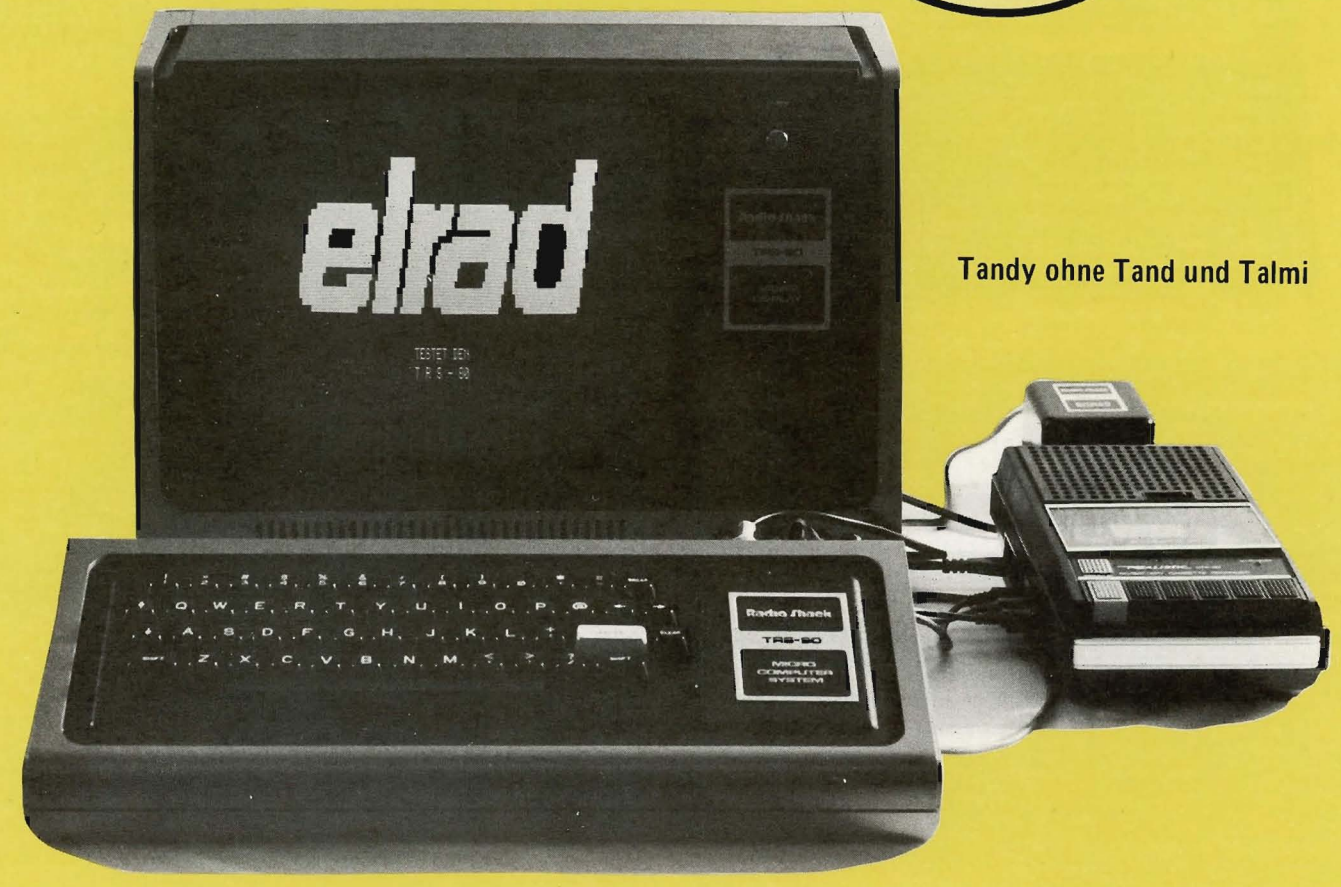
computing today

...jetzt neu

1



Tandy ohne Tand und Talmi

Der TRS-80 auf dem Prüfstand

S. Wittig

Mikrocomputer sind nicht mehr Privileg der Feierabend-Computer-Bastel-Profis. In Deutschland dürften bis Ende 79 ca. 25000 komplette Systeme stehen, in den USA schätzt man für den gleichen Zeitpunkt einen Bestand von 650000! Ohne Zweifel geht der Trend zum komfortablen Black-Box-System für Nur-Programmierer oder gar für die Benutzer-Gruppe, die sich mit Software vom Wühltisch begnügt. Auf die Dauer wird nur der Hersteller erfolgreich sein, der Passendes für diese Käuferschicht zu bieten hat. Die Tandy Corporation hat das offensichtlich klar erkannt, denn ihr TRS-80 hat sich innerhalb weniger Monate von einer in der Fachpresse oft geschmähten grauen Maus zum Marktführer mit 50% Marktanteil (Monatsproduktion zur Zeit 20000 Stück) erhoben. Dieser Bericht enthält Erfahrungen mit dem TRS-80 Level II-System in seiner Grundausbaustufe, bestehend aus dem 16 KByte-Steuergerät mit Tastatur, einem Video-Monitor und einem Kassettenrekorder.

Um bei der Wahrheit zu bleiben: Eigentlich waren es zwei Kassettenrekorder, aber wir wollen nicht vorgreifen, sondern schön der Reihe nach berichten, denn der Mensch ist nun mal vorwiegend seriell organisiert. Zunächst also die Technik.

Interna

Dreh- und Angelpunkt des TRS-80 ist der 8-Bit-Mikroprozessor Z80, der bei 1,78 MHz sozusagen im Schongang gefahren wird. Der TRS-80 ist ein Einplatinen-Computer, wie unser Foto (Bild 1) zeigt, wenn man von der Platine absieht, die die Tastatur trägt.

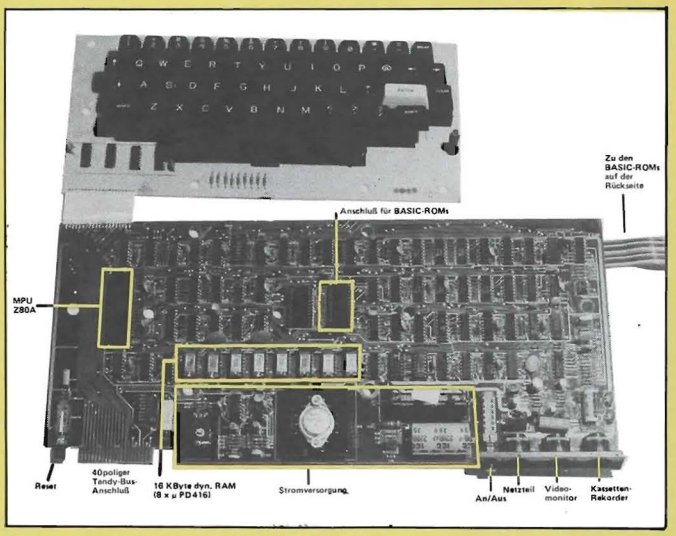


Bild 1. Die Platine des TRS-80.

Das uns zur Verfügung stehende System Level II hat 16 K RAM Arbeits-Speicher. Dieser Arbeitsspeicher kann extern mit Hilfe eines Expansion Interface auf maximal 48 K RAM erweitert werden. (Das ist eine Speichergröße, wie sie vor wenigen Jahren noch Anlagen der Mittleren Datentechnik zum Preis von $3 \cdot 10^5$ DM aufwiesen.) 12 K ROM speichern den Monitor und einen äußerst leistungsfähigen, kommerziell orientierten BASIC-Interpreter von Microsoft. Doch davon später.

