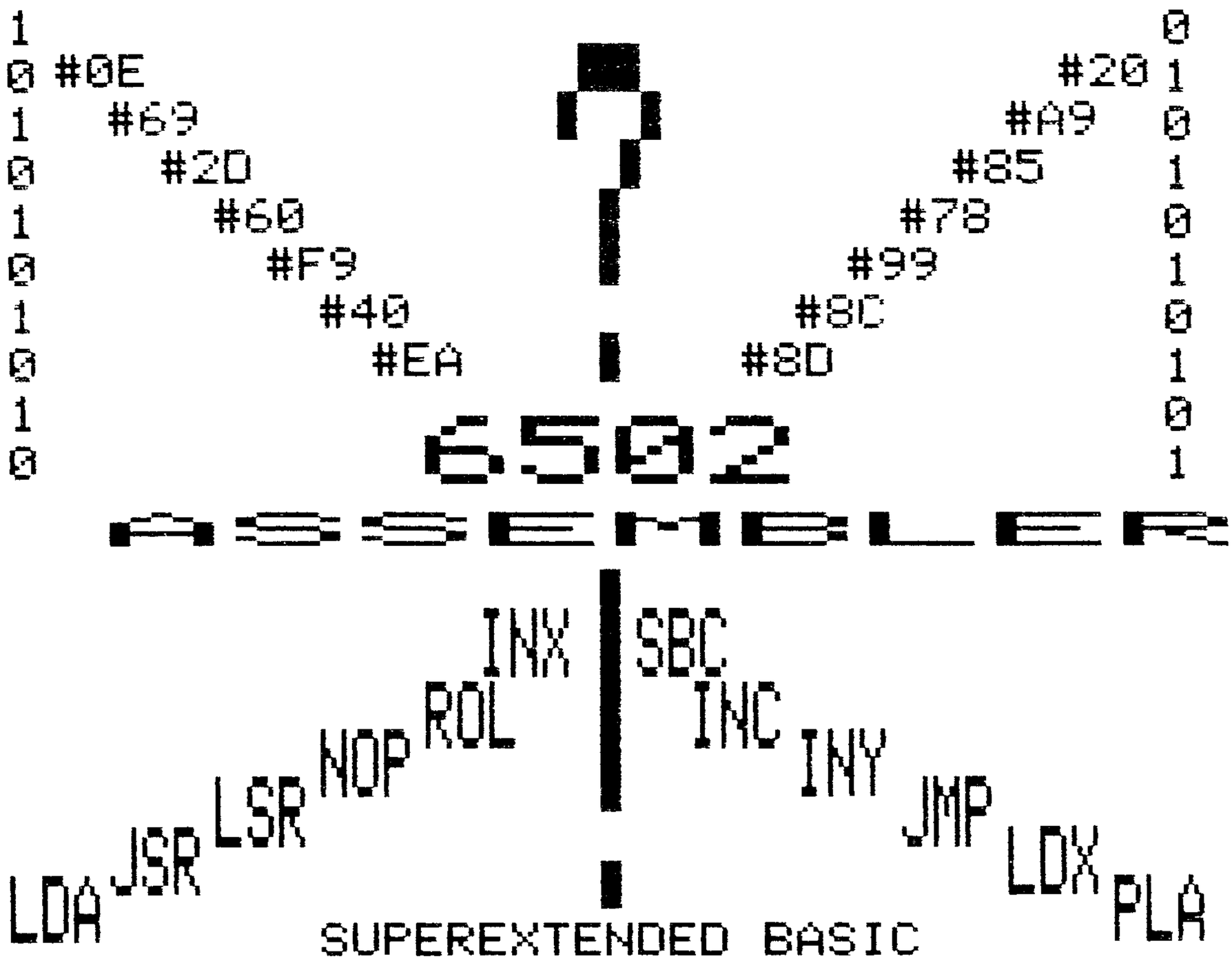
ATMOSPHÄRE

FACHZEITSCHRIFT FÜR ORIC-1 UND ATMOS COMPUTER

Nr.2

SEPTEMBER/OKTOBER 86



INLAND:
Einzelheft : 6,- DM
Jahresabonement: 30,- DM

AUSLAND:
Einzelheft : 7,- DM
Jahresabonement: 36,- DM

VORWORT

Trotz Urlaubs-Schwierigkeiten haben wir diese Ausgabe fertigstellen können. Allerdings fehlt in dieser Ausgabe die "Kleine Bastel-Ecke". Herr Tetzlaf, der sich mit dieser Serie sehr große Mühe gibt, hat es zeitlich nicht mehr geschafft.(Dies soll aber nicht heißen, daß sich die anderen Redakteure keine Mühe machen!!!)
Mit der 2. Ausgabe können wir auch einen neuen Mitarbeiter begrüßen. Ralf Jesse hat die Aufgabe übernommen, Ihnen die Programmierung in Assembler zu vermitteln. Falls zu dieser oder einer anderen Serie Fragen aufkommen, schreiben Sie uns. Trauen Sie sich ruhig. In der Redaktion gilt das Sprichwort "Es gibt KEINE dummen Fragen!!!".
Ansonsten können wir uns nicht über Ihre Mitarbeit beklagen.Der Postbote mußte zwar noch nicht mit einem Wäschekorb kommen, aber das kann ja noch werden. An dieser Stelle möchte ich noch einmal betonen, daß wir nur Programme veröffenlichen können, die uns auf Kassette oder Diskette zugesand werden.Es ist uns aus zeitlichen Gründen nicht möglich die Listings einzutippen.
Auf Wunsch einiger Leser, werden auch wir einen Programm-Dienst einrichten. Für 15,- DM (inkl. Versandk.) können Sie bei uns eine Kassette mit den in der ATMOSPHARE veröffendlichten Programmen beziehen. Auf der ersten Kassette befinden sich die Programme der 1. und 2. Ausgabe + dem Programm DISK-MENÜ aus der alten ATMOSphere.
Einigen Leserr, die uns nach einem größeren Soft- und Hardware-Angebot fragen, möchte ich ein kleines Erlebnis schildern.
Vor kurzem besuchte mich eine Gruppe von ORIC-Usern zu einem Info-Gespräch. Bei dieser Plauderei stellte sich heraus, daß man bei diesen Herren alles mal 8 nehmen mußte. Mit anderen Worten: Einer bestellt sich ein Programm,die anderen 7 bekommen eine Kopie -- Einer bestellt sich die ATMOSPHARE, die anderen 7 lesen mit usw. usw.
