Lüers

MSX

Für Einsteiger

EIN DATA BECKER BUCH

Lüers

MSX

Für Einsteiger

EIN DATA BECKER BUCH

ISBN 3-89011-085-1

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Sehr geehrte liebe Leserin!
Sehr geehrter lieber Leser!
Sehr geehrte liebe Programmiererin!
Sehr geehrter lieber Programmierer!

Vor Ihnen liegt nicht nur ein Buch mit dem Titel 'MSX für Einsteiger', ich darf auch annehmen, daß Sie Ihren Computer bereits in Reichweite aufgebaut haben und nun hoffen, daß dieses Buch Ihnen die vollkommene Bedienung dieses Wunderkästchens erleichtern hilft.

Das freut mich, denn dann ist dieses Buch in den richtigen Händen! Was ich damit sagen will ist folgendes: Genauso wie beim Erlernen einer Fremdsprache gehört der Wille des Lernenden dazu, um erfolgreich abzuschneiden. Es kann nicht angehen, daß man sich mit Literatur zudeckt, nur um dahinter die eigenen Unfähigkeiten zu verbergen. Sie müssen lernen und begreifen wollen, sonst sind Sie beim Thema Computer nicht richtig!

Zwar will ich an dieser Stelle nichts vorwegnehmen, aber neben dem Inhaltsverzeichnis soll Ihnen in diesem persönlichen Anschreiben doch wenigstens mitgeteilt werden, was Sie erwartet: Sie werden in der Lage sein - wenn Sie wollen -, nach dem praktischen Durcharbeiten dieses Buches BASIC-Programme auf Ihrem MSX-Computer schreiben zu können. Das heißt nicht, daß Sie zu diesem Zeitpunkt bereits das vollständige MSX-Vokabular beherrschen werden (für spezielle Anwendungswünsche gibt es auch spezielle Literatur von DATA BECKER z.B. für Maschinensprache oder Grafik & Sound ... und schließlich besitzen Sie ja auch noch eine Gebrauchsanleitung für Ihren MSX-Computer, in der alle MSX-Befehle angesprochen werden).

Dieses Buch erhebt lediglich den Anspruch, bei Ihnen eine gute Basis sowohl für die Computersprache BASIC als auch für das ganze Drumherum beim Computer (Anschlußmöglichkeiten) und um den Computer herum (was ist das überhaupt?) zu schaffen.

Bei diesem 'genußvollen' Lernprozeß wünsche ich Ihnen viel Freude! Haben Sie Probleme oder Fragen, können Sie mich unter 025147478 per BTX erreichen oder aber unter der gleichen Nummer Ihre Computerprobleme (oder die mit diesem Buch) auf Band sprechen. Bitte geben Sie Ihre Adresse und vielleicht auch Ihre Telefonnummer an. Ich melde mich so schnell wie möglich.

Rainer Lüdemann

Bei der Arbeit für dieses Buch haben mich folgende MSX-Firmen mit Rat und Tat unterstützt:

SONY
JÖLLENBECK
YAMAHA
CE-TEC
PHILIPS
SANYO
PANASONIC
GOLDSTAR

Außerdem berieten mich Herr Cole und Frau Heiligenstühler von der MSX-Arbeitsgemeinschaft.

Für das äußere Erscheinungsbild (Karikaturen und Fotos) in diesem Buch zeichnen verantwortlich:

Ronald Goergen
Olaf Schellenberger
Daniel Stoffregen
Andreas Lechtape

Vielen Dank all denen, die während meiner Arbeit an diesem Buch Verständnis zeigten und mich seelisch immer wieder zum Durchhalten ermahnten. Besonders erwähnen möchte ich hier meine lieben Eltern, meinen Bruder Heinz-Eckhard, alle großen und kleinen Computerfreaks aus Münster und Hänschen-BTX-Lindemann.

Vielen Dank!

Gewidmet ist dieses Buch allen meinen lieben Kolleginnen und Kollegen in der Hauptverwaltung und in den einzelnen ZWs von HORTEN.

INHALTSVERZEICHNIS

Was hat MSX für eine Bedeutung?

Sehr geehrte, liebe MSX-Freunde und MSX-Interessenten!

Bisher schauten die Privatleute, die sich nach reiflicher Überlegung zum Kauf eines Home- oder Heimcomputers entschlossen hatten, neidisch auf die größeren Computer, denn für diese Gerätefamilie waren schon vor Jahren vorzügliche Standards geschaffen worden. Die Namen 'CP/M' und 'MS-DOS' sowie der Ausdruck 'IBM-Kompatibilität' gewährleisteten und tun dies gerade heute in immer stärkerem Maße, daß die Programmvielfalt von Computer X auch auf Computer Y ohne Probleme lief und umgekehrt selbstverständlich ebenso. Da gibt es keine Eifersuchtsprobleme nach dem Motto: 'Warum kann ich diese Datenbank ausgerechnet nicht auf meinem Computer laufen lassen?!'.

Wie sah und sieht es heute noch bei den meisten Homecomputern aus? Gerätehersteller X bringt einen Computer heraus, zu dem natürlich die Geräteanschlüsse und auch die Programmvielfalt nur dieses einen Herstellers passen. So zieht der Kauf des Computers X im Normalfall auch ein beachtliches Nachfolgegeschäft nur bei diesem Hersteller nach sich, woran sich manche bereits ein goldenes Näschen verdient haben. Ist der Markt mit Produkt X halbwegs gesättigt, bringt der gleiche Hersteller ein Produkt Y heraus. Damit auch hiernach wieder per Nachfolgegeschäft die Kassen klingeln, werden halt neue Anschlüsse (wieder keinem Standard genügend) am Computer angebracht, und - wo käme der arme Hersteller denn sonst auch hin - die Software von Produkt X läuft selbstverständlich nicht mehr auf dem Computer Y. Ob dieser Hersteller nicht irgendwann einmal mit dieser Politik auf die Nase fallen wird ... warten wir es ab ... die MSX-Computer können nicht nur, sie müssen Sieger am Ende der Wegstrecke sein, denn hier wird niemand mehr für 'dumm' verkauft.

Wie kann so etwas gelingen? Alle japanischen Firmen aus dem HIFI-VIDEO-Bereich haben sich vor einiger Zeit gemeinsam an einen großen Tisch gesetzt und beratschlagt, wie den Wirtschaftsmächten USA und Europa mal wieder ein Schnäppchen mit bombastischer Wirkung geschlagen werden könne (siehe auch die Entwicklung bei Kameras, dem HIFI-VIDEO-Sektor und der Automobilproduktion nicht zuletzt).

Da man den Homecomputermarkt nicht als erster betreten würde, mußte nun gemeinsam etwas besonderes ausgegoren werden. Schließlich einigte man sich darauf, Homecomputer auf den Markt zu bringen, die nicht nur Kompatibilität (Austauschbarkeit) bei der Software (Programme) sondern auch bei der Hardware (Anschlüsse zu weiteren Geräten) gewährleisten sollten.

Man stellte all' dieses unter den Tenor 'MSX', eine Abkürzung für das leistungsfähigste BASIC, das ... mal wieder von 'MICROSOFT', einer äußerst renommierten amerikanischen Softwarefirma, stammt: 'Microsoft Super Extended Basic'.

Doch der MSX-Standard umfaßt mehr: Nicht nur das BASIC ist gleich, auch die Computerchips für Sound und Grafik gleichen bei allen MSX-Computern wie ein Ei dem anderen. Außerdem hat jeder MSX-Computer, soweit zur Zeit zu sehen, eine Standard Centronics Schnittstelle, die Verwendung von kommerziellen CP/M-Programmen (s.o. - eigentlich kommt diese Software ja von den größeren Computern her!) ist zu 100% möglich und last but not least: Das MSX-Diskettenbetriebssystem MSX-DOS ist datenkompatibel zum MS-DOS von IBM ... d.h.: Sie können sich Ihre Daten aus dem Büro nach Hause mitnehmen (Vorsicht! Datenschutz!) und in aller Ruhe auf Ihrem MSX-Computer mit angeschlossener Diskettenstation weiterverarbeiten!!!

Sie merken vielleicht, eine Überlegung der Japaner, die Hand und Fuß hat. Diese Idee muß einfach erfolgreich sein, anders kann es gar nicht sein!

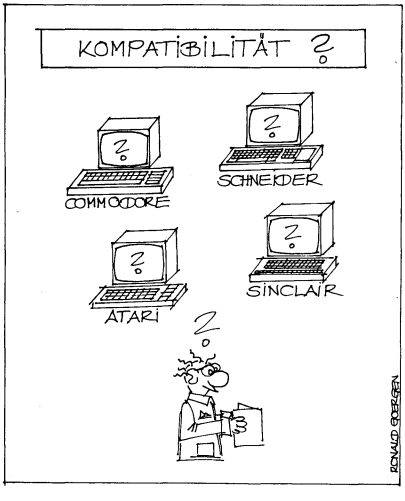
Zwei Punkte seien noch angefügt:

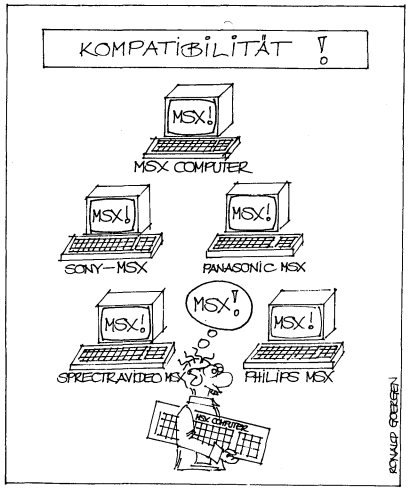
1) Weil die Hersteller von MSX-Computern fast allesamt aus der HIFI-VIDEO-Branche stammen, ist der Anschluß des MSX-Homecomputers z.B. an den Videorecorder, den Bildplattenspieler, die Tonbandmaschine usw. gar nicht mehr fern (Beispiele dafür gab es bereits 1984 auf der HIFIvideo in Düsseldorf und 1985 auf der Hannover Messe en masse zu sehen). Also ein Homecomputersystem, das wahrlich ins häusliche Innenleben hereingehört, ja sich sogar in seine Umgebung schnellstens einpaßt.

2) Nicht nur die Anwendungsmöglichkeiten der MSX-Computer bestechen! Auch technische Daten wie Ausbaubarkeit auf 256 KB RAM, 32 Sprites, 16 Farben, hochauflösende Farbgrafik mit 256*192 Punkten und ein eingebauter dreistimmiger Soundprozessor sprechen für den Computerkenner geradezu Bände!

Warten wir ab, wie sich der Markt umkrempelt. Die Japaner (repräsentiert durch mehr als zwanzig Firmen mit renommierten Namen wie SONY, SPECTRAVIDEO, PANASONIC, SANYO, YAMAHA, TOSHIBA usw.), die Koreaner (u.a. GOLDSTAR und CE-TEC) und auch PHILIPS aus Deutschland/Holland erhoffen, daß bereits 1985 ein Großteil aller verkauften Homecomputer das Kürzel MSX tragen wird. Machen wir es dem europäischen Ausland nach! Toi, Toi, Toi MSX!

P.S.: Die Begriffe 'Homecomputer' und 'Personalcomputer' sind
==== inzwischen so nichtssagend, daß wir bei allen MSX-Computern in jedem Fall auch von Personalcomputern sprechen können.





Die Computerentwicklung mitsamt ihrem Fachchinesisch

Auf den ersten Seiten dieses Buches haben Sie ja bereits darüber Informationen erhalten, warum der MSX-Computerstandard etwas ganz besonderes ist.

In diesem Kapitel will ich Sie nun erst einmal ein klein wenig tiefer in die Computerei einführen. Dabei werden Sie Begriffe wie RAM, ROM, Diskette usw. etwas näher kennenlernen, damit es Ihnen anschließend möglich ist, sich im Kreis der Computerfreaks besser bewegen zu können.

Nicht nur das. Wir werden uns auch über die schnelle Entwicklung der Computer unterhalten bis zum heute vor Ihnen stehenden Endprodukt, Ihrem eigenen MSX-Computer. Nicht zuletzt soll Ihnen dieses Kapitel erste Einblicke in die Anwendung eines Computers und somit auch die Ihres MSX-Rechners vermitteln.

1. Wie schnell ist die Zeit vergangen?

Ein Besuch des Deutschen Museums in München lohnt allemal, nur sollte man dazu genügend Zeit mitbringen. In der Abteilung, die die Entwicklung des Computers eindrucksvoll vorführt, stößt man in einer Ecke auf eine große Rechenmaschine, die der Deutsche Konrad Zuse 1941 gebaut hat. Dies war der erste frei programmierbare Computer. Die Ausmaße entsprechen in etwa einer großen Schrankwand, die Leistungsfähigeit allerdings war im Vergleich zu heutigen Rechnern sehr gering.

Worauf ist einerseits die lahme Leistungsfähigkeit, andererseits die enorme Größe zurückzuführen? Eigentlich beide Male auf den gleichen Ursprung: die mechanische Art und Weise, mit der dieser Rechner an seine Arbeit ging.

Wir werden in diesem Kapitel noch einen kurzen Einblick in die Computerrechenweise vornehmen können, hier wollen wir uns mehr auf die Mechanik konzentrieren: Der Zuse-Rechner arbeitete mit einer Vielzahl von Relais, die in einer Kette zusammengeschaltet Informationen durch zahlreiche An- und Aus-Zustände festhalten konnten.

Man kann sich vielleicht vorstellen, welch ein Krach durch diesen Rechner in Betrieb erzeugt wurde: Andauernd öffneten sich Relais ... 'klack' ... schlossen sie sich wieder. Nicht nur das. Durch die Mechanik entstand ein Verschleiß ohnegleichen. Auf einmal stimmte das durch Lämpchen angezeigte Rechenergebnis nicht mehr - unvorstellbar selbst für einen Taschenrechner in der heutigen Zeit - dann war wieder einmal eines der Relais defekt und die Arbeit mußte von neuem beginnen oder aber eine Wanze (Englisch: bug) war in dieses klappernde Wunderwerk hineingekrochen und hinderte nun ein Relais am Zusammenklappen = dem Schließen des Stromkreises. Aus diesem Ursprung heraus hat sich dann in der Computersprache die Vokabel 'bug' = Fehler herausgebildet (heute sprechen Programmierer beim Korrekturlesen ihrer Werke deshalb auch von 'Debuggen' = Fehlersuchen).

Doch müssen wir auf dem Pfad der Computergeschichte bleiben: Wir haben festgestellt, daß der Zuse-Rechner noch zu langsam und vor allem zu groß war. Einige Jahre später hatten dann die Entwickler die rettende Idee: Aus dem so verschleißfreudigen Relais wurde der Transistor, ein erheblich kleinerer elektronischer Baustein, der zwar dieselbe Arbeit wie das Relais leistete, jedoch nicht mehr mechanisch arbeitete, sondern rein elektronisch (für die Erzeugung des An- bzw. Auszustandes zeichnet eins seiner drei Beinchen verantwortlich).

Ein Computer mit Speicherstellen in Form von Transistoren wurde schnell entwickelt und der Größenunterschied zum Zuse-Rechner war schon gewaltig.

Doch die Erfinder und Entwickler arbeiteten weiter. Schließlich kam man auf die Idee, lediglich elektrische Bahnen, die die Transistoren in sich und untereinander miteinander verbinden, zweidimensional auf ein Papier aufzuzeichnen. Bei einer Anzahl von meist tausenden von Transistoren kann man sich gut vorstellen, wie groß diese Zeichnung aussehen mußte - also was den Miniaturisierungsfortschritt anbetrifft eigentlich ein Rückschritt.

Doch man hatte eine gute Idee: Die Riesenzeichnung der Verbindungen wurde abfotographiert. Bei einem entsprechend guten Filmmaterial hatte man nun mit einem Schlag aus einer mehrere Quadratmeter großen Zeichnung ein Abbild auf einigen Quadratzentimetern geschaffen (daß dies nur ein Negativ war, sollte uns nicht irreführen).

Man hatte also ein ähnliches Vorgehen wie bei der Diafotographie gewählt: Wir stellen uns mit der Kamera auf eine erhöhte Position und wollen ein Bild vom Häusermeer einer Stadt aufnehmen. Dieser Inhalt wird nun auf ein Dia gebannt. Wurde die Schärfe richtig eingestellt und wurden die Regeln zur Belichtungsmessung eingehalten, erscheint bei uns eines Abends genau dieses Häusermeer in all seinen Details wieder auf der Leinwand.

Wie viele Informationen im Vergleich zu An- Aus-Stellungen von Transistoren mögen auf solch einem Dia enthalten sein? Mindestens mehrere Millionen, denn schauen Sie sich einmal ein detailreiches Dia auf der Leinwand etwas genauer an: Jedes kleinste Etwas, das Sie vom Umfeld unterscheiden können, entspricht bereits einem Transistor. Wenn Sie dann noch bedenken, daß jede farbliche Darstellung auf dem Dia aus RGB (Rot, Grün, Blau) zusammengemischt ist, nimmt die Anzahl der wohlunterscheidbaren Deails exponentiell noch weiter zu.

Zurück zu unserer verkleinerten Zeichnung bei den Computerentwicklern in Form eines Negativs: Durch ein besonderes chemisches Verfahren wird die abfotographierte Strichzeichnung der Transistoren stromleitend gemacht ... und schon hat unser nun sogenannte Chip die Leistung von vielen tausend Transistoren. Nicht ganz: Erst muß unser Chip ja noch in Kontakt zu seiner Außenwelt treten: Dafür werden im Vergleich zum winzigen internen Leiternetz wie riesige Baumstämme erscheinende Leiterbahnen rund um den eigentlichen Chip aufgebracht, so daß nun jeder Mensch mit seinem Lötkolben die Verbindung nach draußen hin - vielleicht zu einem anderen Chip - selbst herstellen kann.

Im Vergleich zu Relais und Transistoren ist unser Chip zudem erheblich schneller geworden - bis zu 100000 Additionen können pro Sekunde durchgeführt werden -, denn das engverzweigte Leitungsnetz bietet erheblich kürzere Entfernungen als in der Größenausdehnung einer Schrankwand wie beim Zuse-Rechner.

Wie weit ist nun die Entwicklung dieser Chips bis heute vorangeschritten? In der Entwicklung sind gerade die sogenannten Megabitchips d.h. Mega = 1 Million Transistorfunktionen auf der Fläche eines Stecknadelkopfes. Dieser vorerst letzte Schrei der Entwicklung befindet sich zwar noch nicht in Ihrem MSX-Rechner, aber immerhin auch schon eine Vielzahl von 64 Kilobit-Chips d.h. 64000 Transistorfunktionen auf der Fläche eines Stecknadelkopfes.

2. Was für verschiedene Arten von Chips gibt es?

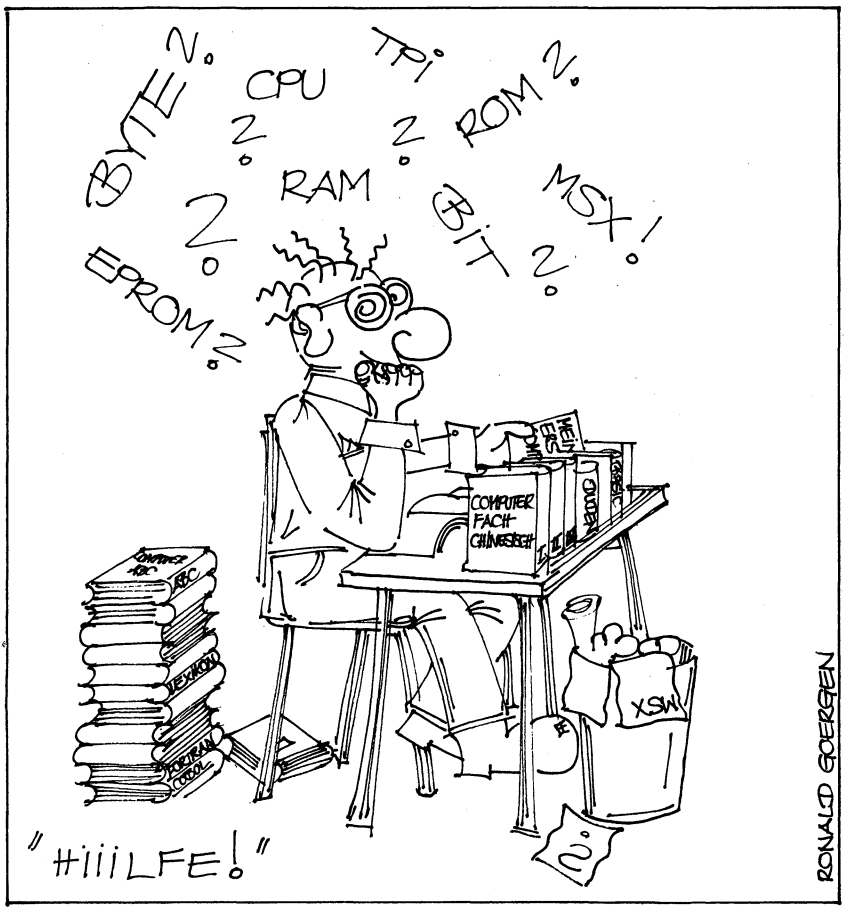
Es gibt verschiedene Möglichkeiten, wofür man die Chips einsetzen kann. Je nach Verwendungszweck - ob Informationen bereits im Chip fest abgespeichert sind oder aber nicht - können wir Chips bereits mit fertigen Programmen kaufen oder aber Chips verwenden, deren Speicher frei programmierbar ist.

Um die Fachbegriffe auch zu verstehen: Im Englischen heißt Speicher 'memory'. Klar, daß also in den Computerfachbegriffen (die fast samt und sonders aus dem Englischen stammen) für die Chips das Wörtchen 'memory' auftauchen muß. Es gibt folgende Arten von Chips:

ROM
PROM
EPROM
RAM

ROM ist eine Abkürzung für Read Only Memory, was soviel bedeutet wie: Nur Lese Speicher. Im ROM sind Programme bereits abgespeichert, die man sofort gebrauchen kann z.B. bei Ihrer Waschmaschine können Sie verschiedene Programme wählen d.h. Sie brauchen diese Programme beim Kauf der Waschmaschine nicht erst mühsam zu programmieren, diese Programme sind bereits im Kaufpreis enthalten (wie würde sonst auch die Wäsche einer Hausfrau aussehen, die sich vorher noch nicht mit der Programmierung auseinandergesetzt hat?!).

In Ihrem MSX-Computer haben Sie bereits eine sogenannte Programmiersprache im ROM d.h. mit dem Kauf Ihres Rechners haben Sie nicht nur den Computer = die sogenannte Hardware, sondern auch gleich ein Programm, die Programmiersprache BASIC = die sogenannte Software miterworben.


BYTE ?
CPU ?
TPI ?
ROM ?
RAM ?
MSX !
BIT ?
EPROM ?
COMPUTER FACH-CHINESISCH
XSM
"HILFE!"
RONALD GOERGEN

Stellen Sie sich Ihr ROM so vor, wie wir vorhin die Entwicklung des Chips beschrieben haben. Allerdings sind hier die Transistoren bzw. die einzelnen Leiterbahnen bereits in ihrem Zusammenspiel von Entwicklern bzw. Programmierern einmal vorbelegt worden. Sie haben bei Ihrem BASIC-ROM also nicht die Möglichkeit, diese Leiterbahnen neu zu beeinflussen, Sie können ihren Aufbau nur auslesen und gebrauchen, halt wie ein Lexikon, nicht aber neu beschreiben.

In der Vorphase der Entwicklung von Computern werden nicht sofort ROMs verwendet (dann wären die Entwicklungskosten noch weit aus höher durch das andauernde Vernichten der im Vorstadium nicht richtig programmierten ROMs) sondern vielmehr EPROMs (Erasable Programmable ROM).

Hiermit ist es möglich, daß der Anwender die Struktur der Leiterbahnen selbst festlegt = ein EPROM brennt und bei Nichtgefallen unter UV-Licht die Informationen wieder löschen (Englisch: to erase) kann. Wörtlich übersetzt ins Deutsche würde EPROM also eigentlich heißen: löschbarer und programmierbarer Nur Lese Speicher.

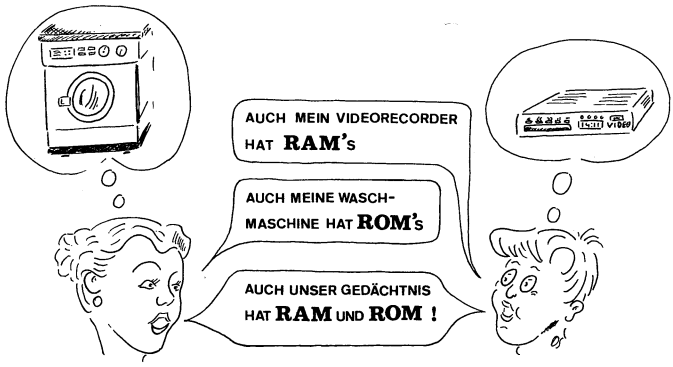
Sie können sich mit Ihrem MSX-Computer nach dem Zukauf einer entsprechenden Erweiterung auch EPROMs selst brennen. So könnten Sie ein selbstgeschriebenes Programm (z.B. das Telefonbuchprogramm weiter hinten in diesem Buch) auf das EPROM speichern und hätten dieses Programm auch noch nach Jahren ohne neues Laden von der Kassette oder der Diskette augenblicklich verfügbar.

Noch etwas zum Aussehen der EPROMs: EPROMs erkennen Sie gut daran, daß auf ihre Oberseite meist kleine Zettelchen aufgeklebt sind. Wenn Sie mal so einen Chip im Bastlerladen sehen, lösen Sie das Zettelchen ruhig einmal ab: darunter sehen Sie dann den eigentlichen Chip, wie eine kleine Maus im großen Käfig. Die Größe des Chips wird durch die vorhin schon angesprochenen Leiterbahnen nach draußen hin bestimmt.

Warum nun dieses Zettelchen oben auf dem EPROM? Sollen hierdurch gar Geheimnisse geschützt werden, die Sie nun entschlüsselt haben? Nein, wir haben doch bereits erfahren, daß das Programm auf dem EPROM nur durch UV-Licht wieder gelöscht werden kann. Wie könnte sonst UV-Licht auf den Chip treffen, wenn nicht durch das kleine Sichtfenster nach Abnehmen des Zettelchens.

Schließlich gibt es noch PROMs und RAMs. Das PROM ist ähnlich dem EPROM, nur das 'E' = die Möglichkeit, wieder und wieder zu löschen und neu zu programmieren, fehlt: Ist ein PROM einmal programmiert = gebraucht worden, muß man sich mit dem Inhalt darauf zufriedengeben oder ... man programmiert halt ein neues.

Die wichtigsten Chips im Computer sind die RAMs, die sich alle eingegebenen Datenmengen merken können und sie auf Befehl oder durch Stromausfall auch sofort wieder vollständig vergessen. Denken Sie an Ihren Videorecorder, bei dem Sie alle paar Tage neue An- und Ausschalt-Zeitpunkte einprogrammieren können. Auch dort sind freiprogrammierbare Speicher = die RAMs für diesen Zweck enthalten.



RAM heißt aber nun nicht 'Read A Memory', sondern 'Random Access Memory', frei übersetzt: Speicher mit direkter Zugriffsmöglichkeit. Zwar können wir auch beim ROM direkt auf die einzelnen Speicherstellen = die Transistoren zugreifen, aber wir können diese nicht wie beim RAM für unsere höchstpersönlichen Zwecke verändern.

Also ist das RAM unser eigentlich frei programmierbarer Speicher: hier können wir unsere Daten, Programmierzeilen oder auch Liebesbriefe ablegen. Jedoch Vorsicht: Wollen wir diese Daten für immer behalten, müssen wir den Computer und damit seine RAMs unter Strom lassen oder aber die dort enthaltene Information auf andere magnetische Speichermedien wie Kassette oder Diskette übertragen. Ähnlich ist es ja auch, wenn Sie Musik hören: Sie vergessen so manche Tonfolge, wenn Sie die Musik nur einmal hören; zeichnen Sie diese allerdings auf Tonband auf, ist die Erinnerung immer wieder abrufbar.

Auch der Mensch hat RAMs und ROMs. Das ROM enthält die Sprachelemente (eigentlich müßten wir von PROMs sprechen, denn was Sie im Augenblick tun, neue Vokabeln mit ihrer Bedeutung zu lernen, würde ja eigentlich heißen: Ihr ROM = Ihren Vokabelschatz zu erweitern = umzuprogrammieren).

Das RAM hingegen ist Ihr Kurzzeitgedächtnis, wo Sie z.B. Termine abspeichern und später diese auch wieder vergessen werden. Allerdings hat der Mensch durch seine Stromzuführng = das Herz eine Art Batteriepufferung für seine RAMs. Während der Computer alle Informationen der RAMs nach dem Ausschalten vollständig vergißt, bleiben beim Menschen manchmal doch noch Reste seiner Kurzzeitinformation über Jahre gespeichert, meist nicht direkt abfragbar, aber im Unterbewußtsein doch noch wie ein Schleier einer Information vorhanden.

3. Computer sind dumm - sie kennen nur aus und ein oder 0 und 1

Sie erinnern sich noch an die Funktionsweise des Zuse-Rechners? Viele Relais, die nur die Zustände an bzw. aus wiedergeben konnten, waren zu mehreren jeweils zusammengeschlossen und so zeigten kleine Lämpchen schließlich nach der Berechnung das Ergebnis an.

Wie arbeitet nun ein Computer - nicht nur unser MSX-Computer - allein mit diesen beiden Zuständen 'an' und 'aus'? Hier müssen wir ein klein wenig in die Schulmathematik einsteigen: Vor vielen hundert Jahren haben sich unsere Vorfahren zusammengesetzt und willkürlich ein Zahlsystem entwickelt, das zehn verschiedene Ziffern kennt:

0 1 2 3 4 5 6 7 8 9

Kommt man beim Hochzählen bei der Ziffer 9 an, kehrt man wieder zur 0 zurück und setzt eine 1 davor (der Zehner mit einer höheren Wertigkeit wird zum Einer ergänzt). Nun werden erneut die Ziffern 0 bis 9 durchgezählt und anschließend ersetzt man den ersten Zehner durch die nächsthöhere Zahl, die 2. So entsteht die Zahl 20 usw., bis man schließlich sämtliche Zustände der Zehner und Einer durchgerechnet hat = 99.

Nun muß man erneut mit der Ziffer 1 (wie beim Umstellen von Einer auf Zehner) beginnen, jedoch nehmen nun zu Beginn des ersten Hunderters Einer und Zehner den Wert 0 ein usw.. So kann man nach und nach die weiteren Stellen ergänzen, seien es nun Tausender, Hunderttausender oder Millionen.

Obgleich wir Menschen uns in einem Lernprozeß schnell daran gewöhnt haben, diese 10 Ziffern richtig und logisch in einem Zahlsystem, dem sogenannten Zehner- oder Dezimalsystem, zu gebrauchen, müssen wir uns doch immer bewußt in der Tatsache sein, daß dies ein logisch aufgebautes Zahlsystem - von Menschenhand geschaffen - ist.

Das Dezimalsystem wurde einstmals ebenso von Menschen erfunden wie die Dampfmaschine, nur durch die Möglichkeit der Menschen, Informationen mündlich oder schriftlich der nächsten Generation weiterzuvermitteln, war es möglich, daß wir dieses System noch heute gebrauchen und es sogar praktisch anzuwenden gelernt haben (12 Uhr, 18.7.1985, 4 Eier ...).

Soviel zur Entwicklungsgeschichte unseres Dezimalsystems. Als sich die Menschheit daran machte, Computer zu erfinden, muße man darüber nachdenken, wie dem Computer auch ein Zahlsystem, am besten unser Dezimalsystem, beigebracht werden könnte. Man hatte hier ja ein Etwas geschaffen, dem man nicht einfach wie bei den Menschen die Information mündlich oder schriftlich weitergeben konnte, hier mußte erst ein Fundament geschaffen werden.

Es gab Etwicklungen, die mit Schaltern arbeiteten, welche mehr als zwei Zustände unterscheiden konnten (es fließt viel Strom, es fließt wenig Strom, es fließt kein Strom). Jedoch, und das war doch die eigentliche Grundidee zur Entwicklung der Computer, manchmal erkannte der Rechner nicht den Unterschied z.B. zwischen wenig und kein Strom. Ergebnis: die Rechnung stimmte nicht mehr, das Resultat war falsch (unvorstellbar, wie ich bereits vorhin sagte, wenn unsere Taschenrechner heute so unsicher rechnen würden).

Also einigte man sich schließlich darauf, den Computer mit nur 2 Zuständen arbeiten zu lassen: an bzw. aus. Hierbei konnte man allerdings dann zu 100 Prozent sichergehen, daß dieses System auch einwandfrei funktionierte! Ein Name für dieses Rechensystem, das nur zwei Zustände kennt, nämlich 'aus' und 'an' oder 0 und 1, war bereits zuvor von Mathematikern bei der Entwicklung unseres Dezimalsystems geprägt worden: Binär- oder Dualsystem.

Wie ist es nun möglich, nur mit den Ziffern 0 und 1 auch größere Zahlwerte zu verarbeiten? Unmathematisch und aus meinem vorherigen Text schließend könnte man sagen: nehmen wir doch einfach so viele An-Zustände oder Einsen, wie groß unsere zu berechnende Zahl ist z.B. für die Dezimalzahl 10 brauchen wir dann 10 angestellte Relais in Folge oder für die Zahl 30000 ...


BIT ≙ binary digit
Binärzeichen
BYTE ≙ besteht aus
8 BIT
ist ein "Wort"
ROM ≙ read only memory
DATA
"SCHÜLER-LEHRER 2"
KLAUS
RONALD GOERGEN

Sicherlich eine Möglichkeit, nur würden wir hierbei ja die Information unseres zweiten Zustandes verschenken: die Null taucht in dieser Version unseres Binärsystems gar nicht mehr auf.

Gehen wir doch einfach mal das Binärsystem so durch, wie wir dies vorhin bereits beim Dezimalsystem Schritt für Schritt getan haben: wir haben zwei Ziffern zur Verfügung, die wir allerdings bei zunehmender Größe der Zahl miteinander verknüpfen können. Die kleinste Binärzahl ist die 0, dann folgt die 1. Da wir nicht wie beim Dezimalsystem über eine größere Ziffer als die 1 verfügen, müssen wir nun schon eine Ziffer voranstellen - also die nächstgrößere Zahl ist die 10 (bitte lesen Sie beim Arbeiten im Binärsystem immer Ziffer für Ziffer - also in diesem Fall nicht 'Zehn' sondern 'Eins-Null').

Danach können wir die kleinste Stelle wieder um 1 vergrößern und es entsteht so die Binärzahl 11. Wieder sind die dargestellten Ziffern bei ihrem Höchstwert angelangt (ähnlich wie bei unserem Dezimalsystem die Zahl 99) und wir müssen erneut eine Ziffer davorsetzen: 100. Daß die weiteren Werte 101, 110 und 111 sind, können Sie sich sicher nun selber errechnen.

Auf den ersten Blick mag Ihnen dieses Zahlsystem, das Binärsystem, sehr kompliziert erscheinen, aber bedenken Sie dabei, daß Sie hier genauso vorgehen, wie bereits beim vertrauten Dezimalsystem seit Jahren gewohnt: Ist eine Stelle im Dezimalsystem bei ihrem höchsten Wert (9) angekommen, wird ganz einfach eine weitere Ziffer davorgeschrieben bzw. die bereits davor stehende Ziffer um 1 erhöht (ist dies auch nicht mehr möglich, wird die Ziffer davor um 1 erhöht ... so daß schließlich nach der 999 im Dezimalsystem die Zahl 1000 folgen kann).

Ist Ihnen vorhin beim Entwickeln der Binärzahlenfolge etwas aufgefallen? Schauen wir uns die Zahlenfolge noch einmal an:

0
1
10
11
100
101
110
111

Wir hatten uns doch vorhin darüber Gedanken gemacht, wie man größere Zahlen aus dem Dezimalsystem in das Binärsystem übertragen kann. Erst hatten wir angenommen, für jede Zahl die entsprechende Anzahl von Einsen nehmen zu müssen, also z.B. für die Zahl 10 auch zehn Einsen. Schauen wir aber nun unsere Folge von der 0 bis zur 111 an, können wir feststellen, daß wir zwischendurch nicht nur die Zustände 1 und 11 errechnet haben, sondern etliche mehr (10, 100, 101 und 110). Nun können wir den Wert unserer Binärzahlenfolge dadurch deutlicher machen, indem wir die entsprechende Zahl des uns bereits mehr vertrauten Dezimalsystems in aufsteigender Reihenfolge danebenschreiben:

Dezimal	Binär
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111

Also brauchen wir für die 7 nicht wie vorhin angenommen sieben Einsen, sondern lediglich drei. Sie werden sich nun aber fragen, wie man denn dieses System auch in die Zehner und Hunderter weiterführen kann ... das würde ja Ewigkeiten dauern, wenn man immer weiter hochzählen müßte. Keine Angst. Wie ich vorhin

bereits klar zu machen versuchte, liegt in diesem logischen System selbstverständlich auch und insbesondere ein mathematisches System verborgen. Wir benötigen dazu die verschiedenen Potenzen der Zahl 2:

```
2^0 =   1
2^1 =   2
2^2 =   4
2^3 =   8
2^4 =  16
2^5 =  32
2^6 =  64
2^7 = 128
```

Wir hatten vorhin in unserer Tabelle errechnet, daß die Binärzahl 100 der Dezimalzahl 4 entspricht. Schreiben wir uns die 100 auf und beginnen von der kleinsten Stelle bis zur höchsten die entsprechenden Zweierpotenzen in aufsteigender Folge darunter zu schreiben:

```
1          0          0

2^2=4      2^1=2      2^0=1

4    +     0    +     0          = 4
```

An den Stellen, wo die Binärziffer 0 steht, brauchen wir die Zweierpotenz nicht zu berücksichtigen. Erscheint aber an einer Stelle die 1 (wie bei 100 die dritte Stelle), müssen wir bei der Umrechnung ins Dezimalsystem auch den entsprechenden Zweierpotenzwert berechnen ... unter der 1 der Binärzahl 100 steht die Dezimalzahl 4, also ist Binär 100 = Dezimal 4.

Wenn mir mehrere Einsen in einer Binärzahl vorfinden, müssen wir die entsprechend dazu gehörende Zweierpotenz addieren z.B.

Binär 111 = Dezimal 2^0 + 2^1 + 2^2 = 1 + 2 + 4 = 7

Vielleicht können Sie sich somit auch vorstellen, daß man mit lediglich acht Binärziffern bereits die Dezimalzahl 255 erzeugen kann (eine Art Magiezahl bei unserem MSX-Computer, wie wir später in diesem Buch noch des öfteren feststellen werden), bei sechzehn Binärstellen gar die Dezimalzahl 65535.

Genug dieser mathematischen Rechenspiele. Es wäre schön, wenn Sie nun einen kleinen Einblick in diese 0-1-Rechenweise Ihres Computers gewonnen haben, denn in der weiteren Arbeit mit Ihrem MSX-Computer werden Sie immer wieder auf dieses Zahlsystem gestoßen werden. Dies ist nicht nur bei Ihrem MSX-Rechner so, auch bei allen anderen Maschinen oder Geräten auf dem Markt, die irgendwo einen Computer eingebaut haben. Nur mit diesem Zahlsystem ist es möglich, einem nicht zum Denken fähigen Apparat zumindest den Vorgang des Rechnens beibringen zu können. Da aber alle Computerlogik auf Rechenvorgängen beruht, ist durch die matheatisch-logische Verknüpfung im Binärsystem das eigentliche Fundament für einen Computer geschaffen worden.

Bevor wir wieder in die Praxis eintauchen, noch ein wichtiger Hinweis: Neben dem Binärsystem und dem Dezimalsystem gibt es unendlich viele andere Zahlsysteme: Zahlsysteme, in denen es acht verschiedene Zustände

0 1 2 3 4 5 6 7 = das Oktalsystem

oder gar sechzehn verschiedene Zustände (0 bis 9 und dann fängt man einfach mit dem ABC an), also

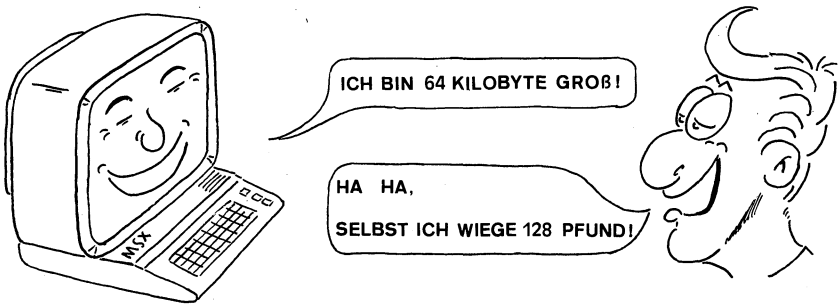
0 1 2 3 4 5 6 7 8 9 A B C D E F = das Hexadezimalsystem

gibt. Die Logik dieser Systeme funktioniert genauso, wie wir sie gerade auch beim Binär- und beim Dezimalsystem kennen- und mathematisch schätzengelernt haben. Später im Buch bei den programmiertechnischen Unterweisungen über die Zahlsysteme, werden Sie schließlich erfahren können, daß all diese Zahlsystemberechnung und -umrechnung von Ihrem MSX-Computer sehr leicht direkt vor Ihren Augen am Bildschirm ausgeführt werden kann.

Wenn Sie dieses Kapitel bis jetzt Seite für Seite nicht nur gelesen, sondern auch verstanden haben, müßten Sie schon eine Menge mehr wissen: Was ist ROM, RAM, PROM, EPROM, Binärsysem, Hexadezimalsystem und wie sind diese ganzen Computer überhaupt zu verstehen (der Aufbau eines Chips aus etlichen Transistoren). Anschließend wollen wir uns noch über Speichergröße (was heißt 64 Kilobyte?), die Funktion des Mikroprozessors und die vielen Geräte um den Computer herum (und deren Funktion) Gedanken machen, bevor es dann endlich damit losgeht, praktisch an unserem MSX-Computer zu arbeiten.

4. Was heißt 64 Kilobyte?

Sie erinnern sich noch daran, was RAM ist? Rchtig! RAM ist ein Speicher in Ihrem MSX-Computer, in dem Sie später Ihre Programme, Texte usw. ablegen können. Wenn Sie Ihren MSX-Cmputer ausstellen oder wenn gar der Strom ausfällt, ist sämtliche Information aus dem RAM-Speicher verschwunden, ähnlich wie beim RAM im Videorecorder: Fällt der Strom aus, blinkt die Anzeige im Videorecorder wie wild, um Sie darauf aufmerksam zu machen, daß alle Programminformationen und meist sogar die vorher eingestellte Uhrzeit vergessen wurden (wie bei einigen Videorecordern gibt es allerdings neuerdings auch bei einigen Computern eine sogenannte Batteriepufferung für den RAM-Speicher - dann verfügt der Computer auch nach dem Ausschalten über die eingespeicherte Information im RAM-Bereich).



Bei den meisten MSX-Computern finden Sie die Bezeichnung 64 K oder 80 KB im Handbuch oder gar auf der Kartonage. Auch andere Nicht-MSX-Computer werden qualitativ mit solchen Zahlen eingeordnet, ähnlich wie bei Autos mit der PS-Zahl. Was bedeutet nun die Abkürzung K oder KB? Ausgeschrieben heißt dies Kilobyte. Ein Kilo sind bekanntlich 1000, also bedeutet die Bezeichnung Kilobyte = 1000 Byte. Hat Ihr MSX-Computer laut Beschreibung 64 KB, so wissen Sie nun, daß in ihn 64000 Byte hineinpassen.

Diese Bezeichnung bezieht sich auf die Größe des RAM-Speichers von Ihrem MSX-Computer. Sie können 1 Byte für einen Buchstaben setzen und wissen somit, wie viele Buchstaben in den RAM-Speicher Ihres MSX-Computers hineinpassen. Seien es nun 64000 oder gar 80000, lange wird es sicherlich bei Ihnen noch dauern, bis Sie diese Speicherplatzmenge für selbstgeschriebene Programme ausnutzen können. Viel eher können Sie aber so viel Speicherplatz für Texte verwenden, nur müssen wir uns dann zuvor darüber im klaren sein, was denn 64000 Buchstaben genau bedeuten.

Genauso, wie Sie nichts damit anfangen könnnen, wenn Ihnen jemand sagt, er hätte eine Fläche von 10 Hektar gekauft - und Sie wissen gar nicht, wie groß ein Hektar ist -, dann hilft Ihnen erst eine praxisnahe Übersetzung wie: Mein Feld ist so groß wie so und so viele Fußballfelder. Genauso müssem wir versuchen, 64000 Buchstaben in eine faßbare Größe zu bekommen: auf einer Seite dieses Buches sind ungefähr 2000 Buchstaben abgedruckt. Also können Sie sich nun selbst ausrechnen, daß bei 64 Kilobyte oder einem Speichervermögen von 64000 Buchstaben ca. 32 Buchseiten mit Text in Ihren MSX-Rechner hineinpassen würden. Ein anderer Computer, der gar 80 Kilobyte RAM-Speicher sein eigen nennt, könnte demnach 40 Seiten in sich abspeichern.

Nur zu Ihrer Information: Wir sprachen vorhin von ROM- und RAM-Speicher. In allen MSX-Computern ist nicht nur RAM sondern auch ROM enthalten. Wie wir vorhin bereits anmerkten, ist im ROM u.a. die Programmiersprache BASIC abgelegt. Um nun auch noch einen Begriff von der Größe dieser Programmiersprache zu bekommen: In jedem MSX-Computer ist der ROM-Speicher 32 Kilobyte groß - das entspricht einer Textinformationsmenge von ca. 16 Buchseiten!

5. Die Funktion des Mikroprozessors

Wir haben nun zwar unsere Speichergröße zu differenzieren gelernt, wissen auch, daß es RAM und ROM gibt, aber das reicht noch nicht so ganz. Irgendein Herzstück muß in dem Computer noch eingebaut sein, das nicht nur in der Lage ist, die vielen Speicherstellen zu verwalten, sondern auch z.B. die Umrechnung von der Dezimaleingabe in das dem Rechner eigene Zahlsystem, das Binärsystem, vorzunehmen.

Dieses elektronische Bauteil heißt Mikroprozessor, wer etwas auf sich hält spricht von Central Processing Unit (oder abgekürzt CPU). Schließlich streiten sich die Gemüter noch, ob man auch Zentraleinheit dazu sagen kann: Früher war die Zentraleinheit Mikroprozessor, RAM und ROM, heute meinen einige, den Fachbegriff CPU mit Zentraleinheit übersetzen zu können (nur damit Sie Bescheid wissen, wenn auch diese Bezeichnung ab und an in den Fachzeitschriften auftaucht).

