 Controller-Ports Serieller I/O-Port Erweiterter Modul- Anschluß
<b>Stromversorgung:</b>	1 Ampere bei 5 Volt Gleichspannung

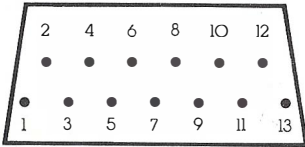
**PIN-Belegung am Computer**

(von außen betrachtet)

**Monitor-Buchse**

 Mehrfarbmonitor über 2 und 4  
 Einfarbmonitor über 2 und 1


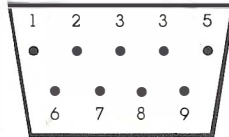
1. Composite Luminance
2. GROUND (Masse)
3. Audio Output/NF
4. Composite Video (FBAS)
5. Composite Chroma (Color + Burst)

**Serielle I/O Buchse**


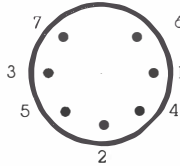
1. Clock Input
2. Clock Output
3. Data Input
4. Ground
5. Data Output
6. Ground
7. Command
8. Motor Control
9. Proceed
10. +5/Ready
11. Audio Input
12. (nicht belegt)
13. Interrupt

**Buchse für Steuerungen**

Port 1 und Port 2



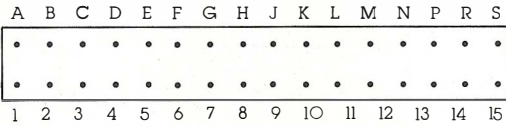
1. (Joystick) Forward Input
2. (Joystick) Back Input
3. (Joystick) Left Input
4. (Joystick) Right Input
5. B Potentiometer Input
6. Trigger Input
7. +5 volts
8. Ground
9. A Potentiometer Input

**Netzteil Anschluß**


1. +5V
2. Shield
3. Ground
4. +5V
5. Ground
6. +5V
7. Ground

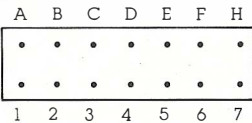


### Modulschacht

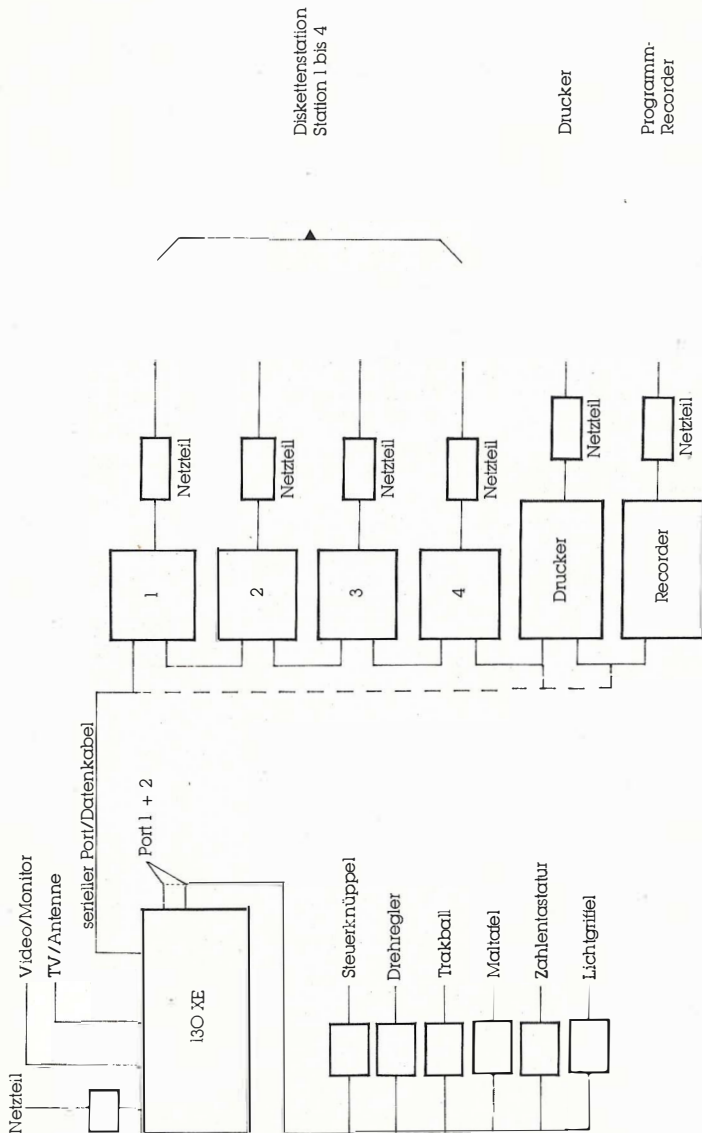


- |          |        |
|----------|--------|
| 1. S4    | A. RD4 |
| 2. A3    | B. GND |
| 3. A2    | C. A4  |
| 4. A1    | D. A5  |
| 5. A0    | E. A6  |
| 6. D4    | F. A7  |
| 7. D5    | H. A8  |
| 8. D2    | J. A9  |
| 9. D1    | K. A12 |
| 10. D0   | L. D3  |
| 11. D6   | M. D7  |
| 12. S5   | N. A11 |
| 13. +5V  | P. A10 |
| 14. RD5  | R. R/W |
| 15. CCTL | S. BO2 |

### ENHANCED CARTRIDGE INTERFACE (ECI)



- |             |          |
|-------------|----------|
| A. Reserved | 1. EXSEL |
| B. IRQ      | 2. RST   |
| C. HALT     | 3. D1XX  |
| D. A13      | 4. MPD   |
| E. A14      | 5. Audio |
| F. A15      | 6. REF   |
| H. GND      | 7. +5V   |


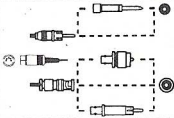
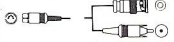





**BLOCKSCHALTBIKD**  
**ATARI Computer-System**  
**130 XE-Handbuch**

 Diskettenstation  
 Station 1 bis 4

Drucker

 Programm-  
 Recorder

Alle über den seriellen Port anzuschließenden Peripheriegeräte werden mit dem Computer und untereinander mit dem speziellen ATARI Datenkabel verbunden. Bitte verwenden Sie bei Bedarf nur die Original ATARI Ersatz- und Zubehörteile.