-- Und so ist sich jeder selbst der Nächste -- Ist das nicht toll? Da die Gelegenheit günstig war, führten sie mir ein Programm vor, daß sie selbst geschrieben hatten. Das Programm war sehr gut doch die Herren haben noch ein großes Problem. Welches? Dreimal dürfen Sie raten! - Richtig. Ein sicherer Programmschutz, denn sie möchten ja für ihre Arbeit entlohnt werden und da stören sie die Raubkopierer!!!

K.D.B.

INHALT:

IMPRESSUM:

Redaktion:
KDB-COMPUTER-VERSAND
Kornstr. 28
5800 Hagen 7
Tel. (02331) 40 06 01
und alle Redakteure

Redakteure:
Birkemeyer Rüdiger
Grigat Klaus
Jesse Ralf
Künzel Wolfgang
Otto Ekkehard
Tetzlaf Gerd

Bankverbindung:
PSchA. Dortmund
BLZ 440 100 46
Kto. 1117 71-469
KENNWORT: ATMOSPHÄRE

CONNEXION Basic-Maschinensprache

von Ekkehard Otto

Unter den ausgebufften Programmierern gibt es zwei verschiedene Typen. Beide sind sich zwar einig darin, daß BASIC nicht alles ist und es Probleme gibt, die nach Maschinensprache schreien (z.B. zeitkritische Anwendungen), in der Durchführung gibt es aber gewaltige Unterschiede. Der "Doityourself"-Programmierer geht davon aus, daß nur gut ist, was er selber schreibt und verzichtet völlig auf die Routinen des Interpreters. ("völlig" ist übertrieben, denn auf "CALL" oder "USR" wird er schon irgendwie zurückgreifen müssen.) Der "faule" Programmierer will das Rad nicht zum zweitenmal erfinden, d.h. den Kopf, den sich die Schreiber des Interpreters (Microsoft) schon zerbrochen haben, will er sich nicht auch noch zerbrechen. Er benutzt immer wenn es geht Routinen des Interpreters, die die gestellte Aufgabe meistern. (Für die meisten Aufgaben gibt es Routinen, die zumindest einen Teil erledigen.) Ich gehöre zur der zweiten Gruppe und will in dieser kleinen Serie zeigen, wie man Maschinenprogramme aus BASIC-Programmen heraus aufrufen kann, wie man dabei Werte übergeben kann und wie man die Routinen des Interpreters nutzen kann. Dabei soll besonders Wert auf die FließkommaRoutinen gelegt werden, denn wer in Maschinensprache Fließkommarechnungen durchzuführen hat, ist praktisch auf diese Routinen angewiesen. Das soll aber keine Serie über Maschinensprache sein, ich gehe davon aus, daß der Leser den Befehlssatz des 6502 (unseres Lieblingsmikroprozessors) kennt und auch schon Maschinenprogramme geschrieben hat. Die Beispiele sind alle mit dem Assembler "ORION" entwickelt worden.