Die wichtigsten Funktionsblöcke sind auf unserem Foto (Bild 1) eingezeichnet. Erwähnt sei hier, daß es den TRS-80 auch in einer Level I-Version gibt, mit einem 4-K-BASIC-Interpreter.

Die Tastatur ist eine 'professionelle' ASCII-QWERTY-Tastatur (53 Tasten) der Mittelklasse. Auf die Vorteile eines separaten numerischen Tastenfeldes (wie man es z. B. von einer weitverbreiteten Liliputaner-Tastatur her kennt) muß man verzichten. Jedoch nicht mehr lange: In den USA ist der TRS-80 mit einer solchen Option bereits angekündigt. Amerika, du hast es (immer vor uns) besser! Die ENTER (CR)-Taste besitzt doppelte Breite und hebt sich infolge ihrer weißen Farbe sehr gut aus ihrer eintönigen schwarzen Umgebung heraus. Leider sind nicht alle Funktionen der Tastatur beschriftet. So kann man z. B. mit SHIFT @ die Programmausführung oder das Auflisten unterbrechen. Eine entsprechende zusätzliche Beschriftung auf der @-Taste wäre hier gut gewesen. Auch die Funktionen des Editors, die teilweise über bestimmte Buchstabentasten gesteuert werden, hätte man auf diesen Tasten kenntlich machen sollen. Z. B. hat die Taste 'L' unter dem Editor die Funktion 'Zeile auflisten'.

Als Video-Bildschirm dient ein abgemagertes Fernsehgerät mit 12-Zoll-Bildschirm und vergrößerter Bandbreite. Kursor-Steuerung (rechts, links, nach unten) ist möglich, ebenso das Löschen des Bildschirms. Die Anzeige erfolgt normalerweise in 16 Zeilen à 64 Zeichen. Die Zeichen sind aus einer 5 x 7-Punkt-Matrix zusammengesetzt. In der Normaldarstellung macht das Schriftbild einen etwas gedrängten Eindruck (Bild 2), sehr gut lesbar ist die Schrift jedoch in einer doppelt breiten Darstellung (Bild 3), die durch Tastendruck (SHIFT →) oder eine BASIC-Anweisung (PRINT CHR$(23)) einstellbar ist.

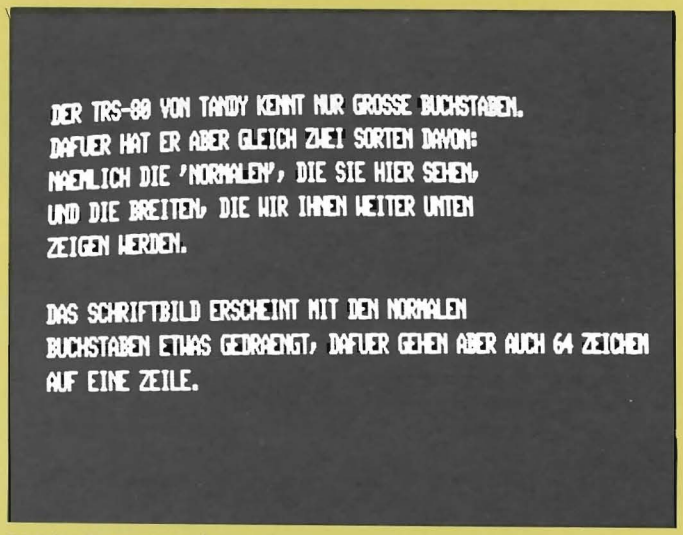


Bild 2. Normalschrift

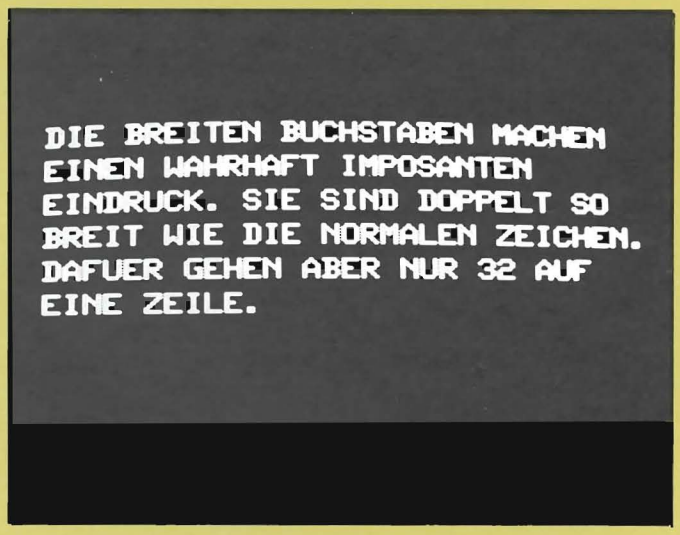


Bild 3. Schrift für diejenigen, die den Gang zum Augenarzt scheuen.

Auch unter Programmkontrolle sind schmale und breite Buchstaben leider nicht mischbar. Ein breiter Buchstabe belegt die Position von zwei schmalen Buchstaben. Das hat eine unangenehme Konsequenz z. B. beim Laden von auf Magnetbandkassetten gespeicherten Programmen: Ist hierbei der Bildschirm auf breite Buchstaben eingestellt, so sieht man statt eines stehenden und eines blinkenden Sterns (was normalerweise das Funktionieren des Ladevorgangs signalisiert) nur den stehenden Stern, denn der nimmt ja jetzt dem blinkenden Stern den Platz weg. Das ist zwar sehr logisch, aber nicht sehr komfortabel. Abhilfe würde hier ein Leerzeichen zwischen beiden Sternen schaffen.

Die Anzeige auf dem Bildschirm erfolgt mittels memory mapping: Den 16 x 64 = 1024 Anzeigepositionen des Bildschirms ist ein 1-K-RAM-Speicher zugeordnet (Adressen: 15360 bis 16383). Schreiben eines ASCII-codierten Zeichens in und Lesen aus diesem Speicherbereich mittels normaler Speicherzugriffs-Instruktionen bewirkt die Ausgabe bzw. das Lesen des entsprechenden ASCII-Zeichens auf den/vom Bildschirm. Aber auch in BASIC ist der Zugriff auf eine bestimmte Position des Bildschirms möglich:

1. Mit der Anweisung
   POKE 15360,65
   schreibt man z. B. ein 'A' in die oberste linke Position des Bildschirms.
2. Das gleiche erreicht man mit der Anweisung
   PRINT @ 0, "A"

3. Mit der Funktion
   PEEK (15360)
   erhält man den dezimalen ASCII-Wert des Zeichens, das gerade auf der obersten linken Position angezeigt wird.

Dieses memory mapping ist besonders wertvoll beim Erstellen von Graphiken auf dem Bildschirm.

Zur Grundausstattung des TRS-80 gehört ein Kassettenrekorder (REALISTIC CTR-80) und ein getrenntes Netzteil (für das Steuergerät), das man tunlichst so weit entfernt vom Monitor aufstellt, wie es das kurze Kabel erlaubt.