**VERBINDUNGSKABEL**

von	nach	Kabel		Artikel-Nr.
Computer	TV-Gerät	Antennenkabel		
Computer	Monitor	Monitorkabel, universell verwend- bar, mit Adaptem Bild: BNC, PL, Cinch Ton: Cinch, Klinke 3,5		89
Computer	Monitor	Monitor-Kabel BNC Bild: BNC Ton: Cinch		8610
Computer	Monitor/TV	Monitorkabel Scart für Scart-Buchse		8611
Computer	Monitor/TV	Monitorkabel AV für AV-Buchse		8612
Computer Disketten-Station Prog.-Recorder Drucker	Disketten-Station Prog.-Recorder Drucker	Datenkabel		81
Computer		Drucker mit Centronics- Schnittstelle	Druckerkabel (Nur zur Verwendung mit ATARI 850 Interface-Modul)	
		Interface-Kabel MPP 1150		1150

**INTERNER ZEICHENSATZ**

**Interner Zeichensatz**

1		2		3		4	
#	CHR	#	CHR	#	CHR	#	CHR
0	Space	32	@	64		80	
1	!	33	A	65		81	
2	"	34	B	66		82	
3	#	35	C	67		83	
4	\$	36	D	68		84	
5	%	37	E	69		85	
6	&	38	F	70		86	
7	'	39	G	71		87	
8	(	40	H	72		88	
9	)	41	I	73		89	
10	*	42	J	74		90	
11	+	43	K	75		91	
12	,	44	L	76		92	
13	-	45	M	77		93	
14	-	46	N	78		94	
15	/	47	O	79		95	
						96	
						97	a
						98	b
						99	c
						100	d
						101	e
						102	f
						103	g
						104	h
						105	i
						106	j
						107	k
						108	l
						109	m
						110	n
						111	o
						112	p
						113	q
						114	r
						115	s
						116	t
						117	u
						118	v
						119	w
						120	x
						121	y
						122	z
						123	
						124	l
						125	
						126	
						127	

1. Bei Grafik-Mode O muß zur Darstellung zuvor die ESC-Taste, CHR\$ 27, gedrückt werden.

**SPEICHERAUFTeilUNG DES XE-SYSTEMS**

Die folgende Tafel zeigt, wie der 6502-Prozessor den Adreßbereich aufteilt. Der maximale Adreßbereich, den der 6502-Prozessor mit 16 Bit ansteuern kann liegt bei \$0000 bis \$FFFF. Dieser Adreßbereich wird folgendermaßen durch die Hardware-Schaltungen aufgespalten:

**SPEICHERAUFTeilUNG**

HEX-ADRESSE	BELEGT DURCH	ANMERKUNGEN
FFFF-D800	OS-ROM oder RAM, wenn ROM abgeschaltet.	1
D7FF-D000	Durch Zugriffe auf diese Speicher-PAGE werden aktive Chip-Selektionen für die Peripherie-Chips erzeugt	
	I/O-Ramaufteilung (memory-mapped)	
	D000-D0FF GTIA	
	D200-D2FF POKEY	
	D300-D3FF PIA	
	D400-D4FF ANTIC	
	D500-D5FF Jede in diesem Bereich angesprochene Adresse aktiviert die CCNTL-Kontrollschaltung des Moduls-Interfaces (wie bei der alten Serie).	
	D100-D1FF, D600-D6FF, D700-D7FF Sind für zukünftige Belegung reserviert	
	OS-ROM physisch vorhanden, kann aber nicht angesprochen werden.	2
CFFF-C000	OS-ROM oder RAM, wenn ROM abgeschaltet.	1
BFFF-A000	RAM oder Modul-Interface, wenn RD5-Leitung durch den Einschub auf +5V angesprochen wird.	
9FFF-8000	RAM oder Modul-Interface, wenn RD4-Leitung durch den Einschub auf +5V angesprochen wird.	
7FFF-5800	RAM	
57FF-5000	RAM, solange nicht im Selbsttest-Modus.	2
4FFF-0000	RAM	

- Der Zugriff auf das OS-ROM kann durch Schreiben eines Wertes von 0 nach Port B des PIA, Bit PBO, abgeschaltet werden. Der Zugriff wird normalerweise durch eine 1 in diesen Bit eingeschaltet. Wird dieses Bit geändert, so dürfen andere Bits des Registers nicht beeinflusst werden.
- Der Selbsttest ROM-Code ist an den Adressen \$D000-\$D7FF im OS-ROM physisch vorhanden. Dieser Bereich wird allerdings für den Zugriff auf die memory-mapped I/O-Devices benötigt. Wird der Selbsttest angesteuert, dann wird das RAM an den Speicherstellen \$5000-\$57FF abgeschaltet. Der Memory-Manager definiert den Speicherzugriff so, daß die Adressen \$D000-\$57FF angesprochen werden. Er benutzt Port B des PIA, Bit PB7, um festzulegen, ob RAM oder ROM in dem Bereich von \$5000-\$57FF angesteuert werden soll. Ist das PB7 unter Spannung gesetzt, so wird RAM angesteuert; andernfalls wird auf das OS-ROM zurückgegriffen. Bei Änderung dieses Bits sollten die anderen Bits des Registers nicht beeinflusst werden.

## OFT BENUTZTE SPEICHERADRESSEN UND DEREN FUNKTIONEN

Viele Speicherzellen erfüllen ganz bestimmte Aufgaben. In diesem Abschnitt sind die Speicherzellen aufgeführt, die für den BASIC-Programmierer von besonderem Interesse sind. Mit Hilfe eines PEEK-Befehls kann der Inhalt einer Speicherzelle gelesen und mit Hilfe eines POKE-Befehls der Inhalt einer Speicherzelle verändert werden.

