DAS ATARI XE HANDBUCH

ATARI®

... wir machen Spitzentechnologie preiswert.

Anmerkung

Dieses Handbuch enthält eine möglichst exakte Beschreibung des Produkts, jedoch keine Garantien für bestimmte Eigenschaften oder Einsatzerfolge. Maßgebend ist, soweit nicht anders angegeben, der technische Stand zum Zeitpunkt der gemeinsamen Auslieferung von Produkt und Handbuch durch ATARI. Da ATARI die Hard- und Software laufend überarbeitet und verbessert, werden vorangegangene oder nachfolgende Releases mit der Produktbeschreibung in diesem Handbuch in der Regel nicht in jeder Hinsicht übereinstimmen.

DAS ATARI XE HANDBUCH

ATARI®

. . . wir machen Spitzentechnologie preiswert.

Vorwort

Mit Ihrer Entscheidung für den neuen ATARI XE-Computer haben Sie eine wohlüberlegte Wahl getroffen. Leistungsfähig und zukunftsicher bietet er 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. Erfahrungsgemäß sind auftauchende Probleme meistens auf fehlerhafte Bedienung oder Nichtbeachtung von Programm-Anleitungen zurückzuführen.

Aber keine Angst: durch Eingabefehler 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 neuesten ATARI XE-Computer viel Spaß – sei es beim Selbstprogrammieren oder bei Benutzung der vielen fertigen ATARI-Computer-Programme.

ATARI Corporation (Deutschland) GmbH

INHALT

Die ATARI XE-Computer-Modelle

Die ATARI XE-Computerserie wurde aus der weltweit erfolgreichen Modellreihe ATARI 600XL/800XL unter besonderer Beachtung einer durchgehenden Systemkompatibilität weiterentwickelt; 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 werden 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.

Modell Übersicht

Serielle I/0 Buchse
Anschluß für ATARI I/0-Kabel zu Peripheriegeräten, wie Diskettenlaufwerken, Programmrecorder, Drucker und Modems.

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

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

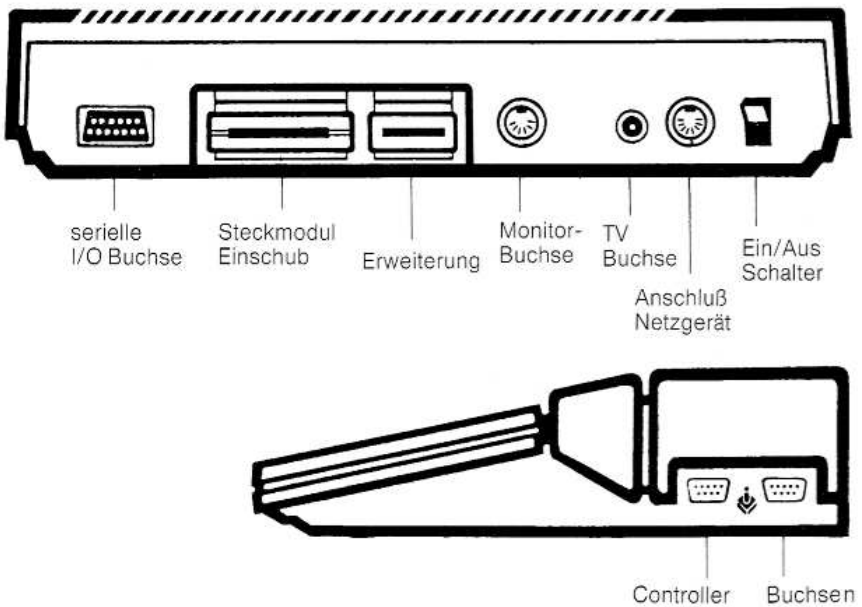
Monitor Anschlußbuchse
Zum Anschluß eines Monochromen- oder Farbmonitors über ein Monitorkabel.

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 Computer Ein und Aus.

Controller Buchsen
Zum Anschluß von Joysticks, Paddle Controllern und weiterer Peripherie. Wird nur ein Controller benutzt, diesen in Port 1 einstecken.



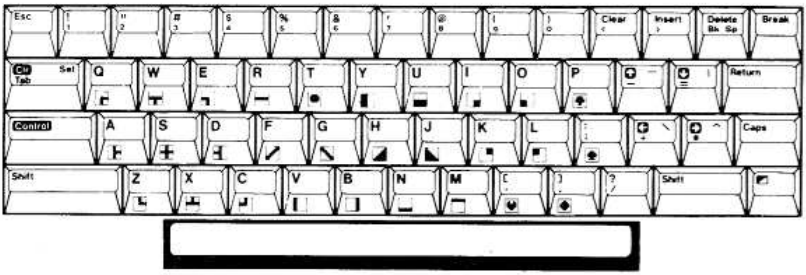
Neben dem Arbeitsspeicher zur Verarbeitung von Daten und Programmen gibt es im Computer noch einen weiteren Speicherbereich, das Betriebssystem. Dieses steuert, rechnet und vergleicht. Um auf beide Bereiche zugreifen zu können und so mit dem Computer zu "sprechen" bedarf es einer Programmiersprache.

Einfache und leicht zu erlernende Programmier-Sprachen 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. Assembler erfordern einen etwas höheren Lernaufwand, ermöglichen aber durch die computergerechte Darstellung auch schnelle Programm-Abläufe.

TASTATUR

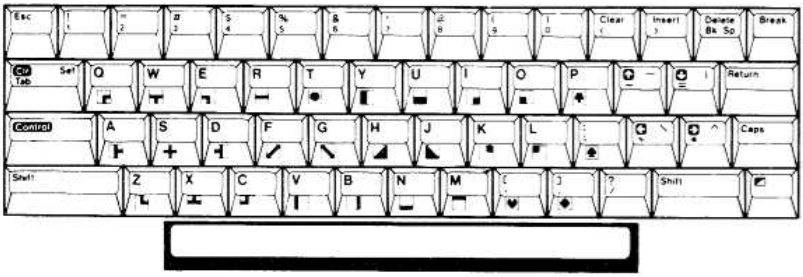
Die Tastatur entspricht einer Schreibmaschinentastatur mit Groß- und Kleinbuchstaben und zusätzlich einigen Graphik-Tasten. Genau genommen enthält die Tastatur drei verschiedene Tasten-Sätze, Normal-, Graphik- und Umlauttastatur.

Normaltastatur



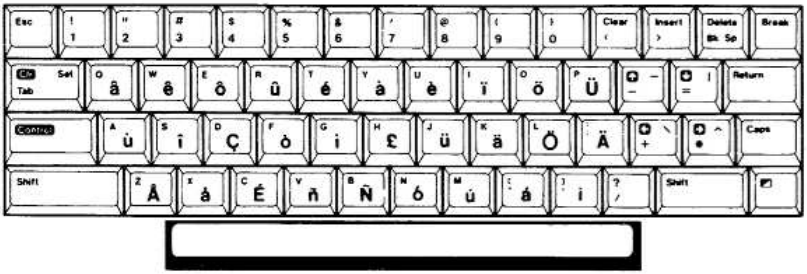
Mit der CAPS-Taste kann jeweils auf Groß- bzw. Kleinbuchstaben umgeschaltet werden.

Graphiktastatur



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

Tastatur mit Umlauten



Auf diesen internationalen Zeichensatz kann umgeschaltet werden durch Eingabe von POKE 756,204 (RETURN). Das gewünschte Sonderzeichen 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:

Reset
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.

Help
ruft in einigen Programmen Erläuterungen und Bedienungshinweise ab.

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

Option
wählt Variationen innerhalb eines Programmes.

Select
erlaubt die Auswahl der Anwendungen innerhalb eines Programmes.

Start
beginnt ein Spiel oder einen Programm-Teil.

BREAK dient zum Abbruch selbstgeschriebener Programme.

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

Mit **CONTROL/INSERT** können Leerzeichen bzw. Zwischenräume eingefügt 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 ständiger 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 Summer 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. Diese Taste finden Sie direkt unterhalb der CAPS-Taste. In manchen Programmen wird sie ATARI-Taste genannt.

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

Peripheriegeräte

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

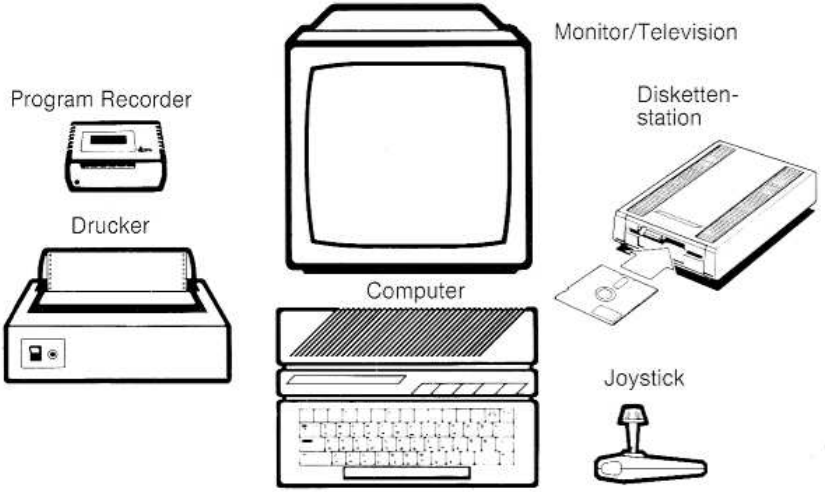
TV-Geräte oder Monitor
Zur Bildschirmdarstellung können Sie sowohl ein TV-Gerät oder einen Computermonitor anschließen. Ein Monitor wird empfohlen, da er eine schärfere Darstellung bietet. Es kann sowohl ein Farbmonitor oder ein monochromer Monitor verwendet werden. Viele Programme verwenden jedoch die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

Programm-Recorder wie z. B. das Modell ATARI XC 12 bieten bei geringen Anschaffungskosten umfassende Möglichkeiten zum Speichern und Laden von Programmen und Daten. Als Datenträger werden Tonband-Cassetten (Compact Cassetten z.B. C30 oder C60) in Ferro-Qualität empfohlen. Chromdioxyd-Cassetten sind aus technischen Gründen weniger geeignet. Die Speicherkapazität beträgt je nach Bandlänge bis zu 100 KBytes, das entspricht ungefähr 60 Schreibmaschinenseiten. Programm-Recorder sind weniger geeignet, wenn es auf schnellen wahlfreien Datenzugriff ankommt.

Disketten-Stationen
ATARI Diskettenstationen bieten schnellen und sicheren 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 entweder direkt anschlußfähig oder über eine Centronics Schnittstelle zu betreiben sind.