Insgesamt hebt man also 4 Bausteine aus der sorgfältigen Verpackung des belgischen Lieferwerks. Drei davon haben ein eigenes Netzanschlußkabel (Monitor, Kassettenrekorder, Netzteil). Hat man diese alle unter dem Tisch verstaut, dann steckt man das Kabel des Netzteils in eine Buchse an der Rückseite des Computergehäuses und das Monitorkabel in eine (nicht vertauschen!) Buchse an der Rückseite des Computergehäuses und das Kabel des Kassettenrekorders in eine Buchse an der Rückseite des Computergehäuses und die drei Netzanschlußkabel in eine hoffentlich vorhandene Dreifachsteckdose und ein Kabel in den 'EAR'-Eingang des Rekorders und ein Kabel in den Eingang REMOTE zur externen Steuerung des Rekorder-Motors, und dann kann man immer noch keines der beiden mitgelieferten Programme (Backgammon und Blackjack) laden, denn dazu muß man durch Probieren am Lautstärkeknopf des Rekorders die dem TRS-80 genehme Ausgangsspannung einstellen. Das machte uns einige Mühe, obwohl man durch den TRS-80 dabei unterstützt wird: Beim Laden von Programmen (500 Baud) erscheinen, wie bereits erwähnt, in der ersten Zeile des Bildschirms zwei Zeichen**, von denen das rechte zu Beginn einer jeden zu lesenden BASIC-Zeile kurz aufblinkt – vorausgesetzt, sie wird gelesen. Zum Schreiben eines Programms muß man das Schreibkabel in die Buchse AUX stecken. Dieses ständige Umstöpseln erschien uns so lästig, daß wir einfach einen weiteren Rekorder benutzten, der ständig angeschlossen und aufnahmebereit war. (Eine elegantere, aber arbeitsintensivere Lösung dieses Problems fanden wir in Kilobaud, Januar 1979, S. 55, wo ein 'TRS-80 Tape Controller' beschrieben wird.)

Der Vollständigkeit halber sei hier noch das Expansion Interface erwähnt, ein weiteres Gehäuse, dessen Innenleben den Anschluß von (lieferbaren!) Druckern und Mini-Floppy-Laufwerken ermöglicht, und/oder zusätzlichen RAM aufnehmen kann. Dieses Expansion Interface wird mit dem 40-Pin-Bus-Anschluß am Hinterteil des TRS-80 verbunden. Es bietet auch noch Anschlußmöglichkeit für einen zweiten Kassettenrekorder und Platz für ein RS232C-Interface, z. B. für den Anschluß eines Modems, Akustikkopplers, Kartenlesers usw.

BASIC

Die Größe des Monitors und des BASIC-Interpreters im Level II (12 K ROM) lassen einiges erwarten – und halten der Erwartung auch stand. Diese BASIC-Version ist mit Sicherheit allen vergleichbaren festgespeicherten BASIC-Interpretern der Preisklasse des TRS-80 Level II überlegen, zumindest was die Vielseitigkeit anbelangt. Wir haben am Ende des Artikels die Möglichkeiten dieses Interpreters sehr kompakt zusammengefaßt und wollen hier deshalb nur schlaglichtartig einige Besonderheiten herausgreifen, wobei wir die Monitor-Kommandos gleich mit einbeziehen. (Das bei Vorhandensein einer Floppy-Disk verfügbare BASIC Level III soll noch leistungsfähiger sein. Das zu testen, hatten wir aber keine Gelegenheit.)

Ca. 900 verschiedene Variablen können definiert werden, mit den Typen ganzzahlig, einfache Genauigkeit, doppelte Genauigkeit und Zeichenkette. Bereiche können definiert werden, wobei der Dimension nur Grenzen durch den vorhandenen Speicherplatz gesetzt sind.

Kommandos

Neben den 'üblichen' Kommandos wie NEW, RUN, LIST fanden wir die automatische Zeilennumerierung (in Abständen von 10) mittels des Kommandos AUTO sehr komfortabel, ebenso die Möglichkeit des Trace, der mit den Kommandos TRON und TROFF gesteuert wird. (Trace arbeitet folgendermaßen: Wenn während der Ausführung eines Programms zu einer neuen Zeile übergegangen wird, dann wird deren Zeilennummer auf den Bildschirm ausgegeben. Dies erleichtert die Fehlersuche beim Entwickeln eines Programms.)

Das Kommando SYSTEM gestattet das Laden von sog. Object Files (d. h. von Programmen, die im Maschinencode geschrieben sind) von einer Magnetbandkassette.

Mit den Kommandos CSAVE und CLOAD kann man selbstgeschriebene BASIC-Programme auf eine Magnetbandkassette ausgeben bzw. von einer solchen laden. Über unsere damit verbundenen Anfangsschwierigkeiten haben wir bereits berichtet. Zur Ehrenrettung des TRS-80 müssen wir aber hinzufügen, daß wir über mehrere Monate intensiver Programmierarbeit hinweg nie einen Lese- oder Schreibfehler erlebt haben. Mit CLOAD ? kann man ein Programm, das auf der Magnetbandkassette gespeichert ist, mit einem im Computer gespeicherten vergleichen. Das empfiehlt sich immer nach dem SAVEn eines Programms.

Ein/Ausgabe

Neben dem PRINT-Kommando zur Ausgabe auf dem Bildschirm gibt es noch das Kommando PRINT TAB, das eine Ausgabe mit Tabulatorfunktion darstellt, sowie ein PRINT @ (PRINT at) zur Ausgabe ab einer bestimmten beliebigen Stelle auf dem Bildschirm. Die Ausgabepositionen des Bildschirms sind von 0 bis 1023 durchnumeriert. PRINT @ 607, "A" z. B. schreibt ein A in die Mitte der 10. Zeile. Zeitweiliges Kopfzerbrechen bereitete uns der PRINT @-Befehl, wenn die Ausgabe auf breite Buchstaben eingestellt war. Ist die Bildschirmposition ungeradzahlig, dann wird zwar der PRINT @-Befehl ohne Murren entgegengenommen und ausgeführt, nur es erscheint nichts auf dem Bildschirm! Nur bei geradzahligen Bildschirmpositionen erscheint das Gewünschte.

Besonders für kommerzielle Anwendungen ist das Kommando PRINT USING nützlich. Mit Hilfe mehrerer Feldschlüssel kann das Ausgabeformat von Texten und Zahlen passend gewählt werden. Ganz einfaches Beispiel ohne und mit PRINT USING:

PRINT SIN (0.505) liefert die Zahl .483807

PRINT USING "# . ###"; SIN (.505)

liefert 0.484

also mit führender Null und drei Stellen mit Rundung.

Neben den bekannten Befehlen INPUT, DATA, READ, RESTORE gibt es auch Versionen der Befehle INPUT und PRINT für den Verkehr mit dem Kassettenrekorder. Beim TRS-80 können Daten auf eine Magnetbandkassette gespeichert und von dieser gelesen werden, ohne daß man mit OPEN und CLOSE entsprechende Dateien öffnen oder schließen muß. Die Kommandos OPEN und CLOSE sind für den Verkehr mit der Floppy-Disk vorbehalten.

Ein kleines Programmbeispiel soll das Speichern und Laden von Daten veranschaulichen (Bild 4).