In BASIC werden Speicheradressen und Inhalte von Speicherzellen dezimal angegeben. Die Speicherzellen sind von 0 bis 65535 nummeriert. Der Wert des Inhaltes einer Speicherzelle kann zwischen 0 und 255 liegen. Wenn größere Werte gespeichert werden sollen, benötigt man zwei aufeinanderfolgende Speicherzellen. Der Inhalt dieser beiden Speicherzellen zusammen berechnet sich aus der Summe des Inhaltes der ersten Speicherzelle und dem 256-fachen des Inhaltes der zweiten Speicherzelle. Um die Spalte zu bestimmen, an der sich der Cursor im Moment befindet, muß zum Beispiel PEEK (85) + 256 \* PEEK (86) berechnet werden. Umgekehrt müssen, um die Spaltenposition des Cursors zu verändern, die beiden Befehle POKE 85, SPAL-INT (SPAL/256) \* 256 und POKE 86, INT (SPAL/256) \* 256 ausgeführt werden. Anschließend befindet sich der Cursor in der Spalte SPAL.

### 17 BREAK-Taste gedrückt (BREAK Key Flag = BRKKEY)

Wenn sich in dieser Speicherzelle eine 0 befindet, wurde die BREAK-Taste gedrückt.

### 77 Farbwechsel Ein/Aus (Attract Mode On/Off = ATRACT)

Wenn in dieser Speicherzelle eine 0 abgelegt wird, dann wird der Farbwechsel auf dem Bildschirm ausgeschaltet. Das geschieht automatisch immer dann, wenn eine Taste der Tastatur gedrückt wird. Wird in dieser Speicherzelle jedoch eine 254 abgelegt, so wird der Farbwechsel eingeschaltet. Das geschieht automatisch, wenn neun Minuten lang keine Taste gedrückt wurde.

### 82 Linke Bildschirmbegrenzung (Left Margin of Text Area = LMARGIN)

Diese Speicherzelle enthält die Nummer der Spalte, an der sich in der Grafikbetriebsart 0 die linke Bildschirmbegrenzung befindet. PEEK (82) liegt immer zwischen 0 und 39. 0 entspricht dem linken Bildschirmrand. Normalerweise ist in dieser Speicherzelle eine 2 abgelegt.

### 83 Rechte Bildschirmbegrenzung (Right Margin of Text Area = RMARGIN)

Diese Speicherzelle enthält die Nummer der Spalte, an der sich in der Grafikbetriebsart 0 die rechte Bildschirmbegrenzung befindet. PEEK (83) liegt immer zwischen 0 und 39. 39 entspricht dem rechten Bildschirmrand. Normalerweise ist in dieser Speicherzelle eine 39 angelegt.

### 84 Momentane Cursorposition (Zeile) (Current Row Cursor Position = ROWCRS)

Der Inhalt dieser Speicherzelle entspricht der Zeile des Bildschirms, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. Der Minimalwert von PEEK (84) ist 0. Wie groß der Maximalwert ist, hängt davon ab, welche Grafikbetriebsart aktiviert ist.

### 85, 86 Momentane Cursorposition (Spalte) (Current Column Cursor Position = COLCRS)

Der Inhalt dieser Speicherzellen entspricht der Spalte des Bildschirms, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. Der Minimalwert von PEEK (85) ist 0. Wie groß der Maximalwert ist, hängt davon ab, welche Grafikbetriebsart aktiviert ist. In den Grafikbetriebsarten 0 bis 7 ist PEEK (86) immer 0.

### 88, 89 Adresse des Bildschirmspeichers (Screen Memory Address = SAVMSC)

Diese Speicherzelle enthält die unterste Adresse des Bildschirmspeichers. Der Inhalt dieser Adresse wird in der oberen linken Ecke des Bildschirms dargestellt.

### 94, 95 Cursoradresse (Cursor Memory Address = OLDADR)

In dieser Speicherzelle ist die Adresse des Cursors gespeichert.

### 106 Oberes Ende des RAM-Speichers (höherwertiges Byte) (Top of RAM Address = RAMTOP)

Der Inhalt dieser Speicherzelle entspricht der 16-fachen Anzahl der 4K RAM-Blöcke, auf die der Computer ausgebaut ist. Die Anzahl der 1K-Blöcke kann mit PEEK (740)/4 berechnet werden.

**186, 187 Zeilennummer, in der ein Programm unterbrochen wurde (Stop Line Number = STOPLN)**

Diese Speicherzellen enthalten die Zeilennummer eines BASIC-Programmes, in der das Programm durch Drücken der BREAK-Taste oder einen STOP-Befehl unterbrochen wurde oder in der ein Fehler auftrat.

**195 Kodexzahl des aufgetretenen Fehlers (Error Number — ERR#AV)**

Wenn ein Fehler auftritt, wird die Kodexzahl des Fehlers in dieser Speicherzelle abgelegt.

**212, 213 Funktionswert der USR-Funktion (USR Function Value = FRO)**

In diesen Speicherzellen kann durch ein mit Hilfe der USR-Funktion aufgerufenes Maschinenprogramm ein numerischer Wert abgelegt werden, der beim Rücksprung in das BASIC-Programm als Funktionswert übergeben wird.

**251 Grad/Bogenmaß-Umschaltung (Radians or Degrees = RADFLG oder DEGFLG)**

Wenn hier eine 0 abgelegt ist, wird im Bogenmaß und wenn hier eine 1 abgelegt ist, in Grad gerechnet.

**656 Cursorposition im Textfenster (Zeile) (Text Cursor Row Position = TXTROW)**

Der Inhalt dieser Speicherzeile entspricht der Zeile des Textfensters, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. PEEK (656) liegt immer zwischen 0 und 3. 0 entspricht der ersten Zeile des Textfensters.

**694 Inverse Zeichendarstellung (Inverse Video Keystrokes = INVFLG)**

Wenn sich in dieser Speicherzelle eine 0 befindet, dann wird, wenn eine Taste der Tastatur gedrückt wird, der ATASCII-Kode eines normal dargestellten Zeichens erzeugt. Falls hier jedoch ein anderer Wert als 0 gespeichert ist, so wird stattdessen der ATASCII-Kode eines invers dargestellten Zeichens erzeugt.