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



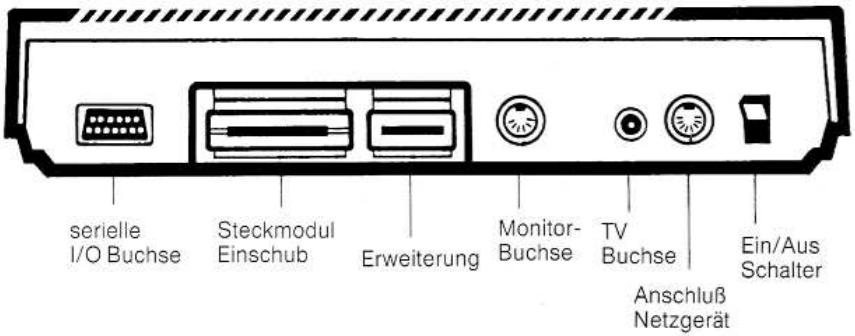
Inbetriebnahme

1. Schieben Sie den runden Stecker des Netzgerätes in die mit 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 etwa 5 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.

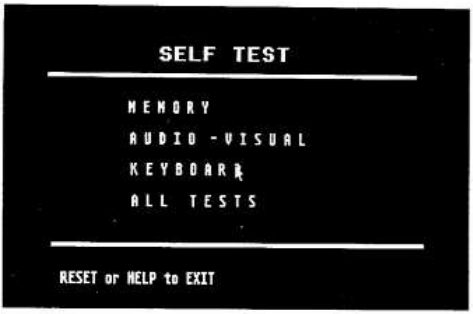
Sie können diesen Test auch selbst starten,

- 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.



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.

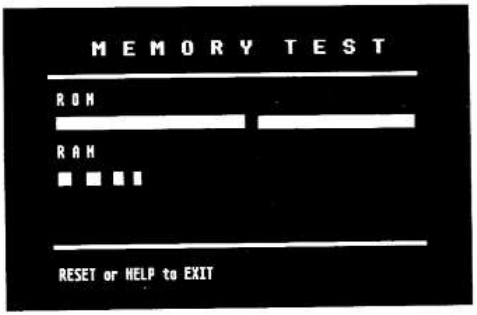
Während des ROM/RAM-Tests werden 40 bzw. 48 KByte des Schreib-/Lesespeichers geprüft. Die verbleibenden 16 KByte sind nur mit spezieller Software zugänglich. 48 kleine Quadrate, von denen jedes 1K RAM repräsentiert und 2 größere Balken (als Zeichen für je eine 16K ROM-Speicherbank), signalisieren die Funktion des Speichers. Auch hier bedeutet Rot, daß etwas nicht stimmt und Sie Ihr Service Center aufsuchen sollten.

Der RAM Test kann bei entsprechender Durchführung zwei verschiedene richtige Ergebnisse zeigen:

1. Nach dem Einschalten erscheint am linken oberen Bildrand die Meldung "READY". Dies bedeutet, daß während der Rechnerinitialisierung das Basic vom Betriebssystem in den Arbeitsspeicher (RAM !) kopiert wird. Dabei werden 8 K ! des Rambereiches belegt. Sollten Sie nun über den Befehl " BYE " in den SELF – TEST MODUS umschalten, bleiben noch 40 KByte, die als Quadrate angezeigt werden.

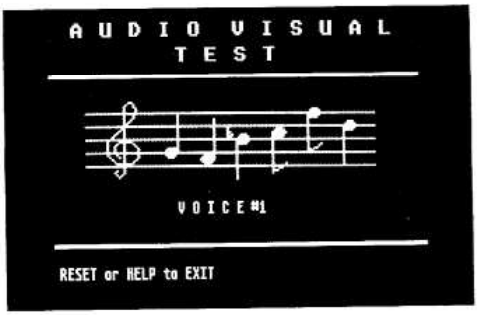
2. Falls Sie Ihren ATARI Computer bei gedrückter OPTION TASTE einschalten, wird das integrierte Basic ausgeblendet. Der entsprechende RAM Bereich bleibt frei. Beim RAM TEST werden nun 48 Quadrate auf dem Bildschirm angezeigt.

Hinweis: Beim ATARI 130 XE Computer wird nur die erste Speicherbank im Self Test-Modus geprüft. Es erscheinen daher ebenfalls nur 40 bzw. 48 Quadrate auf dem Bildschirm.



Speichertest
(Memory-Test)

Ton/Graphik Test
Der ATARI XE-Computer besetzt vier unabhängig von einander arbeitende Tonkanäle und hochauflösende Graphik. Über der Nummer des gerade geprüften Kanals erscheinen fünf Notenlinien mit Violinschlüssel. Für jeden der vier Kanäle werden sechs Noten darestellt und gespielt. Bitte prüfen Sie bei fehlendem Ton die eingestellte Lautstärke Ihres Fernsehgerätes. Die Farben innerhalb eines Testlaufs 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.

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 herstellerspezifischen Dialekten. Das 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 wichtigsten 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 Ihren ATARI Computer wie beschrieben mit einem TV-Gerät oder Monitor verbünden und eingeschaltet. Das eingebaute ATARI BASIC ist aktiviert und meldet sich mit:

READY

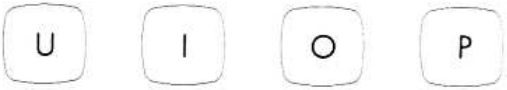
Das kleine (weiße) Quadrat unterhalb von READY heißt "Cursor". Es gibt die Bildschirmposition an, an der das nächste Zeichen stehen wird.

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 114 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.

Einführungskurs in das ATARI BASIC

PRINT

Mit der Anweisung PRINT und einer beliebigen Ergänzung wird vom Computer etwas auf den Bildschirm ausgegeben. 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                (RETURN)
PRINT 13 – 14              (RETURN)
PRINT 4*5                  (RETURN)
PRINT 36/6                 (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
```

Zur Potenzrechnung sind die Tasten SHIFT und ^ zu drücken.

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

Einige Beispiele:

```
PRINT "HALLO"              (RETURN)
PRINT"DAS IST EIN SATZ"    (RETURN)
PRINT "12 * 7"             (RETURN)
```

Der Bildschirm zeigt folgendes:

```
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 dem Bildschirm ausgibt.

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 Betätigung 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 dem BASIC-Befehl RUN gestartet. Nach jeder Eingabe von RUN läuft das Programm erneut ab.

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

Ein Beispiel:

```
10 PRINT "Guten Tag"                    (RETURN)
20 PRINT "WALTER"                       (RETURN)
30 PRINT "WIE GEHT ES DIR ?"            (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 zu schreiben.

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: `20 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 "WALTER"
```

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` (RETURN)

folgende Zeile auf dem Bildschirm:

`20 PRINT "WALTER"` (RETURN)

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` (RETURN)

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:"` (RETURN)

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 dem Bildschirm schreiben, da er kein Programm zum Listen mehr im Speicher hat.

```
READY
NEW                                  (RETURN)
READY
LIST                                 (RETURN)
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 es sich um einen falsch geschriebenen Basic Befehl handelt, 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                           (RETURN)
ERROR LIST 20;30
```

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 welcher der Computer den Fehler entdeckt, muß nicht immer mit der wahren Fehlerposition übereinstimmen.

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 Zeile, 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 Fehlercodes 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 "WALTER"
```

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

```
RUN
HALLO
WALTER
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"
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 waagerecht 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. Näheres finden Sie unter der Rubrik Tips für Fortgeschrittene in diesem Handbuch.

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*2*2
3. Multiplikation bzw. Division
4. Addition bzw. Subtraktion

Beispiele:

4*5+2=22
4*(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 500000000000 (Fünfhundert Milliarden) druckt der Computer 5.OE+11
Die Darstellung bedeutet 5 mal 10 hoch 11, oder 5*1011

Bei dieser Schreibweise wird die 5 als Mantisse und die +11 als Exponent bezeichnet. Beide werden durch die 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 soviele 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: 31459200000

- **bei negativem Exponenten**

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

Beispiel: 4.674E-8
Mantisse : 4.674
Verschiebung des Dezimalpunktes um acht Stellen nach links.
Dezimalzahl: 0.00000004674

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.
5. Es dürfen keine Basic Befehle als Variablennamen verwendet werden.

Mit dem Basic Befehl

LET

kann man einer Variablen eine Zahl oder eine Zeichenfolge zuweisen.

Mit der Zeile:
```
LET ZAHL=27                    (RETURN)
```

wird der Variablen ZAHL der Wert 27 zugewiesen.

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

```
PRINT ZAHL                     (RETURN)
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 TEXT$ wird wie folt der Text STRINGVARIABLE zugewiesen:

```
LET TEXT$ ="STRINGVARIABLE"    (RETURN)
```

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

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

```
PRINT TEXT$
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 TEXT$ auf eine Länge von 20 Zeichen dimensioniert wird, sieht wie folgt aus:

```
10 DIM TEXT$ (20)
```

In der Stringvariablen TEXT$ 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 nicht vorher 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 NAME$ (20)
20 PRINT "BITTE GEBEN SIE IHREN NAMEN EIN"
30 INPUT NAME$
40 PRINT "SIE HEISSEN: ";NAME$
```

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 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. heruntergezählt 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-Anwendung kann eine bestimmte Schrifttweite 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 *** verschachtelte FOR-NEXT-Schleifen
20 FOR AUSSEN=1 TO 4
30 PRINT "Äussere 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.

```
ÄUSSERE SCHLEIFE
INNERE SCHLEIFE
INNERE SCHLEIFE
ÄUSSERE SCHLEIFE
.
.
.
.
```

GOTO

Bis jetzt wurde ein Programm vom Computer immer chronologisch 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 GRÖSSER B ALS B"
60 IF A>B THEN PRINT "A IST KLEINER ALS 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$(5)
20 PRINT "GEBEN SIE DAS CODEWORT EIN"
30 INPUT C$
40 IF C$ >< "ATARI" THEN PRINT "FALSCHES CODEWORT": GOTO 20
50 PRINT "CODEWORT AKZEPTIERT"
```

READ/DATA/RESTORE

Der READ-Befehl ist eine dritte Möglichkeit, neben LET und INPUT, einer Variablen einen Wert zuzuweisen. 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 umformulieren:

```
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.

Beispiel

```
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 6,12,18,24,30,36,42,48,54,60,9,999
```

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 Stringvariablen einlesen.

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

In folgendem Programm werden mit einer READ-Anweisung gleichzeitig zwei Werte aus den DATA-Zeilen gelesen. Dies wird dadurch erreicht, daß hinter dem READ-Befehl zwei Variablen angegeben werden. Es können mehr als zwei Variablen verwendet werden. Jede Variable muß dabei durch ein Komma von der nächsten getrennt werden.