Bei dem Mikroprozessor handelt es sich wieder um einen Chip, der allerdings nicht nur ROM (seine ihm eigene Programmiersprache) sondern auch RAM (eine Anzahl sogenannter Register, in denen all das verarbeitet wird, was im Computer so stattfindet) enthält. Ich will nicht weiter in dieses diffizile Etwas einsteigen, sondern Ihnen nur noch folgendes mitteilen: In Ihrem MSX-Computer ist der Mikroprozessor Z80A von der amerikanischen Softwarefirma Zilog eingebaut. Ein sehr leistungsfähiger und weit verbreiteter aber auch schwierig zu programmierender (da über 600 Befehle möglich) Mikroprozessor.

Sie erinnern sich noch an unser erstes Kapitel über die Rechenweise der Computer mit Nullen und Einsen? Und vielleicht auch noch daran, daß man z.B. mit acht Einsen die Zahl 255 und mit sechzehn Einsen gar die Zahl 65535 darstellen kann? Eine 1 oder eine 0 nennt man ein Bit. Beim Z80A handelt es sich um einen sogenannten 8-Bit-Mikroprozessor, weil er gleichzeitig Binärzahlen in der Größenordnung bis dezimal 255 verarbeiten kann. Es gibt auch andere 8-Bit-Mikroprozessoren wie den 6502, der im ATARI 800 und im C-64 eingebaut ist. Er ist erheblich einfacher mit seinen nur ca. 50 Befehlen zu programmieren ist.

Während beide Prozessoren, der Z80A und der 6502 Daten in der Länge von nur 8 Bit (Dezimal 0 bis 255) verarbeiten können, ist ihr Adreßbus mit 16 Leitungen (= 16 Bit = Dezimalzahlen von 0 bis 65535) ausgestattet. So ist es auch erst möglich, daß ein Speicher von 64000 Byte (oder 64 Kilobyte) genutzt werden kann. Mit lediglich 8 Adreßleitungen (8 Einsen im Binärsystem) könnte man halt nur 255 Byte ansprechen und dafür (eine Achtel Schreibmaschinenseite) würde sich heutzutage wohl kaum jemand einen Computer kaufen.

Noch zwei Dinge:

1. Die nächste Computergeneraion wird in einigen Jahren großenteils mit 16- oder gar 32-Bit-Prozessoren arbeiten. Sie können sich vielleicht nach dem Lesen des vorigen Kapitels besser vorstellen, warum ein 16-Bit-Mikroprozessor (er kann gleichzeitig Zahlen in der Größe von 0 bis 65535 verarbeiten) erheblich leistungsfähiger arbeitet als ein 8-Bit-Mikroprozessor wie unser Z80A, denn er kann - wie gesagt - gleichzeitig 'nur' Zahlen in der Größe von 0 bis 255 verarbeiten. Aber bis zur Überschwemmung des Marktes mit 16-Bit-Mikroprozessoren z.B. mit dem derzeit gängigsten dieser Gattung, dem 68000, zu Preisen von heutigen 8-Bit-Mikroprozessoren wie dem Z80A wird noch einige Zeit ins Land gehen.

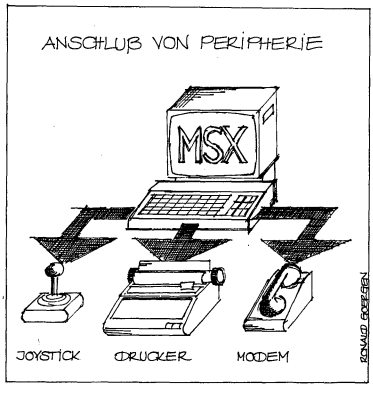
2. Einen weiten Bestandteil des Mikroprozessors sollten Sie sich noch merken: die ALU (Arithmetic Logic Unit) ist die eigentliche Rechenmaschine im Mikroprozessor. Übrigens: Wir sprechen im Mikroprozessor auch vom Rechner und seiner Programiersprache. Dies alles ist nicht zu verwechseln mit der Programmiersprache, die wir nachher in diesem Buch für Einsteiger kennenlernen wollen. Unsere Einstiegsprogrammiersprache, die wir gemeinsam lernen wollen, heißt BASIC und besteht aus Befehlen, die man selbst bei geringen Englischkenntnissen verstehen kann wie z.B. PRINT für Schreiben und LIST für das Auflisten des Programms.

Da BASIC so einfach (!?) ist, nennt man sie eine höhere Programmiersprache. Will man allerdings den Mikroprozessor direkt programmieren, braucht man dazu erheblich mehr Kenntnisse, denn erst eine Vielzahl von schwer verständlichen Speicherorganisationsbefehlen in der Mikroprozessorsprache - auch Maschinensprache genannt -, ergibt die Funktionsweise eines einzigen BASIC-Kommandos. Der Sinn, direkt in Maschinensprache zu programmieren, ist allerdings der, daß man somit bis zu mehrere hundertmal schnellere Programme schreiben kann als in BASIC.

6. Die vielen Geräte um den Comnputer herum

Eigentlich gehören zum Computer nur Speicher (RAM und ROM) sowie der Mikroprozessor.

Sie werden sich fragen, was denn die Tastatur ist. Hierbei handelt es sich bereits um ein Teil der Umgebung des Computers, über die wir in diesem letzten Kapitel der Einführung noch sprechen wollen. Man kann die Umgebung Ihres MSX-Computers um verschiedene Teile erweitern: Nicht nur durch die Tastatur, auch mit Kassettenrecorder, Diskettenstation, Drucker, Monitor, Fernseher usw.. Man bezeichnet diese Zusatzartikel zum eigentlichen Computer auch als Peripherie (ähnlich wie wir von der Peripherie der Großstadt = den Sattelittenstädten ringsum sprechen).



Da unser MSX-Computer bereits eine Tastatur eingebaut hat, können wir ihn eigentlich nicht nur als Computer, sondern gleich schon als eine MSX-Kompaktanlage ansehen, bestehend aus dem eigentlichen Computer und der Tastatur.

Weiterhin gibt es verschiedene Speichermedien wie Kassettenrecorder und Diskettenstation, die Sie direkt an Ihren MSX-Computer anschließen können. Wenn Sie dieses Einführungskapitel bis hierher gelesen haben, wissen Sie, warum zumindest ein Kassettenrecorder als zusätzlicher Speicher neben RAM und ROM dringendst erforderlich ist: Geben wir nämlich Programme oder Texte in unseren MSX-Computer ein, werden diese Informationen nach dem Ausschalten wieder vergessen. Auf einer Kassette (bei den MSX-Computern reicht es, einen normalen Musikrecorder anzuschließen) werden die RAM-Informationen dann in Form von kurz aufeinanderfolgenden Piepstönen (jeweils hell oder dunkel, gemäß den abzuspeichernden Einsen oder Nullen der Bits) auf dem Bandmaterial abgespeichert. Man gibt die Anzahl der auf dem Band abgespeicherten Bits (0 oder 1) pro Sekunde mit dem Wort BAUD an (Ihr MSX-Computer kann auf Kassette pro Sekunde entweder 1200 Bits oder gar 2400 Bits - also 1200 oder 2400 BAUD -abspeichern).

Eine schnellere Abspeicherung ist mit einer sogenannten Diskettenstation auf magnetische Scheiben = Disketten möglich. Hier hat man den Vorteil, daß mit dem Schreib- Lesekopf der Diskettenstation direkt auf das Programm zugesteuert werden kann, das man gerade braucht - ein Vorgang, der sich in Bruchteilen von einer Sekunde abspielt.

Die Unterscheidung vom Recorder zur Diskettenstation ist ähnlich, wie der Unterschied zwischen dem Abrufen eines Musikstükkes auf Kassette oder Schallplatte: Bei der Kassette müssen Sie eventuell vor- und zurückspulen, ein lästiger und zeitraubender Vorgang; bei der Schallplatte können Sie augenblicklich die Nadel genau dort aufsetzen, wo Ihr Lieblingstitel beginnt. Sucht man mit dem Computer veschiedene Datensätze z.B. einen Namen mit Adresse, ist dies somit erheblich einfacher und auch sicherer realisierbar, wenn Sie hierfür eine Diskettenstation verwenden als einen Kassettenrecorder.

Für Sie als Einsteiger sollte allerdings der Kassettenrecorder vorerst vollkommen ausreichend sein, denn selbst wenn Sie einen größeren Teil von Programmen auf Kassette abgespeichert haben, wird es bei Ihnen sicherlich noch nicht so sehr darauf ankommen, Programme in einer Sekunde anstatt in einer Minute zu laden.

Vorerst sollten Sie nur wissen, daß Sie später Ihren MSX-Computer auch professionell mit einer superschnellen Diskettenstation (oder auch Floppy genannt) erweitern können.

Noch ein Tip für Ihren Kassetteneinsatz: Nehmen Sie lieber kurze als lange Kassettenbänder. Es gibt in Computershops meist spezielle Kassetten mit einer Länge von 5 bis 15 Minuten, genau richtig dafür, um nur wenige Programme auf einer Kassette abspeichern zu können ... allerdings nur so werden Sie die Programme auch verhältnismäßig schnell wiederauffinden können. Noch ein weiterer Tip: Die Bandqualität sollte Ferro sein und möglichst kein Chromdioxid oder Metall.

Wollen Sie einen Ausdruck Ihrer Programme oder Daten oder aber später auch von Texten vornehmen, können Sie an eine Buchse Ihres MSX-Computers auch Drucker anschließen. Diese Verbindungsstelle an Ihrem Computer, auch Schnittstelle oder Interface genannt, ist bei Ihrem MSX-Computer eine gewöhnliche, eine sogenannte Standardschnittstelle mit dem Namen Centronics (die Entwicklungsfirma dieses Anschlusses heißt so).

Sie können an die Centronics-Schnittstelle auch verschiedene Schreibmaschinen anschließen, durchgesetzt hat sich allerdings als Computerdrucker der sogenannte Nadel- oder Matrixrucker. Mit ihm wird im Gegensatz zur Schreibmaschine der Buchstabe nicht mit einer festen Type angeschlagen, sondern mit Hilfe von sieben bis neun einzelnen Nadeln Abschnitt für Abschnitt erzeugt. Dies wirkt zwar schwierig, ist aber sehr viel schneller als bei einer angeschlossenen Schreibmaschine. Zudem sind die Nadeldrucker im Gegensatz zur Schreibmaschine dazu fähig, mit Hilfe der einzelnen Nadeln hochauflösende Grafiken zu Papier zu bringen. Nachteil der Nadel- oder Matrixdrucker: da jeder Buchstabe, jede Zahl und jedes Zeichen aus einzelnen Nadelandrucken erzeugt wird, ist das Schriftbild bei weitem nicht so deutlich und klar wie bei einer Schreibmaschine. Gute Beispiele dafür können Sie in meinen DATA BECKER-Büchern finden: die Programme wurden zum großen Teil mit einem Nadeldrucker ausgegeben und die Texte von einer vom Computer ferngesteuerten Schreibmaschine ohne Tastatur, einem sogenannten Typenraddrucker zu Papier gebracht.

Während man Drucker, Kassettenrecorder und Diskettenstation nicht unbedingt zum Einstieg in die Computerei benötigt, ist dies sowohl bei der Tastatur (wie sollten wir auch sonst programmieren?) als auch bei einem Sichtgerät immer der Fall (denn irgendwo müssen wir ja einerseits kontrollieren, was wir eingegeben haben und andererseits die Ergebnisse lesen, die unser MSX-Computer errechnet hat).

Wir unterscheiden bei den Sichtgeräten Fernseher und die speziellen Computermonitoren. Für den Einstieg reicht im Normalfall ein Fernsehgerät, jedoch sollten Sie, falls möglich, nicht den großen 66 cm-Schirm gebrauchen, viel besser und ruhiger im Bild ist da doch ein Portable mit 37 cm.

Ideal ist es zum Programmieren, wenn Sie an Ihren MSX-Computer einen Monitor anschließen, denn hier ist das Bild durch die gesteigerte Wiederholfrequenz (Fernseher 25mal pro Sekunde, Monitor 50 bis 70mal pro Sekunde) flimmerfrei und dadurch bedingt auch sehr viel ruhiger. Sie können bei den einfarbigen Monitoren zwischen grün- und orangefarbigen Bildschirmen wählen, wobei sich die Freaks darüber zanken, welche Monitorfarbe nun gesünder für die Augen sei. Ich selbst arbeite selbst an einem orangefarbigen Monitor, der mir persönlich wegen des höheren Kontrastes augenfreundlicher erscheint.

Soviel zum Einstieg in die Computerei an und für sich. Ich hoffe, Sie haben einen Überblick bekommen und wissen nun wenigstens über das Grundvokabular der Computer Bescheid. Im Anhang sind noch einmal alle nicht nur in diesem Kapitel hier verwendeten Fachbegriffe mit 'Übersetzung' zusammengestellt.

Wir schalten unseren MSX-Computer das erste Mal an

Nun tauchen wir endlich in das unendlich große Meer der Computeranwendung ein, denn um genau dies zu verstehen und zu gebrauchen, deshalb haben Sie sich sicherlich auch Ihren MSX-Computer zugelegt.

In diesem Kapitel wollen wir nicht nur beachten, was geschieht, wenn Sie den Rechner anstellen, sondern auch die schreibmaschinenähnliche Tastatur mit all ihren Sonderfunktionen zu bedienen lernen.

Es ist nicht möglich, an dieser Stelle eine genaue Gebrauchsanleitung dafür zu geben, an welcher Geräteseite Sie das Stromkabel einstöpseln, die Verbindung zum Fernsehgerät oder gar zum Monitor vornehmen sollen, denn es gibt nicht nur ein MSX-Computermodell, sondern derer gleich über zwanzig.

Ich gehe an dieser Stelle davon aus, daß Sie

1. Ihren MSX-Computer mit einem Sichtgerät (Fernseher oder Monitor) bereits verbunden haben

2. an Ihrem Fernseher den entsprechenden Kanal (meist Kanal 36) bereits eingestellt haben

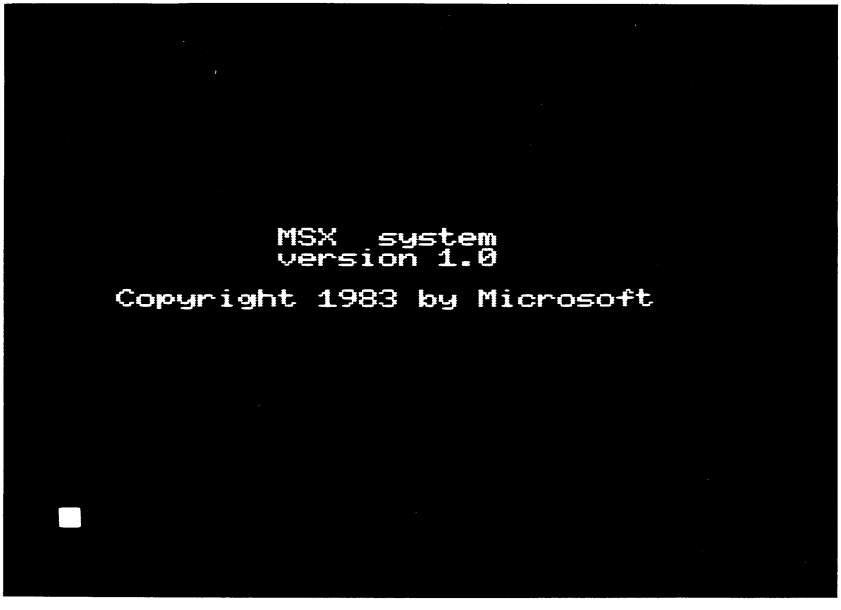
3. darüber Bescheid wissen, wo Sie Ihr Fernsehgerät und Ihren MSX-Computer anzustellen haben

4. sowohl Fernseher als natürlich auch Ihren MSX-Computer bereits ans Stromnetz angeschlossen haben.

Sind all' diese Punkte erfüllt, kann es nun mit der Arbeit am Computer beginnen: Schalten Sie dazu bitte erst den Fernseher an. Kurze Zeit später erscheint ein verrauschtes Bild, denn noch wird kein Bildsignal übermittelt (wie dies der Fall ist, wenn Sie ein Fernsehprogramm empfangen). Ihr MSX-Computer hat einen sogenannten Modulator eingebaut, der es ermöglicht, das im Computer erstellte Bild in die Form eines Fernsehsignals umzuwandeln und per Kabel weiterzuvermitteln.

Wenn Sie nun den An- Ausschalter Ihres MSX-Computers betätigen, sehen Sie - vorausgesetzt Sie haben den richtigen Fernsehkanal (meist Kanal 36) bereits eingestellt -, daß das Signal vom Modulator in den Fernseheingang übermittelt wird, denn einige Sekunden später stabilisiert sich das Fernsehbild, ähnlich wie Sie dies auch beim Einstellen eines Fernsehprogramms gewöhnt sind.

Der Bildschirm färbt sich blau und gibt für ein paar Sekunden eine schriftliche Information preis, zu kurz, um sie zu lesen. Schalten Sie deshalb ruhig noch einmal Ihren Computer wieder aus und dann wieder an. Nun konnten Sie das Schriftbild sicherlich lesen: MSX system version 1.0 Copyright 1983 by Microsoft.



Hinter der Buchstabenabkürzung MSX verbergen sich drei Worte: Microsoft Super Extended. Microsoft ist eine der größten amerikanischen Softwarefirmen (um die in diesem Kapitel verwendeten Fachbegriffe zu verstehen, lesen Sie bitte vorher das technische Einführungskapitel durch oder schauen in der Kurzerläuterung im Anhang nach), die die Programmiersprache BASIC für Ihren MSX-Computer erstellt hat.

Da sich das MSX-BASIC durch seine Vielzahl von Befehlen sehr stark von anderen BASIC-Versionen unterschedet, hat man ihm den Titel 'Super Extended' oder übersetzt 'sehr erweitert' zugedacht.

Nicht nur auf das BASIC ist die Bezeichnung 'Super Extended' bezogen, auch das gesamte Betriebssystem Ihres MSX-Rechners (das auch von Microsoft aus Amerika stammt) kann man als 'Super Extended' einordnen, geht man von der intelligenten Computerlogik aus, die den Anschluß fast jedbeliebigen Geräts an Ihren MSX-Computer zuäßt (siehe hierzu bitte im technischen Einführungskapitel).

Im Titelbild nach dem Einschalten steht neben der Copyright-Meldung (die Rechte für das MSX-Betriebssystem und MSX-BASIC liegen voll und ganz bei Microsoft; erst wenn Vereinbarungen mit Microsoft getroffen werden, kann man MSX weiterverwenden, z.B. zum Einbau in einen neuen Computer - diese Vereinbarung mußten sämtliche MSX-Computervertreiber mit Microsoft vorher abstimmen) auch der Hinweis: version 1.0.

Es wird in den kommenden Jahren - so sind die Pläne von Microsoft - nicht nur MSX Version 1.0 sondern auch erweiterte Versionen von MSX wie Version 2.0 oder gar Version 3.0 geben. Vorteil der MSX-Idee soll es aber sein, daß MSX Version 1.0 voll aufwärtskompatibel zu den nächsten Versionen ist.

Was bedeutet dies? Kompatibilität bedeutet soviel wie Austauschbarkeit d.h. haben Sie ein Programm auf Ihrem MSX-Rechner unter Version 1.0 geschrieben, so läuft dies auf allen anderen kompatiblen Rechnern d.h. auf allen anderen MSX-Rechnern ebenso. Nicht kompatibel sind allerdings die MSX-Rechner zu Computern

von Firmen, die sich bisher noch nicht an den MSX-Standard angeschlossen haben wie z.B. Commodore und Atari. Obgleich auch deren Programme vielleicht auf gleiche Datenträger (Kassette oder Diskette) abgespeichert wurden (die Speichermedien sind also zueinander kompatibel), ist die Aufzeichnung des Programms auf einer MSX-Kassette anders als auf einer Commodore-Kassette.

Aufwärtskompatibel bedeutet nun bei MSX-Version 1.0, daß alle Programme, die Sie auf Ihrem MSX-Computer schreiben, auch auf den nächsten MSX-Versionen 2.0 und 3.0 ohne irgendwelche Probleme laufen werden - ein honoriger Punkt in Beziehung auf Kundenfreundlichkeit (hoffen wir, daß es wirklich so sein wird).

Warum aber nur aufwärtskompatibel? Sie können die Programme Ihres Computers zwar später auf den höheren Versionen auch laufen lassen (nach oben = aufwärts kompatibel) aber nicht umgekehrt deren Programme samt und sonders auf Ihrem MSX-Computer anwenden.

Dies wäre auch gar nicht so leicht möglich, denn wenn man schon eines Tages die Version 2.0 von MSX mit verbesserten Features (und zu einem natürlich höheren Preis als Version 1.0) auf den Markt bringen will, würde es ja gar keinen Fortschritt bedeuten, wenn die dafür geschriebenen Programme genauso auch auf MSX-Computern der Version 1.0 laufen würden - das würde ja im Endprodukt bedeuten: 'technischer Fortschritt ist aus Gründen gewünschter Abwärtskompatibilität nicht mehr möglich'.

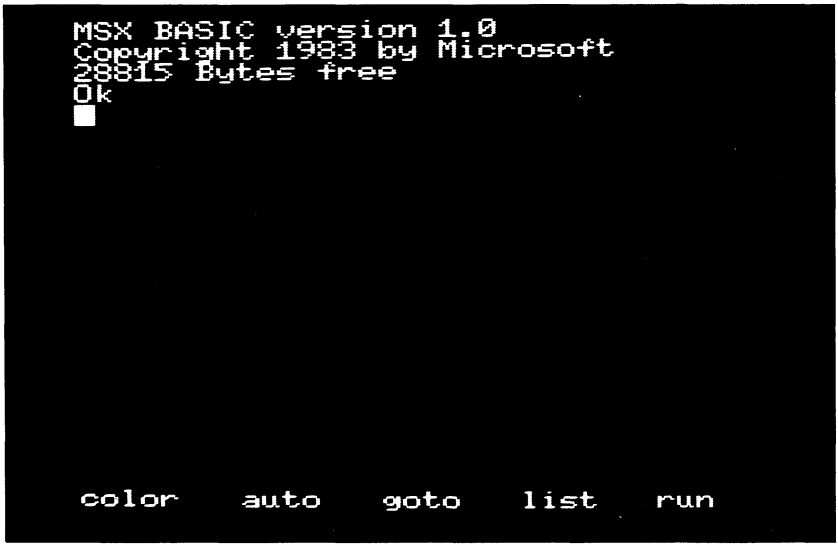
Bei einem zukünftigen MSX-Computer mit MSX-Version 2.0 könnte man somit erwarten, daß dieser zur Version 1.0 nicht abwärtskompatibel wäre und schließlich im weiteren Schritt, wenn eines Tages MSX-Version 3.0 herauskommen sollte, könnte man auch hier nicht mit einer Abwärtskompatibilität rechnen. In diesem Fall würde sich die Kompatibilität - wenn es so sein wird, wie die MSX-Macher heute planen - wieder genauso verhalten, wie im Verhältnis von MSX-Version 1.0 zu einer zukünftigen Version 2.0. d.h. Version 2.0 wäre aufwärtskompatibel zu Version 3.0.

Es wäre nur wünschenswert, wenn beim Erscheinen von MSX-Version 2.0 ein Erweiterungsmodul escheinen würde, das aus MSX-Version 1.0 ebendie Version 2.0 herstellen könnte. Aber hier verlieren wir uns in Spekulationen, die zur Zeit, wo nur MSX-Version 1.0 auf dem Markt ist, sinnloses Kopfzerbrechen bereiten.

Noch eine Bemerkung zu der Versionsbezeichnung: sollte eine neue MSX-Version auf den Markt kommen, die sich nur in Feinheiten (Ausmerzen von Fehlern oder Bugs im Betriebssystem und/oder BASIC) von MSX-Version 1.0 unterscheidet, würde hier nicht gleich die Bezeichnung 2.0 gewählt werden, sondern vielleicht 1.1 oder 1.2.

Soviel zum ersten Bild, das Ihr MSX-Computer nach dem Anstellen auf dem Fernseher erzeugt. Es ist bemerkenswert, welche dominante Stellung die Firma Microsoft im Pulk der MSX-Hardware- und Softwareanbieter einnimmt; wie gerne hätte eine Firma Sony oder Panasonic ihr Firmenlogo beim Einschalten auf dem Bildschirm erscheinen lassen! Aber das ist nun einmal der Hintergrund von Kompatibilität: Sie verlangen ja auch nicht an der Tankstelle VW-Benzin oder Opel-Treibstoff.

In der zweiten Bilddarstellung auf Ihrem MSX-Computer erscheint nun: MSX Basic version 1.0 Copyright 1983 by Microsoft 28815 Bytes free



und wieder ist eine Erklärung notwendig: die erste Zeile dürfte uns aus dem vorherigen Bild noch bekannt sein wenn auch hier nicht mehr von 'MSX system' sondern von 'MSX Basic' die Rede ist.

Wie ich bereits vorhin anführte, hat die Softwarefirma Microsoft nicht nur die eigentliche Grundprogrammiersprache für die MSX-Computer entwickelt, sondern eigentlich das ganze System im Einklang von Hard- und Software d.h. die erstaunlichen Ausbaumöglichkeiten Ihres MSX-Computers hin bis zur Steuerung des Bildplattenspielers wurden den MSX-Computern von der Firma Microsoft in die Wiege gelegt - also ist Microsoft die Entwicklungsfirma des MSX-Systems.

Außerdem hat Microsoft die Programmiersprache MSX-BASIC, die zum Bedienen des MSX-Computers und zum Ausschöpfen all der in ihm schlummernden Fähigeiten Möglichkeiten anbietet, speziell für die MSX-Computer entwickelt - aus diesem Grund taucht zum zweiten Mal der Name Microsoft auf.

Über die Bedeutung der Bezeichnung Copyright sprach ich bereits - also besitzt Microsoft nicht nur die Rechte am MSX-System, sondern auch am MSX-BASIC (wir dürfen aber nicht behaupten, daß Microsoft Erfinder der Programmiersprache BASIC überhaupt ist, obgleich dies den Anschein erweckt, wenn Sie mal auf die Copyrightnotitzen der meisten BASIC-Versionen auch bei Nicht-MSX-Computern achtgeben).

Was bedeutet nun 28815 Bytes free? Erstmal muß eine Entschuldigung angebracht werden, denn Ihr Computer hat vielleicht eine deutschsprachige Tastatur aber er spricht englisch. MSX ist nicht ein deutscher sondern ein internationaler Standard. Da unser Ländle nun einmal nicht das größte ist und die deutsche Sprache auch nicht die führende Weltsprache ist, hat man sich auf englisch geeinigt (daß dies der weiten Verbreitung des MSX-Systems nur förderlich sein kann, merken Sie spätestens im Auslandsurlaub, wenn Sie sich in deutscher Sprache mit Ausländern unterhalten wollen: deutsch wird nur selten verstanden, englisch aber fast immer).

Als kleine Hilfestellung will ich versuchen, zu allen verwendeten englischen Begriffen in diesem Buch das entsprechende deutsche Pendant anzuführen: 28815 Bytes free = 28815 Bytes frei. Und wieder haben Sie höchstwahrscheinlich nicht viel verstanden.

Wir sprachen im technischen Einführungskapitel von Speicherplätzen, ROM und RAM, erinnern Sie sich? Dabei lernten wir, daß die meisten MSX-Computer 80 KB RAM besitzen, was soviel bedeutet wie: 80000 Speicherplätze sind für die gleiche Anzahl von Buchstaben oder Zeichen frei. Damit wir diese Zahl 80000 besser begreifen lernten, haben wir gesagt: auf eine DIN-A-4-Seite passen ca. 2000 Buchstaben, also passen in unseren MSX-Computer ca. 40 DIN-A-4-Seiten.

Nun ist die Verwunderung groß, denn Ihr MSX-Computer teilt Ihnen mit: 'nur' 28815 Bytes (= Speicherplätze für Buchstaben oder Zeichen) frei. Also passen nicht 40 DIN-A-4-Seiten hinein, sondern 'nur' ungefähr 14 Seiten, weniger als die Hälfte.

Ich will Ihnen kurz erklären, wo der anscheinend fehlende Speicherplatz versteckt liegt: Erst einmal hat Ihr MSX-Computer brilliante Grafikmöglichkeiten (wir werden zwar auch in diesem Buch kurz darauf eingehen, aber viel genauer werden Sie darüber in meinem DATA BECKER-Buch MSX Grafik & Sound auf über 450 Seiten informiert). Die auf dem Bildschirm angezeigten Grafiken oder Zeichnungen müssen, da sie ja vom Computer erzeugt werden, irgendwo im Speicher abgelegt werden. Im Kapitel über die Grafik werde ich darauf noch näher eingehen. Hier nur soviel: Der Grafikspeicher in Ihrem MSX-Computer ist 16000 Speicherplätze groß.

Also schon einmal ein Trost: Ihr MSX-Computer hat nicht nur 28815 Bytes, sondern 28815 + 16000 Bytes, das sind 44815! Nun fehlen aber noch immer weit über 30000 von den eigentlich erwarteten 80000.

Im technischen Einführungskapitel unterhielten wir uns über den in MSX-System Version 1.0 innewohnenden Mikroprozessor, den Z80A. Dieser Prozessor, so stellten wir fest, kann gleichzeitig 64000 Speicherplätze verwalten. Außerdem erfuhren wir, daß das MSX-BASIC im ROM der MSX-Computer abgespeichert ist und eine Größe von 32000 Speicherplätzen einnimmt.

Nun wieder zu unserer Rechenaufgabe: Ihr MSX-Computer hat nach dem Einschalten laut Anzeige 28815 Speicherplätze frei. Vorhin erfuhren wir, daß zudem über 16000 Speicherplätze von der Grafik belegt werden = 44815 Speicherplätze. Das BASIC ist noch einmal 32000 Speicherplätze groß, so würde unser Endergebnis lauten: 28815 + 16000 + 32000 = 76815.

Warum ich das BASIC, das doch im ROM liegt, zu unserem freien Speicherplatz dazurechne, wollen Sie wissen? Unser Mikroprozessor kann gleichzeitig nur 64000 Speicherplätze verwalten, also muß ich den BASIC-Speicherraum doch auch mitberechnen (oder hätten Sie anstelle von BASIC lieber freien Speicherplatz?). Sie können allerdings später auch andere Programmiersprachen oder Betriebssysteme auf Ihrem MSX-Computer laufen lassen - dann haben Sie auch wahrlich mehr Speicherplatz frei (wie z.B. bei CP/M = fast 60000 Speicherplätze), aber zur Zeit sollten Sie mit BASIC arbeiten, so lange Sie noch in der Computerei auf der BASIS stehen.

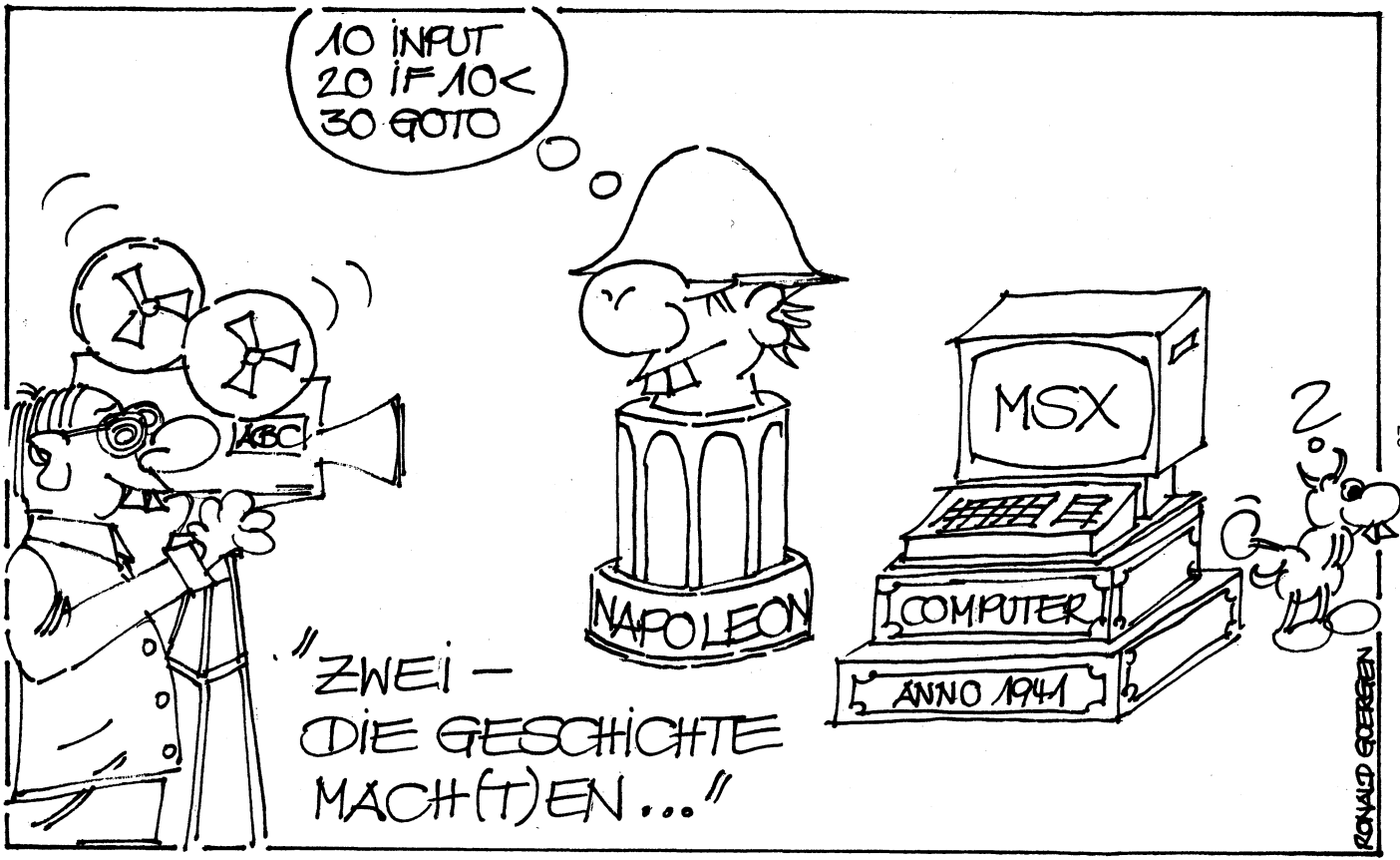
Nun ein leztes Mal zu unserem Rechenexempel: Uns fehlen noch immer gut 3000 Speicherplätze: Diese sind nicht nur in unserer ungenauen Rechnung zu finden (z.B. der Grafikspeicher ist genaugenommen 16384 Speicherplätze groß), sondern auch darin, daß Ihr MSX-Computer bei der Arbeit mit der Programmiersprache BASIC eine größere Zahl von Speicherzellen für seine Arbeit benötigt (z.B. sind hinter der Belegung einiger Tasten jeweils 16 Speicherplätze versteckt).

Sind Sie nun zufrieden?

Nicht ganz, aber seien Sie getröstet einerseits, daß es viele Monate, wenn nicht sogar Jahre dauern wird, bis Sie erst einmal diese 28815 freien Speicherplätze sinnvoll verwenden können. Außerdem gibt es selbstverständlich Tips, wie Sie auch den 'verborgenen' RAM-Speicher in der Größe des BASIC mitbenutzen können (ein bißchen arbeitet Ihr MSX-Computer auch schon so, denn wie wir gesehen haben, verwaltet er ja nicht 'nur' seine 64000 obligatorischen Speicherplätze, sondern immerhin fast 80000). Wie wäre es auch sonst denkbar, wenn demnächst Speichererweiterungen für die MSX-Computer in der Größe von bis zu 256000 Speicherplätzen angeboten werden sollen?

Nun gehen wir aber frisch ans Werk und wollen unsere Tastatur kennenlernen:

Der Hautptteil der Tastatur besteht aus den alphabetischen Zeichen, die Sie auf der Abbildung der Tastaturkappe in Großschrift vorfinden. Betätigen Sie jedoch die Buchstabentasten (dabei kann gar nichts schiefgehen), werden Sie auf dem Bildschirm augenblicklich den entsprechenden Kleinbuchstaben abgebildet sehen. Sie können die Darstellung der Großbuchstaben dadurch unmittelbar in die Wege leiten, indem Sie eine der beiden mit SHIFT gekennzeichneten Tasten links unten bzw. rechts unten von der Tastatur niederdrücken, während Sie gleichzeitig die gewünschte Buchstabentaste betätigen.

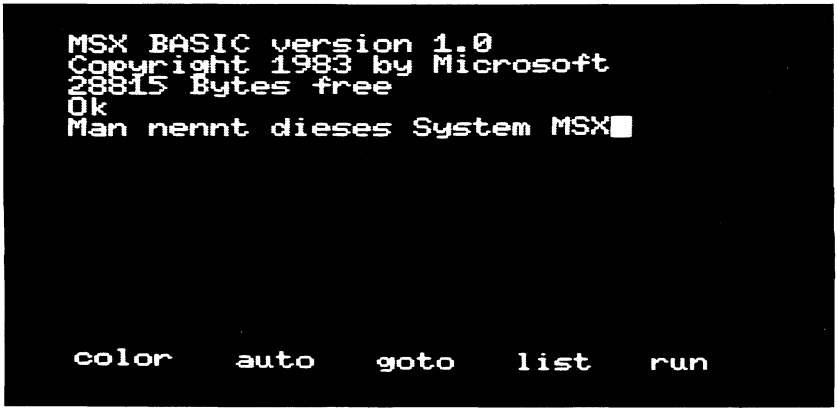

10 INPUT
20 IF 10<
30 GOTO
ABC
NAPOLEON
MSX
COMPUTER
ANNO 1941
"ZWEI –
DIE GESCHICHTE
MACH(T)EN..."
RONALD GOERGEN

SHIFT

Hier geht es nicht darum, erst eine der SHIFT-Tasten und anschließend eine Buchstabentaste niederzudrücken, sondern beide Tastenandrücke gleichzeitig vorzunehmen. Am besten, Sie drücken erst eine der beiden SHIFT-Tasten und halten dieselbige dann so lange niedergedrückt, bis Sie auch den gewünschten Buchstaben niedergedrückt haben, der in Großschrift auf dem Bildschirm dargestellt werden soll.

Am besten, Sie probieren die Umschaltung von Klein- und Großschrift einmal selbst aus, indem Sie folgenden Satz eingeben:

Man nennt dieses System MSX



Leertaste

Vielleicht haben Sie schon von der Schreibmaschine gewohnt die richtige Taste entdeckt, mit der Sie den Leerraum zwischen den Wörtern überbrücken können. Wenn Sie dies noch nicht wissen: Sie betätigen zur Erzeugung eines Leerraums die längliche Taste, die sich am unteren Rand der Tastatur befindet. Es soll Ihnen an dieser Stelle unbenommen bleiben, ob Sie lieber einen größeren Leerraum zwischen den verschiedenen Wörtern erzeugen wollen, in jedem Fall reicht für diesen Trennvorgang das einmalige Andrükken dieser sogenannten Leer- oder Spacetaste voll und ganz aus.

SHIFT und SHIFT

Wir waren vorhin bei den zwei SHIFT-Tasten am linken bzw. rechten unteren Rand Ihrer Tastatur stehengeblieben: warum gleich zwei SHIFT-Tasten? Der Sinn dafür liegt darin begründet, daß Sie ja auch zwei Hände haben; wenn ein Buchstabe groß dargestellt werden soll, der sich auf der rechten Tastaturseite befindet (wie z.B. O, P, K oder L), drücken Sie diesen Buchstaben bequemererweise mit einem Finger der rechten Hand herunter, während Sie mit einem Finger der linken Hand gleichzeitig die linke SHIFT-Taste betätigen.

Andersherum geht es selbstverständlich geauso: Sie wollen die Buchstaben Q, W, A oder S groß darstellen: Drücken Sie hierzu die rechte SHIFT-Taste herunter (selbstverständlich mit einem Finger der rechten Hand), während Sie mit einem Finger der linken Hand den gewünschten Buchstaben auf der linken Tastaturhälfte betätigen. Sie erinnern sich an unseren vorhin eingegebenen Satz?

Man nennt dieses System MSX

Der Satz war sicherlich noch etwas ungewohnt einzugeben (soweit Sie nicht vorher bereits einmal auf einer Computer- oder Schreibmaschinentastatur gearbeitet haben). Besonders ärgerlich (vielleicht auch bequem?) war es gar am Schluß, als Sie die Buchstabenkombination MSX eingeben mußten, gleich drei große Buchstaben unmittelbar hintereinander.

Manchmal kommt es in Texten vor, daß Sie - um etwas deutlicher hervorzheben - ganze Textpassagen nur mit großen Buchstaben schreiben wollen. Ist dies der Fall, wäre ein andauerndes Niederdrücken der SHIFT-Taste mit der Zeit sehr ermüdend.

CAPS

Doch kein Probem für Ihren MSX-Computer: Sie betätigen zu Beginn der Großschreibphase einfach die Taste mit dem Aufdruck CAPS (vielleicht leuchtet dann sogar irgendwo auf Ihrem MSX-Computer zur Bestätigung ein kleines Lämpchen auf!). Von nun an brauchen Sie die SHIFT-Tasten zur Buchstabengroßschreibung nicht mehr zu betätigen, denn alles, was Sie schreiben, wird nun eh in großen Buchstaben auf dem Bildschirm dargestellt.

Ist Ihre 'Großschreibphase' wieder beendet, betätigen Sie bitte ein weiteres Mal die CAPS-Taste (wenn vorher beim ersten Betätigen dieser Taste irgendwo auf dem Computer ein Lämpchen hell erleuchtete, erlischt es jetzt nach dem wiederholten Drükken der CAPS-Taste sofort wieder. Alles ist beim alten und Sie können wieder durch Betätigung der zwei SHIFT-Tasten zwischen Groß- und Kleinschrift schnell hin- und herschalten.

Zur Einübung der Arbeit mit SHIFT, CAPS, Leertaste und der Buchstabentastatur, geben Sie bitte nun folgenden Satz auf Ihrer MSX-Computertastatur ein:

Das Tippen LERNEN macht SPASS

Vielleicht können Sie den Inhalt dieser Aussage nicht ganz bestätigen, insbesondere die Kombination der zwei großgeschriebenen Wörter LERNEN und SPASS. Sei es drum, vielleicht macht Ihnen ja das Kennenlernen der Tastatur Ihres MSX-Computers Spaß. So schwer ist diese Bedienung nämlich auch nicht ... man muß sich halt nur trauen.

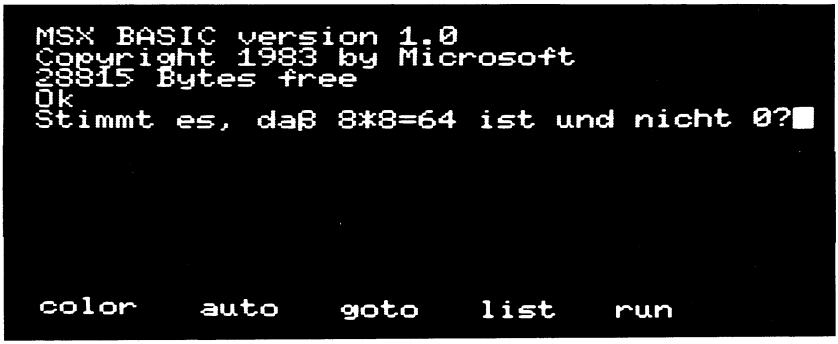
Untere und obere Funktion

Schauen wir uns nun die weiteren Tasten Ihres MSX-Computers an: Auf den meisten Nicht-Buchstaben-Tastaturkappen erkennen Sie unschwer jeweils zwei Zeichen; ein Zeichen ist in der unteren Hälfte der jeweiligen Taste abgebildet, während das andere Zeichen oben steht. Was bedeutet das?

Schauen wir uns dazu die Taste '8' in der oberen Tastaturreihe einmal etwas genauer an: unten steht die Zahl 8 und oben ein Sternchen (dieses Sternchen hat nicht nur bei Ihrem MSX-Cmputer die Bedeutung des uns vertrauten Malpunktes beim Rechnen). Betätigen Sie die Taste, erscheint auf dem Bildschirm die Zahl 8 und ... betätigen Sie diese Taste gleichzeitig mit SHIFT (!), erscheint auf dem Bildschirm der Malpunkt bzw. das computertypische Mal-Sternchen.

Also ist die Aufschrift der zweifach belegten Tasten dadurch anzusteuern, indem Sie diese Tasten gemeinsam mit einer der SHIFT-Tasten drücken. Wichtig zu wissen ist es allerdings, daß die verriegelte SHIFT-Taste mit der Aufschrift 'CAPS' keine Auswirkung auf die Zweifach-Tasten hat, denn allgemein gesehen kommt die Nicht-SHIFT-Funktion der zweifach belegten Tasten (z.B. die Ziffern 0 bis 9) weitaus häufiger vor als die entsprechende SHIFT-Funktion. Um bei gedrückter CAPS-Taste also wiederum das Sternchen über der 8 anzusteuern, müssen Sie eine der SHIFT-Tasten zusätzlich gedrückt halten. Vielleicht probieren Sie zur Übung einmal aus, folgenden Satz in Ihren MSX-Computer samt aller Satzzeichen einzugeben:

Stimmt es, daß 8*8=64 ist und nicht 0?



In diesem Satz tauchen wieder zwei Dinge auf, die speziell Ihre MSX-Computertastatur betreffen:

Deutsche Umlaute und ß

1. Nicht alle MSX-Computer sind speziell für den deutschen Markt tastaturmäßig angepaßt worden. So kann es sein, daß Sie gar nicht wissen, wie Sie den Buchstaben 'ß' ansteuern sollen. In diesem Fall rate ich Ihnen, mein Umlautprogramm aus der MSX-Programmsammlung (ebenfalls bei DATA BECKER erschienen) zu verwenden. Zwar haben Sie auch dann noch nicht eine deutsche Umlauttastatur, aber Sie können durch Drücken auf nicht so häufig gebrauchte Tasten doch die deutschen Umlaute auf Bildschirm und Drucker erzeugen. Hat Ihr MSX-Computer hingegen eine deutsche Umlauttastatur, brauchen Sie sich um die gerade angesprochenen Dinge nicht zu kümmern; seien Sie zufrieden!

Die Ziffer 0 und der Buchstabe O

2. In dem gerade eingetippten Satz tauchte die Ziffer '0' auf. Wenn Sie bisher mit einer Schreibmaschine gearbeitet haben, ist es für Sie klar, den Buchstaben 'O' (groß- oder kleingeschrieben) dafür zu gebrauchen. Nicht so bei Ihrem MSX-Computer und Computern allgemein! Schauen Sie sich einmal die oberste Tastaturreihe an: im Gegensatz zur Schreibmaschine folgt nach der Ziffer '9' die Ziffer '0' (bei den meisten Computern zur Unterscheidung vom Buchstaben 'O' mit einem Querstrich versehen). Gebrauchen Sie die Ziffer '0' später in Ihren Programmen (z.B.

für eine Zeilennummer), sind Sie dazu verpflichtet, die Taste mit der 'Null' (rechts neben der '9') zu betätigen und keine andere.