**702 Umschaltung Groß/Kleinschreibung oder Controlzeichen (Shift/Control Lock Flag = SHFLOK)**

Sinnvolle Inhalte dieser Speicherzelle sind 0 (normale Groß/Kleinschreibung), 64 (Großschreibung) oder 128 (Controlzeichen).

**741, 742 Oberes Ende des freien Speichers (Free Memory Low Address = MEMTOP)**

Die höchste Adresse des freien Speichers kann mit PEEK (741) + 256 \* PEEK (742) — 1 berechnet werden. Der Inhalt dieser Speicherzellen kann sich verändern, wenn die SYSTEM RESET-Taste gedrückt oder ein Ein/Ausgabekanal für den Bildschirm geöffnet wird.

**743, 744 Unteres Ende des freien Speichers (Free Memory Low Address = MEMLO)**

Diese Speicherzellen enthalten die niedrigste Adresse des freien Speichers. Der Inhalt dieser Speicherzellen kann sich beim Drücken der SYSTEM-RESET-Taste verändern.

**752 Cursorunterdrückung (Cursor Inhibit = CRSINH)**

Wenn in dieser Speicherzeile eine 1 abgelegt ist, dann ist der Cursor sichtbar. Jeder andere Wert als 0 macht den Cursor unsichtbar.

**755 Festlegung der Cursor- und Zeichendarstellung (Character and Cursor Control = CHACT)**

In dieser Speicherzelle ist normalerweise eine 2 abgelegt. Durch Verändern des Inhaltes dieser Speicherzelle kann der Cursor undurchsichtig oder unsichtbar gemacht oder die Zeichendarstellung umgekehrt werden.

**756 Auswahl des Zeichensatzes (Character Adress Base = CHBAS)**

Mit Hilfe dieser Speicherzelle wird ausgewählt, welcher Zeichensatz in den Grafikbetriebsarten 1 und 2 benutzt werden soll. Wenn hier eine 224 abgelegt ist, dann wird der Zeichensatz benutzt, in dem die Großbuchstaben und Zahlen enthalten sind. Wenn jedoch in dieser Speicherzelle eine 226 gespeichert ist, so wird der Zeichensatz benutzt, in dem die Kleinbuchstaben und Grafikzeichen enthalten sind.

**764 Zuletzt gedrückte Taste (Keyboard Character = CH)**

In dieser Speicherzelle ist die Kodexzahl der Taste gespeichert, die als letzte gedrückt wurde. Falls keine Taste gedrückt wurde, ist hier eine 255 abgelegt.

### 765 Auswahl der Zeichenfarbe für den XIO-Befehl (Fill Data = FILDAT)

Der Inhalt dieser Speicherzelle bestimmt die Farbe, mit der eine Fläche auf dem Bildschirm mit Hilfe eines XIO-Befehls gefüllt werden soll.

### 766 Controlzeichen anzeigen (Display Control Characters = DSPFLG)

Falls in dieser Speicherzelle eine 0 abgelegt ist, dann wird, wenn ein Zeichen mit einem ATASCII-Kode von 27-31, 123-127, 187-191 oder 251-255 an den Bildschirm übertragen wird, die entsprechende Controlfunktion ausgeführt. Wenn hier jedoch ein anderer Wert als 0 abgelegt ist, so wird stattdessen ein Zeichen auf dem Bildschirm dargestellt.

### 767 Bildschirmausgabe gestoppt (Start/Stop Display Screen = SSFLAG)

Wenn in dieser Speicherzelle eine 255 abgelegt ist, dann ist die Bildschirmausgabe gestoppt. Falls hier eine 0 gespeichert ist, wird die Bildschirmausgabe nicht beeinflusst. Beim Drücken von CTRL-1 wird der Inhalt dieser Speicherzelle verändert.

### 53279 Betätigung der Funktionstasten (CONSOLE Switch Port = CONSOL)

Man kann mit Hilfe eines PEEK (53279)-Befehls feststellen, ob und wenn ja, welche der drei Funktionstasten (OPTION, SELECT und START) gedrückt ist. Vor einem PEEK (53279)-Befehl sollte allerdings der Befehl POKE 53279,8 ausgeführt werden. Nur dann ist vollkommen sicher, daß der gelesene Wert korrekt ist.

Inhalt der Speicherzelle 53279 (dezimal)	Gedrückte Funktionstaste(n)
0	OPTION, SELECT und START
1	OPTION und SELECT
2	OPTION und START
3	OPTION
4	SELECT und START
5	SELECT
6	START
7	Keine

**VERZEICHNIS DER IN ATARI BASIC  
BENUTZTEN BEFEHLSWORTE UND  
FUNKTIONEN**

Reserviertes Befehlswort	Abkür- zung	Funktion
ABS		Berechnet den Absolutwert einer Zahl.
ADR		Berechnet die Adresse, ab der eine alphanumerische Konstante oder Variable im Speicher abgelegt ist.
AND		Logische UND-Verknüpfung zweier Werte (nicht bitweise sondern das Ergebnis ist 0 (Null), falls einer der beiden Werte 0 ist).
ASC		Berechnet den ATASCII-Code eines Zeichens.
ATN		Berechnet den Arcus-Tangens einer Zahl.
BYE	.B.	Übergang vom BASIC in den Selbsttest.
CLOAD	CLOA.	Lädt Daten oder Programme vom Programm-Recorder in den RAM-Speicher ein.
CHR\$		Wandelt entsprechenden ATASCII-Wert (0-255) in Zeichen um.
CLOG		Berechnet den Zehnerlogarithmus einer Zahl.
CLOSE	CL.	Schließt einen Ein-/Ausgabekanal.
CLR		Löscht alle Variablen.
COLOR	C.	Bestimmt ein Farbregister in der Grafik-Betriebsart.
COM		Stellt Speicherplatz bereit. Siehe DIM.
CONT	CON.	Setzt die Programmausführung in der Folgezeile nach Gebrauch der BREAK-Taste oder Eingabe von STOP fort.
COS		Berechnet den Cosinus eines Winkels.
CSAVE		Speichert ein im RAM befindliches Programm auf Cassette ab.
DATA	D.	Teil der READ/DATA-Kombination. Dient zur Speicherung von Werten, die mit Hilfe des READ-Befehls Variablen zugewiesen werden können.
DEG	DE.	Schaltet vom Bogenmaß auf Grad um.