1.) Wohin mit dem Maschinenprogramm?

Die erste Frage, die sich stellt, wenn man ein Maschinenprogramm schreibt, ist : "Wohin damit?". Die naheliegendste Antwort, die auch zu Anfang von vielen Programmierern benutzt wurde, ist die Seite 4, d.h. die Adressen von $400 bis $4FF. (Ich verwende für Hex-Zahlen das übliche Dollarzeichen und nicht "#" wie der ORIC.) Dieser Bereich wird von den Systemvariablen nicht benutzt und BASIC-Programme beginnen ja erst bei $500. Leider hat das viele Probleme bereitet, als die Diskettenstationen auf den Markt kamen, die ja diesen Bereich benötigen. Also: "Finger weg von Seite 4!!"

Es bleiben vier Möglichkeiten offen:

1.) "hinter HIMEM", d.h. HIMEM wird durch DOKE herabgesetzt und der Bereich zwischen dem alten und dem neuen HIMEM kann genutzt werden und vor Überschreiben geschützt. Zum Beispiel reserviert HIMEM#9000 den Bereich von $9000 bis $97FF. Will man auf HIRES verzichten, kann man auch einfach den Bereich von $A000 bis $BB80 benutzen.

2.) Ausnutzung von "Lücken" im Speicher. Es gibt einige Bereiche im RAM, die nicht benutzt werden. Das sind z.B. jeweils die erste Seite der Zeichensätze ($9800-$98FF in HIRES und $B400-$B4FF in TEXT), die allerdings beim Umschalten von TEXT auf HIRES und umgekehrt verschoben werden.

3.) Zwischen BASIC und Variablen. Die beiden ersten Methoden haben den Nachteil, daß das BASIC-Programm und das Maschinenprogramm nicht zusammenhängen und als zwei Teile abgespeichert werden müssen. Dies kann man wie folgt vermeiden. Vorsicht, die Methode klappt nur, wenn das BASIC-Programm wirklich fertig ist und nicht mehr verändert wird. Der Pointer VARTAB ($9C/D) zeigt auf das Ende des Basicprogramms und damit auf den Anfang der Variablen. Wenn man diesen Zeiger erhöht, entsteht zwischen dem BASIC-Programm und den Variablen freier Raum. Das Programm wird damit aber schwer änderbar, weil jedesmal das Maschinenprogramm verschoben wird. Diesen Nachteil vermeidet die vierte Methode.

4.) Vor dem BASIC-Programm, aber nicht in Seite vier. (das geht!) Man setzt mit einem kleinen Vorprogramm, das am normalen BASIC-Anfang steht, den Anfangs-Pointer für BASIC (TXTTAB $9A/B) hoch und gewinnt damit freien Platz. In der Praxis sieht das z.B. so aus:

```
10 DOKE#9A,601:POKE#600,0:RUN
```

Nach Start dieses Programmes schreibt man das eigendliche BASIC-Programm, das dann bei $600 anfängt. Das Programm sollte mit "DOKE#9A,#501" enden, um den ursprünglichen Zustand wieder herzustellen. DOKE#9A,#501 muß auch gegeben werden, wenn das Programm zusammen mit dem Maschinenteil abgespeichert werden soll. Der Platz von $500 bis $5FF ist nun frei verfügbar. Natürlich kann damit auch beliebig mehr Platz reserviert werden.

2.) Wie rufe ich ein Machinenprogramm auf?

Es gibt drei grundverschiedene Wege vom BASIC zur Maschinensprache und jeder Weg kann auf zwei Arten erreicht werden.