Wie ich bereits öfter betonte, kann Ihr MSX-Computer nicht logisch selbständig denken. So unterscheidet er diese in der Form zwar sehr ähnlichen aber in der Funktion keineswegs gleichen Zeichen sehr genau und verzeiht solch einen Eingabefehler nie und nimmer. Also: Wollen Sie Zahlen eingeben, so tun Sie das auch und mischen nicht Zahlen mit Buchstaben zu einem Zeichenbrei! Wollen Sie einem Speicherplatz im Computer den Namen 'A0' (A Null) geben, so tun Sie auch dies und behalten es dann bei, der Speicherplatz 'AO' (A O) hat nur vom Aussehen nicht aber von seiner Funktion etwas mit dem Speicherplatz 'A0' (A Null) zu tun.

In den folgenden zwei Abschnitten über die Tasten GRAPH und CODE unterscheiden sich die verschiedenen MSX-Computermodelle geringfügig voneinander. Also brauchen nicht alle Aussagen für Ihren MSX-Comuter zuzutreffen!

Bisher haben wir zwei Belegungen der Tastatur kennengelernt: Ohne SHIFT (und ohne CAPS) Kleinbuchstaben bzw. das unten stehende Zeichen der zweifach gekennzeichneten Tasten (z.B. die Ziffern '0' bis '9') sowie mit SHIFT Großbuchstaben bzw. das oben stehende Zeichen auf zweifach gekennzeichneten Tasten (z.B. das Sternchen auf der Taste mit der Ziffer '8').

GRAPH

Fast sämtliche Tasten sind noch mit zwei weiteren Funktionen belegt: Drücken Sie die GRAPH-Taste an Ihrem MSX-Computer gleichzeitig mit einer anderen Zeichentaste herunter (ähnlich wie Sie eine der SHIFT-Tasten für Großschreibung mit einem Buchstaben gleichzeitig betätigen), erscheinen auf dem Bildschirm entweder Grafikzeichen (z.B. kleine Kreise und Blocks) oder mathematische Zeichen wie Wurzel und Unendlichkeitsanzeige.

CODE

Drücken Sie hingegen die mit CODE gekennzeichnete Taste mit einer anderen Taste zusammen (bitte wiederum genauso wie beim gleichzeitigen Betätigen von SHIFT und Buchstabentaste zur Großschreibung geschehen), erscheinen auf Ihrem Bildschirm ausländische Buchstaben oder Zeichen.

Bei MSX-Computern, die nicht über eine deutsche Tastatur verfügen, ist dies die einzige Möglichkeit, deutsche Umlaute in Texte direkt einzubauen (im Zeichensatz verfügt nämlich weltweit jeder MSX-Computer auch über die deutschen Umlaute und das 'ß', genauso wie über die speziellen spanischen, französischen usw. Zeichen).

So unscheinbar die Buchstabentasten Ihres MSX-Computers aussehen, Sie können damit - wie gerade angeführt - verschiedene Funktionen direkt ausführen:

1. ohne gleichzeitig eine andere Taste zu betätigen:
   kleine Buchstaben

2. gemeinsam mit SHIFT:
   große Buchstaben

3. gemeinsam mit GRAPH:
   Grafikzeichen und mathematische Sonderzeichen

4. gemeinsam mit CODE:
   internationale Sonderzeichen im Alphabet wie z.B. die deutschen Umlaute

CTRL oder CONTROL

Nicht genug damit. Obgleich Sie zur Zeit noch nichts damit anfangen können, will ich Ihnen an dieser Stelle noch die fünfte und damit letzte Belegung der Tasten vorführen: Drücken Sie die Taste, die mit der Buchstabenkombination CTRL oder CONTROL gekennzeichnet ist gemeinsam mit einer Buchstabentaste: hier werden keine Zeichen direkt auf den Bildschirm ausgegeben, sondern Computersteuerungen vorgenommen.

Drücken Sie z.B. die Taste CTRL (oder CONTROL) mit dem Buchstaben 'L' zusammen ... augenblicklich wird der Bildschirm gelöscht. Oder wie wäre es mit der Kombination CTRL (oder CONTROL) mit dem Buchstaben 'I'? Ihre Schreibarke bewegt sich daraufhin um mehrere Zeichen nach rechts und auf jeden erneuten Tastendruck wiederum einen Abschnitt weiter.

Da Sie durchs Programmieren und erst recht durch das Tastendrükken keinen Computer zerstören können, probieren Sie ruhig weitere Tastenkombinationen aus oder aber Sie testen die CTRL-Funktionen, die wir im Anhang mit kurzer Erläuterung aufgeführt haben.

Editiertasten

Schließlich bietet Ihnen Ihr MSX-Computer eine größere Anzahl von sogenanten Editiertasten an, die wir anschließend in ihrer Funktion noch kurz besprechen wollen.

Was heißt Editieren? Ins Deutsche direkt übersetzt: verbessern oder korrigieren. Bei einem Computer haben Sie Vorteile gegenüber der Schreibmaschine: Wenn Sie einen Fehler machen, können Sie denselbigen leicht vom Bildschirm auf elektronischem Wege entfernen - eben durch diesen sogenannten Editierprozeß -, während Sie bei der Schreibmaschine den Fehler mit Hilfsmitteln (Korrekturband, Tipp-Ex) auf mechanischem Wege zu Leibe rücken müssen.

HOME

Wie geht nun das Editieren auf dem Computerbildschirm elektronisch vor sich? Wählen wir zuerst die rabiate Mehode: tippen Sie schnell ein paar beliebige Zeichen auf Ihrem MSX-Computer ein. Nun drücken Sie bitte die Taste, die mit den Abkürzungen CLS/HOME gekennzeichnet ist.

Ähnliches passiert, wie vorhin, als wir die Taste CTRL gemeinsam mit der Buchstabentaste 'L' niederdrückten: die Schreibmarke kehrt in die obere Bildschirmzeile zurück, allerdings löscht sie noch nicht den Bildschirm.

Die untenstehende Buchstabenkombination auf der soeben gedrückten Taste heißt HOME und bedeutet soviel wie: kehre mit der Schreibmarke (das rechteckige Zeichen, das immer genau diejenige Position anzeigt, wo wir uns gerade auf dem Bildschirm befinden - auch Cursor genannt) zu der Bildschirmposition links oben zurück. Was bedeutet dieser Effekt übertragen auf unsere Arbeit mit der Schreibmaschine? Drehe die Walze noch einmal zurück, damit du links oben zu schreiben (= zu überschreiben) beginnen kannst.

Probieren Sie es aus; der von Ihnen eingegebene Text überschreibt den alten Buchstabe für Buchstabe, Zeichen für Zeichen.

CLS

Drücken Sie die CLS/HOME-Taste allerdings mit einer der SHIFT-Tasten gemeinsam, wird die Funktion CLS ausgeführt, was so viel bedeutet wie: lösche den Bildschrm (die Abkürzung CLS bedeutet ausgeschrieben: CLear Screen = lösche den Bildschirm) und kehre in die linke obere Ecke mit der Scheibmarke zurück. Übertragen auf die Arbeit mit der Schreibmaschine würde das heißen: nimm das beschriebene Blatt aus der Schreibmaschine heraus, spanne ein neues ein und beginne links oben zu schreiben (die CLS-Funktion entspricht übrigens in jedem Detail unserer vorher bereits gedrückten Tastenkombination CTRL (CONTROL) und Buchstabe 'L').

Doch nicht in jedem Fall wollen wir so rabiat mit unseren geschriebenen Texten umgehen. Wie sieht also die 'Korrekturbandmethode' bei Ihrem MSX-Computer aus? Schreiben Sie zu diesem Zweck folgenden Satz auf den mit der CLS/HOME-Taste zuvor gelöschten Bildschirm (bitte mit Fehlern genauso eingeben):

Wen ich MSXX-Programme schreiben könnthe

Ein Lehrer würde beim Anblick dieses Satzes sagen: Sechs! Doch wir sind guten Willens und wollen alles wieder besser machen.

Pfeiltasten

Fehler 1 Korrektur: Sie haben sicherlich das Pfeiltastenkreuz auf Ihrem MSX-Computer bereits gesehen. Betätigen Sie die linke Pfeiltaste, bis Sie sich direkt auf dem Leerzeichen hinter dem ersten Wort (Wen) befinden. Geben Sie nun den fehlenden Buchstaben 'n' ein.

INS

Zwar ist das Wort 'Wenn' nun laut Duden richtig geschrieben, dafür fehlt jetzt das Leerzeichen. Kein Problem für Ihren MSX-Computer: Sie drücken die Taste INS (Insert = Einfügen), wonach die Schreibmarke in der Größe auf die Hälfte schrumpft. Nun können Sie das Leerzeichen durch einmaliges Drücken der Leertaste einfügen.

Weiter bewegen Sie die Scheibmake hinter das falschgeschriebene Wörtchen 'MSXX'. Nun müssen Sie das links neben der Schreibmarke stehende 'X' löschen. Eine Möglichkeit wäre die, das überzählige 'X' drch ein Leerzeichen zu überschreiben. Dies wäre jedoch eine unfeine Methode, weil dann zwischen 'MSX' und dem darauffolgenden Bindestrich eine häßliche Lücke klaffen würde.

BS oder BACKSPACE

Besser ist es, Sie drücken die mit BS oder BACKSPACE (übersetzt: ein Schritt zurück) gekennzeichnete Taste einmal. Wie von magnetischer Kraft wird die rechte Satzhälfte herangezogen und somit verdeckt nun der Bindestrich unser überzähliges 'X'.

DEL oder DELETE

Schließlich müssen wir noch Fehler 3 beseitigen: Bewegen Sie Ihre Schreibmare mit der Rechts-Pfeiltaste auf das überflüssige 'h' im Wörtchen 'könnthe'. Nun betätigen Sie bitte die mit DEL oder DELETE gekennzeichnete Taste (DELETE in deutsch: lösche das Zeichen, wo die Schreibmarke zur Zeit steht) und schon ist auch dieser, unser dritter Fehler ausgemerzt und der vorher so fehlerhafte Satz lautet nun:

Wenn ich MSX-Programme schreiben könnte

Falls Sie bei der Bedienung der Pfeiltasten, der INS-, DEL- oder BS-Taste noch große Probleme hatten, beginnen Sie unser Beispiel noch einmal von vorn und mit einiger Übung wird Ihnen auch die Bedienung dieser Tasten leichter von der Hand gehen.

ESC und SELECT

Mit den Tasten, die mit ESC und SELECT gekennzeichnet sind, werden wir uns in diesem Einsteigerbuch nicht näher beschäftigen - dafür müßten doppelt so viele Seiten zur Verfügung stehen.

Bleibt eigentlich nur noch die TAB-Taste und ein Hinweis zu den Tasten F1 bis F10.

TAB

Die TAB-Taste erfüllt die gleiche Funktion wie vorhin die Tastenkombination CTRL (oder CONTROL) und 'I'. Sie können ähnlich wie bei der Tabulatortaste auf der Schreibmaschine hiermit einen Sprung um mehrere Zeichen nach rechts vornehmen; sinnvoll beim Erstellen von Tabellen oder wenn Sie in der Zeile schneller mit Ihrer Schreibmarke vorankommen wollen.

F1 bis F10

Mit den Tasten F1 bis F5 bzw. F6 bis F10 (letztgenannte 5 Tasteninhalte werden durch gleichzeitiges Niederdrücken gemeinsam mit einer der SHIFT-Tasten angesteuert) sind sogenannte Funktionstasten. Sie können diese Tasten mit Programmierbefehlen belegen (soweit dies nicht schon beim Einschalten des Computers zu Ihrer Zufriedenheit geschehen ist), um davon befreit zu sein, fortwährend die gleichen Tastenkombinationen nacheinander drükken zu müssen.

Probieren Sie auch diese Tasten ruhig einmal aus, selbst wenn Ihnen die Bezeichnungen wie 'PRINT' oder 'LIST' noch nicht sehr viel sagen werden.

Noch ein letzter Hinweis: Fast alle zeichenausgebenden Tasten auf Ihrem MSX-Computer verfügen über eine Wiederholfunktion. Wollen Sie z.B. eine längere Textstelle unterstreichen, so drükken Sie die '-'-Taste etwas länger an und schon beginnt Ihre Schreibmarke, mehrere Bindestriche oder Unterstreichungsstriche in Folge zu produzieren.

Dazu, wie bei den meisten Tastendrücken, vernehmen Sie ein Klicken durch den Lautsprecher des angeschlossenen Fernsehgeräts - als Kontrollfunktion über den richtig erfolgten Tastenandruck in jedem Fall eine große Hilfe.

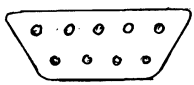
Was können wir an unseren MSX-Computer anschließen?

Wie wir bereits im vorigen technischen Einführungskapitel erfahren haben, besteht ein Computer aus ROM, RAM und dem Mikroprozessor. Während das ROM u.a. die nicht löschbare Programmiersprache BASIC enthält und das RAM den freien Speicherplatz (für unsere Programme und Texte) zur Verfügung stellt, ist der sogenannte Mikroprozessor als Herzstück unseres MSX-Computers dafür verantwortlich, daß nicht nur die Kommunikation zwischen RAM und ROM, sondern auch die Kommunikation zur Außenwelt einwandfrei funktioniert.

Ebenfalls haben wir im vorigen Kapitel im Überblick Kenntnisse darüber bekommen, was denn zum Computer noch alles dazugehören kann, die sogenannte Umwelt oder Peripherie des Rechners. Während die Tastatur bereits vorhanden ist und somit von vornherein aus unserem Nur-MSX-Computer gleich eine Art Kompaktanlage macht, muß man sich die zusätzliche Peripherie käuflich extra dazuerwerben.

Hier wird es nun schwierig für mich, da dieses Buch ja für alle MSX-Computer auf dem Markt konzipiert wurde, und das sind mehr als 20 Rechner von unterschiedlichen Firmen. Während das BASIC überall gleich ist (also das ROM), fangen die Unterschiede jedoch gerade bei der Anzahl der verfügbaren Anschlüsse zur Außenwelt an. Am Schluß dieses Kapitels habe ich deshalb eine Übersicht abgedruckt, in der u.a. angegeben ist, welcher MSX-Rechner über welche Anschlüsse bereits von vornherein verfügt.

1.Zwei Joystickports



JOYSTICK MAL-PAD PADDLE LIGHTPEN MAUS TRACKBALL

A Joystick

Die Aktionen unseres MSX-Computers können zum Teil ferngesteuert werden mit Joysticks. Zu diesem Zwck verfügen alle MSX-Computer über zwei Buchsen, aus denen in der oberen Reihe 5 Pins und in der unteren Reihe 4 Pins ragen.

Wir können an diese Buchsen nicht nur für zwei Spieler sogenannte Joysticks (Steuerhebel zum Spielen und Abfeuern) anschließen, auch andere Steuerinstrumente wie Paddles und sogar ein Zeichenpad sind dort jederzeit willkommen.

Beim Kauf von Joysticks für Ihren MSX-Computer sollten Sie darauf achten, daß Sie sich möglichst spezielle MSX-Joysticks zulegen, denn die MSX-Computer besitzen im Gegensatz zu anderen gängigen Homecomputern die spezielle Möglichkeit, nicht nur mit einem, sondern gleich mit zwei unterschiedlichen Joystickknöpfen die Aktionen auf dem Bildschirm zu beeinflussen. Diese Funktion ist z.B. bei manchen Spielen sehr sinnvoll, wenn Sie mit dem einen Joystickknopf (bei dieser Spielhandlung auch Feuerknopf genannt) die Kanone auf der linken Bildschirmseite und mit dem anderen Feuerknopf die Kanone auf der rechten Bildschirmseite ansteuern können.

Die Bewegung des eigentlichen Joystickhebels wird pro Joystickport nicht nur in den vier Himmelsrichtungen Norden, Westen, Süden, Osten, sondern auch in den Zwischenhimmelsrichtungen (z.B. Nordost, Südwest ...) abgefragt (man könnte auch sagen, daß alle Gradstellungen zwischen 0 und 360 Grad im 45 Grad-Abstand gemessen werden, das sind insgesamt je Joystick 8 Positionen).

Durch das Andrücken des Joystickhebels in die gewünschte Richtung können Sie somit das auf dem Bildschirm befindliche Objekt in die gewünschte Richtung steuern. Ich sprach beim Joystickknopf auch von Feuerknopf. Es ist zwar eine Möglichkeit, den Joystick samt seinen Knöpfen für ordinäre Spiele zu gebrachen, es ist aber selbstverständlich auch möglich, damit interessantere Dinge zu tun z.B. ein Bild zu malen, dasselbige beim Druck auf Joystickknopf Nr.1 auf Kassette abzuspeichern und es bei

Druck auf Joystickknopf Nr.2 wieder in den Bildschirmspeicher einzuladen. Sie können sich selbst weitere Anwendungsmöglichkeiten für Joysticks überlegen, wozu Sie diese Steuergeräte an Ihren MSX-Computer anschließen wollen.

B Malpad

Das zuletzt genannte Beispiel (Zeichnen mit Joystck) kann viel einfacher und besser mit einem sogenannten Pad gelöst werden, das Sie ebenfalls an Ihren MSX-Computer über die Joystickports ohne Mühe anschließen können. Beim Pad haben Sie eine etwa postkartengroße Malfläche zur Verfügung, die in ihrem verkleinerten Aufbau dem Bildschirm entspricht.

Zu dem Pad wird normalerweise ein Plastikgriffel geliefert. Diesen Malgriffel (im Notfall tut es auch der Fingernagel - aber Vorsicht vor Kratzern!) setzen Sie auf das Malpad auf und nun können Sie auf dem Bildschirm die schönsten Gemälde malen, nur mit dem Unterschied zu einem gewöhnlichen Bild, daß Sie hier eigentlich nur den Griffel auf die Maltafel aufdrücken, die im gleichen Augenblick den exakten Aufdruckpunkt dem Computer weitervermittelt.

Sicherlich eine bessere Möglichkeit, seinem künstlerischen Drang ein Medium an die Hand zu geben als mit Hilfe eines Joysticks (der nach dem Loslassen immer wieder in die Mittelposition zurückkehrt); dafür aber ist ein Pad auch nicht nur erheblich teurer als ein Joystick (schließlich müssen alle Punkte auf dem Pad mit Sensoren genauestens abgefragt und die Information an Ihren MSX-Rechner weitervermittelt werden), sondern auch nur speziell für diese eine Tätigkeit, nämlich zum Zeichnen, gedacht ist.

Selbstverständlich verfügt das Pad auch über ein bis zwei Andruckknöpfe, die sich teils an der Außenseite des Pad, teils direkt am Plastikgriffel befinden. Zudem gibt es Software für die Pads, die eine Art Menü auf dem Bildschirm für die verschiedenen Funktionen ausgibt; Sie setzen den Plastikgriffel dann einfach dort auf das Pad, wo die von Ihnen gewünschte Funktion steht (z.B. Abspeichern oder Farbwahl ...).

C Paddle

Artverwandt zum Joystick ist ein drittes Steuerinstrument, das Sie an einen oder beide Joystickports (letzteres bei zwei Mitspielern) anschließen können: das Paddle. Obgleich der Name ähnlich wie Pad klingt, haben diese zwei Steuerungselemente nichts miteinander zu tun. Das Padde besteht aus einem Drehknopf, mit dem Sie einen Gegenstand je nach Drehrichtung nach links oder nach rechts steuern können.

Um sich die Funktion eines Paddles besser vorstellen zu können, will ich Ihnen zwei Beispiele nennen:

1. Sie wollen ein Auto auf dem Bildschirm lenken. Dazu benutzen Sie den Drehknopf des Paddles wie ein Autolenkrad. Drücken Sie an der einen Seite des Paddles den auch hier befindlichen Knopf, fährt Ihr Wagen schneller, lassen Sie ihn wieder los, verlangsamt sich die Geschwindigkeit. Hat Ihr Paddle gar auch noch einen zweiten Knopf, der von Ihrem MSX-Computer anders verstanden wird (wie bei den Joysticks), können Sie hiermit kuppeln.

2. Sie wollen ein Männchen in einem Spiel auf einer Wippe hochspringen lassen. Trifft das Männchen besonders günstig auf der Wippe auf, wird es entsprechend sehr hoch befördert und kann dort Luftballons abtreffen. Aufgrund der Schwerkraft kommt das Männchen schließlich wieder zum Erdboden herunter und Sie müssen nun die Wippe so mit Ihrem Paddle neu justieren, daß das Männchen dort auch wieder günstig auftreffen kann.

Beide Spiele ließen sich zwar auch mit einem Joystick realisieren, wären dann aber bei weitem nicht so praktisch und - insbesondere wie beim Autorennen - nicht so wirklichkeitsgetreu zu bedienen.

An weiteren Steuerinstrumenten, die Sie an Ihre Joystickbuchsen anschließen können, will ich kurz noch drei weitere erklären:

D Trackball und Maus

Der Trackball: Hierbei handelt es sich um ein handtellergroßes Kästchen, das auf den Tisch gestellt wird. Der Name dieses Steuergeräts rührt von einer Kugel (Englisch: ball) her, die sich in dem Kästchen befindet. Sie ist so groß, daß sie nach draußen ragt und von einer darauf gelegten Hand nicht nur berührt sondern auch mit Leichtigkeit in alle Richtungen bewegt werden kann. Die Funktion des Trackballs entspricht somit der eines Joysticks (8 Richtungen der Kugelbewegung werden registriert), wobei der Trackball allerdings sehr viel leichter zu handhaben ist.

Eine ähnliche Technik wie beim Trackball wird seit neuestem in größeren Personalcomputern zur Gewohnheit: die Maus. Dies ist eigentlich auch ein Trackball, nur die Kugel wird erst dadurch in Bewegung gesetzt, wenn das Kästchen auf der Tischfläche irgendwo hinbewegt wird (die Kugel sizt nämlich wie ein Rad an der Unterseite des Kästchens).

Es ergibt sich von selbst, daß sowohl Maus als auch Trackball über mindestens einen Feuernopf verfügen, der (die) gleiche(n) Funktion(en) auslösen kann (können) wie beim Joystick bereits ausführlich erläutert.

E Lightpen

Sie erinnern sich noch an das Malpad? Hierbei benutzen Sie einen Plastikgriffel und vollziehen Ihre kreative Tätigkeit auf einem postkartengroßen Plastikbrettchen mit Sensorzellen.

Der Lightpen ist auch ein Malwerkzeug, nur malen Sie hier nicht auf einer Zwischenstation zum Bildschirm wie dem Malpad, sondern direkt auf der Bildschirmoberfläche. Keine Angst, Ihr Fernsehbildschirm wird hierbei weder verkratzt noch mit den verschiedensten Farben beschmiert. Wie der Name Lightpen (Deutsch: Lichtgriffel) bereits ausdrückt, sitzt in diesem Steuerinstrument eine Fotodiode, die geau feststellen kann, an welchem Bildschirmpunkt Sie gerade Ihre Künste schweifen lassen.

Dies hört sich sehr einleuchtend und sehr klar an, wenn da nicht unser 'normaler' Fernseher wäre: bei bisherigen Lightpens auf dem Zubehörmarkt waren die Ergebnisse nicht in jedem Fall zufriedenstellend, da die dicke Glasscheibe unserer Fernsehgeräte die Lichtstrahlen nach draußen hin derart ablenkt, daß der aufgesetzte Lightpen nicht dort malt, wo dies eigentlich sein soll, sondern meist einige Zentimeter links oder rechts, oben oder unten vom eigentlich angesteuerten Bildschirmpunkt. Besser gelingt dies zwar mit einem Farbmonitor mit abenommener Kontrastscheibe (falls vorhanden), jedoch zu mehr als einer Spielerei (oder einer Menüauswahl) ist dieses System im Augenblick kaum zu gebrauchen. Als Effekt jedoch immer sehr reizvoll.

Beim Schreiben dieser Zeilen war es noch nicht abzusehen, ob in nächster Zukunft ein Lightpen für MSX-Computer auf den Markt kommen wird. Außerdem stand noch nicht fest, ob der Lightpen am Joystickport und/oder am Expansionsbus oder vielleicht gar am Recorderanschluß angeschlossen wird. Letztere Möglichkeiten scheinen mir laut der technischen Unterlagen als wahrscheinlich.

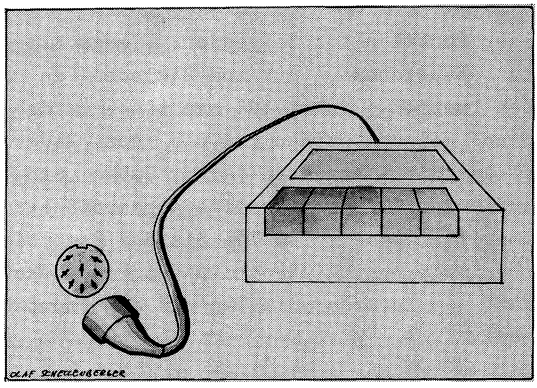
Es gibt sicherlich noch viele andere Möglichkeien (gerade für Bastler, die Steuerungen konrollieren wollen oder für Biologen, die die Messung eines Gewässers überwachen wollen), die Joystickbuchsen Ihres MSX-Computers sinnvoll zu nutzen. Jedoch will ich mit der Erklärung der angeführten Varianten hier Schluß machen und noch einen wichtigen Hinweis anbringen:

Es reicht in keinem Fall aus, wenn Sie eines der Steuergeräte lediglich an Ihren MSX-Computer anschließen. Vielmehr müssen Sie zum Gebrauch ein spezielles Programm dazuverwenden (bei Spielen ist die Joysticksteueröglichkeit meist direkt eingebaut, beim Kauf eines Malpads ist ein entsprechendes Steuerprogramm in jedem Fall immer mit dabei), sonst läuft gar nichts.

Sie brauchen allerdings bei der Selbstprogrammierung und somit der Anwendung dieser Steuerinstrumente für Ihre Programme keine allzu große Angst zu haben: Ihr MSX-Computer verfügt bereits über ein derart reiches Programmiervokabular (= BASIC), so daß Sie direkt Befehle zur Abfrage von Joysticks, Trackball, Paddle

und Malpad verwenden können. Bei Anschluß eines Lightpen hingegen ist dies nicht so einfach möglich, allerdings werden Sie keinen Lightpen für Ihren MSX-Computer erhalten, für den nicht gleichzeitig das entsprechende Programm (= die Software) mitgeliefert werden kann.

2. Der Kassettenrecorderanschluß



Bei allen MSX-Computern ist es möglich, als Speichergerät einen normalen Musikkassettenrecorder anzuschließen. Während am Computer eine achtpolige Buchse zur Verbindung an den Kassettenrecorder vorgegeben ist, mündet das dazu lieferbare Kabel meist in drei Klinkenstecker. Zwei Klinkenstecker haben einen Durchmesser von 3.5 mm und sind meist unterschiedlich farblich gekennzeichnet (rot und weiß), während der dritte Klinkenstecker lediglich einen Durchmesser von 1.5 mm hat.

So schön es auf den ersten Blick erscheint, daß man einen normalen Musikkassettenrecorder an seinen MSX-Computer anschließen kann, so ärgerlich ist es dann doch, wenn man die gerade beschriebenen - nicht unbedingt an jeden Recorder anzuschließenden - Stecker sieht. Schnell ist die Entscheidung getroffen und man kauft sich dann doch den speziell dazu angebotenen Recorder der Computerfirma. So unfolgerichtig dies auch scheinen mag, es ist oft der letzte aber dann auch der richtige Ausweg. Aus diesem Grund müssen wir uns erst einmal über die Funktion der drei Klinkenstecker informieren und uns fragen: Warum hat man nicht ganz einfach einen normalen 5-Pol-DIN-Stecker am Computer verwendet?

Erst einmal zu den drei Klinkensteckern: der kleinere Stecker dient zum An- und Ausschalten des angeschlossenen Recorders (soweit diese sogenannte Remote-Funktion überhaupt vom Recorder angeboten wird). Hierbei handelt es sich zwar nicht um die Auslösung einer STOP-Funktion am Recorder, sondern in Wirklichkeit wird 'nur' die Stromzufuhr zur weiteren Wiedergabe bzw. Aufnahme gesperrt.

Der Sinn dieser Funktion ist folgender: Jedes Programm, das Sie auf Kassette abspeichern, muß zum Einladen in den Computer wiedergefunden werden. Dabei muß man wissen, daß das Programm von Anfang an geladen werden muß und der Ladevorgang nicht irgendwo in der Mitte beginnen darf. Haben Sie nun mehrere Programme auf einer Kassettenseite abgespeichert und wollen Programm Nummer 1 einlesen, spulen Sie das Band bis zum Anfang zurück und beginnen anschließend mit dem Ladeprozeß (indem Sie die PLAY-Taste am Recorder drücken und am Computer den Befehl CLOAD eingeben).

Kurze Zeit später erhalten Sie von Ihrem MSX-Computer die Bestätigung auf dem Bildschirm ausgegeben, daß er das Programm gefunden hat. Schließlich ist das Programm geladen und Sie können nun dasselbige mit dem BASIC-Kommando RUN starten.

Ihr Computer hat es gemerkt, als das Programm zu Ende geladen war und gab augenblicklich das entsprechende STOP-Kommando an den Recorderanschluß weiter. Dieses STOP-Signal wird nun durch das Kabel in den kleineren Klinkenstecker weitergegeben und veranlaßt den Recorder zum augenblicklichen Anhalten.

Was passiert, wenn der Recorder weiterläuft, weil er nicht über diese Klinkenbuchse verfügt? Vorerst nichts (das nächste Programm wird so lange nicht in den Speicher geladen, bis Sie erneut das BASIC-Ladekommando CLOAD erteilen), aber wollen Sie einige Zeit später Programm Nummer 2 laden, müssen Sie das Band eventuell wieder an die gewünschte Stelle zurückspulen. Hätten Sie den kleineren Klinkenstecker mit dem Recorder verbunden, würde das nächste CLOAD-Kommando den Recorder halt erst daraufhin wieder in Gang setzen und Programm Nummer 2 einladen.

Ein weiteres Beispiel, um den Sinn des kleineren Klinkensteckers zu demonstrieren: Sie haben sehr viele Daten auf Kassette abgespeichert, aus denen Ihr MSX-Computer nach von Ihnen vorgegebenen Kriterien einige wenige - vielleicht auch nur eins - heraussuchen soll. Da dieser Suchvorgang ziemlich lange dauert - ein mehrmaliges Einladen nacheinander, da der Computerspeicher nicht gleichzeitig alle Daten aufehmen kann - entscheiden Sie sich dazu, daß Ihr MSX-Computer diese Arbeit eigenständig in der Nacht für Sie durchführen soll.

Auch hier können Sie erst dann den Computer diese Arbeit angehen lassen, wenn von ihm der Recorder ferngesteuert werden kann (mit Hilfe des besagten verbundenen 1.5 mm Klinkensteckers), denn: ist der Speicher zum ersten Mal mit Daten gefüllt, beginnt das Programm mit dem Suchvorgang, der zumindest einige Sekunden dauern kann. Ohne die Computerfernsteuerung würde Ihr Recorder weiterlaufen und somit vielleicht den gesamten nächsten Datensatz überspringen. Mit der Fernsteuerung aber läuft der Recorder erst dann weiter, wenn Ihr MSX-Computer wieder aufnahmefähig für weitere Daten ist.

Haben Sie keine Anschlußmöglichkeit für das Fernsteuerkabel an Ihrem Recorder, müssen Sie aus den oben genannten Gründen genau achtgeben, wann der Computer seinen Datensatz bzw. sein Programm gerade eingeladen hat und dann sofort die PAUSE- oder STOP-Taste am Recorder betätigen.

Nun zu den zwei anderen Kabeln, die über 3.5 mm Klinkenstecker verfügen und in die entsprechenden Buchsen des Recorders eingestöpselt werden sollen: ein Stecker (meist der weiße) gehört in die EAR- oder die Lautsprecherbuchse Ihres Recorders, während Stecker 2 (meist rot) in die MIC-Buchse hineinpaßt.

Warum konnte man nicht einen genormten 5-Pol-DIN-Anschluß verwenden, der doch Aufnahme und Wiedergabe in einem Anschluß vereinigt?

Dies hat folgenden Grund: Würden Sie Aufnahmen mit einem solchen Standardkabel vornehmen, gäbe es sicherlich kein Problem; der Recorder würde die Aufnahme automatisch aussteuern (oder falls nötig steuern Sie selbst aus, wobei der Zeiger in jedem Fall bereits leicht in den roten Bereich hineinragen sollte) und das Programm würde so überspielt werden.

Bei der Wiedergabe über das DIN-Kabel hätten Sie aber ganz sicherlich mit Schwierigkeiten zu rechnen, da für Ihren MSX-Computer Lautstärke nicht gleich Lautstärke ist. Mit dem DIN-Kabel würden Sie nämlich eine feste Lautstärke (da ohne zwischengeschalteten Verstärker erzeugt) in den Computer einspeisen, die zum Lesen des Programms im Normalfall nie aureicht. Also müßten Sie zusätzlich zum DIN-Kabel noch mindestens ein Kabel an den Lautsprecher- oder Kopfhörerausgang Ihres Kassettenrecorders anschließen ... und schon hätten wir auch hier den Kabelsalat.

Deshalb gleich von vornherein dieses Kabel mit den 3 Klinkensteckern an der Anschlußseite zum Recorder hin: der kleinere Stecker zur Fernbedienung (Remote), die beiden größeren für Aufnahme und Wiedergabe.

Seien Sie aber vorsichtig, wenn Sie alle Stecker in den Buchsen Ihres Recorders für ewig und immer verankern wollen: es ist nicht ausgeschlossen, daß Ihr Recorder vielleicht ein Echosignal von der EAR-Buchse erhält, wenn Sie aufzeichnen. In diesem Fall würde es Probleme geben, selbst abgespeicherte Programme wieder in den MSX-Computer einzulesen. Deshalb: gibt es Probleme, selbst abgespeicherte Programme wieder in Ihren MSX-Computer einzulesen, ziehen Sie bei der nächsten Abspeicherung vorher das EAR-Kabel einfach aus der Recorderbuchse heraus.

Vielleicht verstehen Sie mich nun besser, wenn ich den Kauf eines speziell auf Ihren Computer abgestimmten Recorders empfahl. Schließlich kann es nämlich auch noch passieren, daß Sie zwar alle Vorbedingungen rein äußerlich mit Ihrem Recorder erfüllen können, aber die Ausgangsleistung über die EAR-Buchse zu schwach ist. Auch dann können Sie mit solch einem Recorder für die Abspeicherung/Einladung von Programmen nichts anfangen.

Noch 2 Dinge:

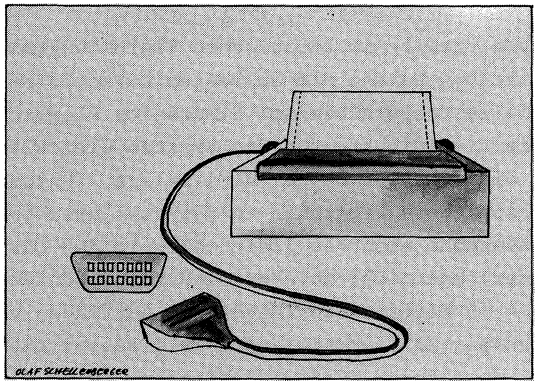
1. Wir sprachen bereits im technischen Einführungskapitel über die Bandqualitäten: Nehmen Sie am besten kurze Ferro-Markenbänder oder spezielle Computerbänder. Einerseits klemmen Markenbänder so gut wie nie (wichtig für den exakten Gleichlauf), andererseits ist ein Computer keine HiFi-Anlage (Chromdioxid- und Metallbänder speichern Programme in der von Ihrem Computer verwendeten Frequenzhöhe nicht so gleichmäßig in der Lautstärke ab wie Ferrobänder) und nicht zuletzt sind kurze Bänder (C10 = 10 Minuten-, C15 = 15 Minuten Bandlänge) zum Auffinden von Programmen oder Daten per Zählwerk in jedem Fall besser geeignet, als 1- bzw. 2-Stunden-Bänder (C60 = 60 Minuten-, C120 = 120 Minuten Bandlänge).

2. Was speichert Ihr MSX-Computer denn nun eigentlich auf Band ab? Sie erinnern sich noch an unser technisches Einführungskapitel? Dort sprachen wir über die Dummheit der Rechner, denn sie verstehen nur zwei Dinge: aus und an oder in Zahlen ausgedrückt 0 und 1.

Wenn Sie auf Kassette nun ein Programm abspeichern, wandelt Ihr Rechner die Speicherinhalte in schrill klingende Piepstöne um, wobei der tiefere Ton für eine 0, der höhere jedoch für eine 1 steht. Nun werden die Buchstaben, Zahlen und Zeichen jeweils in 8 dieser Piepstöne (in einer vorher fest bestimmten Kombination) umgewandelt und so auf Kassette abgespeichert.

Auf den ersten Blick ein komplizierter Vorgang, jedoch denken Sie daran, daß Ihr MSX-Computer nicht nur nach diesem Zahlsystem (sie erinnern sich vielleicht noch an seinen Namen: Binärsystem) rechnen kann, sondern dies auch noch unheimlich schnell tut (bis zu 100000mal pro Sekunde). Somit ist es für ihn ein Kinderspiel, 1200 oder gar 2400 dieser Piepstöne innerhalb von einer Sekunde auf Tonband zu übertragen (auch darüber hatten wir bereits vorhin gesprochen: 1200 Informationen pro Sekunde d.h. 1200 Baud). Nun aber zu weiteren Anschlüssen an Ihrem MSX-Computer.

3. Der Centronics-Parallel-Druckeranschluß



Für den Anschluß eines Druckers ist ein sogenannter Centronics-Parallel-Port in die meisten MSX-Computer eingebaut.

Daß Centronics der Firmenname der Entwicklungsfirma dieses Anschlusses ist, sagte ich bereits im technischen Einführungskapitel (übrigens auch heute noch eine große Druckerfirma). Was heißt aber Parallelanschluß?

Wieder müssen wir hierzu auf das Fundament der Computertechnik zurückgreifen: Ihr MSX-Computer arbeitet wie jeder andere Computer ebenso mit zwei Zuständen - aus und an oder in Zahlen ausgedrückt 0 und 1. Sie wären aber nicht glücklich, wenn Sie mit Computer und Drucker nur diese zwei Zustände ausgeben könnten.

Damit Sie auch andere Zahlen und Zeichen verwenden dürfen, verschlüsselt Ihr MSX-Computer mit Hilfe von acht An- bzw. Auszuständen z.B. jeden Buchstaben unseres Alphabets, jede Ziffer zwischen 0 und 9 sowie viele andere Zeichen, die wir durch einen Tastenandruck auf dem Bildschirm ausgeben können.

Die Ausgabe an den Drucker erfolgt genauso: der Computer gibt über seinen Drucker- bzw. in diesem Fall den Centronicsanschluß für jedes zu druckende Zeichen eine Kombination von 8 Nullen bzw. Einsen aus (codieren), die der Drucker wieder zurückverwandelt (decodieren) und somit schließlich das gewünschte Zeichen zu Papier bringen kann.

Ihr paralleler Centronicsanschluß trägt die Bezeichnung 'parallel', weil hier gleichzeitig alle 8 Zustände an den Drucker übermittelt werden und nicht, wie beim Tonband oder bei vielen anderen Computeranschlüssen, Zustand für Zustand nacheinander (im Gegensatz zur parallelen Datenübertragung spricht man im Fall einer Datenüberteragung in Serie - Zustand nach Zustand - von einer seriellen Datenübertragung).

Also kann man mit Fug und Recht behaupten: Ihr paralleler Centronicsanschluß ermöglicht eine sehr schnelle Datenübertragung zwischen Computer und Drucker (nicht zuletzt haben deshalb auch alle größeren Personalcomputer Centronicsanschlüsse - nur dort sieht die Buchse am Computer meist ein klein wenig anders aus).

Kurz noch etwas dazu, welche Druckertypen Sie denn hier anschließen können: Im technischen Kapitel sprach ich bereits über die Nadel- oder Matrixdrucker, die heutzutage Meistverkauften. Hier wird jeder Buchstabe, jedes Zeichen, jede Zahl von meist 8 Nadeln achtmal hintereinander in der jeweils übermittelten Folge zu Papier gebracht.

Sicherlich ist Ihnen schon klar, daß hinter diesem Nadelandrucksystem wieder die An-Aus-Zustandverschlüsselung verborgen liegt. Bei diesem System werden zwar nicht immer sehr saubere Buchstabenabbilder auf dem Papier erzeugt wie bei einer Schreibmaschine, dafür aber können Sie hier auch mit einer Druckgeschwindigkeit von meist 50 Zeichen pro Sekunde, oftmals viel mehr, rechnen. Dies hängt mit der geringen mechanischen Beanspruchung des Zurückholens bzw. Andrückens der Nadeln auf dem Papier zusammen. Mit einer gewönlichen Schreibmaschine oder auch einem Typenraddrucker (der ähnlich wie eine Schreibmaschine arbeitet) wäre diese Geschwindigkeit nicht zu erreichen (es sei denn, Sie bezahlen dafür etliche 1000 DM), da sich die Typen bei dieser Geschwindigkeit schon in der ersten Sekunde verfangen würden - so schnell könnten Sie überhaupt nicht in ihre Ausgangsstellung zurückgelangen.

Neben dem Nadel- und dem Typenraddrucker gibt es noch Tintenstrahldrucker, Thermodrucker und Laserdrucker, auf die ich kurz eingehen will.

Der Tintenstrahldrucker arbeitet im Prinzip ähnlich wie der Nadeldrucker, nur werden hier anstelle der Nadeln winzige Düsen verwendet, durch die mit Hilfe eines aufwendigen Mechanismus feinste Tintentröpfchen punktweise (!) auf das eingespannte Papier gespritzt werden.

Vorteil dieser recht teuren Drucker: Sie können kaum etwas beim Andruck hören (während ein Matrix- oder gar ein Typenraddrucker manchmal wie ein Hämmerwerk erklingt). Nachteil der Tintenstrahldrucker: Sie können keine Durchschläge erzeugen, denn so hart treffen die Tintenröpfchen nicht auf das Papier auf.

Die Thermodrucker arbeiten noch eine Idee leiser als die Tintenstrahldrucker. Der Schreibkopf erzeugt in der Form des jeweils auszugebenden Buchstabens oder Zeichens Wärme, die auf ein entsprechend empfindliches Spezialpapier übertragen wird. Dieses wärmeempfindliche Spezialpapier hat nun die Eigenschaft, daß überall dort, wo Wärme auf das Papier auftrifft, eine Farbe angezeigt wird (je nach Art des Papiers kann das Ergebnis blau, schwarz oder auch rot aussehen).

Auch eine interessante und recht preisgünstige Technik. Jedoch bedenken Sie, daß das Spezialpapier sehr teuer ist - wenn Sie häufig drucken, wird dann Ihr eigentlich so günstiger Thermodrucker von Tag zu Tag teurer.

Schließlich noch der Laserdrucker: Hier werden ganze Seiten meist innerhalb von einer Sekunde mit Hilfe der modernen Lasertechnik erzeugt.

Sicherlich, für größere Versicherungsunternehmen lohnt sich diese Anschaffung im Werte von mehreren 10000 DM. Für Sie als Einsteiger sind diese Drucker aber vorerst sicherlich noch nicht ganz so wichtig.

Wir haben festgestellt, daß wir an unseren MSX-Computer durch den Standard-Centronics-Parallelausgang mehr oder weniger beliebige Drucker anschließen können. Doch damit ist es noch nicht getan.

Es gibt spezielle MSX-Drucker auf dem Markt, die zwar genauso arbeiten wie jeder andere Drucker auch, der Unterschied liegt jedoch in einem wichtigen Detail verborgen: dem Zeichensatz. Wollen Sie nur normale Texte ausdrucken (allerdings ohne deutsche Umlaute), reicht Ihnen jeder x-beliebige Centronicsdrucker, den Sie an den Druckausgang Ihres MSX-Computers anschließen können.

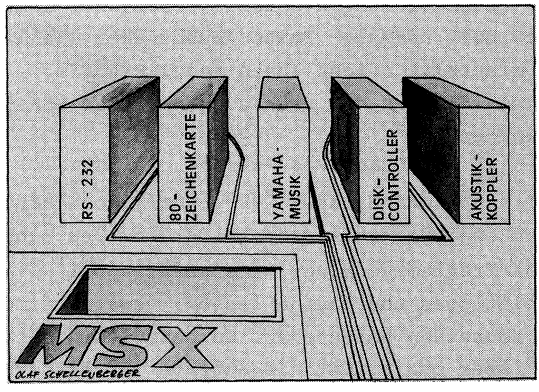
Wollen Sie auf einem Nicht-MSX-Drucker jedoch deutsche Umlaute ausgeben, müssen Sie vorher das Programm 'Deutsche Umlaute' aus meiner 'MSX-Programmsammlung' (ebenfalls bei DATA BECKER erschienen) eintippen und laufen lassen; zwar befinden sch dann die deutschen Umlaute auf Sonderzeichentasten Ihres MSX-Computers, aber nun stimmt die Ausgabe auf Bildschirm und Standard-Centronics-Drucker wieder überein.

Kaufen Sie sich jedoch einen speziellen MSX-Drucker, können Sie nicht nur die auf Ihrer MSX-Tastatur eingegebenen deutschen Umlaute (samt 'ß') auf dem angeschlossenen Drucker direkt ausgeben, Sie können zusätzlich sämtliche anderen Computerzeichen (wie z.B. die vielen Grafiksymbole, die Sie mit der GRAPH-Taste auf Ihrem Computerbildschirm hervorzaubern können) auch auf den Drucker ausgeben.

Gerade für Programmierer, die diese Zeichen in Ihre Listings einbauen, ist somit ein spezieller MSX-Drucker dringend notwendig. Vergessen Sie aber nicht meinen Hinweis von vorhin: Haben Sie bereits einen normalen Centronicsdrucker (oder können Sie einen besonders günstig erwerben), können Sie diesen auch verwenden, vorausgesetzt, Sie geben sich damit zufrieden, nur Buchstaben, Textzeichen und Zahlen (aber keine speziellen MSX-Grafikzeichen) auf Ihrem Drucker ausgeben zu können.

Soviel zum Thema Drucker und Centronics-Parallel-Anschluß.

4. Der Anschluß für Cartridges, auch Expansions- oder Erweiterungsport genannt



Schließlich verfügen alle MSX-Computer noch über ein bzw. zwei Expansionanschlüsse, auch meist als Cartridgeports bezeichnet. Diese(r) Anschluß(üsse) befindet sich meist mit einer Klappe versehen auf der Oberseite oder an der Rückseite Ihres MSX-Computers.