In Zeile 10 werden die Werte der vorbesetzten Variablen A1 und B$ und die Konstante "DAS IST ALLES" auf die Kassette geschrieben. Zeile 20 ist eine Warteschleife, die dem Benutzer Gelegenheit geben soll, das Band zurückzuspulen und die entsprechende Verkabelung des Rekorders vorzunehmen. Die Schleife wird durch Druck auf eine beliebige Taste unterbro-

```
 2 PRINT"REKORDER ZUM SCHREIBEN VORBEREITEN"
 5 A1=-30.334:B$="ZEICHENKETTE"
 10 PRINT#-1,A1,B$,"DAS IST ALLES"
 15 PRINT"BAND ZURUECKSPULEN UND REKORDER ZUM LESEN VORBEREITEN"
 20 A$=INKEY$:IF A$=""THEN 20
 25 A1=0:B$=""
 30 INPUT#-1,A1,B$,L$
 40 PRINTA1,B$,L$
 50 END
READY
>RUN
REKORDER ZUM SCHREIBEN VORBEREITEN
BAND ZURUECKSPULEN UND REKORDER ZUM LESEN VORBEREITEN
-30.334        ZEICHENKETTE    DAS IST ALLES
READY
>_
```

Bild 4. Beispielprogramm für das Speichern und Lesen von Daten auf/von Kassette.

chen. Die Daten werden gelesen (Zeile 30) und auf den Bildschirm ausgegeben (Zeile 40).

Programmanweisungen

Eine Zusammenstellung der BASIC-Anweisungen findet man auf der letzten Seite des Artikels. Besonders hervorzuheben sind hier die Typdefinitionen DEFINT, DEFSNG, DEFDBL und DEFSTR. So bedeutet z. B.

10 DEFSTR L–Z

daß alle Variablen mit den Anfangsbuchstaben L bis Z vom Typ Zeichenkette sind. Nach der Zeile 10 kann man also programmieren: M = "MONTAG", anstatt z. B. M$ = "MONTAG".

Zeichenketten

Zur Verarbeitung von Zeichenketten stehen die üblichen Funktionen wie ASC, CHR$, LEFT$, RIGHT$, MID$ usw. zur Verfügung. Zur Eingabe eines einzelnen Zeichens über die Tastatur dient die Funktion INKEY$. Das wurde schon im Musterprogramm (Bild 4) für das Schreiben und Lesen auf bzw. von Magnetbandkassette gezeigt.

Arithmetische Funktionen

Neben den arithmetischen Funktionen SIN, COS, EXP, LOG usw. ist der Zufallszahlengenerator RND hervorzuheben, dem wir einen gesonderten Abschnitt widmen. In diesem Zusammenhang ist die Zuweisung RANDOM zu erwähnen. Ihr Zweck ist folgender: Da die Zufallszahlen per Programm nach einem bestimmten Algorithmus bestimmt werden (daher 'Pseudo'-Zufallszahlen), erhält man nach dem Anschalten des Computers immer die gleiche Folge von Zufallszahlen. Das kann z. B. bei Spielen sehr langweilig werden. Abhilfe schafft hier die Anweisung RANDOM, die dafür sorgt, daß die Zufallszahlenfolge jedesmal mit einem neuen, unvorhersagbaren Wert beginnt.

Weitere Besonderheiten

Zu diesen Besonderheiten gehören z. B. die graphischen Funktionen, denen wir einen eigenen Abschnitt gewidmet haben. Zu erwähnen wären noch die Befehle OUT und INPUT zum Ausgeben bzw. Lesen von Daten auf einen Port. Die Funktion POS liefert die augenblickliche Position des Cursors. Leider hat der TRS-80 in der hier beschriebenen Version keine programmierbare Uhr, was die Liebhaber von Spielen und die Bastler von Anschlüssen externer Geräte sicher sehr bedauern werden. Ein solcher Timer steht erst mit dem Expansion Interface zur Verfügung (s. unten). Weiter vermißten wir die festgespeicherte Kreiszahl π.

Editor

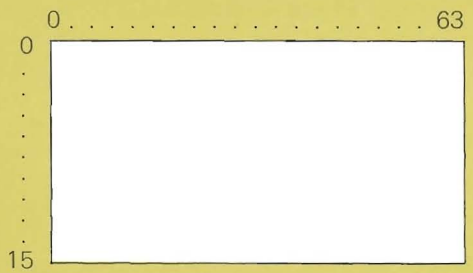
Level II-BASIC verfügt über einen sehr elaborierten Editor, der zum Korrigieren und zum Auflisten von Programmzeilen dient. Die Beschreibung nimmt allein schon in dem nicht sehr ausführlichen Level II-Handbuch 7 Seiten in Anspruch, wir müssen hier aus Platzgründen auf die Besprechung verzichten.

Erweiterungs-Interface

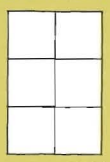
Mit dem Erweiterung-Interface (Expansion Interface) werden die Möglichkeiten von BASIC noch weiter ausgebaut: Ein zweiter Kassettenrekorder kann angesteuert werden, ebenso ein Drucker, bis zu 4 Mini-Floppy-Speicher und bis zu 48 KByte RAM. Außerdem besteht dann die Möglichkeit, hexadezimale und oktale Konstanten zu definieren und noch einiges mehr.

Graphik

Für sein Preisschild bietet der TRS-80 Level II recht gute graphische Möglichkeiten. Zunächst ist, wie erwähnt, der Bildschirm aufgeteilt in 16 Zeilen zu je 64 Zeichen, also in 1024 Felder, die auch in BASIC einzeln adressiert werden können:



Diese Felder sind ansprechbar mit PEEK, POKE und PRINT @. Jedes einzelne Feld aber ist wieder unterteilt in 6 weitere Felder:



so daß bei dieser Einteilung der Bildschirm 48 Zeilen und 128 Spalten, d. h. 6144 adressierbare Positionen hat. Jedes dieser Felder kann mit einem Rechteck weiß ausgefüllt werden. Aus diesen Rechtecken kann man dann graphische Darstellungen zusammensetzen. Ein Beispiel dafür ist der Schriftzug **elrad** auf dem Titelfoto. Zum Schreiben eines solchen 'graphischen Blocks' gibt es die BASIC-Anweisung SET (x, y), wobei x die Spaltennummer und y die Zeilennummer des zu schreibenden graphischen Blocks ist. Mit der Anweisung RESET (x,y) kann man einen graphischen Block auslöschen. Mit dem Statement CLS löscht man den gesamten Bildschirm und bringt den Cursor in die Home-Position (x = 0, y = 0). Mit POINT (x, y) kann man testen, ob an einer Stelle x,y ein graphischer Block gesetzt ist oder nicht.

Damit sind die graphischen Möglichkeiten des TRS-80 jedoch noch nicht erschöpft. Hinter den ASCII-Codes 129 bis 191 verbergen sich verschiedene graphische Zeichen, die durch unterschiedliche Anordnung von einem oder mehreren graphischen Blocks in dem oben beschriebenen Sechser-Feld zustande kommen. Alle diese Zeichen und den zugehörigen ASCII-Code zeigt das Bild 5.