```
10 DIM O$(20), OBST$(20)
20 PRINT "GEBEN SIE EINE OBSTSORTE EIN"
30 INPUT O$
40 RESTORE
50 READ OBST$, PREIS
60 IF OBST$="END" THEN GOTO 100
70 IF O$=OBST$ THEN PRINT "DER PREIS BETRÄGT ";PREIS;" DM": GOTO
20
80 GOTO 50
100 PRINT "FALSCHE SORTE":GOTO 20
120 DATA APFEL, 1.34, APFELSINE, 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 VERIABLEN

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(O)
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),
```

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

Das folgende Programme 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 Matrixen 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.

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(ART,0)=PR
70 PRINT "LAGERBESTAND ";:INPUT LA 80 ART(ART,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 "STÜCK ";:INPUT ST
140 IF ART /ART,1)-ST>0 THEN PRINT "NICHT GENÜGEND ARTIKEL IM
LAGER!":GOTO 110
150 LET PR=ART(ART,0)*ST
160 LET ART(ART,1)=ART(ART,1)-ST
170 PRINT "SUMME: ";PR:" DM"
180 PRINT "NOCH ";ART(ART,1);" STÜCK 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 PROGRAMMES"
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 Unterprogrammablauf 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:

```
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 BIS 5 EIN"
70 INPUT X
80 IF X>1 OR X<5 THEN PRINT "UNGÜLTIGE 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 BIS 5 EIN"
70 INPUT X
80 IF X<5 THEN PRINT "UNGÜLTIGE 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
```

GRAPHICS
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. Lösen 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 #6; erweitert werden.

Beispiel:

```
PRINT #6; "Ich kann es"
```

Der Text muß nun oben auf dem schwarzen Hintergrund in orange 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;"grüner 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-Anweisung 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 H=0 TO 500:NEXT W
80 END
```

Beachten Sie aber, daß Sie bei der POSITION-Anweisung 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 3-8

Die Graphikstufen 3 bis 8 ermöglichen das Zeichnen und Malen auf dem Bildschirm. Die Unterschiede der einzelnen Stufen bestehen in der unterschiedlichen Farbwahlmöglichkeit 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 detailliert 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 nachfolgenden Zahlen bestimmten Stelle des Bildschirms einen Punkt sehen.

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 auf 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 24,16
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. 19,15
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, wird das Ende des Programms verzögert.

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

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

Mit Hilfe des folgende Programms können beliebige Formen auf den Bildschirm gezeichnet werden. Es macht deutlich, daß man bestimmte Figuren durch geeignete Formeln berechnen kann. (Die Anweisung 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 ELLIPSE IN GR.8
20 GRAPHICS 8+16
30 POKE 709,14
40 COLOR 1
50 REM ELLIPSENBERECHNUNG
60 FOR ES=1 TO 400
70 LET I=I+0.05
80 LET X=SIN(I)*80:LET Y=COS(I)*15
90 PLOT X+160,Y85
100 NEXT ES
110 GOTO 110
```

SETCOLOR

In allen Graphikstufen haben Sie die Möglichkeit, in verschiedenen Farben zu malen oder zu schreiben. Diese Möglichkeit kennen Sie bereits von dem Befehl GRAPHICS 0-2 . Man kann die Farben auch 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.

Setcolor (Farbregister)	Default	Helligkeit	Farbe
0	2	8	orange
1	12	10	grün
2	9	4	dunkelblau
3	4	6	rosa,rot
4	0	0	schwarz

Die Default Werte bleiben so lange erhalten, bis ein entsprechender SETCOLOR Befehl gegeben wird.

Beispiel:

GRAPHICS O

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 GR.0 ist die Schriftfarbe 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

Es ergeben sich 16 Farben in 16 Helligkeitsstufen, also 256 Farben. (Anm.:Diese können je nach Typ und Wiedergabequalität des verwendeten Fernsehgerätes bzw. Monitors geringfügig variieren.)

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 H=1 TO 80:NEXT W
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 anzuwählen. Befindet man sich z. B. in der Graphikstufe 3, stehen vier Farbregister für das Bild zur Auswahl.

GRAPHICS 3

Farbregister	Ausgangsfarbe	COLOR Wert
0	Orange	COLOR 1
1	Hellgrün	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 den Bildschirm.

Beispiel:

```
10 LET X=0:LET Y=0:LET H=20:LET N=12
20 FOR I=3 TO < STEP 2
30 GRAPHICS I+16
40 SETCOLOR 0,0,0
50 SETCOLOR 1,15,14
60 REM VIERECK70 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,16+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,F,H
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 Farbregistern 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 Farbregister. Folgende Übersicht zeigt, welcher POKE-Wert für welches Farbregister 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
	2	Textfenster	710
8	2	Hintergrund	710
	1	Graphikpunkt	709
	4	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 Farbabstufungen.

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 1)
9*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 X=0 TO 255
30 PRINT X
40 POKE 712,X
50 FOR W=1 TO 50:NEXT W
60 NEXT X
```

Sie können dieses Programm an jeder Stelle stoppen, indem Sie die CONTROL-Taste gedrückt halten und zusätzlich die Taste "1" 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.

`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

Außerdem können über bestimmte Register einzelne Tasten abgefragt werden. 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 von 0-15	fester Wert

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 aufwendig, die Farbregister zu füllen und sie später auszuwählen. Dabei soll die folgende Tabelle helfen.

	Ausgangsfarbe		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<

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.	Anwendung	Horizontal	Vertikal Zeilen/ geteilt.)	Vertikal (ohne Text) (Zeilen unget.)	Anzahl der	Erforderliche RAM Get.	Unget.
0	Text	40	–	24	1		992
1	Text	20	20	24	5	674	672
2	Text	20	10	12	5	424	420
3	Graphik	40	20	24	4	434	432
4	Graphik	80	40	48	2	694	696
5	Graphik	80	40	48	4	1174	1176
6	Graphik	160	80	96	2	2174	2184
7	Graphik	160	80	96	4	4190	4200
8	Graphik	320	160	192	1/2	8112	8138
9	Graphik	80	–	192	1/16		8138
10	Graphik	80	–	192	9		8138
11	Graphik	80	–	192	16		8138
12	Graphik	40	20	24	5	1154	1152
13	Graphik	40	10	12	5	664	660
14	Graphik	160	160	192	2	4270	4296
15	Graphik	160	160	192	4	8112	8138

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

SOUND
Neben der Graphik bietet der ATARI-Computer eine Vielzahl an Geräusch- bzw. Tonmöglichkeiten. Sie werden durch vier voneinander unabhängigen Tonkanälen/-generatoren erzeugt.

Um einen Ton zu erzeugen, ist eine SOUND-Anweisung mit vier Werten zu schreiben.

Beispiel:

```
SOUND 0,29,10,10
```

Diese Anweisung ergibt das hohe C. Dabei bestimmt die erste Zahl das Tonregister, d. h. welcher Tongenerator den Ton erzeugt. Sie darf die Werte von 0 bis 3 annehmen.

Beispiel:

```
SOUND 0 50,10.6
SOUND 1,100,10,6
SOUND 2,150,10,6
SOUND 3,200,10,6
```

Die zweite Zahl bestimmt die Note (Tonhöhe). Der Wert darf zwischen 0 und 255 liegen.

Beispiel:

```
10 FOR N=0 TO 255
20 PRINT N
30 SOUND 0,N,10,10
40 FOR W=0 TO 00:NEXT W
50 NEXT N
60 END
```

Die dritte Zahl bestimmt die Verzerrung. Der Wert darf zwischen 0 und 14 liegen, wobei 10 ein reiner Ton ist.

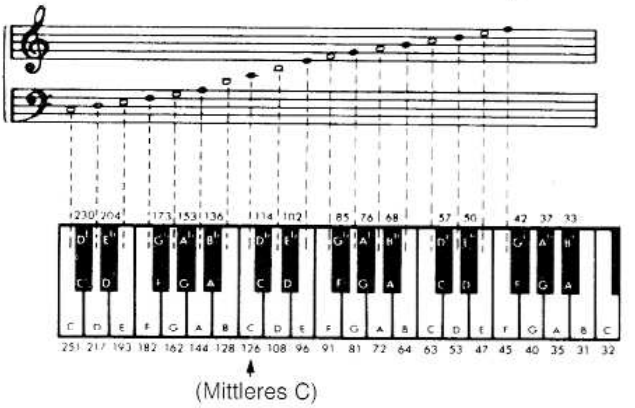
```
10 FOR T=0 TO 14 STEP 2
20 SOUND 0,29,T,10
30 FOR W=0 TO 00:NEXT W
40 NEXT T
50 GOTO 10
```

Die vierte Zahl bestimmt, zusätzlich zum Regler des Fernsehgerätes, die Lautstärke. Sie darf Werte von 0 bis 15 annehmen.

Beispiel

```
10 FOR L=0 TO 15
20 SOUND 0,29,10,L
30 FOR W=0 TO 80:NEXT W
40 NEXT L
50 FOR L=15 TO 0 STEP -1
60 SOUND 0,29,10,L
70 FOR W=0 TO 00:NEXT W
80 NEXT L
90 GOTO 10
```

Beziehung der Klaviertasten zu der Tonleiter.



Abfrage der Steuerungen

Joystick/Steuerknüppel
Der Joystick ist ein einfach zu bedienendes Eingabegerät und darum besonders für Videospiele geeignet. Damit der Joystick in eigenen Programmen verwendet werden kann, sind im ATARI-Basic zwei Anweisungen vorgesehen, um die Werte des Steuerknüppels abzufragen.

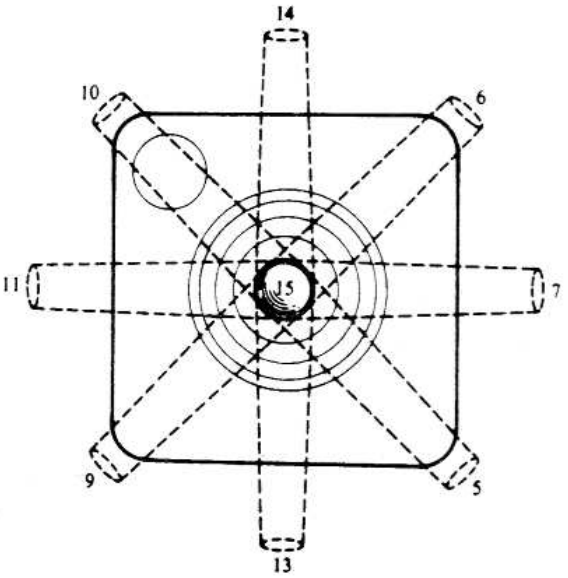
STICK
Es können zwei Joysticks an den ATARI-Computer angeschlossen werden. Diese können mit der STICK-Anweisung abgefragt werden.

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 #6;ST
150 FOR W=0 TO 300:NEXT W
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(0)
```
und
```
STRIG(1)
```

abgefragt. Bei der Abfrage können die Werte 0 und I im Register stehen. Der Wert 0 steht für den gedruckten Zustand, der Wert I 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 "Feuerkopf" des Paddle-Controllers kann wie der des Joysticks die Werte 0 und I im Register stehen haben. Der Wert wird über die PTRIG-Anweisung abgefragt.