Die erste Methode benutzt USR( ) oder &( ). Beide wirken gleich: steht in einem Rechenausdruck USR( ) oder &( ), so wird der Ausdruck zwischen den Klammern zunächst berechnet und in den sogenannten Fließkommaakku geschrieben. Dann erfolgt ein Sprung nach der Adresse, die in $22/3 (USR) bzw. $2FC/D (&) steht. Das Maschinenprogramm kann diesen Wert dann ändern (wie, das kommt im nächsten Abschnitt) und muß dann mit RTS enden. Der Rechenausdruck rechnet dann mit dem eventuell geänderten Ausdruck weiter. Die Sprungadressen können mit DOKE#22,... bzw. DOKE#2FC,... gesetzt werden. Für USR kann man die Sprungadresse auch mit DEF USR=.... setzen.

Die zweite Methode benutzt CALL oder !. Hier wird einfach nach der angegebenen Adresse (CALL) bzw. nach der Adresse, die in $2F5/6 steht (!) gesprungen. Dabei erfolgt keine Übergabe von Parametern. Wie man trotzdem Informationen übergeben kann, wird weiter unten erklärt. Mit CALL oder ! kann man direkt Routinen des Interpreters aufrufen, für die es keine eigenen BASIC-Befehle gibt. So ergibt CALL#F888 (ORIC) bzw. CALL#F8B8 (ATMOS) den gleichen Effekt wie der Druck auf den RESET-Knopf (NMI) auf dem Gehäuseboden. Das Ausrufezeichen wird vom DOS benutzt, es gibt aber verschiedene Wege, es doppelt zu benutzen. (Das kommt auch später).

Für BASIC-Erweiterungen, die ohne "!" auskommen sollen, kann man den Sprungvektor in $1B/C, der vor jedem "READY" angesprungen wird, ändern, oder man kann die Interpreterroutine "CHRGET" ($E2 ff) abändern. Diese beiden Methoden sollen am Schluß besprochen werden.

3. Wie rechnet der Interpreter?

Die Fließkommarechenroutinen des Interpreters benutzen zwei Rechenregister, die Fließkommaakkus FAC1 ($DØ-D5) und FAC2 ($D8-$DD). Die Rechenroutinen erwarten ihre Argumente normalerweise in diesen Akkus. Die erste Gruppe der Routinen hat nur ein Argument.

RECHENROUTINEN mit einem Argument

Name	ORIC1	ATMOS	Beschreibung
ABS	DF31	DF49	FAC1=ABS(FAC1)
INT	DFA5	DFBD	FAC1=INT(FAC1)
SGN	DF12	DF21	FAC1=SGN(FAC1)
SIGN	DFØ4	DF13	A=SGN(FAC1)
CHS	E26D	E271	FAC1=-FAC1
RND	E34B	E34F	FAC1=RND(FAC1)
SQR	E22A	E22E	FAC1=SQR(FAC1)
LOG	DDDØ	DDD4	FAC1=LOG(FAC1)
LN	DC79	DCAF	FAC1=LN(FAC1)
EXP	E2A6	E2AA	FAC1=EXP(FAC1)
DIV1Ø	DDBF	DDC3	FAC1=FAC1/1Ø
MUL1Ø	DDA3	DDA7	FAC1=FAC1*1Ø
PLUS5	DA79	DBØ4	FAC1=FAC1+.5
SIN	E38E	E392	FAC1=SIN(FAC1)
COS	E387	E38B	FAC1=COS(FAC1)
TAN	E3D7	E3DB	FAC1=TAN(FAC1)
ATN	E43B	E43F	FAC1=ATN(FAC1)

Damit können wir als Anwendug bereits eine kleine Routine zum Runden von DM-Beträgen schreiben. (Beispiel 1)

```
ØØ11Ø B4ØØ                    ;**********************************
ØØ12Ø B4ØØ                    ;* BEISPIEL 1                     *
ØØ13Ø B4ØØ                    ;* RUNDUNG AUF HUNDERSTEL         *
ØØ14Ø B4ØØ                    ;**********************************
ØØ2ØØ B4ØØ MUL1Ø=$DDA3
ØØ21Ø B4ØØ DIV1Ø=$DDBF
ØØ25Ø B4ØØ PLUS5=$DA79
ØØ26Ø B4ØØ INT=$DFA5
ØØ3ØØ B4ØØ 2Ø A3 DD           JSR MUL1Ø
ØØ31Ø B4Ø3 2Ø A3 DD           JSR MUL1Ø
ØØ32Ø B4Ø6 2Ø 79 DA           JSR PLUS5
ØØ33Ø B4Ø9 2Ø A5 DF           JSR INT
ØØ34Ø B4ØC 2Ø BF DD           JSR DIV1Ø
ØØ35Ø B4ØF 2Ø BF DD           JSR DIV1Ø
ØØ36Ø B412 6Ø                 RTS
```