Geht man von einem bestimmten Muster aus, so kann man sich den zugehörigen Code leicht nach folgendem Schema ausrechnen: Die 6 Felder einer normalen Schreibposition numeriert man folgendermaßen:

0	1
2	3
4	5

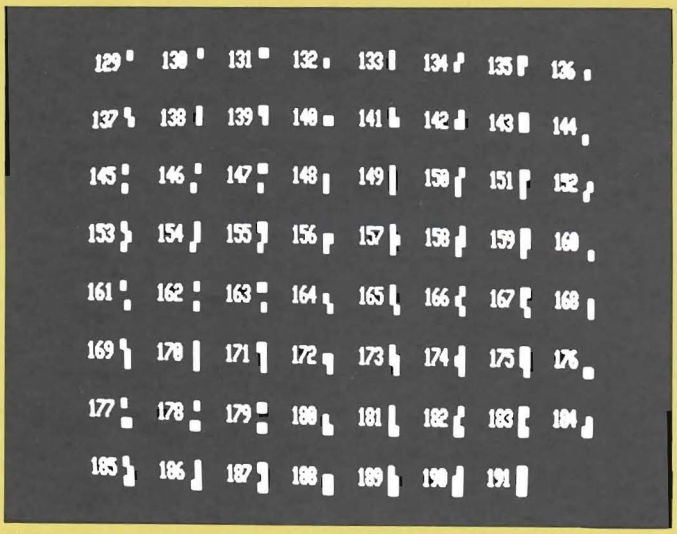


Bild 5. Zusammenstellung aller graphischen Symbole.

Diese Zahlen bedeuten eine Bitposition. Will man z. B. folgendes Muster haben:

so schreibt man eine '1' an die Stellen 0, 3 und 4 einer Dualzahl: 11001. Ihren dezimalen Wert (hier: 25) addiert man zu 128 und erhält damit den passenden ASCII-Code (hier: 153).

Außer zum Zeichnen von dekorativen Mustern können wir uns aber kaum einen sinnvollen Anwendungszweck vorstellen, offensichtlich sind diese Zeichen eher ein zufälliges 'Nebenprodukt' der Entwickler.

Maschinenbefehle

Mit Hilfe der Funktion USR ist es möglich, innerhalb eines BASIC-Programms ein im Maschinencode geschriebenes Programm aufzurufen und auszuführen. Ein Beispiel soll die Vorgehensweise veranschaulichen:

```
10 POKE 16526,0: POKE 16527,125
20 FOR I= 0 TO 26: READ A: POKE 32000+I, A: NEXT
30 B= USR (0)
40 DATA 17, 0,64, 33, 0, 60, 54, 191, 35, 231, 56, 250,
   6, 5, 33, 255, 255
50 DATA 43, 124, 181, 32, 251, 16, 246, 195, 0, 0
```

Das Maschinenprogramm soll in den Speicher ab Adresse 32000 geladen werden. Hexadezimal ist das 7D00. Diese Startadresse muß vor Aufruf des Programms in die Adressen 16526 und 16527 geladen werden, das höherwertige Adreßbyte (hex. 7D = dez 125) in die Adresse 16527. Das passiert in Zeile 10. In Zeile 20 wird der Maschinencode in den Speicher ab Adresse 32000 geladen. In Zeile 30 wird das Programm aufgerufen: Es bewirkt ein Weißschreiben des gesamten Bildschirms und, nach einigen Sekunden, eine Rückkehr zum Monitor, der sich dann meldet mit

MEMORY SIZE?

Das Weißwerden des Schirms geschieht in sehr kurzer Zeit sozusagen 'auf einen Schlag'.

Natürlich kann man auch mit einem BASIC-Programm den gleichen Effekt auf dem Bildschirm erzielen:

```
10 FOR X%=0 TO 127
20 FOR Y%=0 TO 47
30 SET (X%,Y%)
40 NEXT Y%,X%
```

Der 'kleine Unterschied' zwischen den beiden Programmen: Das BASIC-Programm benötigt volle 48 Sekunden, bis der Bildschirm fleckenlos weiß ist!

Doppelte Genauigkeit und ihre Grenzen

Im Normalbetrieb rechnet der TRS-80 mit einer Genauigkeit von 6 signifikanten Stellen. D. h. beispielsweise, die Division 2 durch 3 ergibt das Resultat 0.666667. Darüberhinaus bietet der Tandy aber noch die Möglichkeit des Rechnens mit doppelter Genauigkeit: Hier beträgt die Genauigkeit 16 signifikante Stellen. Will man mit doppelter Genauigkeit rechnen, so muß das zweite Zeichen eines Variablennamens das Nummernsymbol # sein, also z. B. C#. Läßt man beim Rechnen mit doppelter Genauigkeit keine besondere Vorsicht bei der Programmierung walten, so kann man sich selbst leicht ein Bein stellen. Das kleine Programm

```
10 C# = 2/3
20 PRINT C#
```

liefert z. B. die Zahl 0.6666666865348816 und nicht den erwarteten Wert 0.6666666666666667. Warum? Weil die Konstanten 2 und 3 keine 'doppelte' Genauigkeit besitzen. Richtig müßte man programmieren:

```
10 C# = 2.0000000 / 3.0000000
20 PRINT C#
```

wobei die Konstanten 2 und 3 mehr als 6 Nullen hinter dem Dezimalpunkt haben müssen, um als doppeltgenau erkannt zu werden. Dann erhält man das Resultat in der vollen Genauigkeit.

Wem das Eintippen dieser vielen Nullen zu albern ist, der kann auch programmieren:

```
10 A# = 2
20 B# = 3
30 PRINT A#/B#
```

Zum Thema Genauigkeit haben wir mit dem TRS-80 ein paar kleine Versuche unternommen. Und zwar haben wir ein kleines Programm geschrieben, das den Ausdruck

$$\frac{1}{\left(\left(\frac{1}{n}\right)^{1/n}\right)^{n}}$$

für verschiedene Werte von n berechnet. Nun weiß man, was herauskommen muß: nämlich n. Das findet man durch Vereinfachen des Ausdrucks. In der folgenden Tabelle sind einige eingegebene Werte von n und das vom TRS-80 berechnete Resultat gegenübergestellt.

Die Rechnungen wurden mit doppelter Genauigkeit durchgeführt.

n (Eingabe)	Resultat	
	Soll	Ist
1	1	1
1 1	1 1	10.9999870...
1 1 1 1	1 1 1 1	1110.766 . . .
1 1 1 1 1 1	1 1 1 1 1 1	107881.421 . . .
1 1 1 1 1 1 1	1 1 1 1 1 1 1	873081.533 . . .
1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1203956. . .
1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1	1

Man sieht, daß 16 Stellen noch lange nicht heißen, daß das Resultat auf 16 Stellen genau ist. Ist die eingegebene Zahl größer als 6 Stellen, dann kommt völliger Unsinn heraus. Und das Schlimmste ist: Normalerweise weiß man nicht, wann man dem Computer trauen kann und wann nicht mehr.

Ein anderes Beispiel. Wir spielen ein wenig mit der bekannten Beziehung $\sin^2 x + \cos^2 x = 1$. Setzen wir für x Zahlenwerte ein, so sollte uns der Computer immer den Wert 1 berechnen, unab-

hängig von x. Das sagt jedenfalls die Theorie. Die graue Praxis sieht anders aus, wie folgende Tabelle zeigt:

x (Eingabe)	Ergebnis (Soll = 1)
0.1234567890	0.9999999282881018
111111	0.9999999701976776
1111100	0.9378185 . . .
1111110	0.906203 . . .
1111111	1.06218 . . .
10000000	1.499999970197678
11111111	1
12345678	0.4999999701976776
111111111	0
1234567890	0

Da soll man noch Vertrauen zu seinem Computer haben! Das Problem klärt sich auf, wenn man berücksichtigt, daß beim TRS-80 die arithmetischen Funktionen SIN und COS (wie auch alle anderen mit Ausnahme von ABS, FIX und INT) nur Werte mit einfacher Genauigkeit liefern! Um so mehr Vorsicht ist also geboten, wenn man Ausdrücke berechnet, in denen Variable und Konstanten einfacher und doppelter Genauigkeit verwendet werden.