Beispiel:

```
10 SETCOLOR 2,PA1: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
```

Übungsbeispiele

Die folgenden Beispiele (Listings) sollen Ihnen helfen, die im vorangegangenen BASIC-Kurs erworbenen Kenntnisse zu vertiefen. Wir empfehlen, die Programme vollständig abzutippen 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 alle Programmzeilen mit ins 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/19),8
90 NEXT I:SETCOLOR 2,10,10
100 GOTO 40
```

BEISPIEL 2

```
10 REM *** SIRENE 2 ***
20 GRAPHICS 0:? "LÄNGE";:INPUT AA:GRAPHICS 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
```

BEISPIEL 3

```
10 REM *** TUERKLINGEL ***
20 GRAPHICS 18
30 POSITION 1,6:PRINT #6;" ^  ^ TUERIKLINGEL **"
40 FOR I=1000 TO 200 STEP -4
50 IF I>=1000 AND I<<700 THEN SOUND 0,150,10,INT(I-700)/20)
60 IF I>=800 AND I<500 THEN SOUND 1,110,10,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
```

BEISPIEL 4

```
10 REM *** SPIELEN EINER NOTENFOLGE ***
20 DIM A(10)
30 GRAPHICS 0:PRINT "BITTE GEBEN SIE
zehn ZAHLEN EIN":?
40 FOR I=1 TO 10:PRINT "ZAHL";I;" ";:
INPUT B:A(I)=B:NEXT I
50 GRAPHICS 0:POSITION 15,12:? "ICH SPIELE"
60 FOR I=1 TO 10:SOUND 0,A(I),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) verändern.

Vor Ausführung werden Sie zur Eingabe von zehn Werten gefragt.

BEISPIEL 5

```
10 REM *** MUSIK PER TASTATUR ***
20 GRAPHICS 18
30 POSITION 7,5:? #6;"MUSIK"
40 OPEN #1,4,0,"K:"
50 FOR I=1 TO 10000
60 GET #1,A:A=A-65:A=A*10+5
70 SOUND 1,A,10,15:POKE 708,A:NEXT I
80 CLOSE #1
```

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

Beispiel 6

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

Grafikstufe 10 oder 11

Paarweise Veränderung von +1 bis -5

Paarweise Veränderung von +1 buis -5

Beispiel 7

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

Für den Aufbau einer Adressenliste ist dieses Programm nicht vorgesehen. Dazu wären umfangreiche Veränderungen und Programmiererfahrung 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 in Zeile 600 werden zum Datenaustausch mit einem Peripheriegerät genutzt. Ihre ausführliche Besprechung würden den Rahmen dieses BASIC – Kurses überschreiten. Wir verweisen deshalb auf die Literaturempfehlungen.

Beispiel 8

```
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*6.293)
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 O 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:DRAWTO 79,0:DRAWTO
79,39:DRAWTO 0,39:DRAWTO 0,0
40 POKE 752,1:? " SCORE ";P;"
ZEIT ";S;
50 POKE 559,46
60 POKE 704,188
70 I=PEEK(106)-8
80 POKE 54279,I
90 POKE 53277,3
100 POKE 53256,I
110 PMBASE=I*256
120 FOR I=PMBASE+512 TO PMBASE+640
:POKE I,0:NEXT I d
130 POKE PMBASE+512+85,60:POKE
PMBASE+513+85,60
140 FOR I=PMBASE+384 TO PMBASE+512:
POKE I,0:NEXT I
150 POKE PMBASE+384,3:S=999
160 YV=INT(1*RND(0)+2):XV=INT(-4*RND
(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+YM1,0:A=A+1:S=S-1
220 IF S>10 THEN ?"";
230 IF S>100 THEN ?"";
240 POKE 752,1:?" ";S;"";
250 XM=XM+XV:YM1=YM:YM=YM+YV:IF XM>50
OR XM<205 THEN XV=-XV:SOUND 0,20,10,15
;SOUND 0,0,0,0:GOTO 210
260 IF S=0 THEN GOTO 340
270 IF YM>19 OR YM<92 THEN YV=-YV:
SOUND 0,20,10,15:
SOUND 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 POKE
752,2:P=P+1:?" SCORE 2";P;" ZEIT
";: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*RND(0)+2):GOTO 210
340 POKE 53277,0:POKE 53265,0:POKE 532
61,0
350 GRAPHICS 18
360 POSITION 2,3:? #6;"PUNKTE:";P
```

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 ie Taste ESC drücken, dann gleichzeitig CONTROL und CLEAR.

```
370 POSITION 2,9:? #6;"START= Neues
Spiel"
380 POSITION 2,11:? #6;"SELECT=Ende"
390 IF PEEK(53279)=6 THEN CLR :GOTO 20
400 IF PEEK(53279)=5 THEN GRAPHICS 0:END
410 GOTO 390
```

Beispiel 10

```
10 REM *** GRAPHIK-KREISE I ***
20 GRAPHICS 31:COLOR 1
30 POKE 708,91:POKE 709,181:POKE
710,131
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*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:DRAWTO 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:DRAWTO 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:CCOLOR 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)*255)
370 IF A=2 THEN POKE 709,INT(RND(0)*255)
380 IF A=3 THEN POKE 710,INT(RND(0)*255)
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:
SETCOLOR 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:DRAWTO A+15*SIN(ZEI+0.01*
6.283),B+30*COS(ZEI+0.01*6.283)
990 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*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:B129: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*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=3 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$(38):GRAPHICS 7
30 SETCOLOR 0,1,5:SETCOLOR 1,3,1:
SETCOLOR 2,0,6:SETCOLOR 4,0,6
40 A=157:B=10:C=30:D=33:RESTORE
50 X=1:E=A:F=B:G=C:ANDE=21:ANFANG=
1:GOSUB 360
60 X=2:E=A:F=C:G=D:ANDE=21:ANFANG=
1:GOSUB 360
70 FOR I=1 TO 6
80 A$="":READ A$:LANG=LEN(A$)
90 FOR J=1 TO LANG:FOR K=1 TO 50:NEXT K
100 ? A$(J,J);:NEXT J:?:?
110 NEXT I:N=21
120 ?:? "WIEVIELE STREICHHÖLZER";
130 INPUT STREICH
140 IF STREICH ><INT(STREICH) THEN ?:?
"NUR GANZE ZAHLEN, SIE...":?:GOTO 120
150 IF STREICH>1 OR STREICH <4 THEN ? :
?"OHNE MICH, SO GEHT ES JA NICHT":?:
GOTO 120
160 N=N-STREICH
170 X=3:E=A:F=B:G=D:ANDE=21-N:ANFANG=1
:GOSUB 360
180 STREICH=5-STREICH
190 FOR I=1 TO 300:NEXT I
200 ?:? "ICH NEHME";STREICH
210 N=N-STREICH
220 X=3:E=A:F=B:G=D:ANDE=21-N:ANFANG=1
:GOSUB 360
230 IF N<1 THEN 120
240 ?" *** ICH HABE GEWONNEN ***"
250 ?:?"NOCH EIN SPIEL ? (J,N)";
260 A$="":INPUT A$
270 IF A$="J" THEN?"CHR$(125)":GOTO 30
280 GRAPHICS 0:END
290 ?:?"HERZLICHEN GLÜCKWUNSCH"
300 DATA DAS SPIEL BEGINNT !!!!!!
310 DATA WIR HABEN 21 STREICHHÖLZER
320 DATA WER DAS LETZTE HOLZ NIMMT
330 DATA HAT VERLOREN
340 DATA WÄHLEN SIE ZWISCHEN 1 UND 4
HÖLZERN
350 DATA SIE DÜRFEN BEGINNEN
360 COLOR X
370 FOR I=ANFANG TO ANDE
380 PLOT E-7*I,F:DRAWTO E-7*I,G
390 NEXT I
400 RETURN
```

Diese Werte bestimmen die 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
120 GOTO 120
```

Beispiel 14

Hier zeigt Ihr ATARI Computer alle möglichen 256 Farben.

```
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:
NEXT A
70 POKE 1616,0:POKE 512,0:POKE 513,6:
POKE 54286,192
80 GOTO 80
90 DATA 72,173,80,6,24,105,16,141,80,6,
141,10,212,141,26,208,201,240,208,5,169,
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
```

Beispiel 15

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