Zum Testen des Programms kann das nebenstehende Basicprogramm dienen. Nach der Initialisierung steht dann die Rundungsfunktion an beliebiger Stelle mit USR(...) zur Verfügung.

```
100 DEF USR=#B400
110 INPUT X
120 PRINT USR(X)
130 GOTO 110
```

Die zweite Gruppe enhält die Rechenroutinen mit zwei Argumenten, entweder zwischen den beiden FACs oder zwischen FAC1 und einer Variablen im RAM. Dabei bedeutet RAM(A,Y) die Variable, die an der Adresse steht, die durch den Akku A des 6502 (low) und dem Register Y (high) gebildet wird.

RECHENROUTINEN mit zwei Argumenten

Name	ORIC1	ATMOS	Beschreibung
FADDM	DA79	DB22	FAC1=RAM(A,Y)+FAC1
FADD	DA9A	DB25	FAC1=FAC2+FAC1
FADDA	E072	E076	FAC1=FAC1+A
FSUBM	DA80	DB0B	FAC1=RAM(A,Y)-FAC1
FSUB	DA83	DB0E	FAC1=FAC2-FAC1
FMULM	DCB7	DCED	FAC1=RAM(A,Y)*FAC1
FMUL	DCBA	DCED	FAC1=FAC2*FAC1
FDIVM	DDE0	DDE4	FAC1=RAM(A,Y)/FAC1
FDIV	DDE3	DDE7	FAC1=FAC2/FAC1
FDIV2	DDDA	DDDE	FAC1=FAC2/RAM(A,Y)
FPOWM	E231	E235	FAC1=FAC2^RAM(A,Y)
FPOW	E234	E238	FAC1=FAC2^FAC1
FCOMP	DF34	DF4C	Vergleich RAM(A,Y) mit FAC1 RAM < FAC1 => A=1 RAM > FAC1 => A=$FF RAM = FAC1 => A=0

Anmerkung: Die Routinen zwischen den FACs erwarten, daß in $DE das exclusive Oder der beiden Vorzeichen und im Akku der Inhalt von $D0 steht. Normalerweise werden die Werte von den Programmen gesetzt, gelegentlich müsser sie aber im Programm neu gesetzt werden. (Siehe unten, Beispiel 3)

Wie man diese Routinen verwendet und was man dabei beachten muß, sowie weitere Beispiele gibts beim nächsten Mal.

Informationen über den Autor:
Ekkehard Otto (38 Jahre, verheiratet, 3 Söhne 10, 8, 1/2) ist Lehrer für Mathematik und Informatik am Gymnasium. Seinen ersten ORIC hat er 1982 direkt aus England importiert. 1894 kam ein Laufwerk "JASMIN" aus Frankreich dazu. E. Otto beschäftigt sich in erster Linie mit der Maschinensprache und dem Interpreter. (Artikel in c't, TOOLKIT, kommentiertes ROM-Listing)

Seit 1984 ist fur die ORIC-Computer ein 3" Disletten laufwerk erhältlich.Bei senem Erscheinen und zum Teil auch noch heute, war es eine der schnellsten und komfortabelsten Diskettenstationen im Hobby-Computer-Bereich. Größter Mangel ist die (wie fast überall) sehr magere Dokumentation. Außer der Erläuterung der DOS-Befehle werden im Handbuch keine Erklärungen über die Diskettenstation oder das Betriebssystem gegeben.