Fairerweise müssen wir hier nochmals zwei Dinge klarstellen: Erstens ist all dies keine besondere Eigenschaft des TRS-80, sondern ähnliche Ergebnisse würde uns jeder andere vergleichbare Computer oder Taschenrechner liefern. Probieren Sie es einmal! Mit den hier beschriebenen Aufgaben und den gewählten Zahlenwerten bewegen wir uns tatsächlich an der Grenze der Leistungsfähigkeit unseres BASIC-Interpreters. Das Schlimme ist nur, daß man die Grenzen normalerweise ignoriert oder gar nicht kennt. Fazit: Auch der vielseitigen Fähigkeiten eines Computers wie des TRS-80 soll man sich nicht unkritisch bedienen. Etwas Experimentieren verschafft uns ein Gefühl für die Grenzen der Leistungsfähigkeit eines jeden programmierten Algorithmus.

Ein Beispiel für die Vorteile der doppelten Genauigkeit sei hier noch angefügt:

Berechnen Sie einmal folgende einfache Aufgabe auf Ihrem Taschenrechner:

$$-1111111118.0 + 5.911111111 + 1111111119.0$$

Das exakte Ergebnis ist 6.911111111. Ein Privileg 1081 ESR-E (10 Stellen) lieferte das Resultat 6.0, ein TI SR-51-II (10 Stellen) zeigt 6.92, der Tandy aufgrund seiner großen Stellenzahl immerhin 6.911111056804657. Diese Aufgabe mag etwas künstlich erscheinen, solche Probleme (sehr große und sehr kleine Zahlen innerhalb eines zu berechnenden Ausdrucks) treten aber doch recht häufig auf.

Noch ein Hinweis allgemeiner Natur: Das kleine Programm

```
10 A# = EXP(1)
20 PRINT A#
```

liefert dieses eindrucksvolle Ergebnis

```
2.718281984329224
```

von dem allerdings alles hinter der 6. Dezimalstelle nicht mehr richtig ist. Obwohl A# eine Variable mit doppelter Genauigkeit ist, hat das Ergebnis wegen der EXP-Funktion nur einfache Genauigkeit. (Neuerdings liefert Tandy aber ein Programm, mit dessen Hilfe man mathematische Funktionen auch mit doppelter Genauigkeit berechnen kann: Programm-Nr. 26-1704.)

Geplanter Zufall

Level II-BASIC verfügt auch über einen Zufallszahlengenerator. Das ist ein Funktionsunterprogramm, das bei jedem Aufruf eine andere Zahl als Ergebnis liefert, offensichtlich nach den Gesetzen des Zufalls erzeugt. Aber nur offensichtlich, denn bei einem Computer ist nichts dem Zufall überlassen. So benötigt er auch zur Erzeugung von Zufallszahlen einen Algorithmus, so daß man genauer von der Errechnung von Pseudozufallszahlen spricht. Die Funktion RND(0) liefert bei jedem Aufruf eine 'Zufallszahl' zwischen 0 und 1. Die Funktion RND(10) z. B. liefert bei jedem Aufruf eine der Zahlen 1, 2, ..., 9, 10. Das Argument I in RND(I) kann Werte bis zu 32768 annehmen. Schaltet man den TRS-80 an, so erhält man bei wiederholten Aufrufen von RND(100) die Zahlen 77, 79, 8, 53, 5, ... Damit man nun nicht immer nach dem Anschalten des TRS-80 die gleiche Folge von Pseudo-Zufallszahlen erhält (was den Spaß an Spielprogrammen gründlich verleiden könnte), hat Level II-BASIC das Statement RANDOM. Wird RANDOM vor dem ersten Aufruf von RND(1) ausgeführt, dann beginnt die Serie jedesmal mit einer anderen Zufallszahl.

Natürlich interessieren hier zwei Fragen: Erstens, sind die Zufallszahlen wirklich 'zufällig', und zweitens, wie viele Zufallszahlen kann man aufrufen, ehe die Serie von vorn beginnt?

Zur Beantwortung der ersten Frage haben wir verschiedene Tests ausgeführt. Z.B. haben wir 10000 Zufallszahlen zwischen 1 und 10 hintereinander aufgerufen und gezählt, wie oft die einzelne Zahl auftrat. Hier das Resultat:

Zufallszahl	1	2	3	4	5	6	7	8	9	10
Häufigkeit	992	1049	972	999	979	1035	996	1000	982	996

Im (hypothetischen) Idealfall sollte bei gleichverteilten Zufallszahlen jede Ziffer 1000 mal auftreten. Unter Berücksichtigung der natürlichen statistischen Unregelmäßigkeiten kann man aber sagen, daß die Gleichverteilung sehr gut erfüllt ist. (Wie gut die Gleichverteilung erfüllt ist, zeigte uns ein statistischer Test, der sog. Chi-Quadrat-Test, auf den wir hier aber nicht näher eingehen wollen.)

Eine wichtige Eigenschaft von Pseudo-Zufallszahlengeneratoren ist die Periode, d. h. die Länge der Reihe, ehe sie sich wiederholt. Auch das wollten wir herausfinden, mußten aber ein Testprogramm nach mehr als drei Stunden abbrechen. Immerhin hatten wir bis dahin eine Länge von über 100000 Zufallszahlen ohne Periode erreicht, eine für die meisten Zwecke völlig ausreichende Zahl. (Immerhin gibt es einen sehr verbreiteten Mini-Computer, dessen Pseudo-Zufallszahlengenerator die Periode 8192 hat!)

Tempo, Tempo

Wie bereits mit dem Commodore PET-2001 (Elrad, Juli 1978) und dem Heathkit H8 (Elrad, Dezember 1978) haben wir auch den TRS-80 Level II einem kleinen Benchmark-Test unterzogen.

Die Programme BM1 bis BM7 stammen von Tom Rugg und Phil Feldman (Kilobaud, Juli und Oktober 1977), das Programm BM8 wurde von John A. Coll hinzugefügt (Computer Education, Februar 1978). Über einen vergleichenden Test werden wir bald in Elrad berichten.

Der TRS-80 erzielte folgende Zeiten:

BM1: 2,8 Sekunden
BM2: 11,2 Sekunden
BM3: 27,0 Sekunden
BM4: 27,8 Sekunden
BM5: 31,0 Sekunden
BM6: 50,6 Sekunden
BM7: 78,0 Sekunden
BM8: 11,8 Sekunden

Diese Zeiten sind handgestoppt. Der TRS-80 hat ja, wie wir bereits sahen, keine programmierbare Uhr.

Mit diesen Zeiten liegt der TRS-80 zwischen dem PET (der ja immer noch als einer der Schnellsten seiner Klasse gilt) und dem Heathkit H8. Beim Programm BM8 liegen alle drei besprochenen Rechner Kopf an Kopf. Für Zahlenfetischisten: Im Mittel benötigte der TRS-80 für die ersten 7 Programme

das 1,5fache der PET-Zeit. Für diese 7 Programme, wohlgemerkt. Auf welch wackligen Beinen solch ein pauschalierender Zeitvergleich steht, sieht man ja am Programm BM8, das keinen merklichen Unterschied brachte.