Die Bezeichnung Cartridge will ich zuerst erklären: Ein Cartridge ist ein flaches Plastikkästchen, in dem eine kleine Platine mit einem ROM (ein Nur-Lese-Speicher - siehe im technischen Einführungskapitel) steckt. Meist werden Spiele auf Cartridges als Programmspeicher den Kassetten vorgezogen, denn: Sie stecken ein Cartridge lediglich in den Cartridgeanschluß Ihres Computers herein, Sekunden später ist das Programm zum Spielen bereit (im Gegensatz zu einem Kassettenprogramm, bei dem Sie zumindest eine Vielzahl von Sekunden, manchmal auch mehrere Minuten warten müssen, bis Sie mit dem Spiel beginnen können).

Aber nicht nur für die Spielerei ist (sind) der (die) Cartridgeport(s) vorgesehen. Viel wichtiger ist es, daß der Expansionsport eigentlich für sämtliche Erweiterungen Ihres MSX-Compters zuständig ist, z.B. können Sie in diesen Anschluß bei MSX-Computern, die noch keinen Centronics-Parallel-Anschluß ihr eigen nennen, ein Cartridge einstecken, aus dem ein Centronicsanschluß herausgeführt wird (vielleicht sogar gleich mit Kabel).

Wollen Sie 80 Zeichen pro Zeile auf Ihrem MSX-Computer darstellen, können Sie hier eine sogenannte 80-Zeichenkarte einstecken, und schon können Sie kommerziell mit Ihrem Computer zu arbeiten beginnen (eigentlich sind nämlich 80 Zeichen pro Zeile für eine kommerzielle Anwendung unabdingbar).

Wollen Sie später einen Akustikkoppler an Ihren MSX-Computer anschließen (mit diesem Gerät können Sie per Telefon Ihre Programme und Daten in Piepstönen Zustand für Zustand umwandeln - siehe Kassettenrecorder - und an Freunde, Bekannte oder die eigene Firma weiterleiten bzw. von diesen auch wieder empfangen), müssen Sie eine sogenannte RS-232-Schnittstelle mit Hilfe eines entsprechenden Cartridges im Expansionsport erzeugen (der serielle RS-232-Anschluß ist übrigens ähnlich wie der Centronicsanschluß ein Standard auch bei den großen Personalcomputern).

Die Firma Yamaha bietet weiterhin etwas sehr interessantes für Musikfreunde an: Sie stecken in den Cartridgeport ein Plastikkästchen, an welches Sie eine Orgel anschließen können. Nicht nur das: im Kästchen selbst ist ein achtstimmiger Synthesizer eingebaut, mit dem Sie nicht nur jedbeliebiges Geräusch erzeugen können (z.B. Vogelgezwitscher oder Autohupen), sondern dies auch noch gleichzeitig mit 7 anderen Geräuschen gleichzeitig erklingen lassen können.

Es gibt gar schon Programme für dieses Modul, mit denen Sie ein ganzes Orchester auf Ihrem MSX-Computer automatisch nach vorheriger Programmierung (Komposition) spielen lassen können. Der MSX-Computer an und für sich dient hierbei nicht nur als Terminal (zur Eingabe), Sie können auch die komponierten Klänge z.B. auf Kassette abspeichern oder gar Ihre gesamten Partituren (wenn es so weit einmal kommen sollte) auf den am Centronicsport angeschlossenen Drucker ausgeben.

Eine weitere wichtige Anschlußmöglichkeit am Cartridgeport ist die Diskettenstation: Die Diskettenstation erfüllt eine ähnliche Funktion wie der Kassettenrecorder. Sie dient der Datenspeicherung bzw. dem Einladen von Datenmengen.

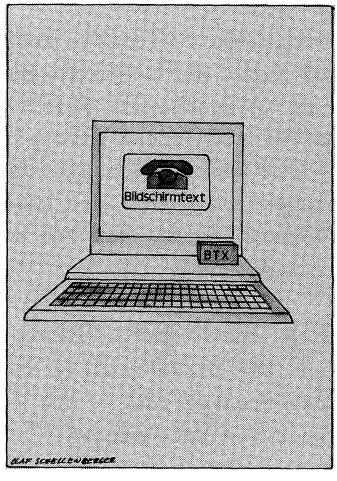
Im Gegensatz zum Tonband in der Kassette wird jedoch hier eine runde magnetische Scheibe zur Datenspeicherung verwendet. Diese Scheibe oder auch Diskette genannt, dreht sich in der Station sehr schnell, so daß ein Magnetkopf in Sekundenschnelle jedes Programm auf der Diskette finden kann.

Wenn auch Vergleiche immer ein wenig hinken, will ich hier doch einen anbringen: Sie können die Diskettenstation mit einer Schallplatte oder CD-Platte vergleichen: wollen Sie ein bestimmtes Musikstück hören, setzen Sie den Saphier genau dort auf die Platte auf, wo sich der gewünschte Musiktitel befindet. Dies geht sehr schnell! Wollen Sie allerdings ein bestimmtes Musikstück auf der Kassette ausfindig machen, müssen Sie zuerst aufs Zählwerk schauen, dann vor- und zurückspulen - ein lästiger und manchmal lang andauernder Vorgang.

Im Vergleich zu diesem musikalischen Beispiel läßt sich das Datenauffinden auf der Kassette (= Musikkassette) im Gegensatz zu der Diskette (= Schallplatte oder CD-Disk) genauso beschreiben.

Nun noch zu ein paar Erweiterungen, die zwar bisher auf Messen gezeigt wurden aber noch nicht offiziell zum Zeitpunkt, während ich diese Zeilen schreibe, auf dem deutschen Markt erhältlich sind:

1. BTX-Modul:



BTX ist die Abkürzung für Bildschirmtext und repräsentiert das neueste Angebot der Deutschen Bundespost zur Steigerung der Kommunikationsmöglichkeiten auch und insbesondere für Otto Normalverbraucher.

Um an diesem neuen Angebot der Post teilnehmen zu können, brauchen Sie einen Fernseher mit eingebautem BTX-Dekoder, einen Telefonanschluß und ein Post-BTX-Modem. Mit Hilfe der Fernbedienung Ihres Fernsehers ist es somit möglich, z.B. die aktuellen Informationsseiten Ihrer Tageszeitung auf dem Bildschirm erscheinen zu lassen oder Prospekte von MSX-Firmen zu bestellen oder das eigene Konto via Bildschirm zu führen oder ... Die Möglichkeiten sind fast unbegrenzt groß.

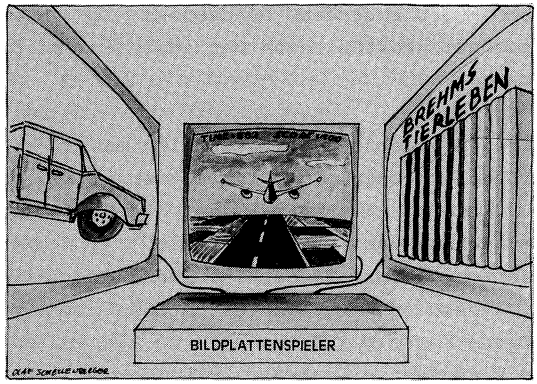
Die Auswahl der Seiten funktioniert wie gesagt mit Ihrer Fernsteuerung in Form von Zahlen. Wollen Sie allerdings an einen Hersteller eine bestimmte Frage über Bildschirmtext stellen oder einem Ihrer Verwandten (der natürlich auch an das BTX-Netz angeschlossen sein muß) einen Brief über BTX schreiben, müssen Sie, der Sie nur mit einer reinen Zahlenfernbedienung ausgestattet sind, passen.

Wollen Sie dies jedoch tun, müssen Sie sich für einige 100 DM eine spezielle BTX-Tastatur dazukaufen, denn die BTX-Fernseher verfügen bereits großenteils über einen Tastaturanschluß.

Nicht so bei Ihnen: Sie besitzen mit Ihrem MSX-Computer ja bereits eine Tastatur. In den Cartridgeport wird einfach das entsprechende BTX-Modul eingesteckt und sofort können Sie mit der Eingabe von der MSX-Tastatur beginnen.

Vielleicht wird es später auch möglich sein, die gesehenen BTX-Seiten mit Hilfe Ihres MSX-Computers auf Kassette oder auf Diskette abzuspeichern. Wie wäre es schließlich, wenn Sie die gesehenen Bilder - wenn auch vielleicht vorerst nur die Texte - direkt auf einen an Ihren MSX-Computer angeschlossenen Drucker ausgeben könnten!? Neben den vielen Möglichkeiten zum Einsatz des MSX-Computers zu Hause oder im Geschäft wäre somit die Kopplung mit BTX sicherlich auch für Sie sehr interessant.

2. Anschluß eines Bildplattenspielers:



Wenn Sie sich die Bildplattenspieler einmal etwas näher anschauen, werden Sie feststellen, daß die Bedienung zur Auswahl einzelner Bilder zwar recht kompliziert vonstatten geht (wozu braucht man das auch?) aber es ist möglich.

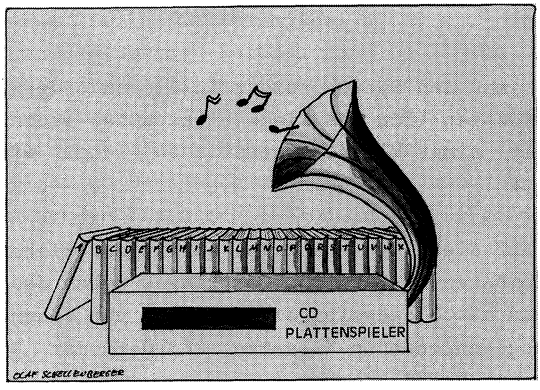
Auch hier können die MSX-Computer über den Cartridgeslot in Aktion treten: so könnte man beispielsweise in einem Autohaus dem Kunden mit Hilfe eines darauf abgestimmten Programms ein auf der Bildplatte abgespeichertes Dia des gewünschten Autotyps in Bruchteilen einer Sekunde auf dem Fernsehbildschirm erscheinen lassen.

Nicht nur der Autotyp wäre dann der gewünschte, auch Karosseriefarbe und besondere Ausstattungswünsche würden mitberücksichtigt werden. Technisch gesehen wäre diese Steuerungsaufgabe über MSX-Computer kein großes Problem, nur man müßte halt von allen möglichen Autotypen in allen Farbtönen die Dias erst einmal auf einer Bildplatte abspeichern, ein recht kostspieliges Verfahren, das sich erst z.B. für sehr große Autofirmen bezahlt machen würde.

Eine volkstümlichere Anwendung eines an Ihren MSX-Computer angeschlossenen Bildplattenspielers wäre vielleicht Brehm's Tierleben: Sie bestimmen, welche Art von Tier Sie suchen (Säugetier, lebt in Nordamerika ...); nachdem Ihr MSX-Computer die Daten verarbeitet hat, steuert er den Bildplattenspieler an und wirft das gewünschte Dia in leuchtenden Farben auf den Bildschirm.

Schließlich eine dritte Anwendungsmöglichkeit für Bildplattenspieler am MSX-Computer: Sie erfreuen sich an einem sogenannten Abenteuerspiel (ein Spiel, das aus einer buchähnlichen Handlung besteht). Immer dann, wenn eine interessante Aktion eintritt, wird Ihnen das entsprechende Bild von der Bildplatte angezeigt. Nicht nur das. Es könnte auch eine Sequenz von Bildern = ein Filmstück angezeigt werden, in dem Sie vielleicht mit Joystick bewaffnet die weitere Bildfolge selbst bestimmen können. Ein realistisches Telespiel wie es kaum besser sein könnte.

3. Anschluß eines CD-Plattenspielers



Ähnlich könnten Sie mit einem CD-Plattenspieler, angeschlossen an Ihren MSX-Computer, arbeiten: Nicht daß Sie die gewünschten Tonsequenzen ansteuern wollten (vielleicht für Musiker ein interessanter Einsatz), es wäre doch denkbar, daß man solch eine CD-Platte ähnlich wie eine Diskette als Datenspeicher verwendet.

Während Sie auf einer Diskette 'nur' 150 DIN-A-4-Seiten abspeichern können, wären das auf einer CD-Platte 250000 DIN-A-4-Seiten.

Wie würde es Ihnen z.B. gefallen, wenn Sie demnächst über Ihren MSX-Computer in Verbindng zum CD-Plattenspieler mal schnell im Brockhaus nachschlagen könnten?

Damit wir uns recht verstehen: Nicht im MSX-Computer-RAM wären diese vielen Informationen abgespeichert, sondern auf der CD-Platte (bzw. beim Bildplattenspieler auf einer Bildplatte).

Ihr MSX-Computer steuert 'nur' das Auffinden der Daten; er liest alles in Sekundenschnelle durch bzw. verfügt über entsprechende Dateiordnungssysteme und steuert somit schließlich auch die Ausgabe auf dem Bildschirm erst dann an, wenn auch wahrlich die gewünschten Daten verfügbar sind.

4. Ein Cartridge, das in den Cartridgeslot geschoben wird, aber in seinem Inneren RAM enthält

Hiermit wäre es möglich, Speichererweiterungen an Ihren MSX-Computer anzuschließen, um den verfügbaren Speicherplatz zu vergrößern.

Auch könnte man den Speicherplatz in diesen sogenannten RAM-Erweiterungsmodulen (für RAM siehe bitte unter technische Informationen) mit einer kleinen Batterie so lange wie gewünscht abrufbar halten.

Man könnte auf diese Art und Weise selbstgeschriebene Programme oder gar Stammdaten in Form von Cartridges neben den MSX-Computer legen und hätte dieselbigen innerhalb von Sekundenbruchteilen (ohne angeschlossenen Kassettenrecorder oder Diskettenstation) zur freien Verfügung.

Das soll erst einmal reichen, um Ihnen die Notwendigkeit und den Sinn des Erweiterungsanschlusses oder Cartridgeports an MSX-Computern deutlich zu machen. Die zuletzt aufgeführten vier Beispiele sind nicht aus der Luft gegriffen, sondern befinden sich z.Z. bei mehreren MSX-Firmen im Entwicklungsstadium.

Noch einmal: dies sollten nur Beispiele sein. Es gibt noch viel mehr Möglichkeiten, diesen Anschluß sinnvoll zu nutzen (Videorecorder steuern, Beleuchtung im Haus beim Verreisen an- und ausschalten ...).

Die weiteren Anschlüsse - abgesehen von Extravaganzen, über die Ihr MSX-Computer vielleicht allein auf dem Markt verfügt -, die jetzt noch bleiben, brauchen nicht weiter ausführlich erläutert zu werden:

An die TV- und/oder Monitorbuchsen können Sie Fernseher (meist auf Kanal 36 - am besten über AV-Kanal) und/oder Monitoren (im Gegensatz zum Fernseher ein ruhigeres Bild, da statt 25 Bilder pro Sekunde 50 bis 70 erzeugt werden) anschließen.

Schließlich ist da noch ein Stecker für die Stromzuführung, wobei Sie nur darauf achtgeben sollten, daß dieser Stecker immer fest in Ihren Computer eingeschoben ist - sonst ist bei unerwartetem Computerausfall = Löschen der Daten und Programme das Wehklagen sehr groß!

Vielleicht haben Sie in diesem Kapitel nicht nur erfahren, daß die vielen unscheinbaren Buchsen rund um Ihren MSX-Computer besondere Funktionen erfüllen, sondern auch ein wenig mehr darüber Kenntnis erworben, wie Sie Ihren MSX-Computer zukünftig sinnvoll einsetzen können. Einen ersten Schritt dahin, nämlich durch die richtige Bedienung der Tastatur und die ersten Befehlseingaben werden Sie anschließend in den folgenden Kapiteln kennenlernen.

BASIC besteht nicht nur aus englischen Vokabeln

Eigentlich ist der Titel dieses Kapitels falsch gewählt, denn er könnte Sie dazu veranlassen, die Hände von dem Erlernen einer Programmiersprache - wie sie Ihr MSX-Computer ja beherrscht - wegzulassen, halt die Flinte zu früh ins Korn zu werfen.

Ich trete hier als Mittler zwischen dem Computer und Ihnen, dem Lernwilligen, auf und versuche, in diesem Kapitel soweit die Scheu vor dem Computer bei Ihnen zu beseitigen, daß Sie mit Freude auch die weiteren Kapitel in diesem Buch lesen.

Bisher sind uns Sprachen nur als Kommunikationsmittel der verschiedenen Völker untereinander bekannt. Während der eine die englische Sprache beherrscht, ist es bei dem anderen die französische und bei einem dritten schließlich die deutsche Sprache. Es reicht allerdings nicht aus, nur die entsprechenden Vokabeln einer Fremdsprache zu lernen, man muß auch in der Lage sein, diese Vokabeln in den Regeln der Sprache richtig anwenden zu lernen; man muß neben den Vokabeln auch noch die Grammatik beherrschen.

In diesen Punkten unterscheidet sich das Erlernen einer Programmiersprache nicht sehr stark vom Erlernen einer gewöhnlichen Fremdsprache. Bei der Menge der zu lernenden Vokabeln und ihrer Bedeutungsfindung setzen allerdings die gravierenden Unterschiede zwischen Fremdsprache und Programmiersprache ein.

Zur Beherrschung des neuen Vokabulars einer Fremdsprache (fachspezifische Begriffe ausgenommen) müssen Sie zumindest 2000 Vokabeln für diesen sogenannten Grundwortschatz Ihr eigen nennen. Eine Menge Vokabeln, für deren Erlernen Sie im Normalfall Jahre brauchen. Bei diesen 2000 Vokabeln gibt es sicherlich auch einige, die mehrere Bedeutungen tragen (wie im Deuschen das Wort Bank: a) als Sitzgelegenheit b) als Sparkasse). Im Regelfall kann man jedoch festlegen, welche Bedeutung das eine von dem anderen Wort deutlich unterscheiden.

Anders bei einer Programmiersprache: Hier gilt es, durchschnittlich nur 100 bis 150 Vokabeln zu lernen. Da Sie aber mit diesem minimalen Wortschatz jedbeliebiges logisches Problem zu lösen in der Lage sein sollten, hat jede dieser Vokabeln eine Bedeutung, die nicht mit dem Begreifen einer fremdsprachlichen Vokabel verglichen werden kann.

Vielmehr haben Sie es beim Erlernen einer Programmiersprache mit Vokabeln zu tun, die nicht nur eine bestimmte Bedeutung im Namen tragen, sondern durch diese Bedeutung auch verschiedenste Funktionen in die Wege leiten können. Man kann die Vokabeln einer Programmiersprache am besten mit den in der Mathmatik verwendeten Fachtermini vergleichen: Schauen Sie sich folgende Zeichenreihe an und lesen Sie sie laut vor:

```
3+5=8
```

Sicherlich sprachen Sie aus:

```
'Drei plus fünf gleich acht'
```

oder so ähnlich. Was ist allerdings in Wirklichkeit der Inhalt dieses Satzes?

```
'Drei,Kreuzchen,fünf,Gleichheitszeichen,acht'
```

und nichts mehr.

Aus Ihren Erfahrungen heraus haben Sie allerdings nicht so gelesen, wie ich dies gerade geschildert habe. Sie haben vielmehr sofort aus der Struktur dieser Zeichenkombination erkannt, daß es sich hier nicht um eine unregelmäßige Folge von Zeichen handelt, sondern vielmehr um eine Aufgabe mit Lösung aus der Mathematik.

Sehen Sie hingegen eine Kette von Gleichheitszeichen unterhalb einer Textüberschrift, deuten Sie diese nicht mehr mathematisch sondern als Unterstreichung.

Während Sie bei einer Fremdsprache die Buchstabenkombination 'dog' mit dem deutschen Wort 'Hund' übersetzen und sonst nichts, haben im Gegensatz dazu mathematische Zeichen und auch alle Vokabeln einer Programmiersprache mehr als nur die reine Übersetzungsbedeutung. Vielmehr steckt hinter den 150 Programmiervokabeln eine Vielzahl von unterschiedlichen Bedeutungsinhalten, die in iher Aufaddition vielleicht schließlich an die Anzahl eines fremdsprachlichen Grundwortschatzes (2000 Vokabeln) ohne Schwierigeiten heranragen.

Hinter jeder Vokabel einer Programmiersprache steckt einerseits als Orientierung die direkte Übersetzung (z.B. PRINT = schreiben), andererseits eine bestimmte Grammatik (wie kann ich PRINT syntaktisch sinnvoll einsetzen) und schließlich eine Bedeutung (warum nehme ich PRINT und nicht INPUT-die Funktion des Befehls im Programmzusammenhang muß dem Anwender ebenso klar sein).

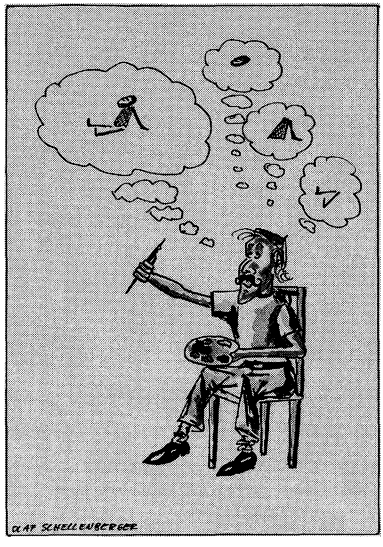
Genauso tragen die mathematischen Grundrechenarten nicht nur den Bedeutungsinhalt (wie plus bzw. minus), sondern wir müssen auch lernen, damit richtig umzugehen. Keine Angst: Genauso wie Sie in der Schule nicht gleich lernen, daß minus mal minus plus ergibt, werden wir die Programmiervokabeln ganz von unten an kennenlernen und nicht gleich in ihrer vollen Wirkungsstärke.

Wie wir vorhin feststellten, ist Ihr MSX-Computer in der Lage, mit Ihnen in seiner speziellen Programmiersprache zu plaudern. Die in Ihrem MSX-Computer bereits eingebaute Programmiersprache trägt den Namen BASIC (Abkürzung für Beginners all purpose symbolic instruction code, was so viel bedeutet wie: Programmiersprache für den Anfänger, mit der verschiedenste Funktionen erfüllt werden, so daß alle nur möglichen programmtechnisch allerdings auch definierbaren Problemfelder einer Lösung zugeführt werden können).

Sie müssen es sich so vorstellen: Ihr MSX-Computer hat das BASIC in seinem ROM gespeichert. Hier sind alle 150 Befehle in ihrer Buchstabenzusammensetzung wie ein Wörterbuch aneinandergereiht abgespeichert, jedoch verfügt jedes Wort zusätzlich über einen Seitenverweis d.h. an welcher Stelle im Speicher die gewünschte Funktionsbeschreibung der angewählten Vokabel zu finden ist.

Glücklicherweise ist diese Programmierspache bereits in Ihren MSX-Computer von vornherein eingebaut; wir brauchen uns halt nicht mehr darum zu kümmern, wie diese Funktionen dem Computer einstmals von den ROM-Programmierern verständlich gemacht wurde (erinnern wir uns an die Ursprünge der Computerei, müßten wir bei dieser Funktionsbeschreibung eigentlich an eine mehr oder minder sehr lange Folge von Aus- und Anzuständen bzw. von 0 und 1 denken).

Für den Computereinsteiger (und nicht nur für den!) ist somit bereits ein erster Schritt gemacht: wir brauchen also nicht mehr unserem MSX-Computer zu verstehen zu geben, wie er denken soll, sondern wir können kreativ tätig werden und sollten uns vielmehr darüber im klaren sein, was für Aufgaben der Computer mit dem bereis vorhandenen Vokabular denn ausführen soll. Wir besitzen also nicht nur Buntstifte und Papier, sondern auch schon eine Vielzahl von fertigen Formen, die wir zum Erstellen unserer Programme oder dem Beispiel folgend des Programmbildes mit verwenden dürfen.



Nun aber frisch ans Werk: Prüfen wir zuerst einmal, ob Ihr MSX-Computer seine Vokabeln auch gelernt hat: Geben Sie die Buchstabenkombination 'MSX' auf der Tastatur ein ... und nichts passiert.

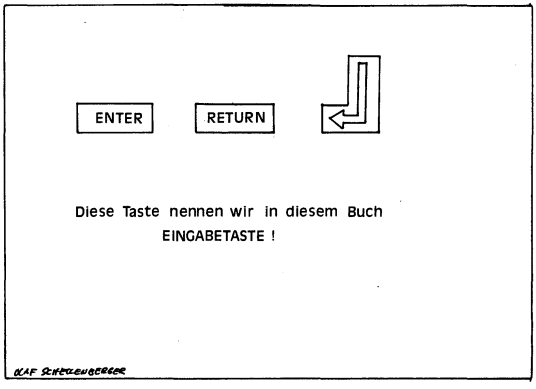
Hierbei müssen Sie zu differenzieren verstehen: In der in Ihren MSX-Computer eingebauten Programmiersprache BASIC ist es so, daß Sie zwei Dinge deutlich voneinander unterscheiden müssen:

1. Das Eingeben von Buchstaben oder Zeichen mit Hilfe der Tastatur. Das Endprodukt dieser Betätigung können Sie gleichzeitig auf dem Bildschirm mitverfolgen.

2. Das Eingeben der auf dem Bidschirm zu erkennenden Zeichenkombination in den Computer zwecks Verarbeitung. Zwar ist der Bildschirmspeicher auch bereits ein Teil des in Ihren MSX-Computer eingebauten RAMs, jedoch dient dieser Speicherbereich nicht direkt zur Verarbeitung, sondern lediglich zur unmittelbaren Wiedergabemöglichkeit oder Abbildung auf dem Bildschirm.

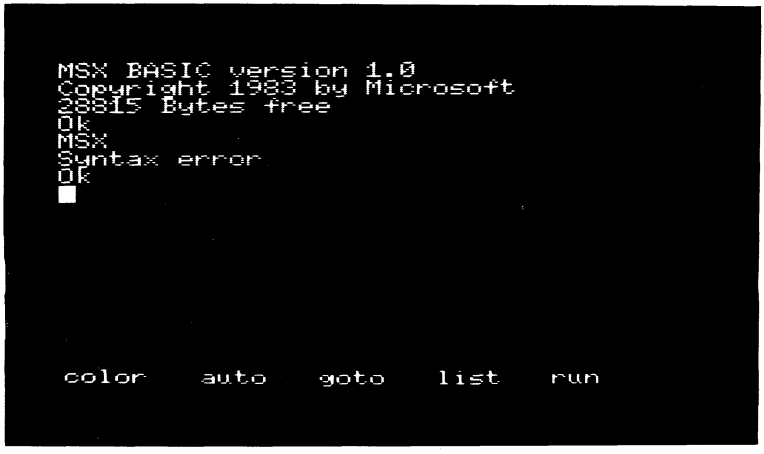
Sie können auch irgendwelche anderen Buchstaben- oder Zeichenkombinationen in Ihren MSX-Computer eingeben; das einzige, was hier passiert ist die gleichzeitige Bildschirmdarstellung, die Sie direkt Zeichen für Zeichen am Bildschirm sehen können.

Wollen Sie unsere gewählte Buchstabenkombination in Ihren MSX-Computer zur Bearbeitung eingeben, müssen Sie eine bei den meisten MSX-Computern etwas größere Taste auf der rechten Tastaturhälfte betätigen (bei einigen MSX-Computern trägt sie den Namen RETURN oder ENTER, manchmal ist hier auch nur ein Pfeil aufgetragen, der zuerst nach unten und dann nach links führt).



Die Bezeichnung ENTER sagt genau das aus, was wir machen wollen: wir sind bestrebt, diese Buchstaben zur Bearbeitung einzugeben. RETURN (zurückkehren) rührt vielmehr von der Wortkombination Carriage RETURN her. Diese Bezeichnung stammt von der Schreibmaschine her und bedeutet soviel wie: kehre mit der Schreibmarke an den Anfang der Zeile zurück aber setze die Eingabe gleichzeitig eine Zeile tiefer fort).

Drücken Sie bitte nun diese Eingabetaste (wir werden der Verständlichkeit halber den Namen Eingabetaste für die ENTER- bzw. RETURN-Taste in diesem Buch beibehalten). Ihr MSX-Computer zeigt sofort Reaktion; nicht nur, daß er (falls ein Fernseher mit Ton an Ihren MSX-Computer angeschlossen ist) einen BEEP-Ton erklingen läßt, er gibt auch eine Meldung aus: Synax error (was soviel bedeutet wie: Irrtum im Synax bzw. in der gewählten Vokabel - die kenne ich nicht).



Obgleich unser Computer die Bezeichnung MSX trägt, ist er nicht in der Lage, etwas mit dieser Vokabel anzufangen - MSX gehört nicht zu den 150 Ihrem Computer bekannten Vokabeln.

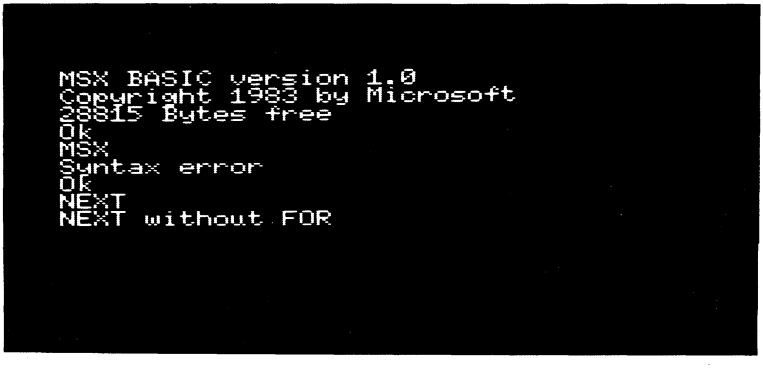
Tippen Sie ruhig ein paar andere englische Wörter auf gut Glück, vielleicht auch Ihren Namen oder den von anderen Personen, in den Computer ein. Fast bin ich mir sicher, daß Sie nur sehr selten eine andere Meldung von Ihrem MSX-Computer ausgegeben bekommen als 'Syntax error', nachdem Sie die Einabetaste betätigen.

'Deprimierend, wie klein der Wortschatz meines MSX-Rechners doch eigentlich ist' werden Sie vielleicht nun sagen. Zudem besitzt Ihr Computer nach der eintönigen Beantwortung Ihrer Eingabe auch jedes Mal noch die Frechheit, die für ihn zufriedenstellende Bezeichnung OK auf dem Bildschirm auszugeben.

Dazu gleich ein Hinweis, um diese Verurteilung augenblicklich zu entschärfen: es gibt Zeitpunkte, wo Sie zwar eine Eingabe auf der Tastatur vornehmen können, diese Eingabe aber gar nicht auf dem Bildschirm erscheint. In diesem Fall würde auch keine OK-Ausgabe erfolgen. OK bedeutet einfach gesagt, daß Ihr MSX-Computer nun wieder in der Lage ist, direkte Eingaben von Ihnen zur weiteren Verarbeitung in Empfang zu nehmen.

Um eine andere Meldung auf Ihre Eingabe hin zu bekommen als nur 'Syntax error', will ich Ihnen an dieser Stelle wenigsens eine der 150 Vokabeln des BASIC verraten: NEXT. Geben Sie also NEXT ein und drücken anschließend die Eingabetaste.

Haben Sie die vier Buchstaben NEXT richtig auf Ihrer Tastatur eingegeben und anschließend die Eingabetaste betätigt, erscheint nun auf dem Bildschirm die Mitteilung 'NEXT without FOR' (es wurde das Wörtchen NEXT verwendet, ohne daß vorher die dazu notwendige Vokabel FOR eingegeben wurde).



Zwar nicht mehr die Syntax Error-Fehlermeldng, aber nun troz der bekannten Vokabel eine andere Fehlermeldung. Sie sollten wissen, daß Ihr MSX-Computer nicht nur über diese zwei sondern über sehr viel mehr verschiedene Fehlermeldungen verfügt. Dies ist nur gut

so, denn so ist es für Sie einfacher, den gemachten Fehler zu erkennen und anschließend zu beseitigen (in der gerade eben ausgegebenen Fehlermeldung 'NEXT without FOR' haben Sie den Tip bekommen, zu dem alleinstehenden Wörtchen NEXT auch noch die Vokabel FOR einzugeben).

Es ist für Sie als Einsteiger nicht immer sehr einfach, diese kurzen englischen Fehlermeldungen zu verstehen. Aus diesem Grund habe ich in meinem Buch 'MSX Programmsammlung' (ebenfalls bei DATA BECKER erschienen) ein Programm mit dem Namen 'Ausführliche Errormeldungen' veröffentlicht. Nachdem Sie dieses Programm in Ihren MSX-Computer eingegeben haben, bekommen Sie bei auftretenden Fehlern Informationen zur Behebung der Fehler in der Länge von bis zu einer Bildschirmseite angezeigt.

Schauen wir uns nun einmal einen Befehl an und versuchen denselbigen in seinem gesamen Funktionsaufbau kennenzulernen: Der Befehl heißt PRINT.

Löschen Sie bitte den Bildschirm mit der CLS-Taste (SHIFT und CLS/HOME drücken), so daß uns bei der Arbeit keine Reste von vorherigen Notitzen stören. Die Schreibmarke müßte nun in der linken oberen Bildschirmecke stehen und blinken.

Zum Kennenlernen des MSX-BASIC-Kommandos PRINT wollen wir zwar alle Möglichkeiten besprechen, aber Schritt für Schritt von unten angefangen vorgehen.

1. Die Bedeutung des PRINT-Befehls ohne Zutaten

Geben Sie PRINT ein und drücken Sie anschließend die Eingabetaste. Vielleicht zum ersten Mal - sollten Sie vorher beim Eintippen von englischen Wörtern nur Fehlermeldungen wie 'Syntax error' erfahren haben - sehen Sie nun, daß Ihr MSX-Computer diese Handlung nicht mit einer Fehlermeldung quittiert hat, sondern lediglich das Wort Ok auf dem Bildschirm ausgibt.

Was hat der Befehl PRINT nun bewirkt? Drücken Sie hierzu noch einmal die Eingabetaste, ohne weitere Befehle einzugeben: die Bildschirmmarke ist lediglich um eine Zeile nach unten an den Anfang der nächsten Zeile gerutscht - sonst nichts. Im Gegensatz dazu die Eingabe PRINT: Hier wird nicht nur Ok auf dem Bildschirm angezeigt, sondern, wenn Sie genau hinsehen, wird zusätzlich auch noch eine Leerzeile produziert d.h. die Schreibmarke hat eine Zeile auf dem Bildschirm übersprungen.

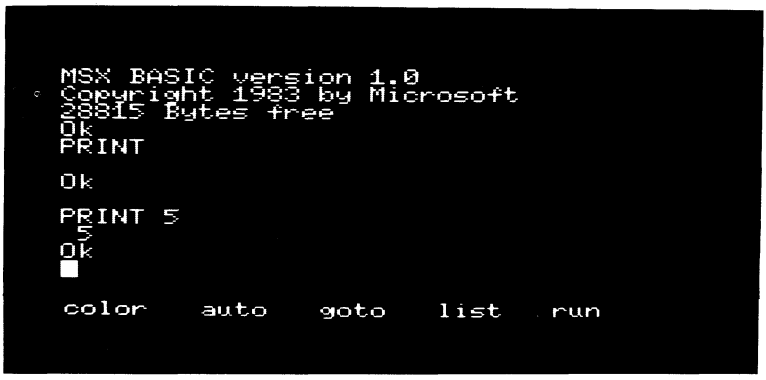
Wir können uns also merken: Gebrauchen wir den Befehl PRINT ohne irgendwelche weiteren Zutaten, so wird auf dem Bildschirm eine Leerzeile ausgegeben.

2. Die Bedeutung des Befehls PRINT mit einer Zahl

Bisher hat Ihr MSX-Computer mit dem Befehl PRINT keine besondere Aktion durchgeführt - es wurde lediglich eine Leerzeile auf dem Bildschirm ausgegeben. Eigentlich hat Ihr MSX-Computer ja auch richtig gehandelt: Sie haben ihm mitgeteilt, daß er schreiben soll (PRINT), aber nicht genauer definiert, was nun eigentlich auf dem Bildschirm geschehen soll.

Wir wollen nun eine Zahl auf dem Bildschirm ausgeben, Nichts leichter als das: Sie geben erneut das Wort PRINT auf der Tastatur ein, drücken aber noch nicht die Eingabetaste, sondern fügen zuvor noch mit der Leertaste ein Leerzeichen an und schreiben schließlich eine Zahl. Erst hiernach betätigen Sie erneut die Eingabetaste.

Ihr MSX-Computer hat die von Ihnen gewünschte Aktion ausgeführt; es wurde eine Zahl auf dem Bildschirm ausgegeben.



Sicherlich für Sie in diesem Zusammenhang auch keine Aktion, die Sie vom Hocker springen läßt. Aber noch sind wir nicht am Ende der Erläuterungen des Befehls PRINT angelangt.

Wir können also durch Gebrauch des Befehls PRINT und einer eingegebenen Zahl diese Zahl ein weiteres Mal (allerdings eine Zeile tiefer als die Eingabezahl) auf dem Bildschirm reproduzieren.

3. Der Befehl PRINT dient zur Ausgabe des Verknüpfungsergebnisses von zwei Zahlen

Wie sind Sie bisher bei Aufgaben vorgegangen, die Sie auf Ihrem Taschenrechner eintippen mußten? Erst die eine Zahl, dann die Verknüpfungsfunktion (z.B. + oder -), anschließend eine weitere Zahl und zuletzt das Gleichheitszeichen. Die Tastaturabfolge bei der Berechnung der Aufgabe 3*4 war demnach:

3 * 4 =

und die Ausgabe auf Ihrem Taschenechner lautete: 12.

Probieren Sie dies auch auf Ihrem MSX-Computer aus. Geben Sie bitte ein:

```
3*4=
```

Nach Drücken der Eingabetaste ist die Schreibmarke lediglich in die nächste Zeile gesprungen, kein Ergebnis weit und breit zu sehen.

Überlegen wir, warum dies nicht funktioniert hat: wir haben vorhin festgestellt, daß Ihr MSX-Computer bereits eine Programmiersprache eingebaut hat, damit er sich mit Ihnen unterhalten kann. Zu dieser Kommunikation beherrscht er ungefähr 150 Vokabeln wie z.B. PRINT zur Bildschirmausgabe.

Bei unserer gerade gescheiterten Rechenaufgabenstellung haben wir aber vergessen, dem MSX-Computer mitzuteilen, daß auch eine Ausgabe des Ergebnisses auf dem Bildschirm vorgenommen werden soll. Also darf unsere Eingabe nicht wie beim Taschenechner nur

```
3*4=
```

lauten, sondern vielmehr

```
PRINT 3*4=
```

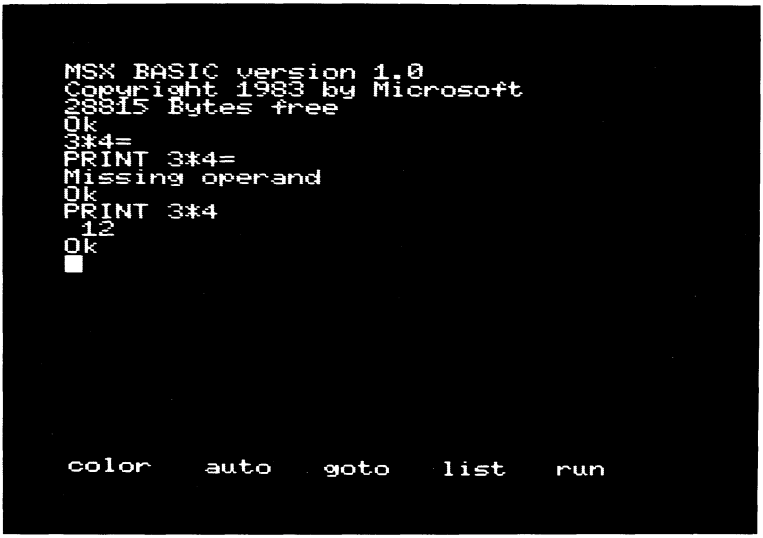
Wenn Sie diese Eingabe getätigt haben, erfolgt schon wieder eine unvorhergesehene Aktion: 'Missing operand' steht als Fehlermeldung unter Ihrer soeben erfolgten Eingabe.

Ich werde Ihnen im nächsten Kapitel, wo es um die Speicherplätze = Schubläden unseres Computers geht, erklären, warum dieser Fehler auftrat.

Sie können ihn beseitigen, indem Sie lediglich eingeben:

```
PRINT 3*4
```

Nun endlich erscheint das richtige Ergebnis auf dem Bildschirm: 12.



Sie können nun selbst einmal überprüfen, ob Ihr MSX-Computer genauso gut und genauso schnell arbeitet wie Ihr Taschenrechner. Ihnen stehen zu diesem Zweck natürlich sämtliche vier Grundrechenarten zur Verfügung wie bei Ihrem Taschenrechner auch: +, -, * und ... hier müssen Sie sich wieder an eine Änderung gewöhnen: Das Teilen führt Ihr MSX-Computer nicht wie bei uns in der Mathematik üblich mit dem Doppelpunkt durch, sondern vielmehr mit einem Querstrich (/). Wollen Sie also z.B. 6 geteilt durch 2 rechnen, heißt Ihre Eingabe in den MSX-Computer:

```
PRINT 6/2
```

Probieren Sie dies ruhig einmal aus ... vielleicht auch noch größere Rechenaufgaben und werden Sie so mit der Eingabe in Ihren MSX-Rechner und den Gebrauch des Befehls PRINT vertraut.

Wir können also zusammenfassen: Rechenaufgaben müssen auf dem MSX-Computer mit dem BASIC-Kommando PRINT eingeleitet werden. Wir dürfen allerdings nach der Rechenaufgabe keine Gleichheitszeichen setzen, denn der MSX-Computer versteht diese Zahleingabe nur so als eine an ihn gestellte Rechenaufgabe.

4. Wollen wir Texte auf dem Bildschirm ausgeben, müssen wir diese mit Anführungszeichen einrahmen

Hoffentlich haben Sie nach diesem erstmaligen erfolgreichen (!) Arbeiten mit Ihrem MSX-Computer ein wenig mehr Vertrauen zu ihm entwickelt, so daß wir noch einen Schritt weitergehen können.

Wir haben bisher mit PRINT Leerzeilen ausgegeben (PRINT), eingegebene Zahlwerte von einer Zeile in die andere kopiert (PRINT 7) und schließlich auch noch zuletzt einfache Berechnungen durchgeführt (PRINT 3*5). Bei all diesen Tätigkeiten war der Befehl PRINT dafür verantwortlich, daß überhaupt eine Ausgabe auf dem Bildschirm erfolgte.

Wie geht nun die Ausgabe von Wörtern und Texten auf dem Bildschirm vor sich? Probieren wir dies einmal aus.

Selbstverständlich müssen wir uns erneut des Befehls PRINT zur Ausgabe auf dem Bildschirm bedienen. Schreiben Sie hiernach ein Leerzeichen und geben dann das gewünschte Wort ein (z.B. die uns bereits vertraute Buchstabenkombination 'MSX') und betätigen anschließend wieder die Eingabetaste (letzteres sollte Ihnen inzwischen zur Gewohnheit geworden sein, denn jede Arbeit, die wir von unserem MSX-Computer erwarten, muß ja auch erst einmal befehligt werden).

Oh, das habe ich nicht gewollt! Unser MSX-Computer gibt nicht MSX in der nächsten Bildschirmzeile aus, sondern die Ziffer 0. Wie das zustande kommen konnte, werden wir im nächsen Kapitel ausführlicher erfahren. Hier nur soviel: Ihr MSX-Computer hat Ihre Eingabe MSX so verstanden, als wenn Sie ihm den Auftrag geben wollten, seine Speicherschublade mit der Bezeichnung MSX auf dem Bildschirm auszugeben. Da wir aber dieser Schublade MSX noch keinen Wert zugewiesen hatten, ist das Ergebnis = leer = 0.

Wollen Sie Texte auf dem Bildschirm mit dem PRINT-Kommando ausgeben, müssen Sie dieselbigen in Anführungszeichen einschließen. Sie können sich diese Vorgehensweise vielleicht so vorstellen, als wenn Sie sich mit Ihrem MSX-Computer unterhalten, ihm mitteilen, daß er die Buchstabenfolge MSX auf den Bildschirm schreiben soll. Wollen Sie eine wörtliche Anrede ausschreiben, müssen Sie dieselbige mit Anführungszeichen einrahmen. Genauso ist es bei Verwendung des Befehls PRINT zur Ausgabe von Texten durch Ihren MSX-Computer: Wollen Sie Texte von Ihrem MSX-Computer auf den Bildschirm kopieren (ähnlich, wie wir dies vorhin mit Zahlen getan haben), müssen Sie diese durch Anführungszeichen eingrenzen z.B. wenn die Buchstabenfolge MSX auf dem Bildschirm ausgegeben werden soll, heißt Ihre Befehlseingabe: PRINT "MSX".

5. Wir können Text- und Zahlausgaben hinter der PRINT-Anweisung beliebig mischen

Wir haben bis jetzt erfahren, daß Texte hinter der PRINT-Anweisung von Anführungsstrichen umgeben sein müssen, während man diese bei einer Rechenaufgabe oder Zahl nicht nur weglassen kann, sondern weglassen muß (während durch 'PRINT 3*4' das Ergebnis dieser Aufgabe errechnet und auf dem Bildschirm ausgegeben wird, würde eine Behandlung dieser Aufgabe wie ein Text 'PRINT "3*5"' nur die Kopie von 3*5 auf dem Bildschirm bedeuten, aber keine Berechnung - probieren Sie dies ruhig einmal aus).

Nun wollen wir ausprobieren, was passiert, wenn wir sowohl einen Text (in Anführungszeichen) als auch eine Rechenaufgabe (ohne

Anführungszeichen) hinter eine PRINT-Anweisung schreiben. Wird dann alles als Text oder aber als Rechenaufgabe verstanden oder aber wird wieder einmal eine der gefürchteten Fehlerkommentare ausgegeben?

Tippen Sie auf Ihrer Tastatur ein:

```
PRINT "MSX" 3*4
```

Oh Wunder! Ihr MSX-Computer hat von sich aus getrennt! Sowohl der Name MSX ist auf dem Bildschirm erschienen als auch durch ein Leerzeichen abgetrennt das Ergebnis der Aufgabe 3*4, nämlich 12.

Probieren Sie aus, ob man dieses Spielchen auch noch weiter fortsetzen kann:

```
PRINT "MSX" 3*4 "Computer"
```

Kein Problem. Alle drei Teile wurden von Ihrem MSX-Computer so behandelt, wie dies auch gefordert wurde. Vielleicht erkennen Sie durch dieses Beispiel noch einmal ganz deutlich die Begründung zur Differenziernng von Texteingaben und Zahleingaben.

Während Ihr MSX-Computer sauber zwischen diesen zwei unterschiedlichen Ausgabearten zu trennen weiß und dies auch auf dem Bildschirm durch die Leerzeichen ganz eindeutig dokumentiert, führt ein anderer Weg zum totalen Fiasko: Geben Sie

```
PRINT 3*4 5*5
```

ein. Statt der erwarteten zwei Ergebnisse (12 und 25) bekommen Sie als Ausgabe nur eines (675), das auch noch falsch ist.