```
10 REM ** ATARI REGENBOGEN FARBEN **
20 DIM C$(24)
30 SETCOLOR 2,0,0:poke 752,1:?" "
40 FOR I=1 TO 24
50 READ D
60 C$(I,I)=CHR$(D)
70 NEXT I
80 D=USR(ADR(C$))
90 END
100 DATA 162,0,173,11,212,201,32,208,
249,141
110 DATA 10,212,142,24,208,232,232,208,
246,142
120 DATA 24,208,240,232
```

Verzeichnis der in ATARI BASIC benutzten Syntaxvereinbarungen, Befehlsworte und Funktionen

Syntaxvereinbarungen für Funktionen und Befehle

><	Hier besteht eine Wahlmöglichkeit. Eine der zwischen diesen Zeichen aufgeführten Angaben muß verwendet werden. Die Zeichen selbst entfallen.
[ ]	Die in eckigen Klammern aufgeführten Angaben können zusätzlich gemacht werden. Die eckigen Klammern selbst entfallen.
...	Die Fortsetzungspunkte deuten an, daß die vorher gemachte Angabe wiederholt werden kann. Die Punkte selbst entfallen.
Zeilen-nummern	Wenn ein Befehl oder eine Funktion in einem Programm verwendet wird, muß eine Zeilennummer angegeben werden.
Andere Zeichen	Alle anderen Zeichen wie z.B. Komma, Semikolon, Anführungszeichen oder runde Klammern werden direkt übernommen.
GROSS-SCHREIBUNG	Alle groß geschriebenen Worte oder Buchstaben werden direkt übernommen.

Allgemeine Definitionen

Die folgenden Abkürzungen werden in den Beschreibungen der Befehle und Funktionen verwendet. Sie werden nicht direkt übernommen, sondern mit ihrer Hilfe wird festgelegt, wie eine an Stelle dieser Abkürzung einzusetzende Aufgabe auszusehen hat. Die Abkürzungen, die Sie hier nicht finden sollten, werden nur für eine bestimmte Funktion oder einen bestimmten Befehl verwendet. Sie werden an der entsprechenden Stelle beschrieben.

Kan	Kanalnummer eines Eingabe- oder Ausgabekanals. Ein numerischer Aus- druck (NumAusdr), in dem allerdings keine Funktionen verwendet werden dürfen, und dessen Ergebnis eine ganze Zahl zwischen 1 und 8 sein muß.
Spal	Nummer einer Spalte des Bildschirms. Ein numerischer Ausdruck. Die erste Spalte hat die Nummer 0. Nicht ganzzahlige Ergebnisse werden gerun- det. Siehe auch "Tabelle der Gra- phik-Modi und Bildschirmforma- te".
Const	Eine numerische oder alphanumerische Konstante. Anführungszeichen dienen hierbei nicht als Begrenzung einer alphanumerischen Konstante, sondern sind ein Teil davon.
Ger	Eine alphanumerische Konstante oder eine Variable, mit deren Hilfe ein Ein- oder Ausgabegerät angesprochen wird. Verwendung findet hier: "C:", "E:", "K:", "P:", "R[n]:", "S:", und "D[n]:Filename[.Ext]".

D[n]	Nummer einer Diskttenstation (D, D1 bis D4, oder D8 als RAM-Disk).
Ausdr	Eine numerische oder boolesche Konstante, Variable oder Funktion, ein Vergleich oder eine Kombination dieser Möglichkeiten.
Zus	Zusatz zu einem Dateinamen, bestehend aus 1 bis 3 Zeichen. Als Zeichen können die Buchstaben von A bis Z und die Ziffern von 0 bis 9 verwendet werden.
Filename	Der Name einer Diskettendatei mit einer Länge von 1 bis 8 Zeichen. Als Zeichen können die Buchstaben A bis Z und die Ziffern 0 bis 9 verwendet werden.
EinGer	Eine alphanumerische Konstante oder Variable, mit deren Hilfe ein Eingabegerät angesprochen wird. Verwendung finden hier: "C:", "E:", "K:", "R[n]:", "S:" und "D[n]:Filename[.Ext]".
SpAdr	Eine numerische Variable, Konstante oder ein Ausdruck, mit dessen Hilfe eine Speicherzelle angesprochen wird. Die Speicherzellen sind von 0 bis 65535 numeriert. NumAusdr Eine numerische Variable, Konstante, Funktion oder eine Kombination dieser Möglichkeiten.
umVar	Eine numerische Variable (keine Feldvariable).
AusGer	Eine alphanumerische Variable oder Konstante, mit deren Hilfe ein Ausgabegerät angesprochen wird. Verwendung finden hier: "C:", "E:", "P:", "R[n]:", "S:" und "D[n]:Filename[.Ext]".
Zeil	Nummer einer Zeile des Bildschirmes. Ein numerischer Ausdruck. Die erste Zeile hat die Nummer 0. Siehe auch "Tabelle der Graphik-Modi und Bildschirmformate".
AlphaVar	Eine alphanumerische Konstante und Variable, ein Variablenteil oder eine Funktion mit einem alphanume- rischen Ergebnis.
Var	Eine numerische oder alphanumerische Variable (keine Feldvariable und kein Variablenteil).

Reserviertes Befehlswort	Abkürzung	Funktion
ABS		Berechnet den Absolutwert einer Zahl. Syntax: ABS (NumAusdr)
ADR		Berechnet die Adresse, ab der eine alphanumerische Konstante oder Variable im Speicher abgelegt ist. Syntax: ADR (Alpha)
AND		Logische UND-Verknüpfung zweier Werte (nicht bitweise sondern das Ergebnis ist 0 (null), falls einer der beiden Werte 0 ist.) Syntax: Ausdr AND Ausdr
ASC		Berechnet den ATASCII-Code eines Zei- chens. Syntax: ASC (Alpha)
ATN		Berechnet den Arcus-Tangens einer Zahl. Syntax: ATN (Ausdr)
BYE B.		Übergang vom BASIC in den Selbsttest. Syntax: BYE
CLOAD	CLOA.	Lädt Daten oder Programme vom Programm-Recorder in den RAM-Speicher ein. Syntax: CLOAD
CHR$		Wandelt entsprechenden ATASCII-Wert (0-255) in Zeichen um. Syntax: CHR$ (NumAusdr)
CLOG		Berechnet den Zehnerlogarithmus einer Zahl. Syntax: CLOG (NumAusdr)
CLOSE	CL.	Schließt einen Ein-/Ausgabekanal. Syntax: CLOSE #Kan
CLR Syntax: CLR		Löscht alle Variablen.
COLOR	C.	Bestimmt ein Farbregister in der Grafik-Betriebsart. Syntax: COLOR NumAusdr
COM		Stellt Speicherplatz bereit. Siehe DIM Syntax: COM >AlphaVar (NumAusdr) NumVar (NumAusdr [,NumAusdr])< [,>AlphaVar(NumAusdr) NumVar (NumAusdr [NumAusdr])<...]

Reserviertes Befehlswort	Abkürzung	Funktion
CONT	CON.	Setzt die Programmausführung in der Folgezeile nach Gebrauch der BREAK-Taste oder Eingabe von STOP fort. Syntax: CONT
COS		Berechnet den Cosinus eines Winkels. Syntax: COS (NumAusdr)
CSAVE		Speichert ein im RAM befindliches Programm auf Cassette ab. Syntax: CSAVE
DATA	D.	Teil der READ/DATA-Kombination. Dient zur Speicherung von Werten, die mit Hilfe des READ-Befehls Variablen zugewiesen werden können. Syntax: DATA Const[,Const...]
DEG	DE.	Schaltet vom Bogenmaß auf Grad um. Syntax: DEG
DIM	DI.	Stellt Speicherplatz für Feld- oder alphanumerische Variablen bereit. Syntax: DIM >AlphaVar(NumAusdr) NumVar(NumAusdr [,NumAusdr])< [,>AlphaVar(NumAusdr) NumVar(Numausdr [NumAusdr])<...]
DOS	DO.	Laden des DOS-Menüs. Syntax: DOS
DRAWTO	DR.	Zeichnet eine Linie vom zuletzt ge- zeichneten Punkt zum angegebenen Endpunkt. Syntax: DRAWTO Spal,Zeil
END		Beendet ein laufendes Programm. Syntax: END
ENTER	E.	Dient zum Laden eines mit LIST ge- speicherten BASIC-Programmes von Cassette oder Diskette. Syntax: ENTER EinGer
EXP		Bezeichnet den Wert einer Exponentialfunktion. Syntax: EXP (NumAusdr)
FOR..TO..NEXT	F	Serie von Befehlen, die sooft ausgeführt werden, bis die Lauf-Variable einen bestimmten Wert erreicht hat. Syntax: FOR NumVar=AnfAusdr TO EndAusdr[STEP AddAusdr] dann: NEXT NumVar

Reserviertes Befehlswort	Abkürzung	Funktion
FRE (O)		Berechnet den noch freien Speicherplatz in Bytes nach Abzug von OS, DOS, Displaylist, Bildschirmspeicher und BASIC mit Daten. Syntax: FRE(0) Zu dem Ergebnis müssen 64k addiert werden.
GET	GE.	Lädt unter Verwendung eines Ein-/Ausgabekanals ein einzelnes Byte von dem angesprochenen Gerät. Syntax: GET #Kan,NumVar
GOSUB	GOS.	Dient zum Aufruf eines Unterprogrammes. Syntax: GOSUB ProZeil
GOTO	G.	Die Programmausführung wird in der genannten Zeilennummer fortgesetzt. Syntax: GOTO ProZeil
GRAPHICS	GR.	Aktiviert eine der Grafik-Betriebsarten Syntax: GRAPHICS NumAusdr
IF...THEN		Wenn eine bestimmte Bedingung erfüllt ist, werden die hinter THEN stehenden Befehle ausgeführt. Syntax: IF Ausdr THEN Befehl oder: IF Ausdr THEN ProZeil
INPUT	I.	Mit INPUT können einer Variablen Wertezugewiesen werden. Syntax: INPUT [#K an>; ,<]Var [,Var...]
INT		Rundet eine Zahl ab auf den ganzzahligen Wert. Syntax: INT(NumAusdr)
LEN		Berechnet die Länge eine alphanumerischen Variablen. Syntax: LEN(Alpha)
LET	LE.	Weist einer Variablen einen Wert zu. Syntax: [LET] Var=Ausdr
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. Syntax: LIST [ProZeil [>- ,< ProZeil]] oder: LIST AusGer [,ProZeil [>- ,<ProZeil]]
LOAD	LO.	Überträgt ein zuvor gespeichertes Programm in den Computer. Syntax: LOAD EinGer ProgName
LOCATE	LOC	Lädt den Code eines Zeichens oder Graphikpunktes, der sich an der angegebenen Position befindet. Syntax: LOCATE Spal,Zeil,NumVar
LOG		Berechnet den natürlichen Logarithmus einer Zahl. Syntax: LOG(NumAusdr)

Reserviertes Befehlswort	Abkürzung	Funktion
LPRINT	LP.	Dient zur Übertragung von Zeichen an einen Drucker. Syntax: LPRINT [Ausdr][>; ,<... [Ausdr]...] oder: LPRINT ["Text"...]
NEW		Löscht das gerade im RAM befindliche Programm und alle Variablen. Syntax: NEW
NOT		Ergibt den Wert 1, wenn der Ausdruck nicht wahr ist, ist er aber wahr, so ergibt sich der Wert 0. Syntax: NOT Ausdr
NOTE	NO.	Stellt fest, an welcher Stelle der Datei auf einer Diskette der Zeiger dieser Datei steht. Syntax: NOTE #Kan,SektorVar,ByteVar
ON		Wird in Verbindung mit GOTO oder GOSUB verwendet zur Ausführung eines Programms ab einer bestimmten Zeile oder eines Unterprogrammes. Syntax: ON NumAusdr GOTO ProZeil [,ProZeil...] oder: ON NumAusdr GOSUB ProZeil [,ProZeil...]
OPEN	O.	Weist einem bestimmten Ein/Ausgabekanal ein bestimmtes Gerät zu. Syntax: OPEN #Kan,Betriebsart, Funktion,Ger