Übrigens haben wir das Programm BM8 auch mal mit dem neuen ITT APPLE II gerechnet. Resultat: Bei 9 Stellen Genauigkeit benötigte der APPLE nur unwesentlich weniger Zeit als der TRS-80.

Schneller, kleiner

Wie wir bereits sahen, gehört der TRS-80 nicht gerade zur 'schnellen Truppe' unter den Mikrocomputern. Um so wichtiger mögen einige Programmierregeln sein, die man beachten sollte, wenn man sein Programm schneller machen will (sofern Sie überhaupt zu den Geschwindigkeits-Aposteln gehören!). Das wird im folgenden an vier Versionen ein und desselben Programms gezeigt.

Version 1

```
10 FOR I = 0 TO 360
20 PRINT I, SIN (I*3.14159/360)
30 NEXT
```

Dieses Programm berechnet den Sinus der Winkel von 0 bis 360^{o} in Schritten von 1^{o} und gibt den Wert auf den Bildschirm aus. Speicherbedarf: 47 Bytes, Zeitbedarf: 44 Sekunden.

Version 2

```
10 C = 3.14159/180
20 FOR I = 0 TO 360
30 PRINT I, SIN(I*C)
40 NEXT
```

Die Konstante 3.14159/180, mit deren Hilfe man das Gradmaß ins Bogenmaß umwandelt, wird hier nur einmal berechnet. Das Ergebnis ist frappierend: Der Speicherbedarf beträgt jetzt zwar 55 Bytes, aber die Laufzeit ist von 44 auf 28 Sekunden gesunken.

Version 3

```
10 C = 3.14159/180
20 FOR I% = 0 TO 360
30 PRINT I%, SIN(I%*C)
```

Die Laufvariable der FOR-Schleife ist jetzt eine Integer-Variable (I%). Auch das bringt etwas ein: Die Laufzeit beträgt jetzt 26 Sekunden, Speicherbedarf: 58 Bytes.

Version 4

```
10 C=3.14159/180:FOR I%=0 TO 360:? I%,SIN(I%*C):NEXT
```

Aus dem Programm ist jetzt ein 'Einzeiler' geworden. Laufzeit: 26 Sekunden, Speicherbedarf: 42 Bytes.

Die Ergebnisse auf eine Blick:

Version	Laufzeit	Speicherbedarf
1	44 Sekunden	47 Bytes
2	28 Sekunden	55 Bytes
3	26 Sekunden	58 Bytes
4	26 Sekunden	42 Bytes

Natürlich sind die Erfahrungen, die wir anhand dieser Programme auf dem TRS-80 gesammelt haben, auch übertragbar auf andere Systeme.

Dokumentation

Das ausgeklügeltste Computer-System kann nur Frustration erzeugen, wenn es nicht von vollständigen und lesbaren Handbüchern begleitet wird. Auch auf diesem Sektor hat man bei Tandy einiges zu bieten. Das Level-II-System wird in einem deutschen Handbuch auf ca. 130 Seiten hinreichend ausführlich beschrieben. Im Gegensatz zur englischen Ausgabe verfügt dieses Handbuch sogar über ein Stichwortverzeichnis.

Spitzenleistung der Tandy-Dokumentation ist jedoch das Handbuch zum Level I-System. Dieses gibt es in Deutsch unter dem tiefstapelnden Titel 'Bedienungsanleitung für Level I'. Hier handelt es sich nämlich nicht nur um eine Bedienungsanleitung, sondern um eine von David A. Lien geschriebene Einführung in BASIC, die so leicht und locker geschrieben ist, wie das nur amerikanische Lehrbuchautoren vermögen. Wenn es auch gewisse Unverträglichkeiten zwischen Level I und Level II gibt, so ist dennoch die Lektüre dieses Buches auch für Level-II-Novizen unbedingt empfehlenswert.

Software

Commodore PET, Tandy TRS-80 und APPLE II, das sind zur Zeit die großen Drei auf dem Hobby-Computer-Markt. Für diese drei ist das Angebot an Software schon sehr reichhaltig. Wir hatten Gelegenheit, uns eine Anzahl von Programmen für den TRS-80 anzuschauen, darunter ein 'Ham Package I', das aus drei Lehrprogrammen zu Themen aus der Elektronik besteht, sowie eine Kassette 'Electronics I', die drei Programme enthält, darunter ein Programm, das wahlweise einen monostabilen oder astabilen Multivibrator mit dem 555 Timer konstruiert und den Schaltplan auf dem Bildschirm zeichnet. Noch beeindruckender war ein Programm zur Konstruktion eines Vorverstärkers mit dem LM381. Den ausgegebenen Schaltplan sehen Sie in Bild 6. Diese Instant Software-Programme sind in Deutschland bei Kilobaud, 7778 Markdorf erhältlich.

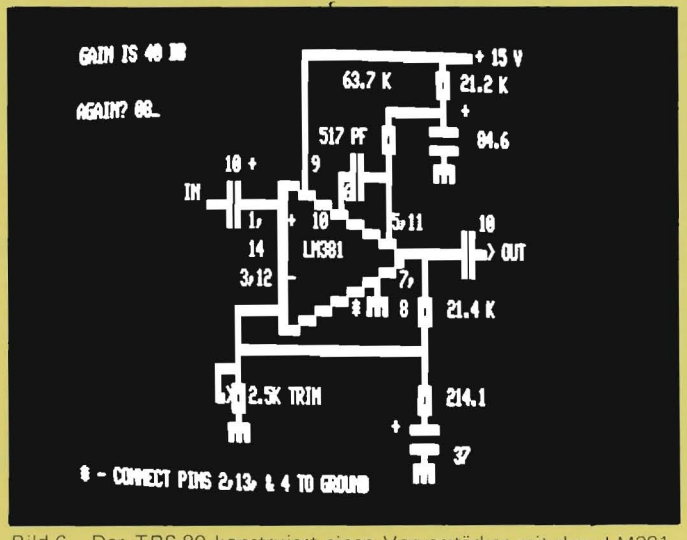


Bild 6. Der TRS-80 konstruiert einen Vorverstärker mit dem LM381.

Fazit

Wenn man versucht, eine abschließende Zuordnung des TRS-80 Level II zu einer bestimmten Gruppe potentieller Benutzer vorzunehmen, dann kann man sagen, daß dieses System geeignet ist für diejenigen Anwender, für die ein starkes BASIC wichtig ist und die sich wenig oder kaum für Mikrocomputer-Hardware interessieren. Design und Dokumentation des TRS-80 sind nicht für den typischen Bastler ausgelegt. Das spiegelt sich auch am Zubehörmarkt wider. Dinge wie A/D-Wandler, Sprach-Ein/Ausgabe-Bausteine, Plotter, S-100-Bus-Adapter, kurz alles für die Kommunikation mit der nicht-druckenden und -speichernden Außenwelt findet man selbst beim Durchblättern der einschlägigen amerikanischen Magazine nur recht spärlich. Dafür aber ist heute schon das Angebot an Software, besonders auch für kommerzielle Anwendungen, erstaunlich umfangreich. Der TRS-80 kann ideal sein für den programmierenden Privatmann, den Selbständigen und den Kleinunternehmer, den es beruhigen mag, zu wissen, daß er mindestens in der nächsten Großstadt einen autorisierten Kundendienst finden kann.