Das ORIC-DOS ist in zwei Teile getrennt. Der erste Teil befindet sich in einem EPROM auf dem Controler-Baustein und wird beim Initialisieren (Betätigen der RESET-Taste am Controler) gestartet. Zunächst werden zwei Maschinenprogramme ins RAM geladen (Umladeroutine und DOS-Befehlsauswertung). Als nächstes wird das Monoflop im Controler, das zum Zugriff auf das DOS dient, auf richtiges 'Timing' geprüft.
(Fehlermeldung: "RV1 adjustment required"). Ist alles in Ordnung, erscheint in der Statuszeile die Aufforderung "insert system disc"

Wird jetzt eine Diskette eingelegt, sucht das DOS nach dem File "SYSTEM.DOS". Ist das Programm nicht vorhanden, blinkt in der Statuszeile "no operating system on Disc". Wird "SYSTEM.DOS gefunden, so wird es in den RAM-Bereich von #7400 bis #A030 geladen und mit der oben erwähnten Umladeroutine in den vom ROM überlagerten RAM-Bereich von #D400 bis #FFFF verschoben.
Da das DOS auch ORIC Systemroutinen mitbenutzt, wird beim Initialisieren gleichzeitig geprüft, ob die ROM-Version 1.0 (ORIC-1) oder 1.1 (ORIC-ATMOS) vorliegt, und die entsprechenden Sprungadressen werden gesetzt.Das zweite Maschinenprogramm, das aus dem EPROM ins RAM geladen wurde,liegt im Bereich von #0480-#04FD. Es wird zum Zugriff auf das DOS benötigt. Die Ansprungadresse für dieses Programm liegt bei #04C4. Da vor alle DOS-Befehle ein '!' zu setzen ist, wird bei der Initialisierung noch in die Speicherplätze #02F5 und #02F6 (Zeiger auf '!'-Routine) die Adresse #04C4 gelegt.
Nach der Initialisierung sucht das DOS den File "BOOTUP.COM" und lädt diesen,falls vorhanden, ein

DER ZUGRIFF AUF DAS DOS

Da das DOS sich im vom ROM überlagerten RAM-Bereich befindet,und der Rechner bei Leseoperationen im Adressbereich von #C000 bis #FFFF normalerweise auf das ROM zugreift,muß das ROM bei DOS-Operationen abgeschaltet werden.Dies geschieht über die Anschlüsse MAP und ROMDIS am Expansion-Bus.
Im Controler der ORIC-Floppy befindet sich deshalb ein Monoflop, das, wenn in der Speicherstelle #0314 das Bit 1 auf '0' gesetzt ist, und gleichzeitig der ROM-Bereich angesprochen wird, die für MAP und ROMDIS notwendigen Impulse erzeugt, und das ROM abschaltet. Damit ist der Zugriff auf das DOS frei, und der entsprechende Befehl kann abgearbeitet werden. Nach der Ausführung wird Bit 1 im Speicher #0314 wieder auf '1' gesetzt.

DIE DISKETTEN-ORGANISATION

Auf der mitgelieferten Systemdiskette befindet sich ein Programm "SYS.COM", mit dem die Systemkonfiguration geändert werden kann. Unter Systemkonfiguration ist die Anzahl und Bauart der angeschlossenen Laufwerke zu verstehen.
Mit dem Controler lassen sich gleichzeitig bis zu vier Laufwerke mit "Shugart-Bus" der Formate 3", 3.5", 5.25" verwalten. Dabei kann es sich um einseitige und/oder doppelseitig schreibbare Laufwerke mit 40 und/oder 80 Spuren je Seite handeln.

Bei der Analyse dieses Programms wurden Hinweise auf den direkten Zugriff zu einzelnen Spuren und Sektoren der Diskette gefunden. Mit dieser Hilfe wurde das Programm "DISC-Monitor" (nächste ATMOSPHÄRE), entwickelt. Mit dieser Software können einzelne Sektoren Spuren und Sektoren gelesen, auf dem Bildschirm dargestellt, geändert und zurückgeschrieben werden

Beim Formatieren unterteilt das DOS die Disketten unabhängig von der Spurenzahl in 16 Sektoren zu 256 Bytes.
Eine einseitig beschreibbare 40 Spur Diskette enthält somit 40*16 = 640 Sektoren zu 256 Bytes. Sektor 1 und Sektor 4 auf Spur Ø sind für die Systemparameter bzw. die ersten 15 Directory-Einträge resaviert. Es bleiben also maximal 638 Sektoren für Programme übrig.
Wie bereits erwähnt werden in Spur Ø Sektor 1 die Diskettenparameter geschrieben. Die Bildschirmkopie eines solchen Sektors ist in Abb.1 zu sehen.