OR		Logischer Vergleich zweier Ausdrücke. Sofern nur einer wahr ist, ergibt sich der Wert 0; eine 1, wenn beide unwahr sind. Syntax: Ausdr AND Ausdr
PADDLE		Bestimmt die momentane Stellung des angegebenen Drehreglers mit einem Wert zwischen 1 und 228. Syntax: PADDLE(NumAusdr)
PEEK		Dient zum Lesen einer RAM- oder ROM- Speicherstelle. Syntax: PEEK(SpAdr)
PLOT	PL.	Zeichnet an der angegebenen Position einen Punkt auf den Bildschirm. Syntax: PLOT Spal,Zeil
POINT	P.	Setzt den Zeiger einer Diskettendatei an eine bestimmte Position. POINT #Kan,SektVar,ByteVar
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. Syntax: POKE SpAdr,ByteVar

Reserviertes Befehlswort	Abkürzung	Funktion
POP		Löscht im Stack die Rücksprungadresse des zuletzt ausgeführten FOR-, GOSUB- oder ON GOSUB-Befehls. Syntax: POP
POSITION	POS.	Setzt den Cursor an eine bestimmte Stelle des Bildschirms. Syntax: POSITION Spal,Zeil
PRINT	PR.	Dient zur Ausgabe von Daten an den oder ? Bildschirm oder ein Gerät. Syntax: PRINT[>Ausdr #Kan<][>: .<... [Ausdr]]... oder: PRINT[#Kan,]["Text"]
PTRIG		Dient zur Festlegung, ob der Feuerknopf des Drehreglers gedrückt ist. Syntax: PTRIG(NumAusdr)
PUT	PU.	Dient zur Ausgabe einer Zahl über den angegebenen Ein-/Ausgabekanal. Syntax: PUT #Kan, NumAusdr
RAD		Schaltet von Grad auf Bogenmaß um. Syntax: RAD
READ	REA.	Weist die bei einem DATA-Befehl angegebenen Werte den angegebenen Variablen zu. Syntax: READ Var,[Var...]
REM	R.	Dient zur Einfügung von Erklärungen in ein Programm. Syntax: REM Kommentar
RESTORE	RES.	Dient zum Rücksetzen des für die DATA-Befehle zuständigen Zeigers. Syntax: RESTORE[ProZeil]
RETURN	RET.	Dient zum Rücksprung zu dem unmittelbar auf den zuletzt ausgeführten GOSUB oder ON GOSUB folgenden Befehl. Syntax: RETURN
RND		Berechnet eine Zufallszahl. Syntax: RND (NumAusdr)
RUN		Dient zum Starten des im RAM befindlichen Programmes bzw. lädt vom benannten Gerät ein und startet sofort. Syntax: RUN[EinGer]
SAVE	S.	Dient zur Übertragung eines im RAM befindlichen BASIC-Programmes an ein bestimmtes Speichergerät. Kann mit LOAD oder RUN wieder geladen werden. Syntax: SAVE AusGer
SETCOLOR	SE.	Bestimmt Farbe und Helligkeit bei der Darstellung eines bestimmten Farbregisters. Syntax: SETCOLOR RegAusdr, FarbAusdr, HellAusdr

Reserviertes Befehlswort	Abkürzung	Funktion
SGN		Dient zur Feststellung, ob eine Zahl positiv oder negativ oder gleich 0 ist. Syntax: SGN(NumAusdr)
SIN		Berechnet den SINUS eines Winkels. Syntax: SIN(NumAusdr)
SOUND	SO.	Schaltet einen Tonkanal an oder aus und bestimmt Tonhöhe, Tonreinheit und Lautstärke. Syntax: SOUND StimmAusdr,HöhAusdr, ReinAusdr,LautAusdr
SQR		Berechnet die Quadratwurzel einer Zahl. Syntax: SQR(NumAusdr)
STATUS	ST.	Dient zur Feststellung, mit welcherm Ergebnis die zuletzt im Zusammenhang mit dem angegebenen Kanal ausgeführte Ein/Ausgabeoperation abgeschlossen wurde. Syntax: STATUS Kan,NumVar
STEP		Wird in Verbindung mit FOR/NEXT-Schleifen gebraucht. Bestimmt die schriftweise Erhöhung der Lauf-Variablen. Syntax: STEP(NumAusdr)
STICK		Bestimmt die momentane Stellung des angegebenen Steuerknüppels. Syntax: STICK(NumAusdr)
STRIG		Dient zur Feststellung, ob der Feuerknopf des Steuerknüppels gedrückt wurde. Syntax: STRIG(NumAusdr)
STOP	STO.	Dient zur Unterbrechung eines BASIC-Programmes. Syntax: STOP
STR$		Wandelt einen numerischen Wert in einen alphanumerischen Wert um. Syntax: STR$(NumAusdr)
TRAP	T.	Dient zum Abfangen und Bearbeiten von Fehlermeldungen. Syntax: TRAP ProZeil
USR		Dient zum Aufruf eines Maschinen-Programmes. Syntax: USR(SpAdr[,NumAdr...])
VAL		Wandelt einen alphanumerischen Wert in einen numerischen Wert um. VAL(Alpha)
XIO	X.	Dient zum Arbeiten mit Ein-/Ausgabe-Kanälen. Syntax: XIO(NumAusdr),Kan, NumAusdr,NumAusdr,Ger

Fehlermeldungen und Erklärungen

Beim ATARI-Computer wird, wenn ein Fehler auftritt, eine Kodezahl auf dem Bildschirm ausgeben. 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 Unterprogramme 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 Zeilenzahl größer als 32767
Die Zeilennummer ist größer als 32767, oder kleiner als 1.

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 Graphikzeichen 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 dimensionierte alphanumerische Variable zuzugreifen.

11 Zahl zu groß
Es wurde versucht, durch 0 zu teilen oder es ist eine Zahl aufgetreten, deren Absolutwert größer ist als 9.9999999999x1097.

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 gelöscht
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 Falsche 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 der CSAVE- oder LIST-Befehls gespeichertes Programm mit Hilfe des LOAD-Befehls zu laden.

128 Abbruch 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 Graphikbefehle 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.

133 XIO-Befehl fehlerhaft
Es wurde versucht, einen Ein/Ausgabekanal zu benutzen, der noch nicht geöffnet war.

134 Falsche IOCB Nummer:
Die Datei-Nummer ist falsch.

135 Kanal nur für Eingabe geöffnet
Es wurde ein End-of-file-Record (EOF) 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 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 auführen.

140 Fehler auf dem seriellen Bus
Möglicherweise ist eine Kassette oder eine Diskette defekt.

141 Cursor außerhalb der Billdschirmbegrenzung

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 Directory 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 Graphikbetriebsart nicht aus
Für die verschiedenen Graphikbetriebsarten wird unterschiedlich viel Speicherplatz benötigt.

160 Unbekannte Diskettenstation
Wenn die Diskettenstation angesprochen wird, können nur die Gerätenamen D:,DI:,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 Dateien 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 Verbindung zwischen den einzelnen Sektoren sind zerstört.

165 Falscher Dateiname
Ein Dateiname beginnt mit einem Kleinbuchstaben oder enthält unzu-lässige 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
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 III Datei an eine unter DOS II oder DOS 2.5 erstellte Datei anzuhängen. Kopieren Sie die DOS III Datei mit DOS 2.5 auf eine DOS II oder DOS 2.5 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.

178 Defekte Sektoren
Die Diskette zeigt während des Zugriffs einen Defekt. Verwenden Sie eine andere. Tritt der Fehler oft auf, kann auch die Diskettenstation defekt sein.

Programme für ATARI – Computer

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 und schwierigeren Problemlösungen, die vielleicht sogar zu verkaufsfähiger Software führen. Deshalb wird niemand erfreut sein, wenn sein Programm von Dritten ohne Genehmigung kopiert und weiterverkauft wird, ohne irgendeine Entschädigung für den Urheber. Verletzungen dieser Urheberrechte 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.

Fragen und Antworten

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.

Einige der häufigsten Fragen sind hier kurz erklärt:

Probleme	**Information**
Wie werden die Umlaute (ä,ö,ü) dargestellt?	Sie sind im intern. Zeichensatz enthalten: Einschalten durch POKE 756,204 und ausschalten durch POKE 756,224. Darstellung durch Drücken der CTRL-Taste mit weiterer Taste.
Wie werden eigene BASIC-Programme ausgedruckt?	Bei betriebsbereitem Drucker einfach LIST"P:" eingeben und RETURN drücken. Einzelne Zeilen durch LIST"P:"10,20, etc.
Wie kann man BASIC-Programme speichern und laden?	Es gibt verschiedene Möglichkeiten:
Programm-Recorder	Speichern LOAD"C" CLOAD ENTER"C"
	Laden SAVE"C" CSAVE LIST"C"
Diskettenstation	Speichern SAVE"D:Name" LIST"D:Name" ENTER"D:Name" Dateiname"
	Laden Das Laden und Starten kann zusammengefaßt werden mit RUN"C: bzw. RUN"D: Dateiname"

Tips für Fortgeschrittene

Vertikales Feinscrollen	Geben Sie folgendes ein: POKE 622,255 (RETURN) OPEN #1,12,0,"E:" Listings und Print Anweisungen werden zeilenweise gescrollt.
Wie kann man die Sonderfunktionstasten im eigenen Programm nutzen?	Ein kleines Programm als Beispiel: 10 A = PEEK (53279) 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 732,0 Wenn N nicht NUll ist, wurde die HELP-Taste gedrückt. Der POKE- Befehl löscht die Speicherstelle.
Cursor abschalten:	Eingabe von POKE 752,1
Cursor einschalten:	POKE 752,0
Tastaturklick ausblenden:	POKE 731,255
Tastaturklick einschalten:	POKE 731,0
Motor des Datenrekorders	Dies funktioniert natürlich nur bei gedrückter PLAY-TASTE
einschalten:	POKE 54018,52
ausschalten:	POKE 54018,60
Farbwechsel am Bildschirm	
unterdrücken:	POKE 77,1
aktivieren:	POKE 77,127

Linken Bildrand des Texteditors verschieben:	POKE 82,n für n=0 ergibt sich eine Darstellungsbreite von 40 Zeichen auf dem Bildschirm. Für n=35 werden beispielsweise nur noch 5 Zeichen am rechten Bildrand dargestellt.
Bildschirmdarstellung ein- und ausschalten:	
aus:	POKE 559,0
ein:	POKE 559,34
Bildschirmdarstellung auf dem Kopf stehend:	POKE 755,4 Mit POKE 755,0 kann der ursprüngliche Zustand wiederhergestellt 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 numeriert. Der Wert des Inhalts einer Speicherzelle kann zwischen O 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 Inhalts der ersten Speicherzelle und dem 256-fachen des Inhalts 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. Dies 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 = LMARGN)