Reserviertes Befehlswort	Abkür- zung	Funktion
DIM	DI.	Stellt Speicherplatz für Feld- oder alphanumerische Variablen bereit.
DOS	DO.	Laden des DOS-Menüs.
DRAWTO	DR.	Zeichnet eine Linie vom zuletzt gezeichneten Punkt zum angegebenen Endpunkt.
END		Beendet ein laufendes Programm.
ENTER	E.	Dient zum Laden eines mit LIST gespeicherten BASIC-Programmes von Cassette oder Diskette.
EXP		Bezeichnet den Wert einer Exponentialfunktion.
FOR...TO...NEXT	F.	Serie von Befehlen, die sofort ausgeführt werden, bis die Lauf-Variablen einen bestimmten Wert erreicht hat.
FRE (O)		Berechnet den noch freien Speicherplatz in Bytes nach Abzug von OS, DOS, Displaylist, Bildschirm Speicher und BASIC mit Daten.
GET	GE.	Lädt unter Verwendung eines Ein-/Ausgabekanals ein einzelnes Byte von dem angesprochenen Gerät.
GOSUB	GOS.	Dient zum Aufruf eines Unterprogrammes.
GOTO	G.	Die Programmausführung wird in der genannten Zeilennummer fortgesetzt.
GRAPHICS	GR.	Aktiviert eine der Grafik-Betriebsarten.
IF...THEN		Wenn eine bestimmte Bedingung erfüllt ist, werden die hinter THEN stehenden Befehle ausgeführt.
INPUT	I.	Mit INPUT können einer Variablen Werte zugewiesen werden.
INT		Rundet eine Zahl ab auf den ganzzahligen Wert.
LEN		Berechnet die Länge einer alphanumerischen Variablen.
LET	LE.	Weist einer Variablen einen Wert zu.
LIST	L.	Stellt ein im RAM befindliches Programm dar. Sofern mit LIST auf Cassette oder Diskette gespeichert, kann es nur mit ENTER wieder geladen werden.

Reserviertes Befehlswort	Abkürzung	Funktion	Reserviertes Befehlswort	Abkürzung	Funktion
LOAD	L.O.	Überträgt ein zuvor gespeichertes Programm in den Computer.	POSITION	POS.	Setzt den Cursor an eine bestimmte Stelle des Bildschirms.
LOCATE	LOC.	Lädt den Code eines Zeichens oder Graphikpunktes, der sich an der angegebenen Position befindet.	PRINT	PR. oder ?	Dient zur Ausgabe von Daten an den Bildschirm oder ein Gerät.
LOG		Berechnet den natürlichen Logarithmus einer Zahl.	PTRIG		Dient zur Feststellung, ob der Feuerknopf des Drehreglers gedrückt ist.
LPRINT	LP.	Dienst zur Übertragung von Zeichen an einen Drucker.	PUT	PU.	Dient zur Ausgabe einer Zahl über den angegebenen Ein-/Ausgabekanal.
NEW		Löscht das gerade im RAM befindliche Programm und alle Variablen.	RAD		Schaltet von Grad auf Bogenmaß um.
NOT		Ergibt den Wert 1, wenn der Ausdruck nicht wahr ist, ist er aber wahr, so ergibt sich der Wert 0.	READ	REA.	Weist die bei einem DATA-Befehl angegebenen Werte den angegebenen Variablen zu.
NOTE	NO.	Stellt fest, an welcher Stelle der Datei auf einer Diskette der Zeiger dieser Datei steht.	REM	R.	Dient zur Einfügung von Erklärungen in ein Programm.
ON		Wird in Verbindung mit GOTO oder GOSUB verwendet zur Ausführung eines Programms ab einer bestimmten Zeile oder eines Unterprogrammes.	RESTORE	RES.	Dient zum Zurücksetzen des für die DATA-Befehle zuständigen Zeigers.
OPEN	O.	Weist einem bestimmten Ein-/Ausgabekanal ein bestimmtes Gerät zu.	RETURN	RET.	Dient zum Rücksprung zu dem unmittelbar auf den zuletzt ausgeführten GOSUB oder ON GOSUB folgenden Befehl.
OR		Logischer Vergleich zweier Ausdrücke. Sofern nur einer wahr ist ergibt sich der Wert 0; eine 1, wenn beide unwahr sind.	RND		Berechnet eine Zufallszahl.
PADDLE		Bestimmt die momentane Stellung des angegebenen Drehreglers mit einem Wert zwischen 1 und 255.	RUN		Dient zum Starten des im RAM befindlichen Programmes bzw. lädt vom benannten Gerät ein und startet sofort.
PEEK		Dient zum Lesen einer RAM- oder ROM-Speicherstelle.	SAVE	S.	Dient zur Übertragung eines im RAM befindlichen BASIC-Programmes an ein bestimmtes Speichergerät. Kann mit LOAD oder RUN wieder geladen werden.
PLOT	PL.	Zeichnet an der angegebenen Position einen Punkt auf den Bildschirm.	SETCOLOR	SE.	Bestimmt Farbe und Helligkeit bei der Darstellung eines bestimmten Farbregisters.
POINT	P.	Setzt den Zeiger einer Disketten-datei an eine bestimmte Position.	SGN		Dient zur Feststellung, ob eine Zahl positiv oder negativ oder gleich 0 ist.
POKE	POK.	Dient zur Speicherung eines bestimmten Wertes in einer bestimmten Speicherzelle. Der erste Wert darf nicht den ROM-Speicher betreffen, der zweite muß zwischen 0 und 255 liegen.	SIN		Berechnet den Sinus eines Winkels.
POP		Löscht im Stack die Rücksprungadresse des zuletzt ausgeführten FOR, GOSUB- oder ON GOSUB-Befehls.	SOUND	SO.	Schaltet einen Tonkanal an oder aus und bestimmt Tonhöhe, Tonreinheit und Lautstärke.
			SQR		Berechnet die Quadratwurzel einer Zahl.