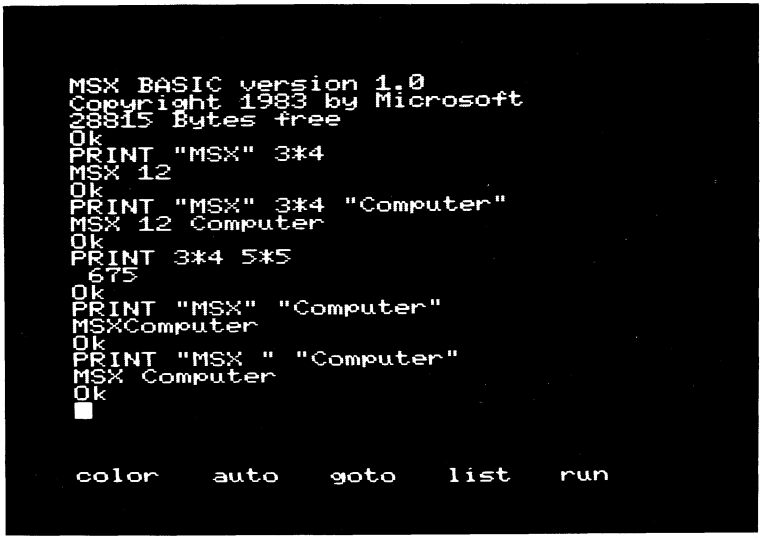
Da Ihr MSX-Computer nur innerhalb von gesetzten Anführungszeichen Leerzeichen akzeptiert, hat er ganz einfach Ihre zwei nebeneinander stehenden Aufgaben lediglich zu einer zusammengezogen: aus 3*4 5*5 wurde somit 3*45*5 und das ergibt nach Adam Riese genau den errechneten Wert 675!

Ähnlich benimmt sich Ihr MSX-Computer, wenn Sie nacheinander mehrere jeweils in Anführungszeichen eingeschlossene Texte mit einer PRINT-Anweisung ausgeben wollen: Geben Sie zur Dokumentation dieses Sachverhaltes folgendes ein:

```
PRINT "MSX" "Computer"
```

Egal, wie viele Leerzeichen Sie zwischen die zwei in Anführungszeichen eingerahmten Wörter 'MSX' und 'Computer' eingegeben haben, beide Wörter schmelzen bei der Ausgabe auf dem Bildschirm ineinander. Probieren Sie allerdings, die Leerzeichen innerhalb der Anführungszeichen einzugeben, werden sie als Teil des Textes gesehen und auch somit als Trennstrich akzeptiert:

```
PRINT "MSX  " "Computer".
```



Es dürfte in diesem Abschnitt deutlich geworden sein, daß es zwar möglich ist, hinter einer PRINT-Anweisung Zahl- und Textausgaben miteinander zu mischen; beachtet man dabei aber nicht gewisse Regeln (Aufgabe hinter Aufgabe ergibt nicht zwei Aufgaben sondern beide Aufgaben verschmelzen zu einer, die vom Wert her gesehen meist viel größer ist), endet die Arbeit in einem Fiasko.

6. Ihr MSX-Computer toleriert zwar das Leerzeichen, es hat für ihn aber keine Bedeutung. Deshalb brauchen wir andere Trennmittel wie den Strichunkt oder das Komma.

Wir können 20 Leerzeichen zwischen unsere zwei Rechenaufgaben von vorhin setzen ... immer wird Ihr MSX-Computer die dann weit auseinandergerücken Zahlen wieder miteinander zu einer größeren Zahl bzw. bei zwei Texten zu einem entsprechend längeren Text zusammenziehen.

Wollen wir diese Aktion aber dennoch durchführen, müssen wir ein anderes Trennungszeichen verwenden. Zuerst wollen wir dies mit dem Semikolon (Strichpunkt) ausprobieren:

```
PRINT 3*4;5*5
```

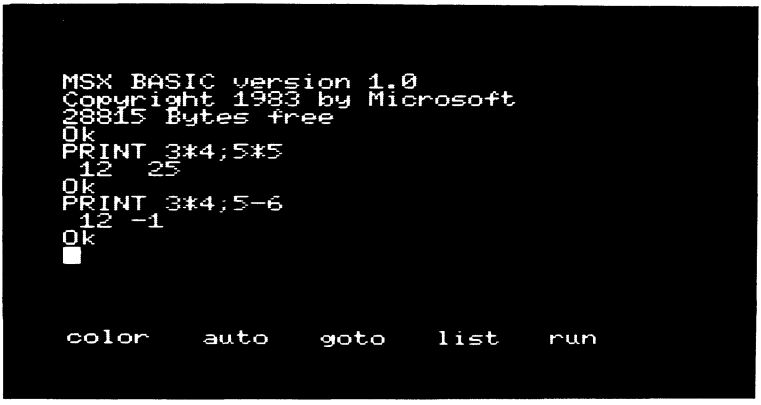
und es hat geklappt. Nun stehen die zwei Ergebnisse allerdings noch gar um ein zweites Zeichen auf dem Bildschirm weiter auseinander. Warum dies? Das eine Leerzeichen rührt vom verwendeten Trennzeichen her (damit wir die beiden Ergebnisse auf dem Bildschirm auch als zwei Einzelergebnisse besser unterscheiden können), während das andere Leerzeichen sich aus einem möglichen Vorzeichen des erzielten Ergebnisses erklärt.

Geben Sie zum Beweis folgenden PRINT-Befehl in Ihren MSX-Computer ein:

```
PRINT 3*4;5-6
```

Nun hat das zweite Ergebnis ein negatives Vorzeichen und genau dieses ist an die Stelle des einen Leerzeichens getreten.

Während das Semikolon zwischen zwei Rechnungen gesetzt also ein Leerzeichen zur Abtrennung erzeugt, sind es bei Textausgaben (wie auch vorhin ohne Semikolon) wiederum 0 Leerzeichen.



Probieren wir nun zum Abschluß unserer ausführlichen Behandlung des PRINT-Befehls noch, ein anderes Zeichen zwischen die unterschiedlichen zur Bildschimausgabe bestimmte Aufgaben zu setzen: Das Komma:

PRINT 3*5,5*5

Nicht nur ein Leerzeichen, sondern derer gleich 12 sind zwischen die beiden Ergebnisse gesetzt worden.

Diese trennen die beiden Ergebnisse allerdings schon so weit voneinander, daß man beinahe den Überblick verliert. Auch bei zwei mit Anführungszeichen eingefaßten Textpassagen beträgt der durch ein Komma erzeugte Abstand 12 Zeichen.

Warum nun dieser große Abstand? Um dies zu erfahren, bitte ich Sie, noch zwei weitere Aufgaben oder Texte (wiederum durch Kommata abgetrennt) hinter den PRINT-Befehl zu schreiben (also insgesamt 4):

PRINT 3*3,4*4,5*5,6*6

Sie sehen, daß Ihr MSX-Computer hinter dem zweiten Ergebnis in die nächste Bildschirmzeile gesprungen ist. So stehen nun die Ergebnisse von 3*3 und 5*5 sowie die Resultate von 4*4 und 6*6 fein säuberlich untereinander.

Der Sinn der Verwendung des Kommas zur Abtrennung rührt also daher, einen sogenannten formatierten Ausdruck auf dem Bildschirm vorzunehmen, Spalte unter Spalte und somit leichter lesbar (stellen Sie sich einmal ein Rechenblatt vor, auf dem die verschiedenen Zahlen kreuz und quer untereinander stehen würden).

Wir haben also schließlich noch eine Differenzierung in der Funktion des PRINT-Befehls in diesem letzten Abschnitt erfahren: Wenn wir Semikolons bzw. Kommata zwischen zwei auszugebende Berechnungen setzen, werden sie voneinander getrennt und nicht zu einer Berechnung verschmolzen (wie ohne eindeutiges Trennungszeichen).

Wir können bei den Trennungszeichen zwischen Semikolon und Komma wählen. Das Semikolon erzeugt zwei Leerzeichen zwischen den ausgegebenen Zahlwerten und kein Leerzeichen bei Texten, während das Komma derer gleich 12 produziert (ideal zur Tabellenerstellung - sowohl bei Zahlwerten = Ergebnissen als auch bei Texten).

Nun haben wir eine Vokabel unseres MSX-BASIC kennengelernt, den Befehl PRINT. Wir haben festgestellt, daß es nicht ausreicht, diesen Befehl nur rein her von seiner Vokabelbedeutung zu verstehen (PRINT = schreiben), sondern wir müssen ähnlich wie in der Mathematik (es ist ein weiter Weg vom Kennenlernen des Minuszeichens bis zum Aufbau eines Verständnisses für die Regel Minus * Minus = Plus) den Befehl von seiner Funktion her kennenlernen.

Hierbei leistet uns gerade der MSX-Computer große Dienste, denn wir können uns durch 'Trial and Error' (Versuch und Fehler, darauf ein neuer Versuch) - wie auch in diesem Kapitel der Anschaulichkeit halber mehrfach so geschehen - an seine Bedeutung herantasten.

Es gibt viele weitere unter den ca. 150 BASIC-Kommandos, die eine derart komplexe Funktionsstruktur aufweisen wie der hier behandelte PRINT-Befehl. Seien Sie aber sicher, daß es nicht von heute auf morgen erforderlich ist, all' diese Befehle vollständig zu beherrschen. Dies wäre ja auch nicht gut, denn so würden Sie vielleicht zu schnell für die weitere Erforschng Ihres MSX-Computers an Interesse verlieren. Es stecken auch vielleicht noch nach Jahren intensiven Gebrauchs Geheimnisse in dieser Wunderkiste verborgen, die Sie auch erst dann begreifen und verstehen lernen.

Der MSX-Computerspeicher enthält viele Schubladen

Im vorigen Kapitel war etwas rätselhaftes passiert, das ich nun auf den nächsten Seiten erklären will: Sie hatten beim Ausprobieren des PRINT-Befehls ohne Anführungszeichen versucht, den Text MSX auszugeben:

```
PRINT MSX
```

Anstelle des Textes wurde aber die Zahl 0 ausgegeben.

Ich hatte Ihnen kurz erläutert, daß der Computer ohne Anführungszeichen diese Buchstabenkombination im Speicher suchen würde und da wir dieselbige vorher noch nicht benannt hätten, wäre der Wert 0.

Um das zu verstehen, müssen wir erst einmal über die Fähigkeiten Ihres MSX-Computers sprechen, wie dieser denn seinen RAM-Speicher überhaupt gebraucht.

Wir hatten im technischen Einführungskapitel bereits erläutert, daß unser MSX-Computer von den RAM-Speicherplätzen genau 28815 beim Einschalten zur Verfügung stellt. Wir hatten auch erfahren, daß man diese Speicherplätze nicht nur zum Abspeichern von Texten (ca. 14 DIN-A-4 Seiten) sondern auch zum Abspeichern von Programmen benutzen kann.

In diesem Kapitel wollen wir uns nun mit der Abspeicherung von Daten in Ihrem MSX-Computer auseinandersetzen: Jeder der 28815 freien Speicherplätze kann ein Zeichen abspeichern, sei es ein Buchstabe oder auch eine Zahl. Wie müssen wir vorgehen, dem Speicher diese Informationen zukommen zu lassen?

Im vorigen Kapitel hatten wir bereits erfahren, daß dies nicht gelingt, indem wir nur unsere Texte einfach wirr und ungeordnet auf den Bildschirm schreiben. Wenn wir nämlich anschließend die Eingabetaste drücken, würde Ihr MSX-Computer lediglich versuchen, den Text mit seinen 150 Vokabeln zu interpretieren. Wie wir im vergangenen Kapitel erfuhren, ist die Ausgabe dann kurz und bündig eine Fehlermeldung, meist Syntax Error.

Wir müssen uns also einer anderen Technik bedienen, um unseren MSX-Computer als Datenbank zu gebrauchen:

1. Der MSX-Computer merkt sich Zahlen in seinen Schubladen

Zuerst einmal soll sich Ihr MSX-Computer Zahlen merken. Wir richten zu diesem Zweck eine Schublade in seinem Speicher ein und geben dieser einen Namen. Da wir in allen Beispielen bisher die Zauberformel bestehend aus den Buchstaben 'MSX' genommen haben, soll es auch hier nicht anders sein: unsere Schublade soll 'MSX' heißen.

Wie in der Mathematik, müssen wir nun eine Gleichung aufstellen. Sie erinnern sich vielleicht noch daran, als Sie in der Schule Gleichungen mit den sogenannten Variablen X und Y lösen mußten? Schließlich konnte man nach einer Berechnung feststellen, daß z.B. X den Wert 5 angenommen hatte und Y den Wert 6. Im Rechenheft stand dann:

```
X=5
Y=6
```

Nun sollen wir unserem MSX-Computer zu verstehen geben, daß die Schublade mit dem Namen MSX den Zahlwert 6 enthält. Kein Problem. So wie Sie gerade die mathematischen Zuweisungen gelesen und verstanden haben (X=5 und Y=6), genau nach derselben Art schreiben Sie jetzt bitte den Zahlwert 6 in Ihre Schublade mit dem Namen MSX:

```
MSX=6
```

Vergessen Sie nicht, wiederum anschießend die Eingabetaste zu betätigen. Wie schön. Keine Fehlermeldung wurde auf dem Bildschirm ausgegeben, sondern nun steht da ganz zufriedenstellend 'Ok'. Unser MSX-Computer hat sich gemerkt, daß die Schublade MSX den Wert 6 enthält!

Glauben Sie mir das nicht? Probieren Sie es doch selbst einmal aus, indem Sie die Schublade MSX auf dem Bildschirm ausgeben. Ausgeben auf dem Bildschirm, das sollte bei Ihnen sofort heißen - soweit Sie das letzte Kapitel genau studiert haben - PRINT.

Also geben wir den Inhalt von Schublade 'MSX' mit dem PRINT-Befehl wieder auf dem Bildschirm aus:

```
PRINT MSX
```

Haben Sie bis hierhin alle Punkte befolgt, sollte nun auf Ihrem Bildschirm die Zahl 6 ausgegeben worden sein. Vielleicht haben Sie noch ein gesundes Mißtrauen, denn eine Zeile vor Ihrer Eingabe haben Sie ja noch selbst geschrieben: 'MSX=6'. Wie leicht (?) könnte Ihr MSX-Computer einfach dort abgeschrieben haben! Um auch diesen Beweis anzutreten (daß nicht gemogelt wurde und die Schublade MSX sich nicht nur auf dem Bildschirm sondern viel tiefer in Ihrem MSX-Computer befindet) löschen Sie den Bildschirm einfach durch Drücken der CLS-Taste (SHIFT und CLS/HOME) und geben erneut den Befehl zur Speicherabfrage ein:

```
PRINT MSX
```

Wieder wurde der Zahlwert 6 ausgegeben. Für Sie als gesunder zweifelnder Zeitgenosse hoffentlich ein schlüssiger Beweis!

Geben Sie nun auch anderen Schubladen Namen und Zahlwert:

```
Lucia=4
Walter=5
Michael=7
Jürgen=6
```

Wenn Sie anschließend mit der PRINT-Funktion die Ausgabe auf dem Bildschirm hervorzaubern, werden Sie keine Enttäuschung erleben.

Der freie Speicherplatz Ihres MSX-Computers wurde von dieser Speicherarbeit nicht sonderlich angefüllt. Es wäre ohne weiteres möglich, wenn Sie die Zuweisung von Zahlen an sogenannte Variablen (die Namen, die wir den Schubladen geben) noch 24 Stunden

weitertreiben würden. Ihr MSX-Computer käme selbst nach mehreren tausend Eingaben noch nicht aus der Puste ... Sie aber vielleicht.

Welche Namen können wir für unsere Variablen eigentlich verwenden? Gelingt es auch mit ganz langen Namen?

```
Mississippi=777
PRINT Mississippi

Ergebnis: 777
```

Doch freuen Sie sich nicht zu früh. Probieren wir es mal mit einer jungen Dame aus, die Mia gerufen wird:

```
Mia=666
PRINT Mia

Ergebnis: 666
```

Wunderbar. Und was ist aus unserer Schublade mit dem Namen Mississippi geworden?

```
PRINT Mississippi

Ergebnis: 666
```

Da hat sich etwas geändert! Statt der erwarteten 777 beträgt der Zahlwert unserer Mississippi-Variablen nur noch 666. Die Erklärung ist ganz einfach: Ihr MSX-Computer toleriert zwar beliebig lange Variablennamen, er unterscheidet allerdings bei seiner Speicherorganisation je Schublade nur die ersten zwei Zeichen. So ist es ihm egal, ob die Schublade von uns nun Mississipi oder Mia genannt wird. Beide Schubladen fangen nun einmal mit dem Buchstaben M und i an und das heißt, Sie sind für unseren MSX-Computer zu 100 % gleich. Klar, daß da auch unsere Variable Mia=666 den Wert der Variablen Mississippi vom ursprünglichen Wert 777 in 666 umgewandelt hat.

Also, Vorsicht bei der Namensauswahl von Variablen! Nur die ersten zwei Zeichen unterscheidet Ihr MSX-Computer!

Es gilt noch zwei weitere Punkte bei der Auswahl der Variablennamen zu beachten:

Im Variablennamen darf keine Bezeichnung vorkommen, die bereits im Vokabular Ihres MSX-Computers enthalten ist. So wäre es nicht möglich, eine Variable 'Sprint' zu nennen; probieren Sie es ruhig einmal praktisch aus:

```
Sprint=5
Syntax error
```

denn in ihr ist die uns bereits vertraute MSX-Vokabel PRINT enthalten.

Zudem müssen wir auch genau darauf achtgeben, daß die ersten zwei Zeichen (die alles entscheidenden, wie wir bei Mississippi und Mia gesehen haben) jeder Variablen nach folgender Regel ausgewählt werden:

Zeichen 1: Nur Buchstaben (Kleinbuchstaben werden von Ihrem MSX-Computer automatisch in Großbuchstaben umgewandelt!)

Zeichen 2: Buchstaben oder Zahlen von 0 bis 9

Von Ihrem MSX-Computer anerkannte Variablennamen wären demnach z.B.: A3, AC, B9, aber natürlich auch X, Y, und C.

Die Zuweisung würde bei folgenden Variablennamen nicht funktionieren: A., 1E und C,.

Wir haben also bisher in diesem Kapitel gelernt, wie wir Schubladen = Speicherplätzen in unserem Computer Namen geben können und ihnen Zahlwerte zuweisen können. Wie sieht es aber nun mit Texten aus, die wir in Schubladen unseres MSX-Computers ablegen wollen?

2. Es gibt nicht nur eine Zahlvariable mit dem Namen MSX, sondern derer gleich drei verschiedene

Wir wollen noch weitere Zahlen in unserem MSX-Computer ablegen, jedoch werden wir in diesem Kapitel darauf achtgeben, ob sich die hohe Stellengenauigkeit unseres Computers bei mathematischen Aufgaben bzw. Zahlen auch reduzieren läßt.

Was heißt das? Bisher hatten wir nur ganze Zahlen in den Speicher geschrieben und sonst nichts. Wenn Sie aber bereits im PRINT-Kapitel beim Ausprobieren der vier Rechenoperatoren das Teilungszechen (/) öfters angewendet haben, wird Ihnen aufgefallen sein, daß unser MSX-Computer auch hinter dem Dezimalpunkt sehr genau rechnet. Bestes Beispiel dafür ist die Berechnung von 2/7.

Bei dieser Aufgabe kommt eine immer wiederkehrende Folge von Zahlwerten hinter dem Komma heraus, mathematisch auch Periode genannt. Die Mathematiker kennzeichen eine solche Zahlenfolge mit einem waagerechten Strich oberhalb der gleichlautenden Dezimalstellen.

Was errechnet unser MSX-Computer bei 2/7? Geben Sie bitte ein

```
PRINT 2/7
```

Ergebnis: 0.28571428571429.

Mit insgesamt 14 Stellen nach dem Komma hat unser MSX-Computer diese Divisionsaufgabe gelöst und nicht nur das! Er hat auch sehr genau Stelle für Stelle berechnet: Wenn Sie genau hinschauen, weist das Ergebnis die periodische Zahl 285714 auf. Die sechsstellige Periode hat Ihr MSX-Computer nicht nur zweimal genau errechnet, die letzten zwei angezeigten Ziffern zeigen sogar, daß Ihr MSX-Computer intern mit noch einer Stelle mehr rechnet - also insgesamt 15 Stellen nach dem Komma (Sie können ruhig einmal diesen Rechentest mit anderen vergleichbaren Nicht-MSX-Computern durchführen. Microsoft-BASIC und damit Ihr MSX-Computer wird dabei am besten abschneiden).

Was ist an den letzten zwei Ziffern so besonderes? Wir wissen, daß die nächste Periode wieder mit den dei Ziffern 285 einfangen müßte. Da Ihr MSX-Computer davon aber nur noch zwei Stellen auf dem Bildschirm anzeigen kann, berechnet er intern auch noch die dritte. Ist diese dritte Zahl kleiner als 5, wird der Wert der zweiten Zahl (8) beibehalten, ist aber der dritte Wert 5 oder gar größer als 5, wird der Wert der zweiten Ziffer aufgerundet. Da dies geschehen ist (der dritte Wert wäre 5), wird einfach der zweite Wert (8) um eins aufgerundet, das ergibt als Ergebnis die 9!

Wie können wir nun diese Berechnung in einer Variablen abspeichern? Wir suchen uns wieder einen Variablennamen (behalten wir ganz einfach den Namen MSX bei) und weisen diesem entweder das Enderebnis Ziffer für Ziffer zu (in unserem Fall MSX=0.28571428571429) oder lassen das Ergebnis während des Eingabevorgangs einfach berechnen:

```
MSX=2/7
```

Wenn Sie nun

```
PRINT MSX
```

eingeben, so erscheint auf dem Bildschirm das entsprechende Ergebnis mitsamt aller Stellen, wie wir sie auch vorhin bei der PRINT-Ausgabe auf dem Bildschirm beobachten konnten.

Sie können sich vielleicht vorstellen, daß die Abspeicherung einer Zahl mit so vielen Ziffern mehr als einen Speicherplatz verbraucht. Glücklicherweise wendet Ihr MSX-Computer eine ganz besondere Art der Abspeicherung an, so daß für diese 15 Ziffern nicht gleich 15 Speicherplätze in Anspruch genommen werden müssen, aber immerhin sind es auch noch derer 8.

Wenn Sie nun aber kein Mathematiker sind bzw. nur das kleine Einmaleins mit Ihrem Computer berechnen wollen, wäre es schade, so viel Speicherplatz sinnlos zu verschenken (zudem rechnet Ihr Computer selbstverständlich erheblich schneller, wenn er nur eine oder vielleicht auch sechs Stellen hinter dem Komma dar-

stellen soll). Reicht Ihnen eine verminderte Genauigkeit hinter dem Komma aus, so müssen Sie den Variablennamen dafür kenntlich machen: Sie setzen ein Ausrufungszeichen hinter den Variablennamen, wenn Ihnen sechs Stellen hinter dem Komma genügen.

```
MSX!=2/7
PRINT MSX!
```

Ergebnis: 0.285714.

Zwei Hinweise zwischendurch:

1. Die Internationalität Ihres MSX-Computers bedingt, daß Sie statt eines Kommas einen Punkt zur Abtrennung vom ganzen Zahlanteil und seinen Dezimalstellen eingeben müssen.

2. Ihr MSX-Computer zeigt bei einem Ergebnis kleiner als 1 keine Ziffer vor dem Punkt an. Genauso wie Sie auch beim Taschenechner bei Zahlen kleiner als 1 lediglich einen Punkt und dann die Dezimalstellen eingeben, zeigt Ihr MSX-Computer diese Werte entsprechend ohne 0 auf dem Bildschirm an.

Die letzte Ziffer, in diesem Fall die 4, braucht nicht aufgerundet zu werden, da, wie wir bereits wissen, hinter der 4 wieder die Periode mit einem Zahlwert beginnt, der kleiner als 5 ist.

Brauchen wir bei unseren Berechnungen noch nicht einmal diese 6 Stellen hinter dem Komma, können wir ein Prozentzeichen hinter dem Variablennamen eingeben also z.B.

```
MSX%=2/7
PRINT MSX%
```

Ergebnis:0.

Das Ergebnis kommt allerdings in dieser Rechenart nur annähernd an den richtigen Zahlwert heran.

Bedenken Sie deshalb bei Divisionsaufgaben immer sehr genau, ob sich der Speicherplatzvorteil und die Rechengeschwindigkeit im Vergleich zur Genauigkeit entsprechend gut miteinander vertragen! Das schöne ist, daß wr mit Hilfe der unterschiedlichen Genauigkeitsangaben gleich dreimal einen Variablennamen vewenden können. Geben Sie zur Kontrolle ein:

```
PRINT MSX
```

Ergebnis: 0.28571428571429

```
PRINT MSX!
```

Ergebnis: 0.285714

```
PRINT MSX%
```

Ergebnis: 0

Bei der ganzzahligen Verareitung von Variablen müssen wir allerdings noch einen weiteren Punkt beachten: Ihre Zahlen dürfen bei dieser Variablenabspeicherung nicht größer als 32767 und nicht kleiner als -32768 sein, sonst gibt Ihr MSX-Computer die Fehlermeldung 'Overflow' (Überfließen = diese Zahl ist jenseits meiner Grenzen) auf dem Bildschirm aus.

Wir können schließlich noch ein weiteres Zeichen hinter den Variablennamen hängen: die Raute (#). Dieses Zeichen bedeutet, daß hier mit doppelter Genauigkeit gerechnet wird, was soviel bedeutet wie 14 Stellen hinter dem Komma.

Sie fragen nun sicher, welcher Unterschied denn dann zu unserem Variablennamen ohne Zusatz besteht (MSX). Die Frage ist berechtigt. Sowohl der Variabenname MSX als auch MSX# beziehen sich auf dieselbe 14-stellige Zahlabspeicherung, jedoch mit dem Unterschied, daß durch Eingabe der Raute hinter der Variablen dieselbige noch einmal für doppelte Genauigkeit besonders kenntlich gemacht wird.

```
PRINT MSX
```

Ergebnis: 0.28571428571429

```
PRINT MSX#
```

Ergebnis: 0.28571428571429

Damit vorerst einmal genug zur Zahlenvariablenabspeicherung. Wie können wir nun auch noch Texte für unsere zahlreichen DIN-A-4-Seiten im MSX-Computer ablegen?

3. Soll der MSX-Computer sich Texte merken, müssen diese in Anführungszeichen eingeschlossen werden

Vielleicht wundern Sie sich, daß wir die Abspeicherung von Texten in Variablen hier gesondert behandeln. Warum sollte es nicht möglich sein, Texte auch in unseren bisherigen Variablentypen abzulegen?

Probieren wir es einfach aus (das ist immer so gut an den MSX-Computern: Trial und Error = Versuch und aus dem möglichen Fehler lernen, indem wir Schlüsse auf dessen Ursache ziehen): Wir wollen das Wörtchen 'dieser' in der alt bekannten Variablen MSX abspeichern:

```
MSX=dieser
```

Schauen wir uns doch gleich einmal an, was sich nun in der Variablen MSX befindet:

```
PRINT MSX
```

Ergebnis: 0

Das war wohl nichts. Bei dieser Eingabe hat Ihr MSX-Computer das Wörtchen 'dieser' als einen anderen Variablennamen verstanden und somit dessen Wert (bisher = 0) einfach in unsere Variable MSX kopiert - so wurde MSX auch 0.

Erinnern wir uns an das Kapitel über den Befehl PRINT: Wenn wir Texte auf den Bildschirm ausgeben wollten, mußten wir dieselbigen mit Anführungszeichen umrahmen. Probieren wir auch dies bei der Variablenzuweisung aus:

```
MSX="dieser"
```

Ergebnis: eine neue Fehlermeldung namens 'Type mismatch' (ins Deutsche übersetzt würde das bedeuten: Du hast den falschen Variablentyp verwendet, die Zahlvariable MSX ist zur Textabspeicherung nicht geeignet!).

Aus diesem Grund müssen wir noch ein anderes Anhängsel für die Kennzeichnung von Variablennamen dazulernen: das Dollarzeichen $.

Fügen wir dieses Zeichen einem Variablennamen hinzu, so entsteht eine sogenannte String- oder Zeichenkettenvariable, die dazu in der Lage ist, Texte von einer Länge bis zu 255 Zeichen zu speichern. Probieren wir dieses aus, indem wir nun unser Beispiel von vorhin mitsamt dem richtigen Variablentyp anwenden:

```
MSX$="dieser"
PRINT MSX$
```

Ergebnis: dieser

Sie können selbstverständlich auch Zahlen in einer Stringvariablen abspeichern. Zu diesem Zweck müssen Sie allerdings auch diese in Anführungszeichen einschließen (sonst wird wieder die Fehlermeldung 'Type mismatch' auf dem Bildschirm angezeigt) und zudem: rechnen läßt sich mit einer Stringvariablen nicht direkt. Probieren Sie es aus:

```
MSX$="33"
PRINT  MSX$+"11"
```

Ergebnis: 3311

Hier wurden nicht mehr die Zahlen mathematisch addiert, sondern die beiden Texte einfach aneinandergefügt: die Stringvariable MSX$ mit dem Text "33" und der eingegebene Textzahlenwert "11" verschmolzen zu 3311.

Ich führte vorhin an, daß Textvariablen 'nur' bis zu 255 Zeichen enthalten dürfen ("dieser" wären z.B. 6 Zeichen).

Wenn Sie das schon nachgeprüft haben, werden Sie festgestellt haben, daß Ihr MSX-Computer bereits bei 200 Zeichen die Flinte ins Korn wirft mit der Ausgabe der Fehlereldung: 'Out of string space' (ich habe nicht mehr Platz frei für weitere Strings zur Abspeicherung).

Diesem Übel können Sie schnell abhelfen, indem Sie den Stringspeicher Ihres MSX-Computers mit dem Befehl CLEAR vergrößern. Geben Sie CLEAR 2000 ein und versuchen Sie nun erneut, mehr als 255 Zeichen im Speicher festzuhalten.

Diesmal wird die Reaktion Ihres Computers gar keine Fehlermeldung mehr sein, aber wenn Sie sich den abgespeicherten String mal etwas genauer ansehen, so werden Sie leicht feststellen können, daß nach dem 255sten Zeichen der Variablenlänge ein abruptes Ende bereitet wurde. Dies ist ein Zustand im Microsoft BASIC, dem Sie auch nicht ohne weiteres abhelfen können.

Sie erkennen somit, daß eine DIN-A-4-Seite mit ca. 2000 Zeichen nicht in eine einzige Textvariable paßt, sondern derer mindestens 8 zur Speicherung benötigt.

Noch zwei Hinweise:

1. Wenn Sie wie vorhin den Befehl CLEAR eingeben, werden augenblicklich in Ihrem Computer alle variablen Zahlwerte und Texte gelöscht. Geben Sie deshalb am besten immer zu Beginn Ihrer Arbeit mit dem MSX-Computer das gewünschte CLEAR-Kommando ein (falls mehr als 200 Textzeichen im Speicher abgelegt werden sollen).

Wenn Sie im übernächsten Kapitel mit dem Erstellen von Programmen beginnen, werden Sie auch dort merken: eine neue eingegebene Programmzeile bzw. ein Programmstart löscht ebenfalls sämtliche im Speicher befindlichen Variablenwerte.

2. Sie haben erfahren, daß Sie drei veschiedene Zahlwerte pro Variablennamen (je nach Schlußzeichen des Variablennamens: %=ohne Dezimalstellen, !=mit 6 Dezimalstellen = einfache Genauigkeit, #=mit 14 Dezimalstellen = doppelte Genauigkeit) und einen Text (Schlußzeichen der Variablen = $) abspeichern können. Weisen Sie dem gleichen Variablennamen allerdings einen neuen Wert zu, wird der alte Wert augenblicklich gelöscht. Merken Sie sich: Pro Variablenname behält sich Ihr MSX-Computer immer nur den zuletzt eingegebenen Zahlwert bzw. Text.

Bevor wir zum Schreiben unseres ersten Programms kommen, müssen wir noch ein paar Befehle mehr kennenlernen. Ich glaube, daß die vorigen Kapitel für dieses Erlernen eine gute Basis bieten und daß Ihnen nun auch gewisse Fehlermeldungen nicht mehr so leicht einen Schreck einjagen können.

Beherrschen Sie die Vokabeln in diesem Kapitel, so sind Sie in einer ähnlichen Situation wie ein Fremdsprachenneuling, der Englisch lernt, aber bereits 200 bis 400 Vokabeln und erste Ansätze der englischen Grammatik beherrscht.

Bei der Erläuterung der BASIC-Vokabeln in diesem Kapitel werde ich allerdings nicht so sehr ins Detail gehen, wie dies im Kapitel über den PRINT-Befehl geschehen ist. Das würde den Rahmen dieses Buches bei weitem sprengen.

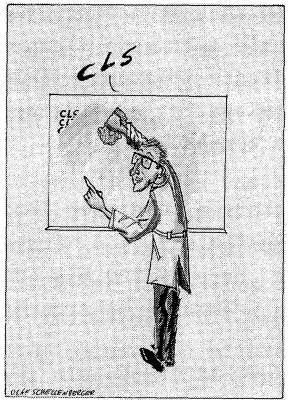
Schließlich erfolgt eine Einweisung, wie Sie nicht nur mit diesen Befehlen praktisch umgehen können, sondern damit auch bereits ein richtiges Programm schreiben können.

Also, bis zum ersten lauffähigen Programm ist es noch ein kleiner Schritt. Packen wir es gemeinsam an.

Bisher haben wir 'lediglich' PRINT in seiner gesamten Anwendungsfülle, CLEAR zum Bereitstellen von Speicherplatz für Strings, die verschiedenen Typen von Variablen und die Bedienung der meisten Tasten auf unserem Computer kennengelernt (auch die, die nicht auf einer gewöhnlichen Schreibmaschinentastatur zu finden sind).

1. Mit dem Befehl CLS löschen wir den Bildschirm

Vorhin hatten wir bereits die CLS/HOME-Taste kennengelernt. Drücken wir diese Taste gemeinsam mit einer der SHIFT-Tasten, wird der Bildschirm gelöscht und unsere Schreibmarke befindet sich augenblicklich in der linken oberen Bildschirmecke.



Da wir bei einem laufenden Programm allerdings ab und zu auch den Wunsch haben, daß der Bildschirm gelöscht werden soll, brauchen wir dazu einen speziellen Befehl: CLS (Englisch CLear Screen = übersetzt: reinige den Bildschirm).

Schreiben Sie etwas auf den Bildschirm, drücken anschließend die Eingabetaste und geben dann die Buchstabenkombination CLS ein. Drücken Sie nun die Eingabetaste, wird augenblicklich der Bildschirm gelöscht. Einziger Unterschied zum Drücken von SHIFT und CLS/HOME: in der oberen linken Bildschirmecke bekommen Sie noch die Quittung Ihres MSX-Computers, daß er den Befehl CLS verstanden und ausgeführt hat: Ok.

2. Die Eingabe von Variablen im Programm geschieht durch INPUT

Im vorigen Kapitel hatten wir nicht nur die verschiedenen Variablentypen kennengelernt, wir hatten diesen auch bereits Werte zugewiesen z.B. Mia=666.

Nicht immer steht es während eines Programmablaufs fest, welchen Wert die Variablen einnehmen sollen (die Variablen heißen schließlich so und nicht anders, weil sie in ihren Inhalten variieren können; einmal ist die Variable Mia=666, beim nächsten Mal schon wieder 777 usw.).

So muß man die Möglichkeit schaffen, daß Werte für Variablen auch durch einen Programmschritt abgefragt werden können (z.B. in einem Programm, wo zu einem Zahlwert der entsprechende Monatsname ausgegeben werden soll, können wir nicht absolut sagen:

Eingabe = 9 = September, sondern wir müssen dem Eingebenden vielmehr die Möglichkeit bieten, zwischen den Zahlen 1 bis 12 auszuwählen).

Dies geschieht durch den Befehl INPUT mitsamt einer Variablen des Typs, den wir als Eingabe wünschen. Verlangen wir eine Zahleingabe, sagen wir z.B. INPUT MSX; wünschen wir hingegen eine Texteingabe, lautet die Programmzeile: INPUT MSX$.

Probieren Sie dies doch einfach einmal aus: Geben Sie ein:

```
INPUT MSX
```

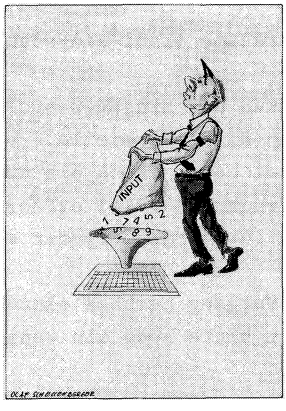
und drücken Sie daran anschließend die Eingabetaste. Auf dem Bildschirm erscheint ein Fragezeichen, das Sie darauf aufmerksam machen soll, daß nun eine Eingabe von Ihnen erwartet wird.

Geben Sie z.B. 6 ein und drücken anschließbend erneut die Eingabetaste. Die ausgegebene Meldung Ok bestätigt Ihnen, daß der Eingabebefehl nicht nur richtig verstanden wurde, sondern daß die Variableneingabe damit abgeschlossen ist.

Mit dem Befehl

```
PRINT MSX
```

bekommen Sie die Bestätigung: Durch das INPUT-Kommando wurde dem Computer etwas ein(IN)gegeben(PUT).



Durch Ihre Zahleingabe haben Sie das gleiche erreicht, als wenn Sie geschrieben hätten:

```
MSX=6
```

Sie können genausogut mit dem INPUT-Befehl Textvariablen oder Strings abfragen:

```
INPUT  MSX$
```

Nach dem Erscheinen des Fragezeichens reicht es aus, wenn Sie den Text ohne Anführungszeichen auf Ihrer MSX-Tastatur eingeben z.B. Eingabe: 'dieser'.

Nach dem Drücken der Eingabetaste können Sie die MSX-Textvariable wieder durch PRINT abfragen und auf den Bildschirm ausgeben. Das Ergebnis müßte erneut das Wörtchen 'dieser' sein.

Zudem können Sie den INPUT-Befehl ähnlich wie einen PRINT-Befehl verwenden; das Fragezeichen erscheint nicht mehr allein, sondern vielmehr zusätzlich auch noch eine Eingabefrage wie z.B.: 'welche Monatszahl wollen Sie als Namen ausgeben?' Zuerst gebrauchen wir den INPUT-Befehl für diesen Fall wie ein PRINT-Kommando, setzen aber hinter die zweiten Anführungszeichen einen Strichpunkt (Semilkolon) und schließen erst dann den gewünschten Variablennamen an.

Ihre Eingabe für obige Frage würde also so aussehen:

```
INPUT "Monatsname (Zahl eingeben) ";MSX
```

Nun erscheint nach Drücken der Eingabetaste nicht nur ein Fragezeichen in der nächsten Bildschirmzeile, sondern vielmehr der Satz mitsamt dem Fragezeichen. Geben Sie nun eine Zahl ein, so werden Sie bei der Abfrage mit PRINT erfahren, daß auch diese Eingabe einwandfrei von Ihrem MSX-Computer akzeptiert wurde.

Wie können wir diesen Vorgang nachher sinnvoll in ein Programnm einbauen? Haben Sie dazu bitte noch ein wenig Geduld.

3. Wenn du mich schlägst, dann schlage ich nicht zurück

Nicht ungewollt ist in diese Aussage in gewisser Weise ein biblischer Charakter einflossen: Wenn du mir auf die linke Backe schlägst, dann halte ich dir auch die Rechte hin.

Worauf es uns aber in diesem Satz ankommt, ist schlicht und ergreifend die 'wenn ... dann'-Beziehung: Wenn ein bestimmter Vorgang abgeschlossen ist, dann erfolgt (in Wort oder Tat) eine Reaktion.

In der englischen Sprache - und auf einem Teil dieses Vokabulars ist ja auch die Sprache unseres MSX-Computers aufgebaut - heißt diese 'wenn ... dann'-Beziehung übersetzt 'IF ... THEN'.

Wir wollen nun einmal versuchen, unser Monatseingabebeispiel von vorhin mit der IF ... THEN-Funktion fortzusetzen. Erst einmal in gut Deutsch: Wenn der Zahlwert der Variablen MSX=9 ist, so schreibe auf den Bildschirm: September. Übersetzt in die Programmiersprache müßten Sie eingeben:

```
IF MSX=9 THEN PRINT "September"
```

Im Deutschen heißt diese Funktion: Wenn MSX=9 dann schreibe auf den Bildschirm das Wort September.

Wir wollen nun beide Befehle nacheinander auf unserem MSX-Computer ablaufen lassen: Geben Sie erst den Befehl

```
INPUT "Monatsname (Zahl eingeben) ";MSX
```

in Ihren MSX-Computer ein und anschließend die gerade eben definierte Programmzeile:

```
IF  MSX=9 THEN PRINT "September"
```

Wenn Sie nun alles richtig gemacht haben, müßte unter der zuletzt eingegebenen Zeile nicht nur das Ergebnis September erscheinen, sondern zum Abschluß des Programmschritts auch noch die Information, daß dieser Programmschritt erfolgreich abgeschlossen wurde: Ok.

Was passiert, wenn wir eine andere Zahl als 9 eingeben? Probieren Sie es aus, indem Sie mit Hilfe der Pfeiltasten die Schreibmarke zur INPUT-Anweisung zurücksteuern und anschließend wieder wie vorhin Schritt für Schritt vorgehen.

Der Unterschied zu vorhin ist der, daß nun nicht der Monatsname September ausgegeben wird (weil die wenn-Bedingung nicht erfüllt wurde) und somit das Wörtchen Ok bereits in der Zeile erscheint, wo vorhin der Monatsname September zu sehen war.

4. Mit BEEP und COLOR erzeugen wir Sound und Grafik

Nach den zuletzt genannten BASIC-Kommandos (INPUT und IF ... THEN - übrigens zwei der wichtigsten und vielleicht auch schwierigsten MSX-Befehle überhaupt) nun wieder etwas leichteres.

Wollen Sie später im Programm darüber Bescheid geben, wenn irgendetwas unvorhergesehenes passiert ist (z.B. es wurde auf die Frage nach dem Monat eine unmögliche Zahl wie 13 eingegeben) oder aber ein Programmteil abgeschlossen ist, teilen Sie dies dem Zuhörer bzw. Zuschauer über den Fernsehlautsprecher am besten durch ein Tonsignal mit.

Drehen Sie die Lautstärke an Ihrem Fernseher ruhig etwas hoch und geben dann den Befehl BEEP ein. Nach Drücken der Eingabetaste können Sie nun einen kurzen hohen Ton vernehmen - später im Programm unser Signalton.

Wenn Sie auch nur ein paar englische Vokabeln beherrschen, das Wort 'COLOR' wird Ihnen als 'Farbe' bereits bekannt sein.

Ihr MSX-Computer stellt Ihnen insgesamt 16 verschiedene Maltöpfe zur Verfügung:

Farbe 0: Transparent
Farbe 1: Schwarz
Farbe 2: Mittleres Grün
Farbe 3: Hellgrün
Farbe 4: Dunkelblau
Farbe 5: Hellblau
Farbe 6: Dunkelrot
Farbe 7: Cyan
Farbe 8: Mittleres Rot
Farbe 9: Hellrot
Farbe 10: Dunkelgelb
Farbe 11: Hellgelb
Farbe 12: Dunkelgrün
Farbe 13: Magentarot
Farbe 14: Grau
Farbe 15: Weiß

Geben Sie nun COLOR mit einer dieser Farbnummern ein und drücken anschließend die Eingabetaste. Augenblicklich ändern sich sämtliche Buchstaben und Zeichen auf dem Bildschirm in die gewählte Farbe (z.B. COLOR 1 = alle Buchstaben auf dem Bildschirm werden in schwarzer Farbe dargestellt).

Seien Sie aber vorschtig, wenn Sie COLOR 4 eingeben. Nach Drükken der Eingabetaste können Sie dann nämlich gar nichts mehr sehen: die Vordergrundfarbe und die Hintergrundfarbe sind nämlich gleich. Am besten, Sie schalten in diesem Fall den Computer erst einmal aus und dann wieder an - schon ist alles wieder in Ordnung (dieses ist eine eher rabiate Methode. Sie können anstattdessen auch mit SHIFT die Taste F1 auf Ihrem Computer betätigen - so kommt die alte lesbare Farbzusammenstellung wieder zustande).

Sie können in der COLOR-Anweisung aber nicht nur die Buchstaben- oder die Vordergrundfarbe angeben, genauso ist dies mit der Hintergrundfarbe möglich. Setzen Sie zu diesem Zweck hinter die Zahl der Vordergrundfarbe einfach ein Komma und geben anschließend vor dem Drücken der Eingabetaste eine zweite Zahl ein - diese gilt dann für die Hintergrundfarbe.

So würde z.B.

```
COLOR 1,15
```

bedeuten: Vordergundfarbe bzw. Zeichenfarbe 1 (schwarz) und Hintergrundfarbe 15 (weiß).

Wie Sie gemerkt haben, wird durch den COLOR-Befehl 'nur' ein Farbregister umgestellt. Der Bildschirm wird weiterhin mit den BASIC-Kommandos CLS oder durch Drücken der CLS/HOME-Taste (gemeinsam mit SHIFT) gelöscht.

5. Nun aber ran ans Programm, damit wir den Befehl FOR ... NEXT verstehen können

Sie erinnern sich noch an unser voriges Kapitel, in dem wir feststellten, daß jeder Variablenname besonderen Vorschriften entsprechend aufgebaut sein muß? Z.B., so teilte ich Ihnen mit, müßte das erste Zeichen jedes Variablennamens ein Buchstabe zwischen A und Z sein.

Sie werden sich vielleicht gefragt haben: Warum ist das so? Warum können wir als erstes Zeichen des Variablennamens nicht eine Ziffer zwischen 0 und 9 eingeben?

Probieren wir das doch einfach einmal aus. Geben Sie ein:

```
1A=33
```

Irgendwie stimmt da was nicht! Nachdem Sie die Eingabetaste gedrückt haben, erscheint nämlich keine Ok-Meldung (wie sonst nach jeder Variableneingabe), sondern der Bildschirmzeiger springt lediglich an den Beginn der nächsten Zeile.

Fragen wir nun den Wert der Variable 1A mit

PRINT 1A

ab, bekommen wir als Ergebnis zwei Zahlen: die 1 (wie Sie der Computer in dieser Pseudovariablen an der ersten Stelle liest) und anschließend noch eine 0, die den Wert der bisher nicht mit einem Zahlwert belegten Variable A repräsentiert.

Wo aber ist unsere Variable 1A geblieben? Geben Sie dazu bitte einen neuen Befehl aus dem MSX-Vokabular ein: LIST und drücken anschließend die Eingabetaste. Hier erscheint auf einmal wieder unsere Pseudevariable '1A', aber nun wird die '1' vom 'A' durch ein Leerzeichen getrennt.