Diese Speicherzelle enthält die Nummer der Spalte an der sich in der Graphikbetriebsart 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 = RMARGN)
Diese Speicherzelle enthält die Nummer der Spalte, an der sich in der Graphikbetriebsart 0 die rechte Bildschirmbegrenzung befindet. PEEK (83) liegt immer zwischen 0 und 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 Bildschirmes, 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 Graphikbetriebsart aktiviert ist.

85, 86 Momentane Cursorposition (Spalte) (Current Columne Cursor Position = COLCRS)
Der Inhalt dieser Speicherzelle 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 Graphikbetriebsart aktiviert ist. In den Graphikbetriebsarten 0 bis 7 ist PEEK (86) immer 0.

88, 89 Adresse des Bildschirmspeichers
(Screen Memory Address = SAVMSC)
Diese Speicherzelle enthält die Anfangsadresse des Bildschirmspeichers. Der Inhalt dieser Adresse wird in der oberen Ecke des Bildschirmes 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 4K-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 Kodezahl des aufgetretenen Fehlers (Error Number – ERRSAV)
Wenn ein Fehler auftritt, wird die Kodezahl 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
Radions or Degreees = RADFLG oder DEGFLG)
Wenn hier eine 0 abgelegt ist, wird im Bogenmaß und wenn hier eine 0 abgelegt ist, in Grad gerechnet.

656 Cursorposition im Textfenster (Zeile)
(Text Cursor Row Position = TXTROW)
Der Inhalt dieser Speicherzelle entspricht der Zeile des Textfensters, in beziehungsweise von der das nächste Zeichen geschrieben oder gelesen wird. PEEK (656) liegt immer zwischen 0 und 3 entspricht der ersten Zeile des Textfensters.

694 Inverse Zeichendarstellung
(Inverse Video Keystrokes = INVFLG)
Befindet sich in dieser Speicherzelle eine 0, dann wird, beim Drücken einer Taste, 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 Speicherzelle 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 Inhalts dieser Speicherzelle kann der Cursor undurchsichtig oder unsichtbar gemacht oder die Zeichendar-stellung umgekehrt werden.

756 Auswahl des Zeichensatzes (Character Adress Base = CHBAS)
Mit Hilfe dieser Speicherzelle wird ausgewählt, welcher Zeichensatz in den Graphikbetriebsarten 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, der die Kleinbuchstaben und Graphikzeichen enthält.

764 Zuletzt gedrückte Taste (Keyboard Character = CH)
In dieser Speicherzelle ist der Kode der Taste gespeichert, die als letzte gedrückt wurde. Falls keine Taste gedrückt wurde, ist hier eine 256 abgelegt.

765 Auswahl der Zeichenfarbe für den XIO-Befehl
(FIII 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 Character = DSPFLG)
Enthält diese Speicherzelle eine 0, 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-I wird der Inhalt dieser Speicherzelle verändert.

53279 Betätigung der Funktionstasten (CONSOLE Switch Port = CONSOL)
Man kann man mit Hilfe eines PEEK (53279)-Befehls feststellen, ob bzw. welche der drei Funktionenstasten (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 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

Speicheraufteilung der XE-Computer

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 $OOOO 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-Selektierungen für die Peripherie-Chips erzeugt. I/O-Raumaufteilung (memory-mapped) DOOO-DOFF 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 Sind für zukünftige D600-D6FF Belegung reserviert. D700-D7FF OS-ROM physisch vorhanden, kann aber nicht verändert werden.	
CFFF-COOO	OS-ROM oder RAM, wenn ROM abgeschaltet.	
BFFF-AOOO	RAM oder Modul-Interface, wenn RD5-Leitung durch den Einschub auf +5V gelegt wird.	
9FFF-8000	RAM oder Modul-Interface, wenn RD4-Leitung durch den Einschub auf +5V gelegt wird.	
7FFF-5800	RAM	
57FF-5000	RAM solange nicht im Selbsttest-Modus.	
4FFF-OOOO	RAM	

1. Der Zugriff auf das OS-ROM kann durch Schreiben eines Wertes von 0 nach Port B des PIA, Bit PB0, abgeschaltet werden. Der Zugriff wird normalerweise durch PB0=1 eingeschaltet. Wird dieses Bit geändert, so dürfen andere Bits des Registers nicht beeinflußt werden.

2. Der Selbsttest ROM-Code ist an den Adressen $DOOO-$D7FF im OS-ROM physisch vorhanden. Dieser Bereich wird allerdings für den Zugriff auf die Eingabe/Ausgabegeräte benötigt. Bei Aufruf des Selbsttests wird das RAM an den Speicherstellen $5000--$57FF abgeschaltet. Der Memory-Manager definiert den Speicherzugriff so, daß die Adressen $DOOO-$57FF angesprochen werden. Er benutzt Port B des PIA, Bit PB7, um festzulegen, ob RAM oder ROM in dem Bereich von $5OOO-$57FF angesteuert werden soll. Ist PB7 aktiv, so wird RAM angesteuert, andernfalls wird auf das OS-ROM zugegriffen. Bei Änderung dieses Bits sollen die anderen Bits des Registers nicht beeinflußt werden.

Ausnutzen aller RAM Konfigurationsmöglichkeiten des 130 XE

Der 130 XE verfügt über 131.072 Bytes Schreib-Lesespeicher (Random Access Memory = "RAM): doppelt soviel wie der ATARI 800 XE Computer.

Die zusätzlichen 65.536 Byte des RAM sind dem Anwender in den meisten Fällen zugänglich. Programme können in dem zusätzlichen Speicherraum größere Datenmengen verfügbar halten. In Verbindung mit einer ATARI Diskettenstation und z.B. DOS 2.5 läßt sich das Zusatz-RAM als RAMDISK nutzen – das ergibt ein sehr schnelles "Laufwerk". (Weitere Informationen über die RAMDISK des 130XE finden Sie in der ATARI Bedienungsanleitung zur Diskettenstation.) Auf die zusätzlichen 65.536 Byte des RAM können Sie im ATARI BASIC durch "Bank Switching" (Speicherbereich-Umschaltung) zugreifen. Der 6502 Prozessor und der ANTIC Videoprozesssor – können lediglich 65.536 Speicherzellen adressieren (für eine Lese- oder Schreiboperation auswählen). Bank-switching schaltet einen 16KByte großen Speicherbereich ab und ersetzt diesen durch einen Speicherbereich (Bank) des zusätzlichen RAM. So wird der adressierbare Speicherbereich vergrößert. Im 130 XE 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 6520 Peripheral Interface-Adapter (PIA) dar, welcher Ein- und Ausgaben des Computers steuert.

Bis 4 und 5 dieser Speicherstelle wählen den oder die Prozessoren, die auf den Zusatzspeicher zugreifen. Nach dem Einschalten des Computers sind beide Bits auf I gesetzt (Bit 4 steuert den CPU Zugriff und Bit 5 den Zugriff des ANTIC auf die Speicherbank.) Wird eines dieser Bits auf 0 gesetzt, 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 selektieren die vier verfügbaren Speicherbereiche aus.

Das Bank-Switching wird über die Speicherstelle 54017 erreicht, die als Port B-Adresse des 6502 PIA (Peripheral Interface Adapter) dient. Wenn ohne Bank-Switching, also nur in dem unteren 64K Bereich, gearbeitet wird, enthält 54017 den Wert 193. Mit dem Basic Befehl POKE kann der Inhalt der Speicherstelle verändert und damit das zusätzliche RAM angesprochen werden, z.B. wählt POKE 54017,255 die erste Bank (Bank 0) aus und läßt den 6502 Prozessor darauf zugreifen, wobei der ANTIC Video Chip nach wie vor mit dem normalen Speicherbereich arbeitet (Modus 2).

Die folgende Formel steuert das Bank-Switching:
POKE 54017, (193 + 4*BANK + 16*MODUS). Nach dem Einschalten des Computers wird der Inhalt dieser Speicherstelle automatisch auf 253 gesetzt, so daß als Default-Werte Bank 3 und Modus 3 angewählt werden: (193 + 4 * 3 +16 * 3) = 253.

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 Bank (VBE)	CPU Bank (CBE)	Bank LSB	Bank MSB	

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.

Garantie und Service

Im Garantiefall wenden Sie sich bitte an den ATARI Händler, bei dem Sie Ihren ATARI Computer gekauft haben. Nur dort kann zweifelsfrei festgestellt werden, ob ein Garantiefall vorliegt. Die Abwicklung erfolgt ebenfalls durch den Händler.

Nicht-Garantie Service
Bitte wenden Sie sich an Ihren Händler oder an eines der von ATARI autorisierten Service Center, welche kostenpflichtige Reparaturen ausführen.

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	L.M Schreiber	3-89090-417-3	Für Anfänger und Fortgeschrittene. Wie man mit dem Computer "spricht". Wie man Spiele entwickelt. Mit über 100 Programm-Bausteinen und vielen fertigen Programmen
Idea Verlag München	D. Inman, K.Inman Der ATARI Assembler	3-88703-025-8	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.Hofacker 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).
DATA Becker Verlag	Adventures und wie man sie auf dem ATARI 600XL 800 XLprogrammiert (Walkowiak)	3-89011-059-2	Eigene Adventure-Programme programmieren
DATA Becker Verlag	ATARI 130 XE/800XL für Einsteiger (Sczepanowsky)	3-89011-033-9	Einfürung in die Programmierung mit ATARI Basic

DATA Becker Verlag	ATARI 600XL/800XL INTERN (Eichler,Grohmann)	3-89011-053-3	Betriebssystem – dokumentation ATARI 8 bit Computer
DATA Becker Verlag	ATARI TIPS & TRICKS (Clausdorf/Dittrich)	3-89011-111-4	