Reserviertes Befehlswort	Abkür- zung	Funktion
STATUS	ST.	Dient zur Feststellung, mit welchem Ergebnis die zuletzt im Zusammenhang mit dem angegebenen Kanal ausgeführte Ein-/Ausgabeoperation abgeschlossen wurde.
STEP		Wird in Verbindung mit FOR/NEXT-Schleifen gebraucht. Bestimmt die schrittweise Erhöhung der Lauf-Variablen.
STICK		Bestimmt die momentane Stellung des angegebenen Steuerknüppels.
STRIG		Dient zur Feststellung, ob der Feuerknopf des Steuerknüppels gedrückt wurde.
STOP	STO.	Dient zur Unterbrechung eines BASIC-Programmes.
STRS		Wandelt einen numerischen Wert in einen alphanumerischen Wert um.
TRAP	T.	Dient zum Abfangen und Bearbeiten von Fehlermeldungen.
USR		Dient zum Aufruf eines Maschinen-Programmes.
VAL		Wandelt einen alphanumerischen Wert in einen numerischen Wert um.
XIO	X.	Dient zum Arbeiten mit Ein-/Ausgabe-Kanälen.



**AUSNUTZEN ALLER RAM KONFIGURATIONSMÖGLICHKEITEN DES 130XE**

Der 130XE verfügt über 131.072 Bytes Schreib-/Lesespeicher (Random Access Memory =» RAM), doppelt soviel wie der ATARI 65XE oder der 800XL Computer.

Die zusätzlichen 65535 Byte des RAM sind dem Anwender in den meisten Fällen zugänglich. Programme können in dem zusätzlichen Speicher-raum größere Datenmengen verfügbar halten. In Verbindung mit einer ATARI 1050 Diskettenstation und DOS 2.5 läßt sich das Zusatz-RAM als RAMDISK nutzen — das ergibt ein sehr schnelles "Laufwerk". (Weitere Informationen über die RAMDISK des 130XE und DOS 2.5 finden Sie in der DOS 2.5: ATARI 1050 Bedienungsanleitung zur Diskettenstation.) Auf die zusätzlichen 65535 Byte des RAM können Sie im ATARI BASIC durch "Bank-Switching" (Speicherbereich-Umschaltung) zugreifen. Die beiden im 130XE verwendeten Mikroprozessoren — der 6502 Hauptprozessor und der ANTIC Videoprozessor — können lediglich 65535 Speicherzellen adressieren (für eine Lese- oder Schreiboperation auswählen). Bank-Switching schaltet einen 16K-Byte großen Speicherbereich ab und ersetzt diesen durch einen Speicherbereich (Bank) des zusätzlichen RAM. So wird der adressierbare Speicherbereich vergrößert. Im 130XE reicht der umschaltbare Speicherbereich von Adresse 16384 bis 32767 dezimal (\$4000 bis \$7FFF hexadezimal). Durch Ändern des Speicherauswahlbytes (Bank-Select Schalter) wird festgelegt, welche Bank in dem 16K Bereich erscheint.

Der Bank-Select Schalter wird als Speicherstelle 54017 angesprochen und stellt Port B des 6502 Peripheral Interface Adapter (PIA) dar, welcher Ein- und Ausgaben des Computers steuert.

Bits 4 und 5 dieser Speicherstelle wählen den oder die Prozessoren, die auf den Zusatzspeicher zugreifen. Nach dem Einschalten des Computers sind beide Bits auf 1 gesetzt. (Bit 4 steuert den (6502) CPU Zugriff und Bit 5 den Zugriff des ANTIC auf die Speicherbank.) Durch Löschen eines der Bits auf 0, adressiert der jeweilige Prozessor die Speicherbank. Sind beide Bits 0, so befinden sich beide Prozessoren im Bank-Modus.

Die Bits 2 und 3 dienen der Auswahl des Bereichs des Zusatzspeichers, der im Bank-Adressbereich (\$4000-\$7FFF) erscheinen soll. Die vier möglichen Bitkombinationen wählen die vier verfügbaren Speicherbereiche aus.

Normalerweise enthält Speicherstelle 54017 den Wert 193. Mit dem BASIC POKE-Befehl kann der Inhalt verändert und damit das Zusatz RAM angesprochen werden. Ein Beispiel: POKE 54017,225 wählt die erste Bank aus und läßt den 6502 Prozessor darauf zugreifen. Der ANTIC "sieht" im Bank-Switch Bereich jedoch weiterhin den Inhalt des Hauptspeichers.

Die folgende Formel steuert das Bank-Switching: POKE 54017,193 + 4\*BANK + 16\*MODUS. Werte für BANK sind im Bereich 0 bis 3 und wählen den Bereich des Zusatzspeichers, der im Bank-Adressraum erscheinen soll.

BANK	Adresse im Zusatzspeicher
0	0 bis 16383
1	16384 bis 32767
2	32768 bis 49151
3	49152 bis 65535

Gültige Werte für MODUS sind im Bereich 0 bis 3 und legen fest, welcher Prozessor auf den Hauptspeicher oder auf BANK zugreift.

MODUS	6502 Zugriff	ANTIC Zugriff
0	BANK	BANK
1	NORMAL	BANK
2	BANK	NORMAL
3	NORMAL	NORMAL

**SPEICHERSTELLE 54017**

Bit Nummer:	7	6	5	4	3	2	1	0
Wert: (wenn 128	64	32	16	8	4	2	1	
Bit gesetzt)	1	1	X	X	X	X	X	1
			Video	CPU	Bank	Bank		
			Bank	Bank	LSB	MSB		
			(VBE)	(CBE)				

(Achtung, Obacht, Vorsicht!!!. Im englischen Original Tabelle ansehen, wegen zusätzlicher Zeilen: Video Bank-, CPU Bank ENABLE = Video Bank, CPU Bank einschalten.)