TRS-80 Level II-BASIC

Sie finden hier eine Zusammenstellung der Möglichkeiten, die Level II-BASIC bietet.

Standard-BASIC-Anweisungen

LET: Zuordnung (kann weggelassen werden). DATA: Werte, die durch READ einer Variablen zugewiesen werden, READ: Liest Daten, die durch DATA zugewiesen wurden, ins Programm ein. IF ... THEN: Erlaubt logische Vergleiche. FOR ... NEXT: Ermöglicht Schleifen. GOTO: Sprungbefehl. PRINT: Ausgabe von Daten auf den Bildschirm. DIM: Dimensionierung von Bereichen. END: Letzte Anweisung eines Programms.

Erweiterte BASIC-Anweisungen

RESTORE: Ermöglicht mehrfaches Lesen von DATA. REM: Erlaubt Kommentare im Programm. GOSUB und RETURN: Ermöglicht Unterprogrammtechnik. On ... GOTO: Für berechnete Verzweigungen. ON ... GOSUB: Berechneter Sprung zu einem Unterprogramm. INPUT: Eingabe von Werten über die Tastatur. FOR ... TO ... STEP ... NEXT: Schleife mit wählbarer Schrittweite. IF ... THEN ... ELSE: Test mit zwei Alternativen. STOP: Beendet laufendes Programm. DEFDBL, DEFINT, DEFSNG, DEFSTR: Dient der Datentyp-Deklarierung (doppelte Genauigkeit bzw. ganzzahlig bzw. einfache Genauigkeit bzw. Zeichenkette). CLEAR: Reserviert Speicher für Zeichenketten. ERROR: Simuliert Fehler. Zum Test der: ON ERROR ... GOTO-Anweisung: Verzweigung im Falle eines Fehlers. RESUME: Wiederaufnahme der Programm-Ausführung nach einer Fehler-Routine. RANDOM: Randomisiert den Zufallszahlen-Generator.

Graphik-Anweisungen

CLS: Löscht den Bildschirm. RESET: Löscht ein Graphik-Rechteck. SET: Schreibt ein Graphik-Rechteck: POINT: Prüft, ob ein Graphik-Rechteck gesetzt ist.

Funktionen/Arithmetische Funktionen

ABS: Absolutwert. ATN: Arcustangens. CDBL: Gibt Wert in doppelter Genauigkeit. CINT: Gibt größte ganze Zahl, die nicht größer als das Argument ist. COS: Cosinus. CSNG: Wandelt aus doppelter in einfache Genauigkeit um. EXP: Natürliche Exponentialfunktion. FIX: Streicht Stellen hinter dem Dezimalpunkt. INT: Gibt größte ganze Zahl, die nicht größer ist als das Argument. LOG: Natürlicher Logarithmus. RND: Pseudo-Zufallszahlengenerator. SGN: Ermittelt Vorzeichen. SIN: Sinus. SQR: Quadratwurzel. TAN: Tangens.

Spezielle Funktionen

ERL: Liefert Zeilennummer eines Fehlers. ERR: Gibt einen Fehlercode. INP: Liest ein Byte von einem Port ein. MEM: Gibt Größe des freien Speichers an. VARPTR: Liefert Adresse einer Variablen.

Systemanweisungen

AUTO: Automatische Zeilennumerierung. CLEAR: Setzt Variablen auf Null. CLOAD: Lädt BASIC-Programm von Magnetbandkassette. CLOAD?: Vergleicht residentes Programm mit Programm auf der Magnetbandkassette. CONT: Fortsetzung der Programmausführung nach BREAK oder STOP. CSAVE: Schreibt BASIC-Programm auf Magnetbandkassette. DELETE: Streicht Programmzeilen. EDIT: Umschalten zum Editor. LIST: Auflisten der Programmzeilen auf den Bildschirm. NEW: Residentes Programm löschen. RUN: Start der Programmausführung. SYSTEM: Zum Laden von Maschinenprogrammen von Magnetbandkassette. TROFF: Trace abschalten. TRON: Trace anschalten.

Editor

Das Editieren wird durch 16 Kommandos, Subkommandos und Funktionstasten ermöglicht.

Zeichenketten-Funktionen

LEFT$, RIGHT$, MID$: Aus einer Zeichenkette können genau spezifizierbare Teile herausgenommen werden. CHR$: Wandelt ASCII-Code in Zeichen um. ACS: Wandelt Zeichen in ASCII-Code um. STR$: Wandelt Zahlen in Ziffernketten um. VAL: Wandelt Ziffernketten in Zahlen um. LEN: Ermittelt die Länge einer Zeichenkette. FRE: Gibt Größe des Speichers für Strings an. INKEY$: Liest einzelne Zeichen von der Tastatur. STRING$: Liefert einen String gleicher Zeichen, Länge wählbar.

Formatierungsanweisungen

TAB: Tabulator. PRINT USING: Dient der Formatierung der ausgegebenen Daten. POS: Liefert Cursorposition. PRINT@: Ausgabe von Daten, beginnend an einer bestimmten Position auf dem Bildschirm.

Logische Operatoren.

AND, OR, NOT.

Maschinensprache-Anweisungen

PEEK: Liest einzelne Bytes aus dem Speicher. POKE: Speichert Werte in spezifizierte Adressen. USR: Verzweigung zu Maschinenprogramm. OUT: Ausgabe eines Bytes auf einen Port.

Erweiterte Ein/Ausgabe-Anweisungen

PRINT#-1: Ausgabe von Daten auf Kassette Nr. 1.
INPUT#-1: Eingabe von der Kassette Nr. 1.

Variable

Typen: Reell (einfache Genauigkeit (!). 6 signifikante Stellen, doppelte Genauigkeit (#), 16 signifikante Stellen, einfache Genauigkeit mit Exponent (E), doppelte Genauigkeit mit Exponent (D)), ganzzahlig (%), Zeichenketten ($) (bis 255 Zeichen).

Namen: Ein oder zwei Buchstaben, oder ein Buchstabe und eine Zahl. Längere Namen sind erlaubt, aber nur die beiden ersten Zeichen werden als der Name der Variablen angesehen. Bei allen Variablentypen sind mehrdimensionale Felder möglich.

Expansion Interface (TRS-80-Zusatzgerät)

Ein zweites Kassettengerät kann angesprochen werden: INPUT#-2, PRINT#-2, CLOAD#-2, CSAVE#-2.

LLIST: Gibt BASIC-Programm auf den Drucker aus. LPRINT: Druckausgabe.

Speicherbedarf des Interpreters

		Bytes RAM
EA-Puffer, Tabellen, Arbeitsbereich		812
Zeilennummer, Zeilenpointer, CR	zusammen	5
Schlüsselwort		1
Zeichen		1
Variable (nach der Wertzuweisung)		
Integer		5
Einfache Genauigkeit		7
Doppelte Genauigkeit		11
Zeichenkette		6 + Länge
Bereiche	mindestens	12

Während der Programmausführung wird Speicherplatz dynamisch zugeordnet.

Es benötigen:	Bytes RAM
Jede aktive FOR ... NEXT-Schleife	16
Jeder nicht beendete Unterprogrammaufruf	6
Jede Klammer in einem Ausdruck	4
Jedes Zwischenergebnis bei der Berechnung eines Ausdrucks	12