Was war geschehen? Ihr MSX-Computer hat sich die Zahl zu Beginn Ihrer Eingabe als Zeilennummer eines richtigen Programms merken wollen. Nun steht in Zeile 1 der Befehl A=33.

Mit dem Kommando

LIST

(hier brauchen Sie sicherlich keine Übersetzung - das Programm soll halt nur gelistet = auf dem Bildschirm abgebildet werden) haben wir uns das Programm angeschaut, mit dem Befehl

RUN

(laß das Programm laufen) plus Eingabetaste können sie es nun starten und somit ablaufen lassen.

Unser Einzeiler hat nur den Bruchteil einer Sekunde zum Ablauf gebraucht und schon wieder erscheint die Meldung Ok.

Sie können nun spaßeshalber mit

```
PRINT A
```

noch einmal prüfend abfragen, ob die Variable A wahrlich den Wert 33 angenommen hat. Es stimmt.

Um den Befehl FOR ... NEXT (für ... das nächste bitte) zu verstehen, haben wir diese Einführung zum Programmschreiben benötigt; eine etwas lange Vorgeschichte. Dafür nun sofort ans Werk: Geben Sie folgendes Miniprogramm in Ihren MSX-Computer ein (drücken Sie bitte hinter jeder Zeile die Eingabetaste):

```
10 FOR A=1 TO 10
20 PRINT A
30 NEXT A
```

Sie sehen, wir haben unseren Befehl FOR ... NEXT ein klein wenig auseinandergerissen: Während in Zeile 10 das Wörtchen FOR auftaucht, befindet sich NEXT in Zeile 30.

Was tut dieses Programm? Am besten, Sie prüfen es selbst einmal, indem Sie den Startbefehl RUN eingeben.

Hui ... 10 Zahlen wurden wie wild auf den Bildschirm geschrieben. Wie ist es dazu gekommen?

In Zeile 10 steht FOR A=1 TO 10: Hiermit geben wir unserem MSX-Computer den Befehl, der Variablen A Werte zwischen 1 und 10 zuzuweisen. Zu Beginn des Programnms hat die Variable A somit den Wert 1.

Nun springt das Programm weiter in Zeile 20: mit PRINT A wird veranlaßt, daß der aktuelle Wert von Variable A auf dem Bildschirm ausgegeben wird, mehr nicht.

In Zeile 30 schließt sich der Kreis: 30 NEXT A. Hier wird mit dem Kommando NEXT ausgesprochen, daß Ihr MSX-Computer wieder zum Kommando FOR zurückspringen soll, allerdings nun für A den nächsthöheren Zahlwert, in unserem Beispiel '2', einsetzen soll.

Danach springt das Programm wieder in Zeile 20 (der aktuelle Zahlwert der Variablen A, im Augenblick=2, wird wieder auf dem Bildschirm ausgegeben), usw., bis schließlich die Ziffer 10 erscheint und es nun kein NEXT mehr gibt (es sollte ja nur FOR A=1 TO 10 gezählt werden).

So einfach ist das. Ändern Sie das Programm ein klein wenig ab, um die Quadratzahlen der Zahlwerte 1 bis 10 zu erfahren: Schreiben Sie dazu in Zeile 20:

```
20 PRINT A*A
```

(Sie können auch die Schreibmarke mit Hilfe der Pfeiltasten in die bereits bestehende Zeile 20 bewegen und dort hinter die Variable A noch die Zeichen '*A' eingeben). Damit sich Ihr MSX-Computer diese geänderte Zeile merkt, müssen Sie wieder die Eingabetaste drücken.

Betätigen Sie anschließend SHIFT und CLS/HOME-Taste und schon kann mit LIST das geänderte Programm angeschaut und mit RUN gestartet werden. Wenn alles richtig funktioniert hat, müßten nun auf dem Bildschirm die Quadratzahlen von 1 bis 10 ausgegeben werden.

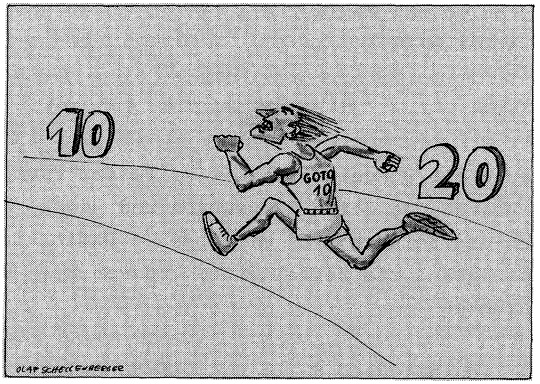
6. Mit NEW können wir unseren Speicher wieder löschen, nicht nur von Variablenwerten, auch von Programmzeilen

Der FOR ... NEXT-Befehl mit dem Überbau, wie denn nun eigentich Programme geschreieben werden können, war schon ein harter Brokken. Deshalb jezt wieder ein leichterer Befehl:

Schauen Sie sich Ihr soeben erstelltes Programm noch einmal mit LIST an, denn gleich wollen wir es aus dem Speicher entfernen. Geben Sie den Befehl NEW (übersetzt: neu beginnen) ein und betätigen die Eingabetaste. Wie bereits vorhin ein paar mal, erfolgt wieder die lapidare Auskunft Ihres MSX-Computers: Ok. Oben auf dem Bildschirm ist das Listing zwar noch zu sehen, aber wenn Sie jetzt noch einmal LIST•eingeben, merken Sie, daß das Programm nicht mehr auf dem Bildschirm erscheint.

Drücken Sie bitte die CLS/HOME-Taste oder geben den Bildschirm-Löschbefehl CLS ein. Wenn Sie jetzt noch einmal versuchen, das Programm zu listen, werden Sie mir bestätigen: NEW hat Ihr erstes Programm aus Ihrem MSX-Computer-RAM auf Nimmerwiedersehen verschwinden lassen.

7. Mit GOTO läuft unser Programm unendlich lang



Eines der primitivsten und geradezu sinnlosesten Programme, das doch immer wieder von Computerfreaks voller Stolz auf den Rechnern eingegeben wird, besteht aus nur zwei Zeilen.

Damit Sie diesen Freaks Paroli bieten können, geben Sie folgenden Zweizeiler doch einfach einmal in Ihren MSX-Computer ein (vorher löschen Sie bitte den Speicher mit NEW):

```
10 PRINT "MSX";
20 GOTO 10
```

Wenn Sie dieses Programm starten und den Text von hier an nicht mehr weiterlesen, können Sie das Programm nicht mehr anhalten und müßten somit Ihren MSX-Computer ausstellen.

Was besagen die zwei Zeilen?

Zeile 10: Mit PRINT wird das Zauberwort MSX auf dem Bildschirm ausgegeben.

Erinnern Sie sich noch, was das Semikolon hinter dem zweiten Anführungszeichen für eine Bedeutung hat?

Richtig, dadurch wird die nächste Ausgabe auf dem Bildschirm direkt hinter die letzte Ausgabe gesetzt und nicht gleich in die darauffolgende Bildschirmzeile. Auf den ersten Blick braucht uns das nicht zu berühren, denn dies ist der einzige PRINT-Befehl und damit die einzige Bildschirmausabe in unserem Miniprogramm, ... wenn da nicht noch in Zeile 20 stehen würde: GOTO 10 (übersetzt: gehe zur 10).

Mit 10 ist hier nämlich die Zeilennummer 10 gemeint, d.h. das Programm fängt wieder neu an und druckt so erneut 'MSX' auf dem Bildschirm aus (direkt neben das 'X' der letzten Bildschirmausgabe von 'MSX') und dies immer und immer wieder, bis wir halt den Netzschalter drücken.

Doch dies ist eine sehr unelegante Möglichkeit, das Programm zu unterbrechen. Viel einfacher gelingt dies, indem Sie entweder CTRL (oder CONTROL)- und STOP-Taste oder aber CTRL (CONTROL)- und die C-Taste gleichzeitig betätigen. Je nachdem, wo sich das Programm gerade befindet, wird dann auf dem Bildschirm angezeigt: 'Break in 10' bzw. 'Break in 20' (Unterbrechnung des Programmablaufs in Zeile 10 oder in Zeile 20).

Wir könnten das Proramm anschließend auch von der gleichen Stelle forfahren lassen, indem wir nun den Befehl CONT (CONT ist die Abkürzung für Continue, was soviel bedeutet wie: weitermachen) eingeben.

Der Befehl, den wir in diesem Kapitel aber eigentlich kennenlernen wollten, GOTO, dürfte durch unser Miniprogramm klargeworden sein: Wir können mit GOTO direkt an den Anfang von jeder beliebigen Programmzeile springen.

Ähnlich wie die BASIC-Vokabel GOTO klingt auch unser nächster Befehl:

8. Mit GOSUB springen wir hin und mit RETURN wieder zurück

Programmierer schimpfen oft auf BASIC gerade aus dem Grund, weil es den Befehl GOTO gibt. Mit diesem Befehl sei es somit leider möglich, 'Spagghettiprogramme' zu schreiben (alles kreuz und quer durcheinander).

Warum wird auf GOTO so geschimpft?

Als Programmierer setzt man sich im Normalfall an den Schreibtisch und überlegt Zeile für Zeile, wie das gewünschte Programm in allen seinen Bestandteilen später zu funktionieren hat. Ist man fertig mit dieser zu Papier gebrachten Vorstellung, gibt man sie Zeile nach Zeile in den MSX-Computer ein und startet das Programm mit RUN.

Im Idealfall läuft das Programm nun nicht nur einwandfrei, sondern es wurden auch alle möglichen WÜnsche bei der Programmierung mitberücksichtigt. Ein Idealfall, wie er vielleicht nur im Buche steht.

Die meisten Möchtegern-Programmierer hingegen setzen sich an Ihren Computer und beginnen ohne große Überlegungen zu programmieren. Kurze Zeit später fällt ihnen ein, daß hier noch etwas fehlt (GOTO) und da auch (GOTO) und schließlich dort auch (GOTO).

Am Tag des Programmschreibens finden sie sich noch in ihrem selbsterstellten Durcheinander zurecht, aber wehe dem, die Fortsetzung der Eingabe soll erst zwei Tage später stattfinden oder - noch viel schlimmer -: ein Fremder soll dieses Progrmm weiterschreiben. Unmöglich, wenn dort solch ein GOTO-Durcheinander herrscht. Die Struktur wurde aufgegeben und nun weiß der Programmierer schließlich nicht mehr ein noch aus.

Besser geht all dies, wenn man sich vorher das Programm in Ruhe am Schreibtisch entwickelt hat. Dort werden die Bausteine nach und nach aneinandergefügt und fertig ist das Endprodukt.

Bezogen auf unseren Zweizeiler von vorhin würde das bedeuten: Die Bildschirmausgabe der Buchstabenkombination MSX ist ein eigenes Programm, ein sogenanntes Unterprogramm, das vom Hauptprogramm abgegrenzt wird.

```
10 GOSUB 100
20 RUN
100 PRINT "MSX";
110 RETURN
```

Zwar wurde unser Zweizeiler nun zu einem Vierzeiler, aber die Zeilen 100 und 110 zeigen uns deutlich an, daß sich hier ein Unterprogramm befindet.

Der RETURN-Befehl in Zeile 110 schließlich besagt: Kehre an die dem GOSUB-Befehl folgende Zeile im Programm zurück und setze von dort aus die Programmausführung fort (Zeile 20).

In unserem kleinen Programm wirkt die Übersicht durch dieses Unterprogramms vielleicht noch nicht so stark. Schauen Sie aber in unsere größeren Programme, die wir für verschiedene Anwendungen im nächsten Kapitel starten und schon werden Sie mit mir einer Meinung sein: Unterprogramme erleichtern die Kontrolle über den Programmablauf erheblich!

9. REM-Zeilen erinnern uns - sie werden vom Computer aber einfach übersehen

Nun noch zu unseren letzten zwei Standardbefehlen, bevor wir beginnen, die ersten größeren Programme zu schreiben und zu verstehen:

Wenn Sie an den Beginn einer Zeile die Buchstaben REM schreiben, übersieht Ihr MSX-Computer diese Zeile und fährt mit dem Programm eine Zeile tiefer fort.

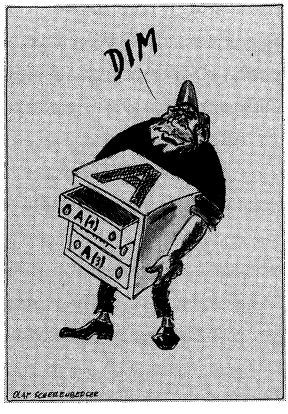
Für uns haben REM-Zeilen allerdings eine sehr wichtige Bedeutung: wir können uns hinter der Buchstabenfolge REM Dinge notieren, die bei der Lesbarkeit unserer Programme sehr hilfreich sind.

Verändern wir unser GOSUB-Beispielprogramm von vorhin mit Hilfe der REM-Zeilen:

```
10 REM Programm zur GOSUB-Demonstration
20 REM Sprung ins Bildschirm-Ausgabe-Unterprogramm
30 GOSUB 100
40 REM Programmneustart
50 RUN
100 REM Bildschirm-Ausgabe-Unterprogramm
110 PRINT "MSX";
120 RETURN
```

Selbst wenn Sie dieses Programm noch nach Jahren lesen, werden Sie schnell verstehen, was hier geschehen ist. Die REM (Abkürzung für Remark = Erinnerung)-Zeilen helfen uns dabei, unsere Programme übersichtlicher zu gestalten.

10. Wir können zwar nur vier Variablen A1 nennen, mit DIM geben wir aber trotzdem eine Anzahl von viel mehr A1-Variablen vor



Sie erinnern sich bestimmt noch an das Kapitel zum Thema Schubladen bzw. Variablen im MSX-Computer. Zwar hatten wir dabei festgestellt, daß wir vier verschiedene Variablen mit dem gleichen Namen verwenden können (String $ für Texte, doppelte Genauigkeit mit #, einfache Gebnauigkeit mit ! und ganze Zahlen mit %), aber halt nicht mehr.

Mit dem Befehl DIM hingegen legen wir für eine Variable eine sogenannte Dimension (Ausbreitung) fest, so daß wir denselben Variablennamen auch 100- oder 1000mal mit unterschiedlichen Werten verwenden dürfen, aber halt immer mit einer Indexzahl versehen, die zusätzlich angegeben werden muß.

Diese Indexzahl wird nach der Dimensionierung einfach in Klammern dem Variablennamen hinzugefügt. DIM MSX(1000) zu Beginn des Programms eingegeben würde heißen, daß es 1000 verschiedene Variablen mit dem Namen MSX gibt, die mit unterschiedlichen Zahlwerten belegt werden können: MSX(1), MSX(2) ... bis MSX(1000).

Vorteil dieser Methode ist es, für bestimmte Programmzusammenhänge nicht immer wieder neue Variablennamen suchen zu müssen, sondern den Grundvariablennamen beizubehalten. Außerdem können wir auf diese Art und Weise direkt im Programm verschiedene Variableninhalte abrufen, ohne jedes Mal einen neuen Variablennamen definieren zu müssen d.h. Sie könnten für die Indexzahl wiederum eine Variable verwenden, die Sie mit Hilfe einer FOR ... NEXT-Schleife langsam aber sicher hochsetzen.

Beispiel: Gib alle 100 Namen aus, die abgespeichert wurden:

```
10 DIM MSX$(100)
...
100 FOR N=1 TO 100
110 PRINT MSX$(N)
120 NEXT N
```

Vorausgesetzt, wir hätten 100 verschiedene Namen in der dimensionierten Variable A$ abgespeichert, würde dieser Vierzeiler zur Ausgabe reichen.

Anders ohne die Möglichkeit der Dimensionierung: Wir müßten alle 100 Variablen einzeln im Programm, Zeile für Zeile, zum Ausdruck bringen:

```
10 PRINT A$
20 PRINT A1$
30 PRINT A2$
...
```

Eine Tortur ohnegleichen.

Nun haben wir in diesem Kapitel zwar nicht alle BASIC-Vokabeln kennengelernt, aber immerhin doch schon einmal die entscheidenden Schlüsselwörter.

Wir dürften in unserem Grundwortschatz nun so sicher sein, daß wir die folgenden Programme (es sind nie mehr als 30 Zeilen pro Programm) verstehen können.

Wollen Sie einen Schritt weitergehen, so schauen Sie sich doch die ausführlichen Programme in meiner 'MSX-Programmsammlung' und in 'MSX Grafik & Sound' (ebenfalls bei DATA BECKER erschienen) etwas genauer an.

Nun aber in medias res. Auf den folgenden Seiten finden Sie BASIC-Programme, die sich fast voll und ganz aus dem Vokabelschatz zusammensetzen, den wir im letzten Kapitel abgehandelt haben.

An manchen Stellen werden Sie feststellen, daß Ihnen dieser oder jener Befehl vorher in diesem Buch noch nicht begegnet ist. In diesem Fall bin ich im Erläuterungstext zum Programm dann noch einmal entsprechend tiefgehender darauf eingegangen.

Für jemanden, der BASIC als seine alltägliche Programmiersprache bezeichnet, bieten die abgedruckten Listings nichts besonderes. Für diese Leute (zu denen Sie ja in Zukunft ganz sicherlich nun auch gehören werden) habe ich zwei weitere MSX-Computerbücher bei DATA BECKER veröffentlicht: MSX Programmsammlung und MSX Grafik & Sound. Dort finden Sie z.B. sowohl Programme, die aus Ihrem MSX-Computer eine kleine Orgel machen, als auch Programme zur Sporttabellenerstellung, Plattenverwaltung sowie zahlreiche Spiele usw. (es sind fast 40 mehr oder weniger sehr lange Programme - manche sind länger als 20 Seiten!).

Das Problem bei solch langen Programmen sehe ich vor allem darin, daß die Computerneulinge sich ohne jegliches Wissen von BASIC (das Sie ja nach dem Studieren dieses Buches bereits in größeren Zügen vorweisen können) an diese seitenlangen Listings herantrauen, ohne auch nur einen Bruchteil davon zu verstehen, was sie dort mühsam über mehrere Seiten abschreiben müssen. Kein Wunder, daß dann die Fehlersuche fast unmöglich wird, denn - davor sind auch Sie nicht gefeit - Tippfehler begehen wir alle. Nur das Problem beim Abtippen von Programmlistings ist halt, daß ein Computer meist auch nicht den kleinsten Tippfehler verzeiht.

Nun aber zu den Programmen in diesem Buch.

Noch zwei Hinweise vorweg:

1. Wenn Sie die Programme in Ihren MSX-Computer eingeben, vergessen Sie bitte nicht, am Ende jeder Zeile die Eingabetaste zu betätigen.

2. Bei den Erläuterungen der Programme Zeile für Zeile habe ich die REM-Zeilen jeweils übersprungen, denn was soll ich Ihnen zu Zeilen etwas erklären, die selbst nichts anderes tun, als Erklärungen auszusprechen.

1. Telefonbuch

Programm Nummer 1 soll Ihnen zeigen, wie Ihr MSX-Computer als Telefonbuch eingesetzt werden kann.

```
10 REM Telefonbuch
20 REM Festlegen des Stringspeicherplatzes
30 CLEAR 2000
40 CLS
50 INPUT "Wie viele Telefonnummern ";A
60 REM Dimensionierung fuer Nummern und Namen
70 DIM NA$(A),NU$(A)
80 CLS
90 REM Aufbau des Telefonverzeichnisses
100 FOR N=1 TO A
110 INPUT "Name ";NA$(N)
120 INPUT "Nummer ";NU$(N)
130 PRINT
140 NEXT N
150 REM Suche von Telefonnummern
160 CLS
170 INPUT "Name bitte eingeben ";A$
180 FOR N=1 TO A
190 REM Name gefunden, so Ausgabe der Nummer
200 IF A$=NA$(N) THEN GOSUB 230
210 NEXT N
220 GOTO 160
230 REM Unterprogramm: Ausgabe der Nummer
240 PRINT "Die Telefonnummer: ";NU$(N)
250 PRINT
260 INPUT "Bitte Eingabetaste druecken ";B$
270 RETURN
```

Das Programm läuft folgendermaßen ab:

Erst werden Sie nach der Anzahl der einzugebenden Teilnehmer gefragt. Anschließend führen Sie die Eingaben durch und schließlich findet Ihnen Ihr MSX-Computer all die Nummern der Telefonteilnehmer, von denen Sie vorher Namen eingegeben hatten.

Nun schauen wir uns das Programm Zeile für Zeile an:

Zeile 30: Hiermit wird Speicherplatz für die Stringvariablen reserviert, in die Sie nach dem Start des Programms mit RUN die zu speichernden Gesprächsteilnehmer und deren Telefonnummern eingeben können.

Ich habe absichtlich diesen Speicherplatz (2000 Buchstaben) nicht zu groß gewählt, denn bei einer durchschnittlichen Namens- und Telefonnummernlänge von insgesamt 20 Zeichen, werden Ihnen 100 Teilnehmereintragungen wohl reichen. Bedenken Sie nämlich: Dieses Programm bietet keine Möglichkeit der Abspeicherung der eingegebenen Daten auf Kassette oder Diskette. Beschränken Sie sich deshalb bei der Eingabe, sonst war schließlich alle Arbeit umsonst.

Zeile 40: Der Bildschirm wird gelöscht.

Zeile 50: In der Schublade A wird die Anzahl der im folgenden einzugebenden Telefonnummer und somit auch die Anzahl der dazugehörenden Gesprächsteilnehmer abgelegt.

Zeile 70: Hier wird nun nicht nur die Anzahl der Variablen für die Telefonnummer (die Stringvariable NU$(A)) sondern auch die der Teilnehmer (die Stringvariable NA$(A)) festgelegt. Obgleich die Telefonnummer eine Zahl ist, mußten wir wegen des Querstriches (0251/47478) und der ersten Stelle (0) eine Stringvariable wählen.

Zeile 80: Erneutes Bildschirmlöschen.

Zeile 100-140: Abfrage des Teilnehmernamens (Zeile 110) und seiner Telefonnummer (Zeile 120).

Da hier nicht nur ein Teilnehmer sondern etliche mehr abgefragt werden sollen (laut Ihrer Eingabe in die Schublade A - siehe Zeile 50), wird die Form der FOR ... NEXT-Schleife zum mehrmaligen Durchlaufen der gleichen Funktion (aber mit unterschiedlichen Variableninhalten) gewählt.

Die Abfrage der Daten findet so oft statt, bis der Wert von Variable A erreicht ist.

Zeile 160-170: Nach dem Bildschirmlöschen wird nach dem Namen des gewünschten Teilnehmers gefragt. Dieser Name wird in der Variable A$ abgespeichert.

Zeile 180-220: Nun werden erneut mit einer FOR ... NEXT-Schleife sämtliche bereits abgespeicherten Namen untersucht. Wenn einer dieser Namen (NA$(N)) dem Namen des gewünschten Teilnehmers zu 100% entspricht, verzweigt das Programm ins Unterprogramm in Zeile 230 ff..

Wenn hingegen die Suche erfolglos verlief, läßt der Programmschritt in Zeile 220 die gleiche Abfrageprozedur von neuem starten.

Zeile 230-270: In dieses Unterprogramm wird nur dann verzweigt, wenn bereits die Übereinstimmung der Variablen A$ (gewünschter Teilnehmer) und NA$(N) (abgespeicherter Teilnehmer) eindeutig festgestellt wurde. In diesem Fall wird in Zeile 240 die Ausgabe der gewünschten Telefonnummer veranlaßt.

Zeile 260 erfüllt die Funktion, daß erst nach Drücken der Eingabetaste mit dem weiteren Programmablauf fortgefahren werden kann, sonst könnte es passieren, daß Ihr MSX-Computer sofort wieder ins Hauptprogramm zurückkehrt und dabei den Bildschirm löscht (Zeile 160), ohne daß Sie die Ausgabe der Telefonnummer auf dem Bildschirm wahrnehmen konnten.

2. Vokabelprogramm

Hiermit ist es möglich, daß Sie Ihr MSX-Computer Vokabeln abfragt.

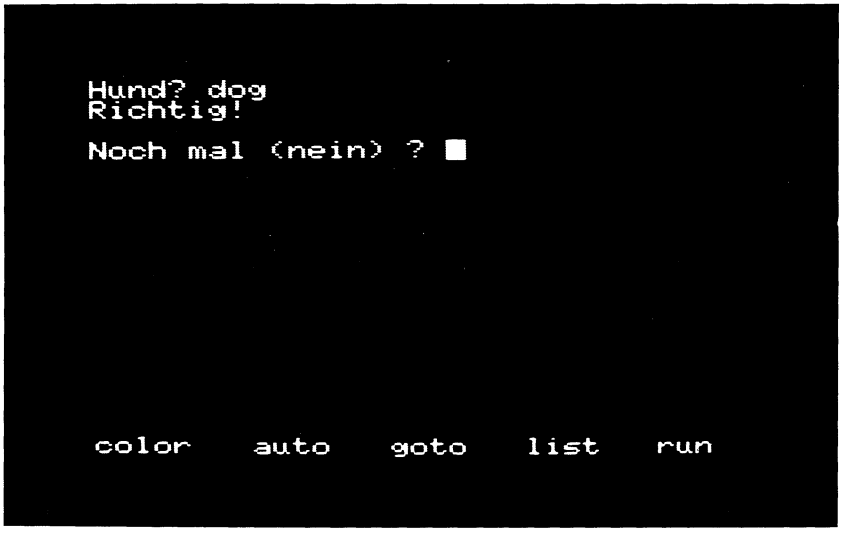
```
10 REM Vokabelprogramm
20 CLEAR 2000
30 CLS
40 INPUT "Wie viele Vokabeln ";A
50 DIM A$(A),B$(A)
60 CLS
70 FOR N=1 TO A
80 INPUT "Deutsche Bedeutung ";A$(N)
90 INPUT "Englische Bedeutung ";B$(N)
100 PRINT
110 NEXT N
120 CLS
130 INPUT "Deutsch - Englisch (ja) ";B$
140 IF B$="ja" THEN GOSUB 190 ELSE GOSUB 260
150 PRINT
160 INPUT "Noch mal (nein) ";B$
170 IF B$="nein" THEN END
180 GOTO 120
190 B=RND(1)*A+1
200 CLS
210 PRINT A$(B);
220 INPUT AN$
230 IF AN$=B$(B) THEN PRINT "Richtig!"
240 IF AN$<>B$(B) THEN PRINT "Falsch!"
250 RETURN
260 B=RND(1)*A+1
270 CLS
```

```
280 PRINT B$(B);
290 INPUT AN$
300 IF AN$=A$(B) THEN PRINT "Richtig!"
310 IF AN$<>A$(B) THEN PRINT "Falsch!"
320 RETURN
```

Das Programm läuft folgendermaßen ab: Zuerst werden Sie nach der Anzahl der zu speichernden Vokabeln gefragt und anschließend dazu aufgefordert, die englische und die dazu passende deutsche Bedeutung einzutragen.

Danach können Sie auswählen, ob Ihnen Ihr MSX-Computer die deutsche oder die englische Vokabel zur Abfrage vorlegen soll.

Schließlich sucht er Ihnen mit seinem Zufallsgenerator Vokabeln heraus, die Sie daraufhin - möglichst richtig - in der anderen Sprache eingeben sollen.



Nun schauen wir uns das Programm Zeile für Zeile an:

Zeile 20: Hiermit wird Speicherplatz für die Stringvariablen reserviert, in die Sie nach dem Start des Programms mit RUN die zu speichernden Vokabeln eingeben sollen.

Ich habe absichtlich diesen Speicherplatz (2000 Buchstaben) nicht zu groß gewählt, denn bei einer durchschnittlichen Länge pro Vokabel von 20 Buchstaben, werden Ihnen 100 Vokabeleintragungen wohl reichen. Bedenken Sie nämlich: das Programm bietet keine Mögklichkeit der Abspeicherung der eingegebenen Daten auf Kassette oder Diskette. Beschränken Sie sich deshalb bei der Eingabe, sonst war schließlich alle Arbeit umsonst.

Zeile 30: Löschen des Bildschirms

Zeile 40: In der Schublade A wird die Anzahl der im folgenden einzugebenden Vokabeln und somit auch die Anzahl der dazugehörenden Übersetzungen abgelegt.

Zeile 50: Hier wird nun nicht nur die Anzahl der Stringvariablen für die deutschen Bedeutungen (A$(A)) sondern auch die der englischen Bedeutungen (B$(A)) festgelegt.

Zeile 60: Erneutes Bildschirmlöschen.

Zeile 70-110: Abfrage der deutschen Bedeutung (Zeile 80) und der englischen Bedeutung (Zeile 90).

Da hier nicht nur eine Vokabel, sondern etliche mehr abgefragt werden sollen (laut Ihrer Eingabe in die Schublade A - siehe Zeile 40) wird die Form der FOR ... NEXT-Schleife zum mehrmaligen Durchlaufen der gleichen Funktion (aber mit unterschiedlichen Variableninhalten) gewählt. Die Abfrage der Daten findet so oft statt, bis der Wert von Variabe A erreicht ist.

Zeile 120-130: Nach dem Löschen des Bildschirms werden Sie gefragt, ob Sie 'Deutsch - Englisch' abgefragt werden wollen (in diesem Fall bitte 'ja' eingeben) oder ob Sie lieber 'Englisch - Deutsch' abgefragt werden wollen (in diesem Fall etwas anderes als 'ja' eingeben).

Ihre Antwort wird in der Stringvariablen B$ abgespeichert.

Zeile 140: Hier wird nun entschieden, in welches Unterprogramm Ihr MSX-Computer verzweigen soll. Haben Sie in Zeile 130 mit 'ja' geantwortet, dann springt Ihr MSX-Computer in das Unterprogramm in Zeile 190 (Abfrage 'Deutsch - Englisch'), andererseits (Abfrage 'Englisch - Deutsch' wird gewünscht) springt er in das Unterprogramm in Zeile 260.

Wie Sie sehen ist hier unser bereits bekannter Befehl IF ... THEN erweitert worden um die Funktion ELSE. Was bedeutet ELSE? Wir können uns die Programmzeile 140 folgendermaßen ins Deutsche übersetzen: Wenn B$="ja" dann springe ins Unterprogramm in Zeile 190, andererseits springe ins Unterprogramm in Zeile 260.

Bei unserer 'wenn ... dann'-Abfrage gibt es immer zwei Möglichkeiten: Trifft die wenn-Aussage zu, wird direkt der Befehl hinter dem Wörtchen 'dann' ausgeführt; ist dies nicht der Fall, wird der Befehl hinter dem Wörtchen ELSE (übersetzt: andererseits) ausgeführt. Fehlt ELSE in einer IF-Anweisung und trifft die wenn-Aussage nicht zu, wird das Programm in der nächsten Programmzeile fortgesetzt.

Zeile 150-180: Nach der Vokabelabfrage wird der Übersichtlichkeit halber eine Leerzeile auf dem Bildschirm erzeugt (Zeile 150), nachgefragt, ob weiteres Vokabellernen erwünscht wird (Zeile 160) und aufgrund dieser Eingabe entweder das Programm beendet (Zeile 170 - der neue Befehl END besagt: Hier hat der Computer die Ausführung abzubrechen) oder mit der Abfrage fortgefahren werden soll (Zeile 180).

Zeile 190-210: Wieder taucht ein neuer Befehl auf: RND(1). Er erzeugt eine Zufallszahl, die zwischen 0 und 1 liegt.

Mit der Formel in Zeile 190 wird der Zufallswert so multipliziert (und um 1 erhöht), daß alle eingetragenen Vokabeln mit gleicher Wahrscheinlichkeit abgefragt werden können.

Danach wird in Zeile 200 der Bildschirm gelöscht und in Zeile 210 die durch Zufall herausgesuchte Vokabel ausgegeben.

Das Semikolon hinter dem PRINT-Befehl besagt, daß die Antwort auf die Frage direkt hinter der letzten Bildschirmausgabe beginnen kann.

Zeile 220-250: Hier wird nun überprüft, ob die Frage richtig beantwortet wurde (Übereinstimmung der Variablen AN$ und B$(B)) oder ob die Antwort falsch gegeben wurde (Zeile 240: keine Übereinstimmung zwischen den Variablen AN$ und B$(B)). Daraufhin wird wieder in das Hauptprogramm zurückverzweigt (Zeile 250).

Zeile 260-280: Mit der Formel in Zeile 260 wird der Zufallswert so multipliziert (und um 1 erhöht), daß alle eingetragenen Vokabeln mit gleicher Wahrscheinlichkeit abgefragt werden können. Danach wird in Zeile 270 der Bildschirm gelöscht und in Zeile 280 die durch Zufall herausgesuchte Vokabel ausgegeben. Das Semikolon hinter dem PRINT-Befehl besagt, daß die Antwort auf die Frage direkt hinter der Ausgabe der Bedeutung gestartet wird.

Zeile 290-320: Hier wird nun überprüft, ob die Frage richtig beantwortet wurde (Übereinstimmung der Variablen AN$ und A$(B)) oder ob die Antwort falsch gegeben wurde (Zeile 310: keine Übereinstimmung zwischen den Variablen AN$ und A$(B)). Daraufhin wird wieder in das Hauptprogramm zurückverzweigt (Zeile 320).

3. Das Schachbrettproblem

Programm Nummer 3 soll Ihnen zeigen, wie Ihr MSX-Computer eine größere Rechenaufgabe in wenigen Sekunden lösen kann.

```
10 REM Das Schachbrettproblem
20 REM 64 Felder hat das Schachbrett
30 FOR N=1 TO 64
40 PRINT "Feld ";N
50 REM Potenzformel
60 A=2^(N-1)
70 PRINT A;"Reiskoerner"
80 REM In Schublade B liegt aller Reis
90 B=B+A
```

```
100 PRINT "Insgesamt ";B;" Reiskoerner"
110 PRINT
120 NEXT N
```

Das bekannte Schachbrettproblem stützt sich auf eine Geschichte, in der einem Geschäftsmann die Wahl gelassen wurde, entweder einen größeren Geldbetrag zu zahlen oder 'einfach' ein Schachbrett mit Reiskörnern nach einer mathematischen Regel zu füllen. Die Reiskörner sollten auf den 64 Feldern des Schachbretts nach folgender Regel verteilt werden: Auf das erste Feld lege 1 Reiskorn, auf das zweite 2 Reiskörner, auf das dritte Feld 4 Reiskörner, auf das vierte Feld 8 Reiskörner usw..

Auf den ersten Blick scheint das Problem sehr einfach zu lösen zu sein, denn wer weiß schon, wieviel Reiskörner in einer Pakkung stecken - da kommt es doch auf eins mehr oder weniger nicht an?! Wenn Sie sich allerdings am Ende des Programmablaufs auf dem Bildschirm anschauen, wie groß die Zahl der Reiskörner angewachsen ist, können Sie sich vielleicht vorstellen, daß diese Zahl noch nicht einmal mit Reiskörnern ausgeglichen werden könnte, wie es sie heutzutage auf der ganzen Welt gibt. Der Geschäftsmann hatte sich also geirrt.

```
Feld  9
 256 Reiskoerner
Insgesamt  511  Reiskoerner

Feld  10
 512 Reiskoerner
Insgesamt  1023  Reiskoerner

Feld  11
 1024 Reiskoerner
Insgesamt  2047  Reiskoerner

Feld  12
 2048 Reiskoerner
Insgesamt  4095  Reiskoerner

Feld  13
 4096 Reiskoerner
Insgesamt  8191  Reiskoerner
■

color   auto   goto   list   run
```

Da Sie bei diesem Programm nur den Befehl RUN einzugeben brauchen und somit lediglich als Beobachter fungieren, will ich an dieser Stelle nichts über den Programmablauf sagen. Dies kann viel besser in der genauen Programminformation, die nun folgt, nachgelesen werden:

Zeile 30: Hier wird eine FOR ... NEXT-Schleife eröffnet, damit alle 64 Schachfelder nacheinander berechnet werden können.

Zeile 40: Diese Zeile veranlaßt die Ausgabe der gerade behandelten Feldnummer auf dem Bildschirm.

Zeile 60: Nach dieser Rechenformel wird die Anzahl der Reiskörner pro Feld bestimmt. Der Pfeil nach oben (↑) ist das Potenzierungszeichen und ist auf Ihrer MSX-Tastatur durch gleichzeitiges Niederdrücken der Tasten SHIFT und 6 zu erreichen.

Zeile 70: Hier wird die Anzahl der Reiskörner ausgegeben, die sich auf dem zur Zeit aktuellen Feld befinden.

Zeile 90: In der Variablen B werden sämtliche Reiskörner abgespeichert und die Werte bereits mit Reiskörnern belegter Felder dazuaddiert.

Zeile 100: Hier erfolgt die Bildschirmausgabe sämtlicher Reiskörner auf dem Schachbrett.

Zeile 110-120: In Zeile 110 wird eine Leerzeile der Übersichtlichkeit halber ausgegeben, während in Zeile 120 der Rücksprung zu Zeile 30 eingeleitet wird, um nacheinander alle 64 Schachfelder zu berechnen.

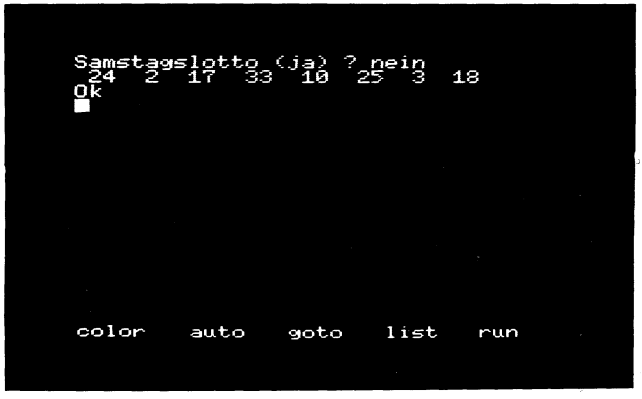
4. Lottozahlen

Programm Nummer 4 soll Ihnen zeigen, wie Ihr MSX-Computer Ihnen dabei hilft (ohne Gewähr!), Ihre Entscheidung für Samstag bzw. Mittwoch zur Auswahl der richtigen Lottozahlen abzunehmen und rechtzeitig in die Wege zu leiten.

```
10 REM Lottozahlen
20 CLS
30 INPUT "Samstagslotto (ja) ";A$
40 IF A$<>"ja" THEN GOTO 180
50 FOR N=1 TO 7
60 A%(N)=RND(-TIME)*49+1
```

```
70 NEXT N
80 FOR N=1 TO 7
90 FOR M=1 TO 7
100 IF N=M THEN GOTO 120
110 IF A%(N)=A%(M) THEN GOTO 50
120 NEXT M
130 NEXT N
140 FOR N=1 TO 7
150 PRINT A%(N);
160 NEXT N
170 END
180 FOR N=1 TO 8
190 A%(N)=RND(-TIME)*38+1
200 NEXT N
210 FOR N=1 TO 8
220 FOR M=1 TO 8
230 IF N=M THEN GOTO 250
240 IF A%(N)=A%(M) THEN GOTO 180
250 NEXT M
260 NEXT N
270 FOR N=1 TO 8
280 PRINT A%(N);
290 NEXT N
300 END
```

Auch in diesem Programm (wie bereits im Vokabelprogramm) wird auf die Zufallsfunktion (RND) der Programmiersprache BASIC zurückgegriffen.



Das Programm läuft folgendermaßen ab: Zuerst werden Sie gefragt, ob Sie das Samstagslotto (6 Zahlen plus Zufallszahl aus 1 bis 49) oder das Mittwochslotto (7 Zahlen plus Zufallszahl aus 1 bis 38) spielen wollen.

Daraufhin werden 7 bzw. 8 Zahlen ausgewählt, die allerdngs vor der Bildschirmausgabe noch genau miteinander verglichen werden, damit nicht zwei oder mehr gleiche Zahlen ausgegeben werden können. Anschließend werden diese Glückszahlen auf den Bildschirm ausgegeben.

Nun schauen wir uns das Programm Zeile für Zeile etwas genauer an:

Zeile 10-30: Nach dem Löschen des Bildschirms (Zeile 20) wird abgefragt, ob Sie Samstagslotto oder Mittwochslotto spielen wollen.

Geben Sie in Zeile 30 das Wörtchen 'ja' ein, versteht es Ihr MSX-Computer so: Sie wollen Samstagslotto spielen. Geben Sie irgendetwas anderes in Zeile 30 ein (z.B. 'nein'), versteht es Ihr MSX-Computer so, daß Sie Mittwochslotto spielen wollen und verzweigt deshalb in den Programmteil ab Zeile 180 (ausgehend von der Abfrage in Zeile 40).

Zeile 50-70: Hier werden nun die 7 Zufallszahlen für das Mittwochslotto erzeugt. Zur Abspeicherung haben wir Zahlvariablen gewählt, die nur ganze Zahlwerte annehmen können.

Mit der Funktion RND(-TIME) wird auch wahrlich eine Zufallszahl erzeugt. Dies gelingt folgendermaßen: Ihr MSX-Computer hat eine eingebaute Uhrvariable, die sich ständig ändert und durch den Variablennamen TIME abgefragt werden kann. So haben wir jedes Mal eine unterschiedliche Uhrzeit und damit jedesmal eine andere Zahl.

Die Multiplikation der Variablen A%(N) mit 49 geschieht selbstverständlich aus dem Grund, weil ja Zahlen zwischen 1 und 49 ausgewählt werden sollen.

Zeile 80-130: In diesem Zeilenblock werden sämtliche erzeugten Zufallszahlen miteinander vergleichen. Sind zwei Zufallszahlen gleich (Zeile 110), werden neue Zufallszahen erzeugt, indem in Zeile 50 zurückgesprungen wird.

Zeile 140-170: Hier findet die Ausgabe der sieben unterschiedlichen Zahlen statt, die inklusive der Zufallszahl zur Eingabe auf dem Lottoschein verwendet werden können. Viel Glück!

Zeile 180-200: Hier werden nun die 8 Zufallszahlen für das Mittwochslotto erzeugt. Zur Abspeicherung haben wir Zahlvariablen gewählt, die nur ganze Zahlwerte annehmen können.

Die Multiplikation der Variablen A%(N) mit 38 geschieht selbstverständlich aus dem Grund, weil ja Zahlen zwischen 1 und 38 ausgewählt werden sollen.

Zeile 210-260: In diesem Zeilenblock werden sämtliche erzeugten Zufallszahlen miteinander vergleichen. Sind zwei Zufallszahlen gleich (Zeile 240), werden neue Zufallszahlen erzeugt, indem in Zeile 180 zurückgesprungen wird.

Zeile 270-300: Hier findet die Ausgabe der acht unterschiedlichen Zahlen statt, die inklusive der Zufallszahl zur Eingabe auf dem Lottoschein verwendet werden können. Viel Glück!

5. Umrechnungsprogramme

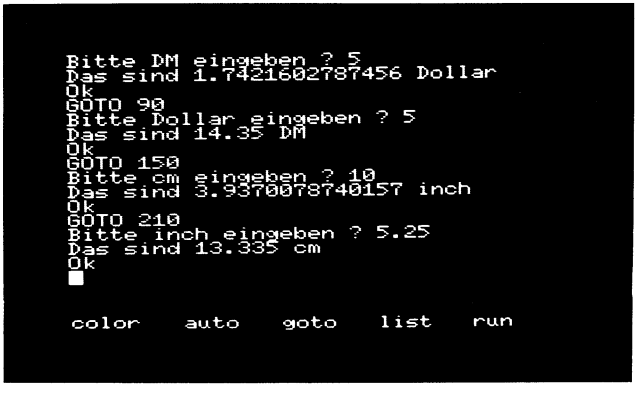
Programm Nummer 5 besteht eigentlich aus 4 einzelnen Programmen. In diesen Programmen wird Ihnen gezeigt, wie Ihr MSX-Computer sich Umrechnungen merken kann: Von DM in Dollar, von Dollar in DM, von cm in inch und von inch in cm. Die Reihe ließe sich beliebig fortsetzen.

```
10 REM Umrechnungsprogramm DM-Dollar
20 CLS
30 INPUT "Bitte DM eingeben ";A
40 B=A/2.87
50 PRINT "Das sind";B;"Dollar"
```

```
60 END
70 REM Umrechnungsprogramm Dollar-DM
80 CLS
90 INPUT "Bitte Dollar eingeben ";A
100 B=A*2.87
110 PRINT "Das sind";B;"DM"
120 END
130 REM Umrechnungsprogramm cm-inch
140 CLS
150 INPUT "Bitte cm eingeben ";A
160 B=A/2.54
170 PRINT "Das sind";B;"inch"
180 END
190 REM Umrechnungsprogramm inch-cm
200 CLS
210 INPUT "Bitte inch eingeben ";A
220 B=A*2.54
230 PRINT "Das sind";B;"cm"
240 END
```

Sie können jedes Programm einzeln starten, indem Sie mit GOTO die entsprechende Zeilennummer eingeben und danach die Eingabetaste betätigen.

1. Umrechnung DM in Dollar: RUN
2. Umrechnung Dollar in DM: GOTO 90
3. Umrechnung cm in inch: GOTO 150
4. Umrechnung inch in cm: GOTO 210



Nun schauen wir uns das Programm Zeile für Zeile an:

Zeile 20-60: Nach dem Löschen des Bildschirms (Zeile 20) wird nach dem gewünschten DM-Betrag gefragt. Dieser Betrag wird durch den aktuellen Dollarstand - im Augenblick, in dem ich diese Zeilen schreibe, hat der Dollar den Wert 2.87 DM - geteilt.

Schließlich erfolgt in Zeile 50 die Bildschirmausgabe des Dollarwertes.

Zeile 80-110: Nach dem Löschen des Bildschirms (Zeile 80) wird nach dem gewünschten Dollar-Betrag gefragt. Dieser Betrag wird mit dem aktuellen Dollarstand multipliziert.

Schließlich erfolgt in Zeile 110 die Bildschirmausgabe des DM-Betrags.

Zeile 140-180: Nach dem Löschen des Bildschirms (Zeile 140) wird nach dem gewünschten cm-Wert gefragt. Dieser Wert wird durch den Inchmaßstab (2.54) geteilt.

Schließlich erfolgt in Zeile 170 die Bildschirmausgabe des Inchwertes.

Zeile 200-240: Nach dem Löschen des Bildschirms (Zeile 200) wird nach dem gewünschten Inch-Wert gefragt. Dieser Wert wird mit dem Inchmaßstab (2.54) multipliziert.

Schließlich erfolgt in Zeile 230 die Bildschirmausgabe des cm-Wertes.

Steuerbefehl für die MSX-Musikprogrammiersprache

Neben einer Vielzahl von Sound- und vor allen Dingen Grafikbefehlen bietet das MSX-BASIC auch zwei eigene programmierbare Sprachen, die uns die Ansteuerung jedbeliebiger Effekte mit Sound- und Grafikprozessor sehr erleichtern:

Für Grafik die GML (Graphic Macro Language)
Für Sound die MML (Music Macro Language)

Während die Grafiksprache mit dem Befehl DRAW arbeitet, den wir an anderer Stelle in diesem Buch ausführlich erläutert haben, bedient sich die hier besprochene Musiksprache des Befehls PLAY, was soviel bedeutet wie: 'Spiele Noten'.