Bits 0, 1, 6 und 7 müssen immer gesetzt (1) sein. Sind VBE oder CBE gesetzt, wird der Hauptspeicher angesprochen. Ist ein Bit gelöscht (0), greift der jeweilige Prozessor im Adressbereich von \$4000-\$7FFF auf die gewählte Speicherbank zu.

**C. ATASCII Zeichensatz**



Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastensan Schlag	Europäisches Zeichen
0	0		Control ,	á
1	1		Control A	ù
2	2		Control B	Ñ
3	3		Control C	É
4	4		Control D	Ç
5	5		Control E	ô
6	6		Control F	ò
7	7		Control G	ì
8	8		Control H	£
9	9		Control I	ï
10	A		Control J	ü
11	B		Control K	ä
12	C		Control L	Ö
13	D		Control M	ú
14	E		Control N	ó
15	F		Control O	ö
16	10		Control P	Ü
17	11		Control Q	â

**Anmerkungen:**

1. ATASCII bedeutet ATARI ASCII. Buchstaben und Zahlen entsprechen dem ASCII-Zeichensatz. Einige der Spezialzeichen haben jedoch eine andere Funktion.

2. Wenn nicht anders angegeben, stellen die Werte von 128 bis 255 die inversen Zeichen der Werte 1 bis 127 dar.

3. Addieren Sie 32 zum Wert der Großbuchstaben um den entsprechenden Kleinbuchstaben zu erhalten.

4. Um den ATASCII-Wert zu erhalten, geben Sie im Direkten Modus PRINT ASC ("...") ein. Setzen Sie zwischen die Anführungszeichen den Buchstaben oder das Graphiksymbol.

5. Die normal dargestellten Zeichen werden weiß auf schwarzem Grund, inverse Zeichen dagegen schwarz auf weißem Grund dargestellt.

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastensanschlag	Europäisches Zeichen
18	12	—	Control R	ú
19	13	+	Control S	î
20	14	•	Control T	é
21	15	■	Control U	è
22	16	▮	Control V	ñ
23	17	⌄	Control W	ê
24	18	⌅	Control X	â
25	19	▮	Control Y	à
26	1A	⌆	Control Z	Å
27	1B	⌇	Esc Esc	
28	1C	↑	Esc Control -	
29	1D	↓	Esc Control =	
30	1E	←	Esc Control +	
31	1F	→	Esc Control *	
32	20	□	Space bar	
33	21	!	Shift 1	
34	22	"	Shift 2	
35	23	#	Shift 3	
36	24	\$	Shift 4	
37	25	%	Shift 5	
38	26	&	Shift 6	
39	27	:	Shift 7	
40	28	(	Shift 9	
41	29	)	Shift 0	
42	2A	*	*	
43	2B	+	+	
44	2C	?	,	
45	2D	...	-	
46	2E	°	.	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastensschlag	Europäisches Zeichen
47	2F	⎵	/	
48	30	0	0	
49	31	1	1	
50	32	2	2	
51	33	3	3	
52	34	4	4	
53	35	5	5	
54	36	6	6	
55	37	7	7	
56	38	8	8	
57	39	9	9	
58	3A	;	Shift ;	
59	3B	;	;	
60	3C	<	<	
61	3D	=	=	
62	3E	>	>	
63	3F	?	Shift /	
64	40	@	Shift 8	
65	41	A	A	
66	42	B	B	
67	43	C	C	
68	44	D	D	
69	45	E	E	
70	46	F	F	
71	47	G	G	
72	48	H	H	
73	49	I	I	
74	4A	J	J	
75	4B	K	K	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenschlag	Europäisches Zeichen
76	4C	L	L	
77	4D	M	M	
78	4E	N	N	
79	4F	O	O	
80	50	P	P	
81	51	Q	Q	
82	52	R	R	
83	53	S	S	
84	54	T	T	
85	55	U	U	
86	56	V	V	
87	57	W	W	
88	58	X	X	
89	59	Y	Y	
90	5A	Z	Z	
91	5B	[	Shift ,	
92	5C	\	Shift +	
93	5D	]	Shift .	
94	5E	^	Shift *	
95	5F	_	Shift -	
96	60	◆	Control .	
97	61	a	a	
98	62	b	b	
99	63	c	c	
100	64	d	d	
101	65	e	e	
102	66	f	f	
103	67	g	g	
104	68	h	h	
105	69	i	i	



Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenschlag	Europäisches Zeichen
106	6A		j	
107	6B		k	
108	6C		l	
109	6D		m	
110	6E		n	
111	6F		o	
112	70		p	
113	71		q	
114	72		r	
115	73		s	
116	74		t	
117	75		u	
118	76		v	
119	77		w	
120	78		x	
121	79		y	
122	7A		z	
123	7B		Control ;	Ä
124	7C		Shift =	
125	7D		Esc Control < OR Esc Shift <	
126	7E		Esc Delete Bk Sp	
127	7F		Esc Tab	
128	80		Control ,	
129	81		Control A	
130	82		Control B	
131	83		Control C	
132	84		Control D	
133	85		Control E	
134	86		Control F	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenanschlag	Europäisches Zeichen
135	87		<input checked="" type="checkbox"/> Control G	
136	88		<input checked="" type="checkbox"/> Control H	
137	89		<input checked="" type="checkbox"/> Control I	
138	8A		<input checked="" type="checkbox"/> Control J	
139	8B		<input checked="" type="checkbox"/> Control K	
140	8C		<input checked="" type="checkbox"/> Control L	
141	8D		<input checked="" type="checkbox"/> Control M	
142	8E		<input checked="" type="checkbox"/> Control N	
143	8F		<input checked="" type="checkbox"/> Control O	
144	90		<input checked="" type="checkbox"/> Control P	
145	91		<input checked="" type="checkbox"/> Control Q	
146	92		<input checked="" type="checkbox"/> Control R	
147	93		<input checked="" type="checkbox"/> Control S	
148	94		<input checked="" type="checkbox"/> Control T	
149	95		<input checked="" type="checkbox"/> Control U	
150	96		<input checked="" type="checkbox"/> Control V	
151	97		<input checked="" type="checkbox"/> Control W	
152	98		<input checked="" type="checkbox"/> Control X	
153	99		<input checked="" type="checkbox"/> Control Y	
154	9A		<input checked="" type="checkbox"/> Control Z	
155	9B	EOL	<input checked="" type="checkbox"/> Return	
156	9C		<input checked="" type="checkbox"/> Esc Shift Delete Bk Sp	
157	9D		<input checked="" type="checkbox"/> Esc Shift >	
158	9E		<input checked="" type="checkbox"/> Esc Control Tab	
159	9F		<input checked="" type="checkbox"/> Esc Shift Tab	
160	A0		<input checked="" type="checkbox"/> Space bar	
161	A1		<input checked="" type="checkbox"/> Shift 1	
162	A2		<input checked="" type="checkbox"/> Shift 2	
163	A3		<input checked="" type="checkbox"/> Shift 3	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastensanschlag	Europäisches Zeichen
164	A4	⌘	☑ Shift 4	
165	A5	⌘	☑ Shift 5	
166	A6	⌘	☑ Shift 6	
167	A7	■	☑ Shift 7	
168	A8	<	☑ Shift 9	
169	A9	)	☑ Shift 0	
170	AA	*	☑ *	
171	AB	+	☑ +	
172	AC	,	☑ ,	
173	AD	-	☑ -	
174	AE	.	☑ .	
175	AF	/	☑ /	
176	B0	0	☑ 0	
177	B1	1	☑ 1	
178	B2	2	☑ 2	
179	B3	3	☑ 3	
180	B4	4	☑ 4	
181	B5	5	☑ 5	
182	B6	6	☑ 6	
183	B7	7	☑ 7	
184	B8	8	☑ 8	
185	B9	9	☑ 9	
186	BA	:	☑ Shift ;	
187	BB	;	☑ ;	
188	BC	<	☑ <	
189	BD	=	☑ =	
190	BE	>	☑ >	
191	BF	?	☑ Shift /	
192	C0	@	☑ Shift 8	
193	C1	A	☑ A	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenschnalag	Europäisches Zeichen
194	C2	B	<input type="checkbox"/> B	
195	C3	C	<input type="checkbox"/> C	
196	C4	D	<input type="checkbox"/> D	
197	C5	E	<input type="checkbox"/> E	
198	C6	F	<input type="checkbox"/> F	
199	C7	G	<input type="checkbox"/> G	
200	C8	H	<input type="checkbox"/> H	
201	C9	I	<input type="checkbox"/> I	
202	CA	J	<input type="checkbox"/> J	
203	CB	K	<input type="checkbox"/> K	
204	CC	L	<input type="checkbox"/> L	
205	CD	M	<input type="checkbox"/> M	
206	CE	N	<input type="checkbox"/> N	
207	CF	O	<input type="checkbox"/> O	
208	D0	P	<input type="checkbox"/> P	
209	D1	Q	<input type="checkbox"/> Q	
210	D2	R	<input type="checkbox"/> R	
211	D3	S	<input type="checkbox"/> S	
212	D4	T	<input type="checkbox"/> T	
213	D5	U	<input type="checkbox"/> U	
214	D6	V	<input type="checkbox"/> V	
215	D7	W	<input type="checkbox"/> W	
216	D8	X	<input type="checkbox"/> X	
217	D9	Y	<input type="checkbox"/> Y	
218	DA	Z	<input type="checkbox"/> Z	
219	DB	[	<input type="checkbox"/> Shift ,	
220	DC	\	<input type="checkbox"/> Shift +	
221	DD	]	<input type="checkbox"/> Shift .	
222	DE	^	<input type="checkbox"/> Shift *	
223	DF	█	<input type="checkbox"/> Shift -	

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenanschlag	Europäisches Zeichen
224	E0	◐	<input checked="" type="checkbox"/> Control	
225	E1	a	<input checked="" type="checkbox"/> a	
226	E2	b	<input checked="" type="checkbox"/> b	
227	E3	c	<input checked="" type="checkbox"/> c	
228	E4	d	<input checked="" type="checkbox"/> d	
229	E5	e	<input checked="" type="checkbox"/> e	
230	E6	f	<input checked="" type="checkbox"/> f	
231	E7	g	<input checked="" type="checkbox"/> g	
232	E8	h	<input checked="" type="checkbox"/> h	
233	E9	i	<input checked="" type="checkbox"/> i	
234	EA	j	<input checked="" type="checkbox"/> j	
235	EB	k	<input checked="" type="checkbox"/> k	
236	EC	l	<input checked="" type="checkbox"/> l	
237	ED	m	<input checked="" type="checkbox"/> m	
238	EE	n	<input checked="" type="checkbox"/> n	
239	EF	o	<input checked="" type="checkbox"/> o	
240	F0	p	<input checked="" type="checkbox"/> p	
241	F1	q	<input checked="" type="checkbox"/> q	
242	F2	r	<input checked="" type="checkbox"/> r	
243	F3	s	<input checked="" type="checkbox"/> s	
244	F4	t	<input checked="" type="checkbox"/> t	
245	F5	u	<input checked="" type="checkbox"/> u	
246	F6	v	<input checked="" type="checkbox"/> v	
247	F7	w	<input checked="" type="checkbox"/> w	
248	F8	x	<input checked="" type="checkbox"/> x	
249	F9	y	<input checked="" type="checkbox"/> y	
250	FA	z	<input checked="" type="checkbox"/> z	
251	FB	◑	<input checked="" type="checkbox"/> Control ;	
252	FC		<input checked="" type="checkbox"/> Shift =	
253	FD	◒	Esc Control 2	

<b>Dezimal Kode</b>	<b>Hexadezimal Kode</b>	<b>ATASCII Zeichen</b>	<b>Tastensanschlag</b>	<b>Europäisches Zeichen</b>
254	FE		<input checked="" type="checkbox"/> Esc Control Delete Bk Sp	
255	FF		<input checked="" type="checkbox"/> Esc Control >	







ATARI Corp. (Deutschland) GmbH  
Frankfurter Straße 89-91 · 6096 Raunheim

Jegliche Rechte vorbehalten  
© 1985 ATARI Corp. (Deutschland) GmbH  
Änderungen vorbehalten