DATA Becker Verlag	Das Basic Trainingsbuch zu ATARI 600 XL/ 800XL (Voss)	3-89011-057-6	
DATA Becker Verlag	Das Schulbuch zu ATARI 600XL/ 800XL (Voß)	3-89011-045-2	
DATA Becker Verlag	PEEKS & POKES zu ATARI 600 XL/ 800XL (Koch)	3-89011-082-7	
DATA Becker Verlag	Strategiespiele und wie man sie auf dem ATARI 600 XL/800XL programmiert (Schneider)	3-89011-077-0	
Sybex-Verlag	ATARI Basic Handbuch (J.Reschke)	3-88745-083-3	Handbuch für ATARI Basic und Basic XL
Sybex-Verlag	ATARI Programm Sammlung (S.R. Trost)	3-88745-068-X	Praktische Sammlung einer Fülle von Problemlösungen für den Alltagsgebrauch
Sybex Verlag	Das ATARI Profibuch (Reschke,J. Wiethoff A.)	3-88745-605-X	Sehr ausführliche Dokumentation des XL/XE Betriebs-systems.
Sybex Verlag	Fortgeschrittene 6502 Programmierung (Rodney Zaks)	3-88745-047-7	
Sybex Verlag	Programmierung des 6502 Pro-zessors. (R.Zaks)	3-88745-600-9	
Vieweg Verlag	Ausgewählte Basic Computerspiele (Hrsg.H.Schumny)	3-528-04307-5	

Vieweg Verlag	Mathematikprogr. für den ATARI 800 XL (H.Schumny)	3-528-04422-5	Quadr. Gleichungen, Entfernungsber., Bernoulli-Exp., Galton-Brett, Kurvendiskussion,u.a.
	Diskette z.Buch	3-528-02702-9	
Vogel Buchverlag	Chip SPECIAL The Best of ATARI XL-XE	Best.Nr.0480	
Vogel Buchverlag	Das ATARI Spielebuch (James/Gee/Ewbank)	3-8023-0788-7	
Vogel Buchverlag	Was der ATARI alles kann (Peschetz A.&J.)	3-8023-0795-X	
Vogel Buchverlag	Start mit ATARI-Basic (Hettinger H./Heinz A.)	3-8023-0827-1	
Vogel Buchverlag	Die ATARI Hitparade (Hettinger H./Krauß W.)	3-8023-085-7	

Weitere Neuerscheinungen sind z.Zt. in Vorbereitung. Bitte fragen Sie im Fachhandel nach. Für die Richtigkeit und Vollständigkeit der gemachten Angaben wird keine Gewähr übernommen.

Englischsprachige Literatur zum ATARI Computer-System

Verlag	Verfasser/Titel	ISBN	Inhalt
Reston Publishing Company Inc. Reston,VA USA	H.Kohl,T.Kahn L.Lindsay, P.Cleland ATARI Games and Recreations	0-8359-0242-0	Kurze Systemeinführung mit Programmiertips für Spiele, Ton und Graphik
W.Hofacker 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, VA USA	D.Inman,K.Inman	0-9359-0236-6	Für BASIC-erfahrene Anwender ohne Vorkenntnisse in ATARI ASSEMBLER/Editor.
W.Holzacker Holzkirchen	S.D.Roberts How to programme your ATARI Computer in 6502 Maschinenlanguage	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.	0-942386-15-9	Spezielle Information über das Programmieren in Maschinensprache (6502)
Compute! Publications Inc. Greensboro, NC USA	B.Wilkinson, K.OBrien, 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.Holzacker Holzkirchen	T.E.Rowley ATARI BASIC Learning by using	3-92-1682-86-X	Programmbeispiele für Ton und Farbe und Kontroller.
Compute! Publications Inc. Greensboro, NC 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
Creative Computing Morris Plains NJ, USA	D.Small,s.Small G.Blank The creative ATARI	0-916688-34-8	Umfangreiche Systemdarstellung mit Listings und Software-Besprechung
Compute! Publications Inc. Greensboro, NC, USA	Compute!"s Second book of ATARI	0-942386-06-x	Neue, bislang nicht veröffentlichte Artikel über das ATARI System. Programmiertechniken, Grafik, Spiele und andere Anwendungen. Maschinensprache.
Compute! Publications, Inc. Greensboro,NC USA	Compute!"s First book of ATARI Graphik	0-942386-08-6	Alles über Grafik,Nutzung der 256 Farben. Player-Missile etc.
Compute! Publications Inc. Greensboro, NC USA	Compute!"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

Technische Daten des ATARI 130 XE

Prozessor:	6502C Mikroprozessor, Taktfrequenz 1,79 MHz
Spezial-Bausteine:	GTIA Chip – Graphik-Darstellung POkEY CHIP – Tonerzeugung und serieller Bus I/O ANTIC CHIP – Bildschirmdarstellung FREDDY CHIP – Speicherverwaltung
Speicher:	131.072 Byte RAM 24.576 Byte ROM (Betriebssystem + ATARI BASIC Interpreter)
Bildschirm:	11 Graphik-Modi 256 Farben 320 x 192 Bildpunkte 40 Spalten x 24 Zeilen Textdarstellung 5 Text-Modi
Tonerzeugung	4 unabhängige Tonkanäle (Stimmen) 3 ½ Oktaven Tonumfang
Tastatur	Schreibmaschinentastatur 62 Tasten mit HELP-Taste und 4 Spezialfunktionstasten Internationaler Zeichensatz 29 Graphik Tasten
Programmierung:	ATARI BASIC Interpreter eingebaut Softwarekompatibel zu allen ATARI XL, XE Computern
Anschlüsse:	Steckmodul Anschluß TV Anschluß Monitor Anschluß 2 Controller Ports Serieller I/O Port Erweiterter Modul-Anschluß
Stromversorgung:	1 Ampere bei 5 Volt Gleichspannung

Technische Daten des ATARI 800 XE

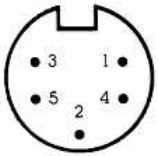
Prozessor:	6502C Mikroprozessor, Taktfrequenz 1,79 MHz
Spezial-Bausteine:	GTIA Chip – Graphik-Darstellung POKEY CHIP – Tonerzeugung und serieller Bus I/O ANTIC CHIP – Bildschirmdarstellung FREDDY CHIP – Speicherverwaltung
Speicher:	65.536 Byte RAM 24.576 Byte ROM (Betriebssystem + ATARI BASIC Interpreter)
Bildschirm:	11 Graphik-Modi 256 Farben 320 x 192 Bildpunkte 40 Spalten x 24 Zeilen Textdarstellung 5 Text-Modi
Tonerzeugung	4 unabhängige Tonkanäle (Stimmen) 3 ½ Oktaven Tonumfang
Tastatur	Schreibmaschinentastatur 62 Tasten mit HELP-Taste und 4 Spezialfunktionstasten Internationaler Zeichensatz 29 Graphik Tasten
Programmierung:	ATARI BASIC Interpreter eingebaut Softwarekompatibel zu allen ATARI XL, XE Computern
Anschlüsse:	Steckmodul Anschluß TV Anschluß Monitor Anschluß 2 Controller Ports Serieller I/O Port Erweiterter Modul-Anschluß
Stromversorgung:	1 Ampere bei 5 Volt Gleichspannung

PIN-Belegungen am Computer

(von außen betrachtet)

Monitor-Buchse

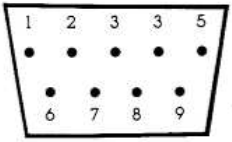
Mehrfarbmonitor über 2 und 4
Einfarbmonitor über 2 und 1



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

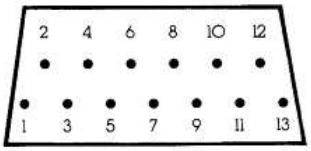
Buchse für Steuerungen

Port 1 und Port 2



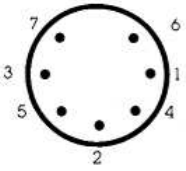
1. (Joystick) Forward Input
2. (Joystick) Back Input
3. (Joystick) Left Input
4. (Joystick) Right Input
5. B Potentiometer Input
6. Trigger Input
7. + 5 volts
8. Ground
9. A Potentiometer Input

Serielle I/O Buchse



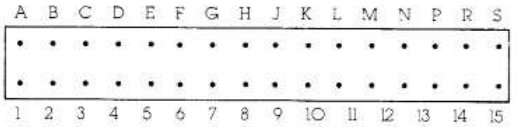
1. Clock Input
2. Clock Output
3. Data Input
4. Ground
5. Data Output
6. Ground
7. Command
8. Motor Control
9. Proceed
10. + 5/Ready
11. Audio Input
12. (nicht belegt)
13. Interrupt

Netzteil Anschluß



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

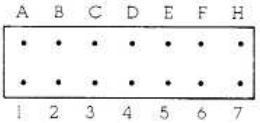
Modulschacht



1. S4
2. A3
3. A2
4. A1
5. AO
6. D4
7. D5
8. D2
9. D1
10. D0
11. D6
12. S5
13. + 5V
14. RD5
15. CCTL

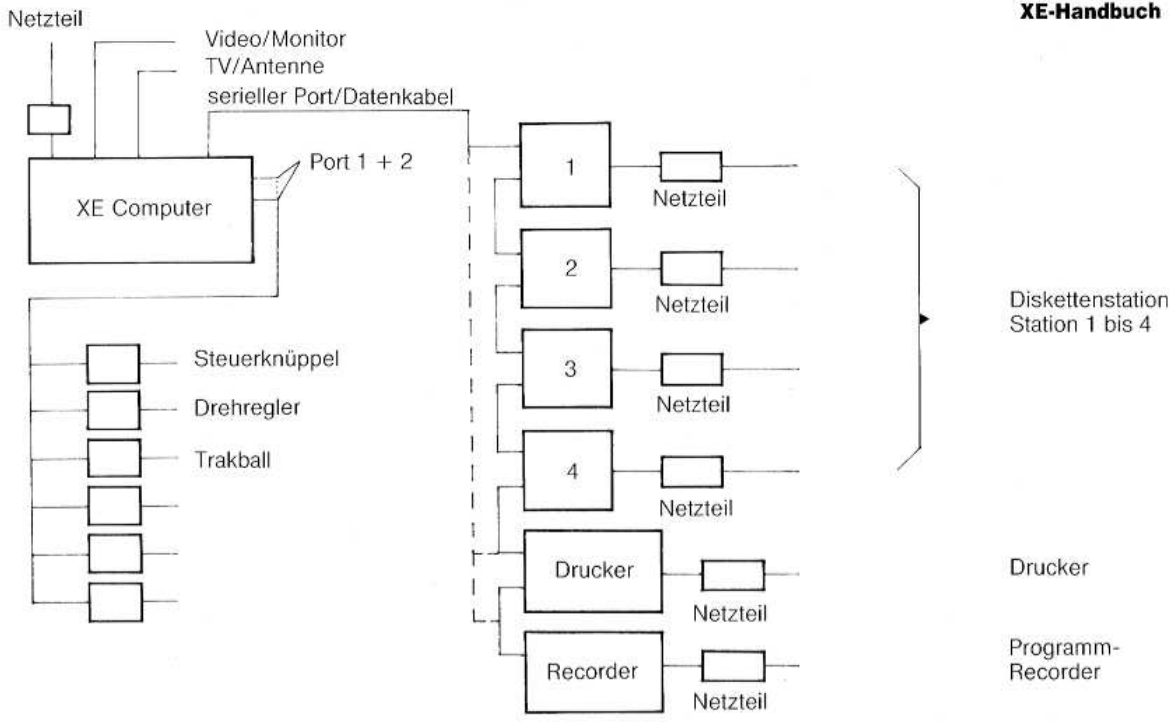
A. RD4
B. GND
C. A4
D. A5
E. A6
F. A7
H. A8
H. A9
K. A12
L. D3
M. D7
N. A11
P. A10
R. R/W
S. BO2

ENHANCED CARTRIDGE INTERFACE (ECI)



A. Reserved
B. IRQ
C. HALT
D. A13
E. A14
F. A15
H. GND

1. EXSEL
2. RST
3. DIXX
4. MPD
5. Audio
6. REF
7. + 5V



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.

Interner Zeichensatz

1				2				3				4			
#	CHR	#	CHR	#	CHR	#	CHR	#	CHR	#	CHR	#	CHR	#	CHR
0	Space	16	0	32	@	48	P	64		80		96		112	p
1	!	17	1	33	A	49	Q	65		81		97	a	113	q
2	"	18	2	34	B	50	R	66		82		98	b	114	r
3	#	19	3	35	C	51	S	67		83		99	c	115	s
4	$	20	4	36	D	52	T	68		84		100	d	116	t
5	%	21	5	37	E	53	U	69		85		101	e	117	u
6	&	22	6	38	F	54	V	70		86		102	f	118	v
7	'	23	7	39	G	55	W	71		87		103	g	119	w
8	(	24	8	40	H	56	X	72		88		104	h	120	x
9	)	25	9	41	I	57	Y	73		89		105	i	121	y
10	*	26	:	42	J	58	Z	74		90		106	j	122	z
11	+	27	;	43	K	59	[	75		91 ①		107	k	123	
12	,	28	<	44	L	60	\	76		92		108	l	124	|
13	–	29	=	45	M	61	]	77		93		109	m	125 ①	
14	–	30	>	46	N	62	^	78		94		110	n	126 ①	
15	/	31	?	47	O	63	–	79		95		111	o	127 ①	

1. Bei Grafik-Mode 0 muß zur Darstellung zuvor die ESC-Taste, CHR$ 27, gedrückt werden.

C. ATASCII Zeichensatz

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenanschlag	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	Õ
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	Tastenanschlag	Europäisches Zeichen
18	12		Control R	û
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	å
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	Tastenanschlag	Europäisches Zeichen
47	2F		/	
48	30		0	
49	31		1	
50	32		2	
51	33		3	
52	34		4	
53	35		5	
54	36		6	
55	37		7	
56	38		8	
57	39		9	
58	3A		Shift ;	
59	3B		;	
60	3C		<	
61	3D		=	
62	3E		>	
63	3F		Shift /	
64	40		Shift 8	
65	41		A	
66	42		B	
67	43		C	
68	44		D	
69	45		E	
70	46		F	
71	47		G	
72	48		H	
73	49		I	
74	4A		J	
75	4B		K	

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

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

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenanschlag	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	
177	B1		1	
178	B2		2	
179	B3		3	
180	B4		4	
181	B5		5	
182	B6		6	
183	B7		7	
184	B8		8	
185	B9		9	
186	BA		Shift ;	
187	BB		;	
188	BC		<	
189	BD		=	
190	BE		>	
191	BF		Shift /	
192	C0		Shift 8	
193	C1		A	

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

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

Dezimal Kode	Hexadezimal Kode	ATASCII Zeichen	Tastenanschlag	Europäisches Zeichen
254	FE		Esc Control Delete Bk Sp	
255	FF		Esc Control >	

ATARI

ATARI Corporation (Deutschland) GmbH
Frankfurter Straße 89-91 • 6009 Raunheim

C072018-004
Printed in Taiwan
K. I. 3. 1989