Sowohl die Grafik- als auch die Musiksprache arbeiten mit Strings, was bedeutet, daß die nach dem Befehl PLAY bzw. DRAW folgenden Befehlszeichen in Anführungsstriche eingeschlossen werden müssen. Dies jedoch hat den großen Vorteil, daß wir die einmal definierten Klänge verschiedenen Variablen zuweisen können.

Sämtliche Teilbefehle, die in dem dem Befehl PLAY nachfolgenden String enthalten sein können, müssen nur im Ausnahmefall durch Semikolons bzw. Kommata voneinander abgetrennt werden. Wenn diese Abtrennung erfolgen muß, verweisen ich selbstverständlich in diesem Kapitel auf die zwingende Notwendigkeit. Ist dies aber nicht ausdrücklich vorgeschrieben, können Sie alle Zeichen im PLAY-String direkt ohne Leerzeichen hintereinander schreiben. Der Übersichtlichkeit halber haben wir aber in diesem Kapitel zusätzliche Leerzeichen eingefügt, jedoch können Sie auch alles ohne Leerzeichen hintereinander schreiben und sparen dadurch natürlich im Laufe von größeren Programmen viel Speicherplatz.

So wie wir das bereits in der Schule gelernt haben, so können wir unserem MSX-Computer auch Töne entlocken. Geben Sie

```
PLAY "CDE"
```

ein und es erklingen die Noten C,D und E. Bis zu 255 Noten können Sie auf diese Art nacheinander in einen String schreiben.

Sollen gar zwei Tonreihen parallel zueinander erklingen, müssen Sie nur die Tonreihen wie zwei einzelne Strings voneinander durch ein Komma trennen:

```
PLAY "CDE","DEF"
```

und es erklingen parallel zueinander die Tonpaare CD, DE und EF. Auch hier wäre es theoretisch möglich, daß Sie 255 Töne nacheinander parallel zueinander durch einen PLAY-Befehl erklingen lassen.

Die Höhe der Gefühle bei den MSX-Computern heißt letztendlich Dreiklang. Der dritte Tonkanal wird hierbei wiederum von den zwei anderen Tonkanälen durch ein weiteres Komma getrennt:

```
PLAY "C","E","G"
```

Nicht mehr der Rede wert ist an dieser Stelle erneut der Hinweis: bis zu 255 Noten lassen sich parallel zueinander durch jeweils einen PLAY-Befehl pro Musikkanal abspielen!

Zur Verfügung stehen uns beim Gebrauch des PLAY-Kommandos die allgemein gebräuchlichen Tonbezeichnungen:

```
C D E F G A B
```

Was bei einem Klavier die schwarzen Tasten sind (jeweils ein halber Ton höher bzw. tiefer als die benachbarten weißen Tasten) sind in der Music Macro Language die Plus- bzw. Minuszeichen.

Eine vollständige Tonleiter ohne Halbtöne würde so aussehen:

```
PLAY "CDEFGAB"
```

und im Vergleich dazu eine Tonleiter mit sämtlichen Halbtönen:

```
PLAY "CC+DD+EE+FF+GG+AA+BB+"
```

Ihnen wird aufgefallen sein, daß hier etwas nicht stimmt und da sehen Sie ganz richtig:

1) Der Ton "E+" entspricht dem Ton F
2) Der Ton "B+" entspricht dem Ton C

Genau sieben Ganztöne und fünf Halbtöne lassen sich in einer Oktave unterbringen und ein erhöhtes "E" gibt es halt nicht, anstatt dessen ist die nächsthöhere und damit gleichwertige Note das "F".

Wollen wir über das "B" hinaus einen Halbton höher spielen, gibt es wiederum keinen durch eine entsprechend schwarze Taste zu erzeugenden Ton; der nächste mögliche Ton ist halt wieder das "C", allerdings um eine Oktave höher als die, in der wir gerade musizieren.

Der Vollständigkeit halber hier unsere richtige Oktave und noch eine andere Möglicheit, wie wir eine Oktave samt allen möglichen Halbtönen erklingen lassen können:

```
1) PLAY "CC+DD+EFF+GG+AA+B"
2) PLAY "CD-DE-EFG-GA-AB-B"
```

Wie in der musikalischen Notation üblich, können wir an Stelle des Pluszeichens (für eine Halbtonerhöhung) auch ein Kreuzchen setzen, so daß unsere Tonfolge von vorhin auch so aussehen kann:

```
1) PLAY "CC#DD#EFF#GG#AA#B"
```

Wollen wir einen Halbtonschritt anzeigen, muß das entsprechende Plus-, Minus- oder Kreuzchenzeichen, wie Sie den aufgeführten Beispielen entnehmen können, immer hinter dem Notennamen stehen, andernfalls würde sich das Zeichen auf die vorherige Note (falls vorhanden) beziehen.

Nicht alle möglichen Halbtöne ergeben zu den bereits vorhandenen Tönen eine neue Tonhöhe. Wie steht es denn mit der Tatsache, daß wir zu Anfang in unserer Tonfolge für das erhöhte "B" wieder ein tiefes "C" erhielten?

Wir müssen unserem MSX-Computer also auch Bescheid darüber geben, in welcher Oktave der Ton jeweils zu erklingen hat; in unserem Beispiel vorhin war dazu keine Angabe erfolgt, so daß der Computer seinen Standardwert übernommen hat d.h.: Spiele in Oktave 4. In Wirklichkeit können wir aber über gleich 8 Oktaven verfügen, die wir im PLAY-Befehl durch das Einfügen des Buchstabens "O" und einer Zahl zwischen 1 und 8 deutlich machen. Probieren wir also einfach mit diesem neu dazugelernten PLAY-Kommando, das erhöhte "B" ("B+") auch wahrlich erhöht d.h. in einer Oktave höher als "C" wiederzugeben:

```
PLAY "CC+DD+EFF+GG+AA+B O5 B+"
```

Wunderschön, es klappt ... und es klappt noch einmal gespielt nicht mehr! Gehen Sie mit der Schreibmarke auf dem Bildschirm hoch und lassen die Melodie ein weiteres Mal erklingen: Diesmal wird die gesamte Tonfolge in Oktave 5 gespielt, was heißt, daß unsere letzte Note wieder zum tiefen "C" zurückkehrt. Ihr MSX-Computer hat sich sofort nach Lesen des "O5"-Befehls eingeprägt, daß von nun an alle Töne in Oktave 5 zu erklingen haben. Wir könnten nun einerseits Oktave 5 durch Oktave 6 ("O5" wird zu "O6") ersetzen, aber das wäre nicht unbedingt im Sinne des Erfinders, weil es von nun an (bis Oktave 8) immer höher hinaus geht.

Besser gelingt es uns, wenn wir vor die zuerst gespielten Töne die Oktavennummer 4 einfügen, während vor dem zuletzt gespielten Ton die Angabe "O5" (eine Oktave höher als "O4") beibehalten wird:

```
PLAY "O4 CC+DD+EFF+GG+AA+B O5 B+"
```

Nun ist es richtig und Sie können nach Belieben die Oktaven wechseln (von Oktave 1, die mehr oder minder brummig klingt, bis hinauf zu Oktave 8 in den höchsten Tönen). Es sei Ihnen anheimgestellt, ob Sie nun vor jedem Notennamen die Oktave durch Einfügen einer entsprechenden "O"-Zahl verändern, oder aber nur einmal vor der dazu bestimmten Notenfolge. Ihr MSX-Computer akzeptiert selbstverständlich beide Möglichkeiten.

Wir hatten zu Beginn dreistimmige Kompositionen (Dreiklang) ertönen lassen. Der eingegebene Oktavenwert gilt in jedem Fall, wie auch die im weiteren Verlauf dieses Kapitels angegebenen Tonveränderungen, immer nur für den einen gerade angewählten Tonkanal. Soll die Änderung in sämtlichen Tonkanälen stattfinden, müssen wir sie auch entsprechend dort einzeln anbringen: Somit würden also folgende 3 Tonkanäle den selben Dreiklang in verschiedenen Tonhöhen erzeugen:

```
PLAY "O4 C O5 C O6 C","O4 E O5 E O6 E","O4 G O5 G O6 G"
```

Merken Sie sich aber in jedem Fall: Wenn Sie ohne Oktavenänderungseingaben in einer Oktave bleiben wollen, springen Sie über das "B" (denn "B+" ist das tiefe "C") und das "C" ("C-" ist das hohe "B") nie hinaus.

Für die Notennamen-Unkundigen ist es auch möglich, daß Sie in der Music Macro Language entsprechend der Tonhöhe mit Zahlen arbeiten können: Die Zahlen werden in dem PLAY-Befehl durch ein vorangestelltes "N" (für Number = Nummer) eingegeben. Sie können sämtliche Noten aller zur Verfügung stehenden Oktaven also auch mit Zahlen anspringen, d.h., das "C" der ersten Oktave ist Nummer 1 ("N1") und das "B" der achten = höchsten Oktave Nummer 96 ("N96"). Da jede Oktave aus 12 Tönen besteht ("CC+DD+EFF+GG+AA+B") und wir über 8 Oktaven verfügen könnnen, gibt es nachweislich - wie auch schon angegeben - 8*12=96 Töne. Zur Orientierung: Unser "C" in Oktave 4 ist "N36" und die gesamte Oktave würde mit Zahlangabe so aussehen:

```
PLAY "N36 N37 N38 N39 N40 N41 N42 N43 N44 N45 N46 N47"
```

Sie werden sich nun sicherlich fragen, warum man zu der übersichtlichen Eingabe der Notennamen auch noch diese recht unübersichtliche Zahleingabe benötigt. Dies hat mehrere Gründe:

1) Wir brauchen uns um keine Oktaveneingabe zu kümmern. Die Eingabe erfolgt absolut mit Zahlwerten, was soviel heißt wie: Je höher die Zahl, desto höher auch der Ton.

2) Wenn wir im weiteren Verlauf dieses Kapitels kennenlernen, wie ein String vom Computer bearbeitet werden kann (Aufteilung in bzw. Zusammensetzen durch PLAY-Teilstrings), ist es allemal einfacher, mit Zahlwerten zu arbeiten als mit Notennamen, die dem Computer in ihrer Reihenfolge nicht so leicht zu erklären sind (die Folge "CDEFG" ist ja noch eindeutig und errechenbar, da sie dem Alphabet folgt, wie sieht es aber mit den nachfolgenden Notennamen "A" und "B" aus?). Hier haben wir Menschen diese willkürliche Folge auswendig gelernt bis sie uns in Fleisch und Blut überging. Einem Computer (der wohlgemerkt immer nur so schlau ist, wie die Leute, die ihn programmiert haben) ist gar nicht so leicht zu erklären, daß auf eine G wieder ein A folgt.

Bisher spielte unser MSX-Computer die Noten mit jeweils gleich langen Tönen wie ein Kind, das seine ersten Geigentöne beigebracht bekommt. Variieren läßt sich aber ein Ton nicht allein in der Tonhöhe, sondern auch in der Tonlänge: Jeder Ton erklingt nach dem Einschalten des Computers als eine Viertelnote, was mit dem nun zu besprechenden PLAY-Befehl "L" heißen würde:

```
PLAY "L4"
```

Während "L4" für eine Viertelnote steht, können wir auch "L8" vor einen Notennamen setzen, was soviel bedeutet wie: die folgenden Noten (bis zur nächsten Tonlängenänderung durch "L") sind Achtelnoten. "L1" würde bedeuten: Ganze Noten, "L2" halbe Noten ... bis hin zu "L64" = Vierundsechzigstel Noten (eine ganze Note besteht in diesem Fall aus 64 Noten mit der Tonlänge 1/64).

In unserem folgenden Beispiel erklingen sämtliche drei Tonkanäle geichlang, aber alle Töne haben halt eine andere Tonlänge:

```
Kanal 1 = 8mal 1/8 Note
Kanal 2 = 4mal 1/4 Note
Kanal 3 = 1mal 1/1 Note
```

Um den Unterschied deutlich und damit hörbar zu machen, weisen wir jedem Kanal eine andere Oktave zu:

```
PLAY "O4 L8 CDECDECD","O5 L4 CDEC","O6 L1 C"
```

Ähnlich wie beim Oktavenbefehl, merkt sich unser MSX-Computer auch die Tonlänge bis zur nächsten Änderung des Tonlängenwerts. Sollen Tonlängenänderungen aber nur für einzelne Töne gelten, lassen wir die "L"-Angabe weg und geben anstatt dessen hinter dem Notennamen die gewünschte Tonlänge ein. Also hätten folgende Töne die gleiche Tonhöhe und -länge:

```
PLAY "C32"
PLAY "L32 C"
```

Im zweiten Beispiel wird durch "L32" vorgegeben, daß auch alle zukünftigen Töne (bis zur nächsten "L"-Eingabe) ebenfalls als 32stel Noten zu erklingen haben.

Es ist auch möglich, den Buchstaben für Länge "L" bei einzelnen Notenlängenänderungen wegzulassen. In diesem Fall müssen wir allerdings zur Abtrennung zwischen den Zahlwerten (einerseits für Tonhöhe, andererseits für Tonlänge) ein Semikolon setzen:

```
Eingabe von Notennamen: PLAY "C8 DE"
Eingabe von Notennummern: PLAY "N36;8 N38 N40"
```

Wir können die Länge einer Note auch noch auf eine andere Weise verlängern: Wie in der Musikschreibweise besagt ein Punkt, der einer Note folgt, daß dieselbige um die Hälfte länger zu spielen ist: So werden aus einer ganzen Notenlänge eineinhalb Notenlängen, aus einer halben Notenlänge eine dreiviertel usw:

```
PLAY "L4 C." = eine Dreiviertelnte
PLAY "L1 C." = eine ganze und eine halbe Note
```

Zum Musikmachen brauchen wir auch die Möglichkeit, keine Musik zu machen. Dieses Unterbrechen (Pause, im Englischen heißt Pause "Rest") werden in unseren Tonfolgen durch den Buchstaben "R" (für Englisch = "Rest") gekennzeichnet. Dem "R" folgt wiederum eine Zahlangabe zwischen 1 (für eine ganze Pause) und 64 (für eine vierundsechzigstel Pause):

```
PLAY "C R4 DE"
```

Ton "C" erklingt, eine Viertel Pause lang ist Stillschweigen und schließlich hören wir die Töne "D" und "E".

Besonders sinnvoll ist die Eingabe von extrem kurzen Pausenwerten, wenn mehrmals der gleiche Ton in Folge erklingt: Während sich im folgenden Beispiel 4 Töne wie ein langer Ton anhören:

```
PLAY "CCCC"
```

bewirkt die kurze Pause ("R64"), daß wir sämtliche 4 Töne in ihrem Anschlag voneinander unterscheiden können:

```
PLAY "C R64 C R64 C R64 C"
```

Würden Sie dieses Vorgehen jedoch bei der Eingabe mehrstimmiger Musik heranziehen, hätten Sie alsbald unter der Asynchron-Wiedergabe der drei Tonkanäle zu leiden.

Wir können auch den Buchstaben "R" ohne Zahlangabe in den String einbauen. So nimmt der Buchstabe "R" den ihm zugeordneten Standardwert ein, was soviel bedeutet wie: Die Pause dauert eine viertel Notenlänge.

Bevor wir zur direkten Änderung des Tones (Hüllkurve usw.) kommen, müssen wir uns noch mit dem Tempo und der Lautstärke und ihrer Bedeutung auseinandersetzen: Bisher haben wir Viertel-, Halb- oder gar Vierundsechzigstel-Noten kennengelernt und dabei so getan, als wenn z.B. eine Viertelnote in jedem Musikstück so und so viele Bruchteile einer Sekunde zu erklingen habe. Dies stimmt aber so nicht, denn das vorgegebene Tempo allein (rhythmische Bewegung des Dirigenten mit seinem Taktstock) bestimmt auch die Schnelligkeit des gespielten Musikstückes und seiner Noten.

Befindet man sich allerdings in einem vorgegebenen Tempo, haben die Noten relativ zueinander wieder die gleiche Länge:

eine halbe Note ist genauso lang wie zwei Viertelnoten
eine ganze Note ist so lang wie acht Achtelnoten usw.

Unser MSX-Computer hat beim Einschalten den Wert 120 für das Tempo vorgegeben. Diesen Wert können wir jederzeit zurückrufen durch Eingabe des Buchstabens "T" ohne eine darauf folgende Zahleingabe. Andernfalls ist es möglich, das Tempo unseres Musikstückes zu erhöhen (bis "T255") oder halt zu verlangsamen (bis "T32"). Schauen bzw. hören wir uns dies an folgenden Beispielen einmal an:

```
PLAY "T CDE"        Tempo = 120 = normal
PLAY "T240 CDE"     Tempo = 240 = schnell
PLAY "T60 CDE"      Tempo = 60 = langsam
```

Die Lautstärke unserer Musik läßt sich durch Eingabe des Buchstabens "V" (für Englisch "Volume" = Lautstärke) und einer darauf folgenden Zahl zwischen 0 und 15 eingeben. Geben wir keinen "V"-Befehl nach dem Einschalten des Computers ein oder setzen lediglich "V" ohne Zahlangabe in einen Musikstring, erklingen die darauf folgenden Töne bis zur nächsten Lautstärkeänderung mit "V"=8 bzw. Lautstärke=8.

Lauter werden die Töne bei Einfügung von "V" bis 15, leiser bei Einfügung von "V" bis 1 bzw. "V" bis 0, was bedeuten würde: kein Ton erklingt:

```
PLAY "V CDE"        Lautstärke = 8 = normal
PLAY "V15 CDE"      Lautstärke = 15 = laut
PLAY "V1 CDE"       Lautstärke = 1 = leise
PLAY "V0 CDE"       Lautstärke = 0 = aus
```

Wir können jeden Ton mit Hilfe von verschiedenen Hüllkurven in seinem Klang verändern. Zum Beispiel ist es möglich, daß ein Ton nicht sofort erklingt, sondern langsam lauter wird (in seiner Lautstärke bis zum Höchstpunkt ansteigt). Eine andere Möglichkeit wäre die, einen Ton kurz nacheinander auf- und abschwingen zu lassen.

Um diese Tonänderungen durchzuführen, haben wir mit dem Buchstaben "S" (für Englisch "Shape") acht verschiedene dieser sogenannten Hüllkurven zur freien Verfügung. Um eine Hüllkurve erklingen zu lassen, ist es sinnvoll, die Notenlänge auf 1 (L1=ganze Note) einzustellen, die V(olume=Lautstärke)-Eingabe vollkommen zu übergehen (Ihr MSX-Computer arbeitet nämlich bei Hüllkurvenanwahl stetig mit V16) und natürlich noch ein paar Töne hinzuzufügen (damit überhaupt etwas zu hören ist, denn die Hüllkurve erklingt nicht ohne Ton sondern nur mit bzw. durch den Ton). Folgende Eingabe wäre z.B. denkbar:

```
PLAY "S10 L1 CDEFG"
```

Hören Sie sich folgende Hüllkurven einmal genauer an:

S1, S4, S8, S10, S11, S12, S13, S14

Die Töne "C", "D" und "E" durch eine Hüllkurve zu einem Hubschrauberschnattergeräusch verändert würde beispielsweise so aussehen:

```
PLAY "S8 CDE"
```

oder das Anschlagen von Trommeln:

```
PLAY "S1 CDE"
```

Sie können zudem jede Hüllkurve noch in ihren Feinheiten verändern, indem Sie eine besondere Abstimmung mit Hilfe des Buchstabens "M" und einer darauf folgenden Zahl zwischen 1 und 65535 vornehmen.

So bekommen unsere "Trommeln" vom vorherigen Beispiel ein wenig mehr Volumen und Klang, wenn wir sie folgendermaßen ansprechen:

```
PLAY "S1 M2000 CDE"
```

Damit haben wir alle Möglichkeiten aufgeführt, um den PLAY-Befehl, oder sagen wir lieber "die PLAY-Programmiersprache" sinnvoll zu nutzen. Der Übersichtlichkeit halber haben wir hier noch einmal alle Parameter aufgeührt:

C D E F G A B	Tonname
+ - # '	Tonhöhenveränderung um einen halben Ton
O	Auswahl zwischen 8 Oktaven
N	Zahlanwahl für Tonhöhe (1 bis 96)
L	Tonlänge (1 = 1/1 bis 64 = 1/64)
.	Tonlänge um die Hälfte verlängern
R	Pause (1 = 1/1 bis 64 = 1/64)
T	Tempo (32 bis 255)
V	Lautstärke (0 bis 15)
S	Hüllkurve (1,4,8,10,11,12,13,14)
M	Abstimmung der Hüllkurven (1 bis 65535)

Es ist Ihnen sicherlich schon aufgefallen, daß Ihr MSX-Computer eine Tonfolge (auch gleichzeitig in drei Stimmen) spielen kann, während die Schreibmarke bereits zur weiteren Eingabe bereit ist. Dies hängt ganz einfach damit zusammen, daß Ihr MSX-Computer einen extra Speicher besitzt, der sich eine Tonfolge von bis zu 24 Tönen (dreistimmig) merken und suksessive abspielen kann, ohne den weiteren Programmablauf zu stören. Wird allerdings dieses Merklimit überschritten, kann es lange dauern, bis Ihr Programm in normaler Geschwindigkeit weiterläuft.

Außerdem wird Ihnen, wie bereits zu Beginn dieses Kapitels kurz darauf hingewiesen, aufgefallen sein, daß wir es bei den PLAY-Befehlen mit normalen Strings zu tun haben. So ist es möglich, wie im folgenden Programm gezeigt, vorgegebene Strings einfach der Reihe nach zu spielen:

```
10 A$="CDE"
20 B$="FGA"
30 PLAY A$,B$
```

Wir können diese Strings aber auch in bestehende PLAY-Anweisungen direkt einbinden:

```
PLAY "CDE X B$;"
```

Hierbei muß der Unterstring durch ein "X" eingeleitet und durch ein ";" (Semikolon) abgeschlossen werden.

Wir können auch direkt Variablen in unseren PLAY-Befehl vom Programm her einlesen (ähnlich wie bei der Einbindung des Unterstrings geben Sie vor der Variablen ein "=" ein und schließen den Variablennamen mit einem ";" ab), z.B. erklingen anschließend sämtliche 96 Töne von "N1" bis "N96":

```
10 FOR N=1 TO 96     20 PLAY "N =N;"     30 NEXT N
```

Schneller läuft das Programm, wenn wir sowohl das Tempo als auch die Tonlänge auf die schnellstmöglichen Werte festlegen:

```
20 PLAY "T255 L64 N =N;"
```

In Zeile 20 wurde die übernommene Variable "N" (aus der FOR ... NEXT - Schleife von Zeile 10) durch ein Gleichheitszeichen und ein Semikolon, beides zur Abtrennung vom übrigen String, umgeben.

Noch ein Hinweis: Durch die Tastenkombination CTRL und STOP zum Abbruch eines Programms sowie durch das BEEP-Kommando werden alle PLAY-Parameter (Oktave, Lautstärke ...) wieder auf ihren Urzustand (Oktave = 4, Lautstärke = 8 ...) zurückgesetzt. Das ist manchmal dazu ganz gut zu wissen, wenn Sie nicht mehr wissen, wie Sie aus Ihrer PLAY-Abänderung herauskommen sollen.

Die PLAY-Programmiersprache ist nur ein Soundbefehl Ihres MSX-Computers, den Sie hiermit kennengelernt haben. Dieser Abschnitt wurde als Beispiel fast wortwörtlich aus meinem Buch 'MSX Grafik & Sound' (ebenfalls bei DATA BECKER erschienen) übernommen. Dort finden Sie auch recht lange Programme, die den von Ihrem MSX-Computer erzeugten Sound gebrauchen (z.B. das Programm: Musik auf Klavier).

Zwar stecken hinter dem DRAW-Befehl, wie nicht zuletzt die Länge dieses Kapitels beweist, eine Vielzahl von Unterkommandos. Diese Unterkommandos dienen lediglich einem Zweck: dem Zeichnen von Linien. Trotzdem kommt dieses Befehlsspektrum einer echten Grafikprogrammiersprache wie LOGO nur um Bruchteile nahe (lediglich das LOGO-Unterprogramm Schildkrötengrafik ist mit dem Befehl DRAW der MSX-Computer nachvollziehbar).

Nach dieser Untertreibung der Fähigkeiten des DRAW-Befehls (der Hersteller des MSX-BASIC, die amerikanische Softwarefirma Microsoft, spricht beim Befehl DRAW von der GML = Graphic Macro Language) gehen wir ran in medias res und schauen uns diesen Befehl mal etwas genauer an:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 U8 R8 U8 R8 D16 L16"
40 GOTO 40
```

Bevor wir auf die in diesem Programm verwendeten DRAW-Befehle genauer zu sprechen kommen, sei darauf hingewiesen, daß es sich bei der Verwendung des DRAW-Befehls um ein Tätigsein in der Grafikebene handelt. Grafik läßt sich allerdings nur auf SCREEN 2 und SCREEN 3 abbilden, andernfalls gibt es die Fehlermeldung: Illegal function call.

Gehen wir die Zeichen hinter dem DRAW-Befehl Teil für Teil durch: Die Anführungszeichen verraten uns, daß es sich hier um einen String handelt (ähnlich wie A$="Hans"). Im Augenblick ist das noch nicht wichtig, aber warten Sie bitte einen Augenblick ab, dann werden Sie die Möglichkeiten der Stringbehandlung bei DRAW noch zu schätzen lernen.

Zu Beginn teilen wir unserem MSX-Computer mit, daß er mit seinem Grafikcursor zum Bildschirmmittelpunkt (128, 96) gehen soll. Mit 'BM 128,96' erreichen wir genau dies; das 'B' bedeutet ausgeschrieben BLIND, was so viel heißt wie: ziehe aber keine Linie auf dem Bildschirm, bis du Koordinate 128,96 erreicht hast, während das 'M' ausgeschrieben eigentlich heißen würde: MOVE d.h. bewege den Grafikcursor von seiner letzten Position aus zu

der nachstehenden Koordinate hin (128, 96).

Ausgeschrieben würde 'B M 128,96' also heißen: BLIND MOVE TO 128,96 (Bewegung des Cursors ohne Zeichnung zur Koordinate 128,96).

Haben Sie selbst noch mäßige Erinnerungen an das Erlernen der englischen Sprache in der Schule, wird Ihnen das Übersetzen der vier Richtungen nicht gerade sehr schwer fallen:

nach oben = Up
nach unten = Down
nach links = Left
nach rechts = Right

Da wir Zeit und Speicherplatz bei der Eingabe sparen wollen, reicht es unserem MSX-Computer, wenn wir nur den Anfangsbuchstaben der entsprechenden Richtung eingeben und zwar

'U' für Up
'D' für Down
'L' für Left
'R' für Right

Zu jeder Richtungsangabe braucht unser MSX-Computer nur noch eine Zahl, um zu wissen, wie weit die Reise denn gehen soll. Dabei besagt z.B. die Eingabe 'R8' schlicht und einfach: gehe 8 Bildschirmpunkte nach rechts.

Obgleich wir nun erst ein paar der DRAW-Unterbefehle kennengelernt haben, können wir nun unser Beispiel deuten:

```
DRAW "B M 128,96 U8 R8 U8 R8 D16 L16"
```

Gehe an Koordinatenposition 128,96, ohne bis dahin eine Linie zu zeichnen; bewege den Cursor 8 Punkte aufwärts, dann 8 Punkte nach rechts, noch einmal 8 Punkte aufwärts, wiederum 8 Punkte nach rechts und von dort aus 16 Punkte nach unten und 16 Punkte nach links, fertig ist unsere kleine Grafik, eine Treppe.

Zur Schreibweise hier nur so viel: Damit Sie alles besser lesen können, habe ich in diesem Kapitel Leerzeichen zwischen die einzelnen DRAW-Unterbefehle gesetzt. Wenn Sie wollen, können Sie darauf gerne verzichten; Sie können auch Semikolons an jeder Stelle dort einsetzen, wo wir ein Leerzeichen zur Abtrennung gebraucht haben. Nur an einer Stelle muß ein Komma zur Abtrennung eingegeben werden: Wenn eine Bildschirmkoordinate (X,Y) mit zwei Zahlen angesprochen wird, muß zwischen diesen Zahlen zur Abtrennung ein Komma gesetzt werden.

Gehen wir weiter in unseren DRAW-Unterbefehlen: Wir brauchen uns nicht nur mit den vier Himmelsrichtungen zufrieden zu geben, wir können auch Zwischenwerte verwenden (in der Wetterkunde spricht man dann z.B. von SO oder Südosten bzw. NW oder Nordwesten ...). Da auch diese Richtungen im MSX-BASIC der Einfachheit halber nur aus einem Buchstaben bestehen, haben die Entwickler dieser DRAW-Unterbefehle eine blendende Idee gehabt: Gehen wir das Alphabet vom Buchstaben 'E' angefangen der Reihe nach durch: Die Zwischenhimmelsrichtungen heißen somit einfach:

```
Nord-Ost  = E
Süd-Ost   = F
Süd-West  = G
Nord-West = H
```

Mit Hilfe unserer acht Himmelsrichtungen können wir nun auch ein Haus malen (kennen Sie das noch aus der Kinderzeit? Ein Haus zeichnen, ohne den Stift abzusetzen und ohne eine Linie zweimal zu zeichnen?):

Übersetzen wir uns eine mögliche Reihenfolge beim Zeichnen in den DRAW-Befehl:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 R8 U8 L8 E4 F4 G8 U8 F8"
40 GOTO 40
```

Aber in unserem MSX-Kommando DRAW steckt noch sehr viel mehr drin! Wie wäre es zum Beispiel, wenn wir das dargestellte Häuschen etwas größer oder gar kleiner darstellen wollen? Ganz einfach, werden Sie sagen: Soll das Häuschen doppelt so groß erscheinen, verdoppeln wir ganz einfach die Zahlangaben hinter dem DRAW-Befehl, so daß schließlich in Zeile 30 steht:

```
30 DRAW "B M 128,96 R16 U16 L16 E8 F8 G16 U16 F16"
```

So klappt es wohl auch, ist aber laut der Meinung der Softwareingenieure von MICROSOFT noch zu aufwendig. Anstatt dessen verwenden wir einen weiteren Buchstaben mit Zahlangabe, der den Vergrößerungs- bzw. Verkleinerungsfaktors unserer Zeichnung angibt. Der hierzu verwendete Buchstabe ist das "S", was so viel bedeutet wie Skalenfaktor. Wollen wir in normaler Größe zeichnen (d.h. DRAW "R8", was soviel bedeutet wie: bewege den Grafikcursor um acht Punkte nach rechts), müssen wir Skalenfaktor vier ("S4") anwählen. Für eine doppelt so große Darstellung (Größenverhältnis = 2:1) muß der S-Wert zweimal 4 = 8 betragen (DRAW "S8"). Soll auf die Hälfte verkleinert werden (Größenverhältnis = 1:2) muß der S-Wert 2 betragen (die Hälfte des Ausgangswerts = 4/2 = 2). Wir können dieses Vergrößerungs- bzw. Verkleinerungsspielchen mit Werten von 1 bis 255 und dem Buchstaben "S" weiterführen. Dabei ist es unserem MSX-Computer egal, ob wir über die Bildschirmbegrenzung hinausschießen (was leicht einmal vorkommen kann, wenn wir einen zwei- oder gar dreistelligen Skalenfaktor benutzen). In diesem Fall wird die extrem außerhalb des Bildschirmrahmens liegende Zeichnung am Bildschirmrand angezeigt, wie z.B. in:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 R8 U8 L8 D8"
40 DRAW "B M 128,96 S80 R8 U8 L8 D8"
50 GOTO 50
```

Unterbrechen Sie bitte die Bildschirmdarstellung durch Drücken der Tasten CTRL und STOP und starten Sie es von neuem mit RUN. Zu Ihrem großen Erstaunen wird nun unser Quadrat gar nicht mehr abgebildet, anstatt dessen nur unser übergroßes Quadrat. Wie ist

das nun zu erklären? Wird der Skalenfaktor nicht rechtzeitig wieder zurückgesetzt (in unserem Fall auf Normalgröße = "S4"), bleibt er auch beim Einsatz weiterer DRAW-Befehle erhalten. Dies hat den Vorteil, daß man den Skalenfaktor für oftmaliges Benutzen nicht in jeder Zeile neu definieren muß! Also müßte Zeile 30 in unserem Programm ein klein wenig geändert werden, wollen wir es immer wieder von neuem in seinem Urzustand ablaufen lassen:

```
30 DRAW "B M 128,96 S4 R8 U8 L8 D8"
```

Einen weiteren Buchstaben, den wir in den DRAW-Befehl einsetzen können, ist das A = wie Angel = Winkel. Hiermit ist es möglich, unsere erstellten Zeichen zu drehen und zwar im 90-Grad-Rhythmus:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 R8 U8 R8"
40 GOTO 40
```

Hier ist noch nichts gedreht worden, damit Sie sich merken können, wie unsere Zeichnung im Urzustand ausgesehen hat: eine Treppenstufe, die vom Mittelpunkt her aufwärts führt. Nun verändern wir Zeile 30 mit unserem neuen A-Kommando:

```
30 DRAW "B M 128,96 A1 R8 U8 R8"
```

Wie auch immer man unsere Zeichnung nun deuten will: die Treppenstufe führt abwärts: das Rechts-Kommando wurde ebenso wie das Aufwärts-Kommando um 90 Grad im Uhrzeigersinn gedreht, so daß nun aus der Funktion R (nach rechts) eine Funktion D (nach unten) wurde; ebenso verhält es sich mit der U(aufwärts)-Funktion: Sie wurde auch um 90 Grad im Uhrzeigersinn gedreht, so daß aus aufwärts ein rechts wurde. Also können wir folgende Regel aufstellen:

```
A = 1 d.h. U = R  (aufwärts = rechts)
           R = D  (rechts = abwärts)
+90 Grad   D = L  (abwärts = links)
           L = U  (links = aufwärts)
```

Hätten wir einen Turm gezeichnet, so würde derselbige jetzt auf der ursprünglich rechten Seite flachliegen.

Geben wir höhere Werte für das Kommando A im DRAW-Befehl ein, so vergrößert sich der Winkel immer mehr (im Uhrzeigersinn), jedoch immer nur in 90 Grad-Schritten:

```
A = 2  d.h.   U = D  (aufwärts = abwärts)
              R = L  (rechts = links)
+180 Grad     D = U  (abwärts = aufwärts)
              L = R  (links = rechts)

A = 3  d.h.   U = L  (aufwärts = links)
              R = U  (rechts = aufwärts)
+270 Grad     D = R  (abwärts = rechts)
              L = D  (links = abwärts)
```

Mit Hilfe eines kleinen Unterprogramms ist es gar möglich, daß wir unsere Zeichnungen mit dem DRAW-Befehl auch um jeden beliebigen anderen Winkel drehen.

Wozu brauchen wir denn diesen DRAW-A-Befehl? Haben Sie es gelernt, damit umzugehen, wäre es nun z.B. möglich, sehr einfach einen kleinen Kompaß zu zeichnen. Dafür brauchen Sie nur einmal die Pfeilform zu definieren, und dieselbige dann durch einen vierfach umdefinierten Richtungsfaktor mehrmals zu zeichnen:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 U16 F4 H4 G4"
40 N=N+1
50 ON N GOSUB 70,90,110
60 GOTO 60
70 DRAW "A1"
80 GOTO 30
90 DRAW "A2"
100 GOTO 30
110 DRAW "A3"
120 GOTO 30
```

Obgleich immer von neuem unser Pfeil mit seiner Pfeilspitze nach oben (wie in Zeile 30 angegeben) gezeichnet werden soll, beeinflußt der immer größer werdende A-Faktor im DRAW-Befehl die Richtung letztendlich. Auf diese Art und Weise wird die Richtung verdreht und ein Pfeil nach oben (Norden), einer nach rechts (Osten), einer nach unten (Süden) und einer nach links (Westen) entsteht und hilft uns leicht, das gewünschte Kompaßaussehen zu erstellen. Denken Sie einmal nach, um wie vieles ausführlicher das Programm geworden wäre, wenn jedesmal die Richtung hätte umgedreht (und damit auch umgerechnet) werden müssen.

Wenn wir zu weiteren Unterkommandos des DRAW-Befehls gelangen, wird Ihnen deutlich werden, wie einfach und luxuriös es doch im Endprodukt ist, mit dem Kommando DRAW zu arbeiten, wenn man weiß, wie.

Mit Hilfe des Buchstabens N beauftragen wir unseren MSX-Computer, nach Beendigung seines Zeichenauftrags an den Ursprungspunkt zurückzukehren.

Um diesen Zeichenbefehl praktisch demonstrieren zu können, vereinfachen wir unseren vorhin gezeichneten Kompaß, indem wir die Pfeilspitze weglassen und das seinerzeit so lange Programm ist zu etwas mehr als einem Einzeiler geschrumpft:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96 N A0 U8 N A1 U8 N A2 U8 N A3 U8"
40 GOTO 40
```

Zu Beginn des DRAW-Befehls bekommt der MSX-Computer den Auftrag, bei Bildschirmposition 128,96 (Bildschirmmittelpunkt) mit der Zeichnung zu beginnen. Der anschließend eingegebene Buchstabe N besagt, daß der folgende Zeichenbefehl zwar ausgeführt werden soll, daß aber hiernach wieder zum zuletzt angesprochenen Grafikpunkt (in unserem Fall dem Mittelpunkt bei 128,96) zurückgekehrt werden soll.

Wie wir im vorigen Abschnitt (A-DRAW-Unterbefehl) gelernt haben, besagt die steigende Angabe für A, daß der Befehl U (aufwärts) nacheinander als gedreht um 90, 180 und schließlich 270 Grad verstanden wird. So entsteht letztendlich unsere Windrose.

Wollen Sie es größer haben? Das dürfte, wenn Sie bisher zum DRAW-Kommando alles aufmerksam gelesen haben, kein Problem mehr sein: Setzen Sie ein 'S' für den Skalenfaktor vor die eigentliche Zeichenroutine und geben anschließend eine Zahl größer als 4 ein zum Beispiel:

```
30 DRAW"B M 128,96 S12 N A0 U8 N A1 U8 N A2 U8 N A3 U8"
```

Bereits zu Beginn dieses Kapitels haben wir die DRAW-Unterbefehle M und B angesprochen: Mit M und zwei darauf folgenden Koordinaten (die durch ein Komma voneinander getrennt sein müssen) geben wir an, daß der Grafikcursor sich von der letzten Bildschirmposition dorthin bewegen sollen (M = to move = bewegen), allerdings, indem er in der vorher eingestellten Farbe die Linie zeichnen soll.

Wünschen wir eine Bewegung des Grafikcursors ohne Zeichnung zum angegebenen Zielpunkt, setzen wir vor das M noch ein B, was so viel bedeutet wie: gehe blind zum Zielpunkt = ziehe keine Linie.

Da bei Verwendung von DRAW "B M" nur der Grafikcursor bewegt, nicht aber gezeichnet wird, kann dieser Befehl gut dazu verwendet werden, ihn zur punktweisen Positionierung zu gebrauchen.

Wir können aber noch anders mit den Befehlen M und B umgehen: Wir müssen bei der absoluten Adressierung davon ausgehen, daß unser Bildschirm mit der Position 0,0 in der linken oberen Bildschirmecke beginnt (wollen wir in der Mitte des Bildschirms etwas zeichnen, müssen wir uns im Rahmen der zur Verfügung stehenden Bildpunkte (256 * 192) den Mittelpunkt ausrechnen (128,96) und geben für diese Stelle den Befehl, mit der Zeichnung zu beginnen). Einfach gesagt: absolut heißt, 0,0 ist oben links, 255,191 ist unten rechts und 128,96 ist der Bildschirmmittelpunkt. Bei der relativen Adressierung hingegen bestimmen wir uns einen neuen Anfangspunkt, von wo aus (STEP) nun so und

so viele Schritte nach rechts, oben, links oder unten gegangen werden sollen. Diese relative Adressierung hat den Vorteil, daß wir die bereits erstellten Zeichnungen auch an ganz anderen Bildschirmstellen wieder auftragen können.

Wir verwenden bei der Koordinateneingabe wie gehabt die Zahlwerte, geben aber zwecks relativer Adressierung ein Pluszeichen (+) oder ein Minuszeichen (-) zusätzlich ein. Liest dies unser MSX-Computer, weiß er sofort, daß der ursprüngliche Punkt nun als neuer Nullpunkt anzusehen ist. Als Beispiel wollen wir ganz einfach vier Kästchen nebeneinander zeichnen:

```
10 COLOR 1,15
20 SCREEN 2
30 DRAW "B M 128,96"
40 FOR N=1 TO 4
50 DRAW "B M +0,+8 R4 U4 L4 D4"
60 NEXT N
70 GOTO 70
```

Einfach und einleuchtend, nicht wahr? Verändern wir Zeile 50 so, daß eine Diagonale aus den Kästchen zusammengesetzt wird und zur Abbildung gelangt:

```
50 DRAW "B M +8,+8 R4 U4 L4 D4"
```

Wenn Sie die relative Adressierung erst einmal bei viel komplizierteren Zeichenstrukturen verwenden, werden Sie auch diese Feinheit des MSX-DRAW-Grafikkommandos zu schätzen lernen.

Gab es vorher mißmutige Spötter, die sich über die Bezeichnung, der DRAW-Befehl beinhalte eine vollständige Programmiersprache, aufgeregt haben? Ich hoffe, Sie können sich langsam beruhigen und finden diesen DRAW-Befehl in seiner vielseitigen Anwendungsmöglichkeit nun auch ein klein wenig phaszinierender.

Kommen wir zum nächsten DRAW-Unterbefehl: wir mischen einfach eine der 15 (bzw. 16) Farben in unsere Zeichnung mit ein. Dieser Vorgang wird mit dem Buchstaben "C" (für englisch COLOR = Farbe) und einer nachfolgenden ein- bzw. zweiziffrigen Zahl ausgelöst.

Damit alles wieder aus unserem vorherigen Beispiel in den Ursprungszustand versetzt wird, geben wir zu Beginn den Befehl DRAW "S4 A0" als eine Art Resetknopf ein:

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "S4 A0"
40 DRAW "B M 128,96 C15 R16 C1 U16 C15 L16 C1 D16"
50 GOTO 50
```

Nachdem wir unseren Grafikcursor wieder elegant "mittig" (128,96) positioniert haben, besagt "C15", daß ab jetzt in weißer Farbe gemalt wird und zwar 16 Punkte nach rechts; anschließend wird vor jeder Bewegung jeweils die Farbe geändert, so daß schließlich auf dem Bildschirm zwei schwarze ("C1") und zwei weiße ("C15") Balken unser Quadrat umschließen.

Den gleichen Effekt kann man erzielen, wenn man zwischenzeitlich immer von neuem den COLOR-Befehl benutzt, jedoch hier heißt es, VORSICHT! Andernfalls wird die Schriftfarbe auf dem Textbildschirm halt gleich mitverändert (während der COLOR-Befehl nämlich auf Grafik und Text Einfluß ausübt, gilt dies bei Verwendung besonderer Grafik-Farbe-Unterkommandos halt nur auf die einzelne Grafik bezogen). Schauen wir uns trotzdem einmal an, wie unser obiges Programm unter Hinzuziehung des COLOR-Befehls ausschaut:

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "S4 A0"
40 DRAW "B M 128,96"
50 COLOR 15
60 DRAW "R 16"
70 COLOR 1
80 DRAW "U 16"
90 COLOR 15
100 DRAW "L 16"
110 COLOR 1
120 DRAW "D 16"
130 GOTO 130
```

Ganz schön lang ist das Programm nun geworden, obwohl es nichts anderes tut als unser Programm mit dem DRAW-C-Unterkommando zuvor. Wir müßten gar abschließend noch eine Zeile in der Form

```
COLOR 15
```

einfügen, um zur Grundfarbe zurückzukehren.

Noch eine Warnung! Hüten Sie sich davor, bei Ihrem MSX-Computer willkürlich die Farbe in zu kleinen Punktabständen zu verändern! Die MSX-Computer vertragen zwar 256*192 Grafikpunkte, die Unterscheidung der Farbe spielt sich jedoch nur in einer Matrix von 32*192 ab, andernfalls überlagern sich die Farben wie Farbkleckse in dem folgenden Programm:

```
10 COLOR 1,4
20 SCREEN 2
30 DRAW "B M 128,96 C15 R4 C1 U4 C15 L4 C1 D4"
40 GOTO 40
```

Obgleich hier, wie in unseren vorherigen Beispiel, laut Programm zwei weiße und zwei schwarze Linien zu sehen sein sollten, ist dies leider Gottes aufgrund des Ineinanderlaufens der zu eng beieinander liegenden Farben nicht der Fall (Untergrundfarbe plus weiß plus schwarz d.h.: da ist mindestens eine Farbe zu viel).

Bis hierhin haben wir eigentlich alle Kommandos d.h. Buchstaben kennengelernt, die unserer DRAW-Programmiersprache innewohnen, als da sind:

R U L D	für die 90 Grad Richtungen
E F G H	für die 45 Grad Zwischenrichtungen
A	für den Drehfaktor um jeweils 90 Grad
M	für das Grafikcursorbewegen mit Zeichnung
B M	für das Grafikcursorbewegen ohne Zeichnung
C	für die Farbveränderung
S	für die Vergrößerung bzw. Verkleinerung
M 128,96	für die absolute Adressierung
M +50,-50	für die relative Adressierung

Zuletzt noch zu zwei Kommandos, die zwar keine DRAW-Unterbefehlserweiterungen sind, die uns aber das Programmieren der Grafiken mit Hilfe des DRAW-Befehls erheblich erleichtern.

1) Wir können Variablen in unsere DRAW-String einfließen lassen, indem wir folgendes Format anstelle einer Zahleingabe verwenden:

= Variablenname ;

Variablenname dürfte klar sein; vergessen Sie aber keinesfalls, das dem Variablennamen ihm vorhergehende Gleichheitszeichen und das ihm nachfolgende Semikolon zusätzlich einzugeben! Andernfalls wüßte Ihr MSX-Computer gar im Zweifelsfalle nicht darüber Bescheid, was er zu tun hat, z.B. würde er bei Verwendung der Variablen A versuchen, die nachfolgende Zeichnung im 90 Grad Rhythmus zu drehen. Also geben Sie immer darauf acht, daß die Syntax bei der Eingabe stimmt, wie im folgenden Beispiel:

```
10 COLOR 1,4
20 SCREEN 2
30 FOR N=4 TO 100 STEP 4
40 DRAW "B M 128,96 S=N;R4 U4 L4 D4"
50 NEXT N
60 DRAW "S4"
70 GOTO 70
```

Wau! Innerhalb von nicht einmal zwei Sekunden hat Ihnen Ihr MSX-Computer eine Vielzahl von Quadraten und immer größer werdenden Quadraten auf den Bildschirm gebracht. Während die innerste Zeichnung noch in Originalgröße aufgebaut wurde, nahm die Vergrößerungsrate stetig zu, bis schließlich mit "S=100" alles fünfundzwanzigmal so groß abgebildet wurde (d.h. 25*4 = anstelle von 4 Punkten wurden 100 Punkte gezeichnet - was sich zwischen kleinstem und größten Quadrat getan hat, können Sie am Bildschirm suksessive verfolgen).

Diese Variablenzuweisung mit "=" und ";" können wir in allen unseren DRAW-Unterkommandos anwenden, wie z.B. beim Zeichnen unserer Windrose bei Verwendung des Winkelbefehls mit "A":

```
10 COLOR 1,15
20 SCREEN 2
30 FOR N=0 TO 3
40 DRAW "B M 128,96 A=N;U8"
50 NEXT N
60 DRAW "A0"
70 GOTO 70
```

Wie Sie bei dem letzten Programm sicherlich bemerkt haben, setzen wir am Ende (hier Zeile 60 des Programms) die veränderten Parameter auf ihren Normalstatus zurück. Dann kann es beim Neustart nicht zu Komplikationen kommen. Der Normalstatus heißt eigentlich vollständig:

```
A0 (Winkel = 0 Grad)
C15 (weiße Vordergrundfarbe)
S4 (keine Vergrößerung bzw. Verkleinerung)
B M 0,0 (Grafikcursor am Ausgangspunkt d.h. in der Ecke
        links oben)
```

2) Wir können bereits erstellte DRAW-Zeichenketten in neue DRAW-Zeichenketten als Unterzeichenketten einbauen. Um dies zu demonstrieren, fangen wir nicht gleich an zu zeichnen, sondern erstellen uns ein Zeichentableau:

In Variable A$ speichern wir unsere Treppe vom Anfang ab ("R8 U8 R8"),

in Variable B$ unser Quadrat ("R8 U8 L8 D8") und schließlich

in Variable C$ unser Hexenhäuschen ("R8 U8 L8 E4 F4 G8 F8").
Die Syntax für die Verschachtelung der Stringvariablen wird mit dem Buchstaben "X" eingeleitet und wiederum mit einem Semikolon zur Kennzeichnung abgeschlossen:

```
X Variablenstring ;
```

Unsere Aneinanderkettung der drei Strings sieht dann letztendlich so aus:

```
10 COLOR 1,15
20 SCREEN 2
30 A$="S4 R8 U8 R8"
40 B$="R8 U8 L8 D8"
50 C$="S8 R8 U8 L8 E4 F4 G8 U8 F8"
60 DRAW "B M 128,96 X A$;X B$;X C$;"
70 GOTO 70
```

Zwar sieht das Ergebnis nicht gerade begeisternd aus, aber vielleicht zeigt es Ihnen in der verketteten Anwendung unserer zuvor gezeichneten Teilstücke zumindest, wie leistungsfähig und geradezu variabel man bei der Verkettung verschiedener DRAW-Teilstrings vorgehen kann.

Wie bereits zu Beginn dieses Kapitels angedeutet, hier noch einmal der Hinweis: Denken Sie daran, daß die DRAW-Strings genauso leicht zu behandeln sind wie normale Zeichenketten (A$="Hugo"). Also müssen Sie auch daran denken, daß ein String bei MSX-Computern "nur" 255 Zeichen lang sein darf! Außerdem müssen Sie eigentlich mit dem Befehl CLEAR vorher Speicherraum für die Strings reservieren (z.B. für vier Strings mit einer Länge von je 200 Zeichen müssen Sie vorher auf alle Fälle eingeben: CLEAR 800).

Abschließend noch ein Hinweis:

Für die Zahlangabe zur Richtungsbestimmung sind Sie nicht dazu verpflichtet, nur Werte einzugeben, die sich im möglichen Bildrahmen darstellen lassen (256*192 Punkte). Vielmehr können Sie Werte zwischen +32767 und -32768 eingeben. Wofür dies jedoch gut sein soll, ist zumindest fraglich (eventuell könnte dazu ein Grundgedanke der MSX-Konstrukteure Anlaß gewesen sein, möglichst eine Vielzahl von Fehlermeldungen (und damit das unweigerliche Löschen des Grafikbildschirms) vermeiden zu helfen).

LIST

Abbildung des Programms auf dem Bildschirm. Anzeige kann 'eingefroren' werden, wenn man zwischendurch die STOP-Taste betätigt. Durch nochmaliges Drücken der STOP-Taste wird die Anzeige fortgesetzt. Man kann mit LIST auch eine ganz bestimmte Zeile auf dem Bildschirm abbilden, indem man die Zeilennummer hinzufügt z.B.: 'LIST 10' zeigt nur Zeile 10 an; 'LIST -10' zeigt sämtliche Zeilen bis einschließlich Zeile 10 an; 'LIST 10-100' zeigt die Zeilen zwischen 10 und 100 auf dem Bildschirm an (einschließlich der Zeilen 10 und 100).

DELETE

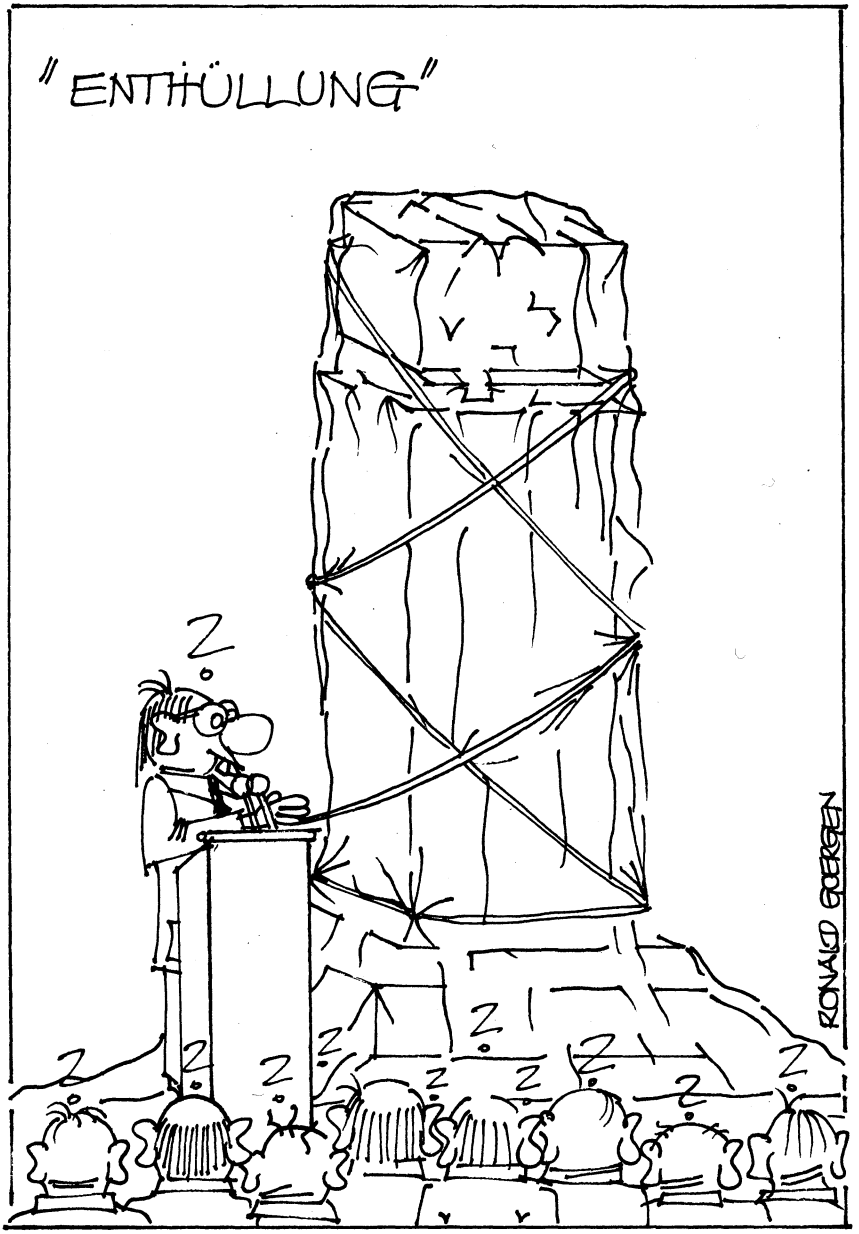
Löschen von Programmzeilen. Man kann mit DELETE eine einzelne Zeile löschen (z.B. zum Löschen von Zeile 20: 'DELETE 20') oder auch eine größere Anzahl von Zeilen (mit 'DELETE 10-100' werden sämtliche Zeilen zwischen 10 und 100 (einschließlich der Zeilen 10 und 100) gelöscht). 'DELETE -20' löscht alle Programmzeilen vom Anfang des Programms bis einschließlich Zeile 20.

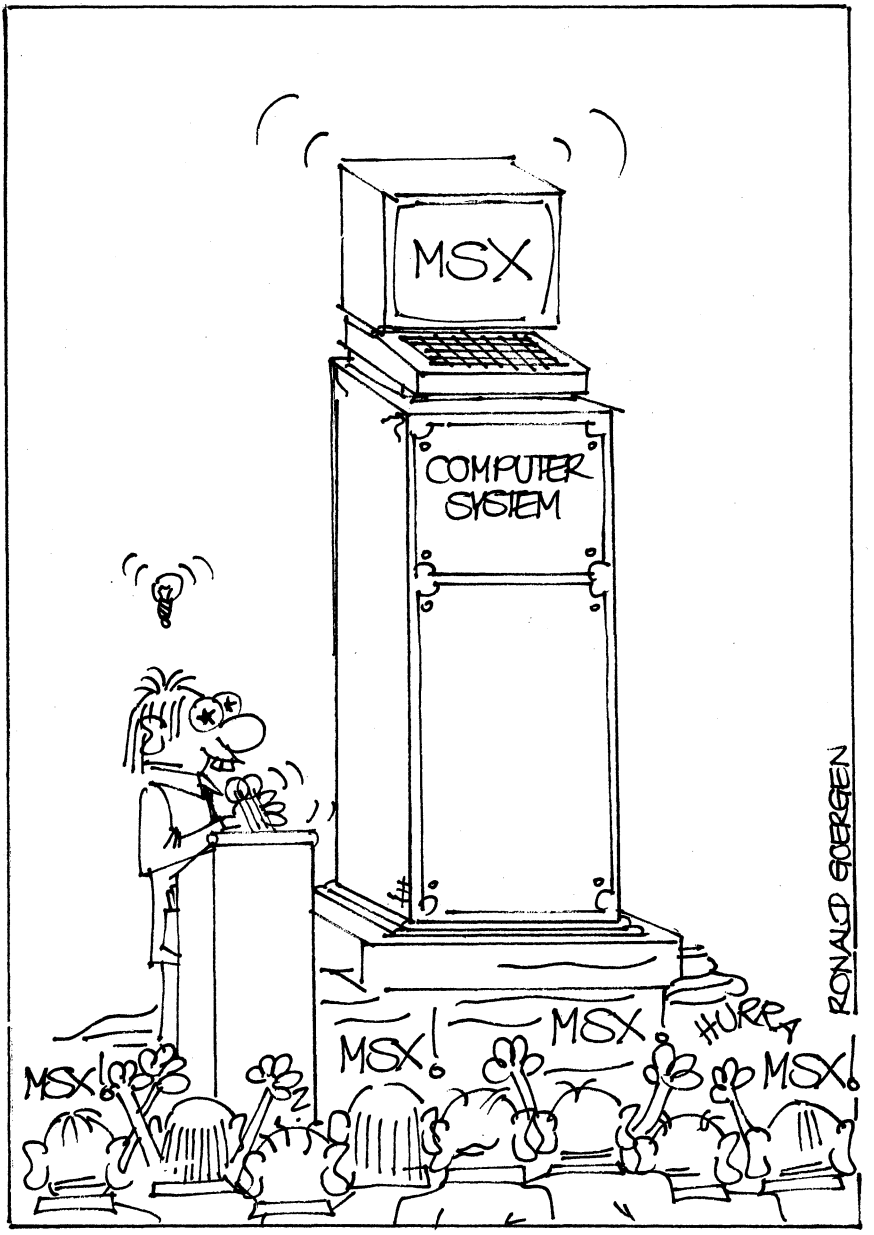
AUTO

Nach Eingabe dieses Befehls wird Ihnen eine automatische Zeilennummerierung vorgegeben d.h. Sie brauchen nicht mehr selbst die Zeilennummern in aufsteigender Reihenfolge einzugeben; nach Drücken der Eingabetaste wird automatisch die nächste Zeilennummer angezeigt. Setzen Sie eine Zahl hinter den AUTO-Befehl, wird mit dieser Zahl=Zeilennummer die automatische Zeilennummernausgabe eingeleitet ('AUTO 100' bedeutet demnach: die automatische Nummerierung beginnt mit Zeile 100). Setzen Sie zudem noch ein Komma und eine weitere Zahl hinzu, wird diese Größe als Zeilenabstand interpretiert.

RENUM

Nachträgliche Umnummerierung der Zeilen im Speicher. Man kann diesen Befehl mit drei Parametern erweitern: 'RENUM 10,100,5' heißt: Nummeriere das Programm ab Zeile 100 neu, starte mit Zeilennummer 10 und wähle einen Zeilenabstand von jeweils 5.


"ENTHÜLLUNG"
RONALD GOERGEN


MSX
COMPUTER SYSTEM
MSX!
MSX!
MSX!
HURRA
MSX!
RONALD GOERGEN

PANASONIC

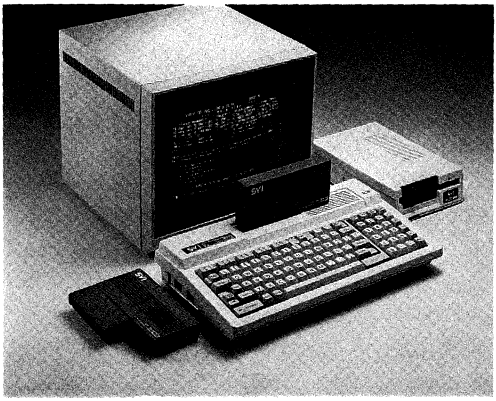
Name: CF-2700

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 2
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX
Tastatur: Hubtasten

Zubehör: Recorder, Vierfarbplotter, Thermoprinter

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß'

SPECTRAVIDEO



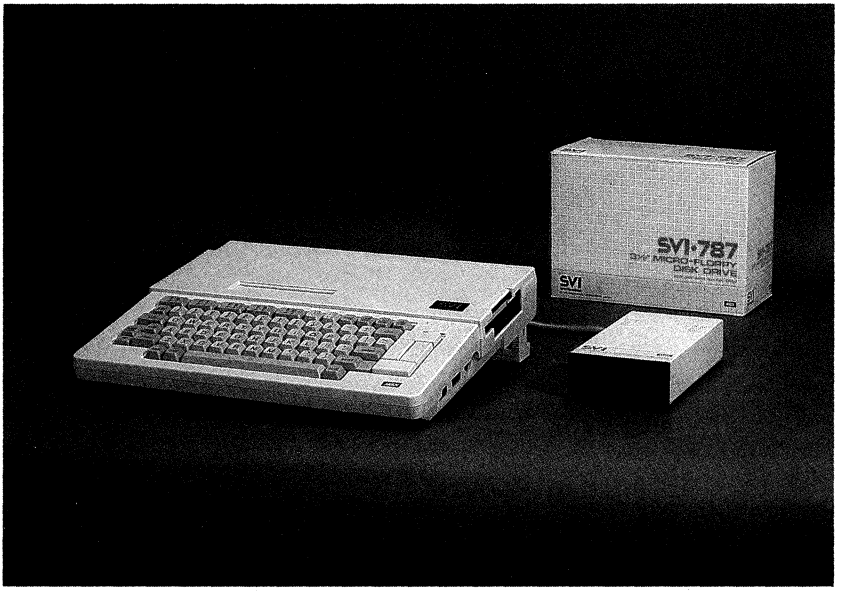
Name: SVI-728

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 1
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Recorder, 80-Zeichenkarte, RS-232-Schnittstelle,
Diskettenstation 5.25' inkl. CP/M 2.2

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß', eingebaute Zehnertastatur

SPECTRAVIDEO



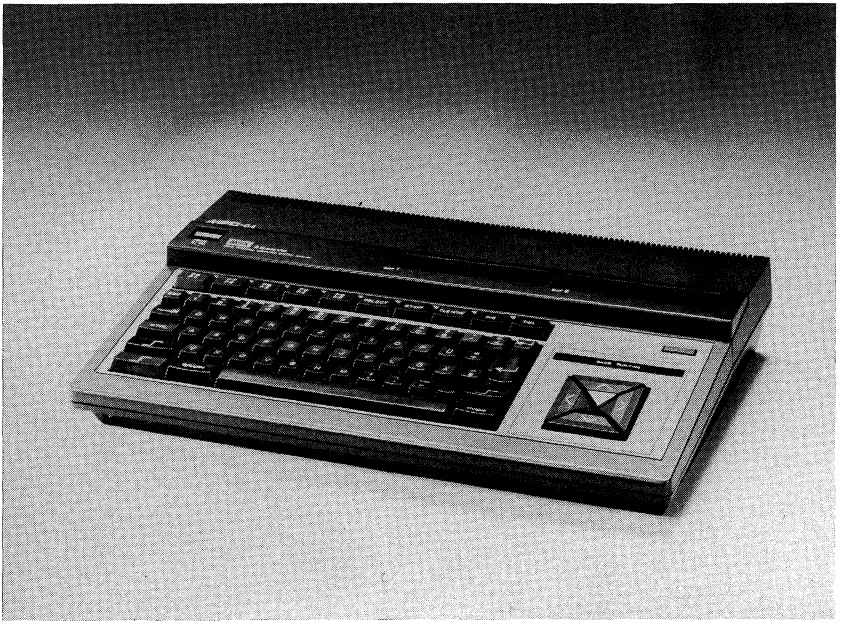
Name: XPRESS

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 80*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 1
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: 3.5' Diskettenstation

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß',
eingebaute RS-232-Schnittstelle, eingebaute
80-Zeichenkarte, eingebaute 3.5' Diskstation
inkl. CP/M 2.2, tragbar

SANYO



Name: MPC64

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 2
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Lichtgriffeleinschub, 64 KB RAM-Erweiterung, Diskettenstation 5.25'

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß', eingebaute RS-232-Schnittstelle, Lightpenanschluß

GoldStar

Name: FC-200

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 1
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß',
Lightpenanschluß

CE-TEC

Name: MCP-80

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 1
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Recorder, Quick Disk 2.8', Diskettenlaufwerk 5.25',
Matrixdrucker, Joystick, Grafiktablett

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß'

PHILIPS

Name: VG-8020

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 2
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Monitor, 40- und 80-Zeichen Matrixdrucker, Diskettenstation 3.5', Recorder, Joysticks

PHILIPS

Name: MSX-8010

Prozessor: Z80A
ROM: 32 KB
RAM: 48 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 2
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Monitor, 40- und 80-Zeichendrucker, Diskettenstation 3.5', Recorder, Joystick

SONY

Name: HIT-BIT 75 D

Prozessor: Z80A
ROM: 32 KB
RAM: 64 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde
Expansionsport: 2
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Recorder, Vierfarbplotter, Diskettenstation 3.5',
Joystick, Monitor, Data Cartridge

Besonderheiten: deutsche Tastatur mit Umlauten und 'ß'

YAMAHA

Name: CX5M

Prozessor: Z80A
ROM: 32 KB
RAM: 32 KB + 16 KB
grafische Auflösung: 256*192 Punkte
Textdarstellung: bis zu 40*24 Zeichen
Farben: 16
Sound: 8 Oktaven, dreistimmige Akkorde und s.u. Besonderh.
Expansionsport: 1
Centronicsanschluß: eingebaut
Joystickanschluß: 2
BASIC und Betriebssystem: Microsoft MSX

Zubehör: Music Keyboards und Music Cartridges

Besonderheiten: eingebauter 8-stimmiger Synthesizer mit
MIDI-Interface und -Anschlüssen

CONT: Progammablauf wird nach Unterbrechung durch CTRL-STOP oder das BASIC-Kommando STOP bzw. END fortgesetzt

LIST: Abbildung des Programms auf dem Bildschirm

NEW: Löschung des Programms im Speicher

RUN: Start der Programmausführung

CLEAR: Reserviert RAM-Speicherplatz für Stringvariablen (beim Einschalten: 200 Speicherplätze)

DIM: Festlegung eines Feldes für Variablennamen = Möglichkeit zur Dimensionierung

END: Anzeige für Programmende

FOR ... NEXT: Einleitung und Abschluß einer Programmschleife, die den Wert einer Variablen bei jedem Durchlauf um 1 erhöht

GOTO: Sprung in eine bestimmte Programmzeile

GOSUB ... RETURN: Sprung in ein Unterprogramm und Rückkehr aus demselbigen ins Hauptprogramm

IF ... THEN ... ELSE: IF = wenn
THEN = dann
ELSE = andererseits

INPUT: Einlesen von Daten in vorgegebene Variable; kann auch zusätzlich zu Beginn eine Bildschirmausgabe enthalten

PRINT: Ausgabe von Variablen und Texten auf den Bildschirm

REM: Die Anweisungen hinter diesem Befehl gelten lediglich zur Information, sie werden nicht vom Computer verarbeitet

STOP: Programmunterbrechungsbefehl

TIME: Abfrage der MSX-Zeitvariable

BEEP: Gibt einen Signalton aus

RND( ): Erzeugt eine Zufallszahl zwischen 0 und 1

CLS: Löschen des Bildschirms

COLOR: Auswahl der Vordergrund- (1.Parameter) und der Hintergrundfarbe (2.Parameter)

Die 16 Farben der MSX-Computer

Farbe 0: Transparent
Farbe 1: Schwarz
Farbe 2: Mittleres Grün
Farbe 3: Hellgrün
Farbe 4: Dunkelblau
Farbe 5: Hellblau
Farbe 6: Dunkelrot
Farbe 7: Cyan
Farbe 8: Mittleres Rot
Farbe 9: Hellrot
Farbe 10: Dunkelgelb
Farbe 11: Hellgelb
Farbe 12: Dunkelgrün
Farbe 13: Magentarot
Farbe 14: Grau
Farbe 15: Weiß

CTRL+B: Sprung zum Wortanfang

CTRL+C: gleiche Funktion wie CTRL+STOP = Programmunterbrechung

CTRL+E: löschen der Zeile von Schreibmarke bis Ende

CTRL+F: Sprung zum nächsten Wort

CTRL+G: gleiche Funktion wie der Befehl BEEP

CTRL+H: gleiche Funktion wie die BS-Taste

CTRL+I: gleiche Funktion wie TAB

CTRL+J: gleiche Funktion wie Pfeiltaste nach unten

CTRL+K: gleiche Funktion wie HOME-Taste

CTRL+L: gleiche Funktion wie CLS-Taste

CTRL+M: gleiche Funktion wie ENTER-Taste

CTRL+R: gleiche Funktion wie INS-Taste

CTRL+U: löscht die Zeile, wo die Schreibmarke steht

CTRL+^: gleiche Funktion wie Pfeiltaste nach oben

CTRL+_: gleiche Funktion wie Pfeiltaste nach unten

F1 bis F10: Funktionstasten, die mit folgenden Befehlen beim Einschalten des MSX-Computers belegt sind:

F1: color
F2: auto
F3: goto
F4: list
F5: run
F6: color 15,4,4
F7: cload"
F8: cont
F9: list.
F10: run

BS: Rückschritt mit der Schreibmarke und Löschen des dort stehenden Zeichens

STOP: gemeinsam mit CTRL: Unterbrechung des Programms

INS: Einfügen von Zeichen in bestehenden Text

DEL: Löschen des Zeichens an der Schreibmarke und Heranziehen des Textes nach links

Pfeiltastenkreuz: zur Steuerung der Schreibmarke in die angezeigte Richtung

SHIFT: erzeugt gemeinsam mit einer Buchstabentaste gedrückt einen Großbuchstaben, ansonsten obenstehendes Zeichen

CAPS: Umschaltung auf Großbuchstaben (ohne SHIFT)

CTRL: erzeugt gemeinsam mit einer Buchstaben- oder manchen Zeichentasten gedrückt einen Steuercode

GRAPH: erzeugt gemeinsam mit einer Buchstabentaste gedrückt ein Grafik- oder Mathematikzeichen

CODE: erzeugt gemeinsam mit einer Buchstabentaste gedrückt einen internationalen Code (z.B. deutsche Umlaute)

Akustikkoppler: Gerät zur Übertragung von Daten per Telefon

Aufwärtskompatibilität: Programme laufen auch auf größeren Computermodellen

BASIC: Programmiersprache, die verhältnismäßig leicht zu erlernen ist (weil sofort Fehler erkannt werden und man sich die Befehle gut merken kann)

Baud: Geschwindigkeit, mit der Bits pro Sekunde vom Computer zur Peripherie übertragen werden

Betriebssystem: Programm, das die Steuerung zwischen Computer und Peripherie regelt

Binärsystem: Der Computer arbeitet mit diesem Zahlsystem, das lediglich die Ziffern 0 und 1 kennt

Bit: Kleinste Informationseinheit für den Computer, auf der die gesamte Datenverarbeitung beruht (8 Bit=1 Byte= 1 Buchstabe oder Zeichen)

Bug: Fehler im Programm (eigentlich Wanze, die früher die mechanischen Rechner ab und zu außer Betrieb setzte)

Byte: Entspricht 8 Bit. 1 Byte ist z.B. 1 Buchstabe oder ein Zeichen. Man rechnet beim Computerspeicher mit jeweils 1024 Byte=1 Kilobyte

Cartridge: Plastikkästchen, in dem eine Platine steckt, die z.B. ein Programm enthält

Centronics: Firma, die den Druckeranschluß für Ihren MSX-Computer entwickelt hat

Chip: Elektronischer Baustein, der auf fotographischem Wege hergestellt, eine Unmenge von Transistoren enthält

CP/M: Betriebssystem für kommerzielle Programme

CPU: Herzstück des Computers, das sämtlichen Datenverkehr steuert

Cursor: Aktuelle Schreibmarke auf dem Bildschirm

Debuggen: Befreien der Programme von Fehlern (eigentlich: Entfernen der Wanzen=bugs aus dem Computer)

Dezimalsystem: Zahlsystem, mit dem wir rechnen (Ziffern zwischen 0 und 9)

Diskette: Magnetbeschichtete Platte zum Speichern von Daten. Maße 5.25' und 3.5'

Diskettenstation: Gerät zum Abspielen von Disketten

Drucker: Gerät zum Schreiben von Daten auf Papier

Dualsystem: Der Computer arbeitet mit diesem Zahlsystem, das lediglich die Zahlen 0 und 1 kennt

Editieren: Korrigieren von Fehlern direkt auf dem Bildschirm

EPROM: Speicherbaustein, auf dem Daten festgehalten werden können=Brennen des EPROMS. Löschen des EPROMS durch UV-Licht

Errormeldung: Fehlermeldung, die auf dem Bildschirm ausgegeben wird

Expansionsport: Anschlußbuchse der MSX-Computer, die zur Erweiterung dient

Feuerknopf: Meist auf Joysticks zur Spielsteuerung gedacht

Floppy: Bezeichnung für Diskette bzw. Diskettenstation

Funktionstasten: Tasten, die bereits mit einem Kommando belegt sind (leichter zu handhaben, als das Kommando buchstabenweise einzugeben)

Grafik: Darstellung von Zeichnungen

Hardware: Alle mechanischen und elektronischen Teile des Computers (Gegenteil: Software)

Hexadezimalsystem: Zahlsystem, das aus 16 Ziffern bzw. Zeichen besteht (0 bis 9 und A bis F)

Interface: Verbindungsstelle zwischen Computer und angeschlossenen Geräten (Peripherie)

Joystick: Steuerinstrument mit frei beweglichem Knüppel und Feuerknöpfen (besonders für Spiele geeignet)

Kilobit: 1024 Bit=128 Byte; Maßzahl für Speicherbausteine

Kilobyte: 1024 Byte; Maßzahl für Größe des Computerspeichers

Kompatibilität: Austausch von Hard- bzw. Software zwischen verschiedenen Computern möglich

Lichtgriffel: Zeichengerät, das mit einer Fotodiode ausgestattet direktes Zeichnen auf dem Bildschirm ermöglicht

Lightpen: siehe unter Lichtgriffel

Malpad: Gerät, das auf einer postkartengroßen Fläche die Erstellung von Grafik ermöglicht (Aufbringen der Grafik mit einem Plastikgriffel)

Maschinensprache: Programmiersprache, mit der Ihr Computer eigentlich rechnet (0 und 1)

Matrixdrucker: Auch Nadeldrucker genannt, weil er mit Hilfe von Nadeln Buchstaben, Zeichen und Grafiken in Form einer Matrix zu Papier bringt

Maus: Steuerinstrument, das aus einem kleinen Kästchen mit innenliegender Kugel besteht (bei Bewegung dieses Kästchens auf dem Schreibtisch bewegt sich der Cursor)

Mega: Bezeichnung für ca. 1 Million Bits

Microsoft: Amerikanische Softwarefirma, die sowohl das MSX-Betriebssystem als auch das MSX-BASIC entwickelte

Mikroprozessor: siehe unter CPU

Modulator: Durch dieses Bauteil wird das Computerbild in ein auf dem Fernseher darstellbares Signal umgewandelt

Monitor: Spezielles Datensichtgerät. Bietet durch höhere Bildwiederholung eine flimmerfreie Darstellung

MSX: Abkürzung für Microsoft Super Extended: Bezeichnung für BASIC und Betriebssystem der untereinander kompatiblen MSX-Computer

MSX-Zeichensatz: Während die Buchstaben bei den MSX-Computern dem Standard entsprechen, sind die übrigen Zeichen (z.B.Grafik) MSX-speziell

Oktalsystem: Zahlsystem, das aus 8 Ziffern besteht (0 bis 7)

Pad: siehe unter Malpad

Paddle: Steuerinstrument, das über einen Drehknopf und meist einen Feuerknopf verfügt

Parallel-Anschluß: Hier werden gleichzeitig 8 Bit übertragen (Gegenteil: seriell=gleichzeitig 1 Bit)

Peripherie: Alle Geräte, die zusätzlich mit dem Computer verbunden werden können (z.B. Drucker,Floppy)

Programmiersprache: Anzahl von Befehlen, die es ermöglichen, den Computer zu steuern. Es gibt die sehr schwierige computereigene Maschinensprache und höhere Sprachen wie BASIC

PROM: Speicherbaustein, auf dem Daten festgehalten werden können. Löschen des PROMs ist nicht möglich

RAM: Speicherbaustein, auf dem Daten festgehalten werden können. Löschen des RAMs, wenn kein Stromzufluß mehr besteht (deshalb auch RAM=Schreib-Lesespeicher)

Relais: Elektromechanischer Schalter. Später ersetzt durch Transistor und schließlich Chips

Remote: Fernsteuerungsmöglichkeit z.B. der PLAY-Funktion beim Kassettenrecorder

RGB: Rot-Grün-Blau. Kennzeichen guter Farbmonitoren

ROM: Nur-Lese-Speicher, der auch bei Stromausfall nicht seine Daten verliert (Unterscheidung EPROM, PROM)

RS-232-Schnittstelle: Verbindung zu anderen Geräten; die Datenübertragung erfolgt bitweise

Schreibmarke: Aktueller Cursor auf dem Bildschirm

Serielle Datenübertragung: Die Datenübermittlung erfolgt bitweise (Gegenteil parallel= 8 Bit)

Software: z.B. Programmiersprache, Betriebssystem und eigene Programme

Sound: Klänge oder Geräusche, die vom Computer mit einem speziellen Soundchip erzeugt werden

Speicher: Elektronische Bausteine, die Daten festhalten können: RAM, ROM, EPROM, PROM

Steuercode: Durch Tastendruck kann z.B. der Bildschirm gelöscht werden (meist Buchstabe mit CTRL)

String: Schublade bzw. Variable, in der Texte abgespeichert werden können

Synthesizer: Elektronisches Bauteil, das künstlich Töne und Geräusche wiedergeben kann

Transistor: Elektronisches Schaltelement (vorher Relais), später ersetzt durch Chip (mehrere tausend Transistoren auf einer Stecknadel)

Variable: Teil des Speichers (Schublade), in dem ein bestimmtes Data enthalten ist

Z80A: Mikroprozessor bzw. CPU bzw. Zentraleinheit, die in unserem MSX-Computer enthalten ist

Zeichensatz: Satz von Buchstaben und anderen Zeichen, die unserer Computer von vornherein beherrscht

Zentraleinheit: siehe unter CPU

1.Umrechnung Dezimal-Binär

Aufgabe: Umrechnung von 255 Dezimal in die entsprechende Binärzahl

```
PRINT BIN$(255)
```

Ergebnis: 11111111

2.Umrechnung Binär-Dezimal

Aufgabe: Umrechnung von 11111111 Binär in die entsprechende Dezimalzahl

```
PRINT &B11111111
```

Ergebnis: 255

3.Umrechnung Dezimal-Hexadezimal

Aufgabe: Umrechnung von 255 Dezimal in die entsprechende Hexadezimalzahl

```
PRINT HEX$(255)
```

Ergebnis: FF

4.Umrechnung Hexadezimal-Dezimal

Aufgabe: Umrechnung von FF Hexadezimal in die entsprechende Dezimalzahl

```
PRINT &FF
```

Ergebnis: 255

5.Umrechnung Dezimal-Octal

Aufgabe: Umrechnung von 255 Dezimal
in die entsprechende Octalzahl

```
PRINT OCT$(255)
```

Ergebnis: 377

6.Umrechnung Octal-Dezimal

Aufgabe: Umrechnung von 377 Octal in
die entsprechende Dezimalzahl

```
PRINT &O377
```

Ergebnis: 255

7.Umrechnung Binär-Hexadezimal

Aufgabe: Umrechnung von 11111111 Binär in
die entsprechende Hexadezimalzahl

```
PRINT HEX$(&B11111111)
```

Ergebnis: FF

8.Umrechnung Hexadezimal-Binär

Aufgabe: Umrechnung von FF Hexadezimal in
die entsprechende Binärzahl

```
PRINT BIN$(&HFF)
```

Ergebnis: 11111111

Der Computer: Wir können ihn nicht ignorieren, denn er greift immer stärker in unser Arbeitsleben ein

Auf der einen Seite ist es noch gar nicht so lange her, daß sich die Menschheit ernsthaft Gedanken darüber macht, was passieren könnte, wenn die Computer in immer stärkerem Maße um sich greifen: die abermals aufgeschobene Volkszählung sowie der Ausweis im Scheckkartenformat seien hier nur als zwei Beispiele genannt.

Auf der anderen Seite muß uns bewußt werden, daß diese moderne Technik der Neuzeit eigentlich nicht erst mit der Entwicklung des Taschenrechners in den siebziger Jahren begonnen hat, sondern bereits viele Jahre vorher: Damals (1941) benutzte man noch Computer mit Relais, die nur den Bruchteil der Leistung heutiger kleinster Rechenmaschinen vollbringen konnten. Zudem waren diese wuchtigen ersten Computer nicht nur so groß wie manch eine kleine Wohnung, sie waren zudem durch die ihnen innewohnende Mechanik auch noch sehr störanfällig und unheimlich laut. Ein Besuch im Deutschen Museum in München führt diese erstaunliche Entwicklungsgeschichte jedem Besucher nur zu deutlich immer wieder vor Augen.

Also: Wirkt sich das Zeitalter der Computer - manche Leute sprechen ja bereits von der 2.industriellen Revolution - nun erst seit kurzer Zeit oder aber über einen viel längeren Zeitraum auf die Menschheit positiv oder negativ aus? Denken wir darüber nach, daß bereits eine Generation von jungen Leuten nicht mehr rchtig im Kopf rechnen kann (weil der Taschenrechner einfach so hingenommen wird und dadurch die tagtägliche nötige Übung geradezu verkümmert), so müssen wir uns fragen, ob es nicht ein Fehler war, daß wir uns ohne es zu merken immer mehr in den Bannkreis dieser Maschinen haben reinziehen lassen.

Nun teilt sich unser Volk in zwei Lager: Die einen bejubeln den technischen Fortschritt und fühlen sich bei der rasanten Weiterentwicklung gerade der Computertechnik so richtig pudelwohl. Die anderen würden am liebsten alle Maschinen und jegliche Computer auf unserem Erdboden zerstören, denn wie schon Orwell andeutete: Die Computerwelt droht, das Elend eines Überwachungsstaates, den

wir als kleine Erdbewohner nicht mehr überschauen können, in kürzester Zeit herbeizuführen.

Beide Seiten haben in ihrer Argumentation in gewisser Weise recht ... die einen, die sich im Fortschritt sonnen - sie vergessen nur die Gefahr, die die anderen sehen: Verstehen wir nichts von Computern, beherrschen uns alsbald diese teuflisch durch Menschenhand schlau gemachten und in Wirklichkeit doch so dummen Maschinen ... die anderen, die die Entwicklung der Computer als Bedrohung der Menschheit ansehen - sie vergessen nur, daß es auch schon eine erste industrielle Revolution gab, wo die Arbeiter (heute kann man zu recht sagen 'zum Glück vergeblich') gegen die ersten Dampfmaschinen anzurennen versuchten.

Wir dürfen nicht vergessen, daß gerade Deutschland als eine der größten Wirtschaftsnationen der Erde zum Großteil vom Export abhängig ist. Da helfen in keinem Fall nur ökologische Anbaumethoden und Fabriken, in denen von jedem Arbeiter an jedem Tag haargenau die gleichen Aufgaben stereotyp ausgeführt werden müssen. Hier muß einerseits, um Konkurrenz auf dem Weltmarkt bieten zu können, andererseits, um eintöniges Arbeitsleben interessanter zu gestalten und die primitiven Aufgaben auch der primitiven Maschine zu überlassen, die moderne Technik und damit gerade die Computer- und Roboterindustrie so schnell wie möglich ihren Einzug halten.

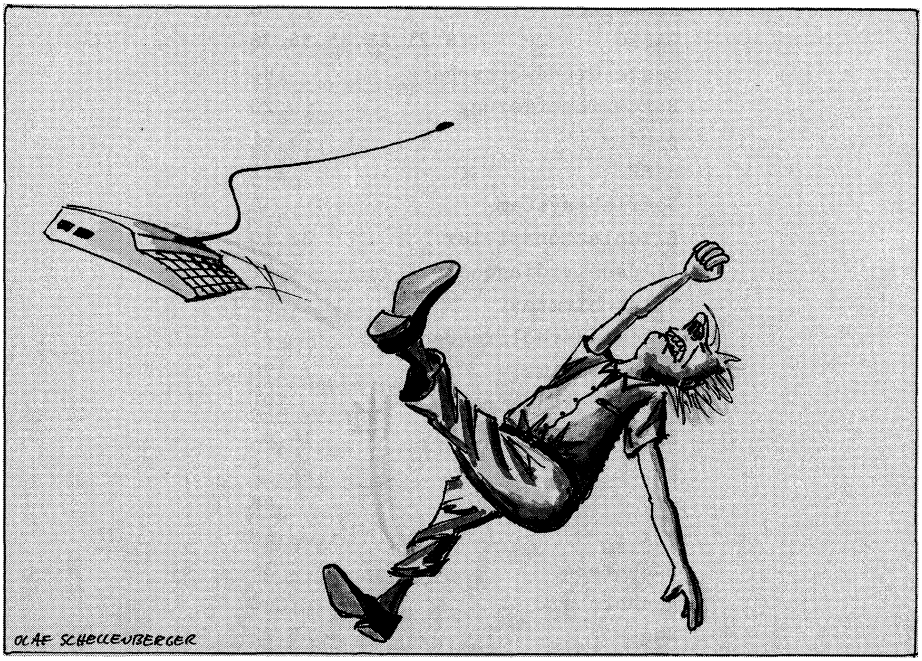
So weit sind wir gekommen: Wir brauchen die Computer zukünftig immer mehr, wir müssen sie allerdings nicht nur anzuwenden sondern sie auch - und dies an erster Stelle - verstehen lernen.

Gerade auf diesem Weg kommt uns zur zur Zeit eine immer mehr um sich greifende Entwicklung entgegen, die wir nicht mit Schaudern ablehnen, sondern die wir zu akzeptieren lernen sollten: Seit ca. 2 Jahren gibt es für einen Obulus, der nicht einmal so hoch liegt wie manch ein theoretisch ausgerichteter Volkshochschulkursus, einen privaten Einstieg in dieses Raumschiff der Zukunft: Ein verkleinertes Computersystem, von Spöttern gerne als Heimcomputer beschimpft (obgleich vor 10 Jahren jedes Rechenzentrum glücklich gewesen wäre, solch einen 'kleinen' Computer sein eigen nennen zu dürfen), hat seinen Einzug gehalten.

Sicherlich ist auch ein 500 DM-Schein bereits ein Geldbetrag, für den man Leistung sehen will. Doch was heißt hier Leistung? Ist es nicht von vielen Menschen geradezu überheblich, für diesen Geldbetrag gleich die Leistung eines Großrechnersystems zu erwarten? Sollte mit diesem Geldbetrag nicht der Erwerb eines viel höheren Guts als Rechtfertigung ausreichen: Durch autodidaktischen Bildungserwerb 'vor der Maschine' weicht die Angst 'vor der Maschine' und es entwickelt sich langsam ein gefesteter Blick in die Zukunft gemeinsam 'mit' und 'durch die Maschine'.

Rainer Lüers

Veröffentlicht in: AUSBILDUNG AKTUELL 4/85, Informationsdienst des Düsseldorfer Ausbildungskreises e. V.


OLAF SCHELLENBERGER

Stichwortverzeichnis

DATA BECKER bringt die richtigen Bücher für Ihren MSX-Computer

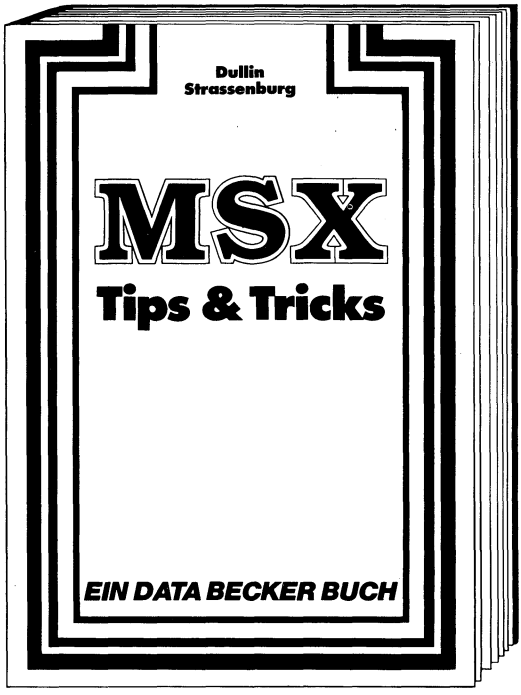
MSX-Computer haben zwei ganz elementare Vorzüge: zum einen ein hervorragendes Preis-/Leistungs-Verhältnis, zum andern darüber hinaus außergewöhnliche Grafik- und Soundfähigkeiten. Das vorliegende Buch behandelt gerade diese Möglichkeiten der MSX-Rechner, umfassend und ausgezeichnet dargestellt. Viele nützliche Beispielprogramme, die den Text gelungen abrunden.

MSX Grafik & Sound, 463(!) Seiten, DM 39,–

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

DATA BECKER bringt die richtigen Bücher für Ihren MSX-Computer



Einfach Spitze, was man aus den MSX-Rechnern herausholen kann: Zeichensatzgenerator, 14 Bildschirmseiten im Direktzugriff, inverse Zeichendarstellung, Windows mit MSX, Text und Grafikhardcopy, Joystickmalprogramm, Benutzung von Systemroutinen, Peeks und Pokes, Abspeicherung von BASIC-Zeilen, Tokens, Listschutz, DATA-Zeilengenerator, Variablendump, Textprogramm, Menuegenerator u.v.m. Ein absoluter Hit für jeden MSX-Anwender!

MSX Tips & Tricks,
ca. 300 Seiten, DM 49,–

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

DATA BECKER bringt die richtigen Bücher für Ihren MSX-Computer

Von den Grundlagen der Maschinenspracheprogrammierung über die Arbeitsweise des Z-80-Prozessors und einer genauen Beschreibung seiner Befehle bis zur Benutzung von Systemroutinen wird in diesem Buch jeder Bereich aufgegriffen und ausführlich – mit vielen Beispielen – erklärt. Ein fundierter Einstieg in die Maschinensprache, und dabei wirklich leichtgemacht! Wenn Sie mehr aus Ihrem MSX machen wollen, dann ist dieser Weg optimal.

MSX Maschinensprachebuch, ca. 300 Seiten, DM 39,–

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

DATA BECKER bringt die richtigen Bücher für Ihren MSX-Computer

Spitzenprogramme vom Disassembler bis zum Sporttabellenprogramm, vom spannenden Superspiel bis zum kompletten Anwendungsprogramm. Mit wichtigen Programmiertips und -tricks. Schwerpunkte bilden beispielsweise Hexdump, Grafik- und Soundeditor, Balkendiagramme, Variablenreferenzliste, Computerschrift, deutsche Umlaute und etliche spannende Themen mehr. Für alle MSX sowie Spectravideo 318 und 328.

MSX Programmsammlung, 195 Seiten, DM 39,–

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

DAS STEHT DRIN:

Gebrauchsanleitungen zu Computern sind eine Sache für sich. Zwar versucht jede Firma, die besten Computer zu entwickeln und vielleicht auch noch dazu die passenden Programme, aber...

Dieses Buch ist von keinem Computerhersteller geschrieben, sondern von jemandem, der es sich zur Lebensaufgabe gemacht hat, anderen Menschen beim Lernen zu helfen, insbesondere was das Thema Computer anbetrifft.

In diesem Buch finden Sie

- die Lernbasis zur Programmiersprache MSX-BASIC
- Informationen über die Erweiterungen Ihres MSX-Computers
- vieles um Ihren MSX-Computer drumherum
- die Bedienung der MSX-Tastatur in allen Einzelheiten
- eine Auflistung der MSX-Computer
- einige Programme, die den Weg zum Ziel versüßen helfen

All dies wird nicht trocken dargeboten, sondern mehr vom Gedanken geprägt: Lernenwollen soll auch Spaß machen!

UND GESCHRIEBEN HAT DIESES BUCH:

Rainer Lüers ist ausgebildeter Lehrer, Computer-Fachtrainer bei einem der führenden Warenhausunternehmen und erfahrener Autor verschiedener DATA BECKER-Bücher (MSX Programmsammlung, MSX Grafik & Sound, CPC BASIC-Programme).

ISBN 3-89011-085-1