Spur: ØØ Sektor: Ø1 **Abb.1**

```
A1ØØ 28 28 ØØ ØØ ØØ 28 ØØ ØØ ((...(..
A1Ø8 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A11Ø Ø3 ØØ Ø4 ØØ 7A Ø2 Ø4 ØØ ....z...
A118 54 65 73 74 64 69 73 63 Testdisk
A12Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A128 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A13Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A138 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A14Ø 4F 72 69 63 2Ø 44 4F 53 Oric DOS
A148 2Ø 56 31 2E 31 2Ø 2Ø 2Ø  V1.1
A15Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A158 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A158 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A16Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A168 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A17Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A178 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
```

DISKETTENPARAMETER:

Die Bytes Ø-7 enthalten die Systemkonfiguration nach folgendem Schema

- Bytes Ø bis 3 = Spurenzahl der Laufwerke Ø-3 auf der ersten Seite
- Bytes 4 bis 7 = Spurenzahl der Laufwerke Ø-3 auf der zweiten Seite

Die Bytes #1Ø - #17 enthalten Infos über die Belegung der Diskette

- Byte #1Ø = rächster freier Sektor der Diskette
- Byte #11 = Nummer der Spur, auf der sich der Sektor befindet
- Bytes #12 urd #13 = Sektor und Spur des ersten Directory-Sektors
- Bytes #14 und #15 = Anzahl der freien Sektoren auf der Diskette
- Bytes #16 urd #17 = Anzahl der von Programmen oder Dateien belegter Sektoren.

In den Bytes #18 bis #2Ø ist der Diskettenname mit maximal neun Zeichen festgehalten.
Die Bytes #4Ø bis #4B beinhalten den Hinweis auf die verwendete DOS-Version.

DAS INHALTSVERZEICHNIS

Abb.2 ist die Screen-Kopie eines Directory-Sektors.
In Byte 3 ist die Anzahl der Einträge im Directory-Sektor festgehalten. In einem Sektor können max. 15 Eintragungen untergebracht werden. Sind mehr als 15 Programme auf der Diskette, muß das Inhaltsverzeichnis erweitert werden, d.h. ein weiterer Directory-Sektor wird angelegt. Der Hinweis auf diesen Folgesektor befindet sich in diesem Fall in Byte Ø (Spur) und 1 (Sektor). Beim letzten Sektor des Inhaltsverz.
haben diese den Wert 'ØØ'.

Spur: ØØ Sektor: Ø4 **Abb.2**

```
A1ØØ ØØ ØØ Ø2 48 41 4C 54 2Ø ...HALT
A1Ø8 2Ø 43 4F 4D Ø2 ØØ Ø7 ØØ  COM....
A11Ø ØA ØØ 8Ø 53 4F 46 54 2Ø ...SOFT
A118 2Ø 43 4F 4D Ø2 ØØ ØD ØØ  COM....
A12Ø 1Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A128 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A13Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A138 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A14Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A148 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A15Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A158 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A16Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A168 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A17Ø ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
A178 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
```

Der Name des ersten Files ist in den nächsten neun Bytes festgehalten. Hierbei enthalten die ersten sechs Bytes den Programmnamen und die restlichen drei den Namenszusatz.Der Punkt,der bei der Eingabe über die Tastatur zwischen Namen und Zusatz gesetzt wird, erscheint nicht im Inhaltsverzeichnis, sondern dient dem DOS legiglich zur Erkennung der beiden Namensteile. Die beiden Bytes nach dem Namen geben die Anzahl der vom Programm belegten Sektoren an. In den nächsten vier Bytes sind Sektor und Spur des ersten und letzten Programmsektors eingetragen. Das letzte Byte des File-Eintrags betrifft die Schutzparameter des Programms. Dabei bedeutet:

- ØØ = kein Schreibschutz
- #ØØ = Schreibschutz
- #CØ = das File hat einen Schreibschutz und wird im Directory nicht angezeigt.

Danach folgt der Eintrag für das nächste Programm,der aralog dem ersten Programmeintrag aufgebaut ist.

Spur:ØØ Sektor: Ø7 Abb.3

```
A1ØØ ØØ ØA FF ØØ Ø  Ø5 B8 Ø6 ........
A1Ø8 Ø2 ØØ F5 1E Ø  Ø1 ØØ 9D ........
A11Ø 2Ø 2Ø 2A 2A 2A 2A 2A 2A   ******
A118 2A 2A 2A 2A 2A 2A 2A 2A ********
A12Ø 2A 2A 2A 2A 2A 2A 2A 2A ********
A128 3B Ø5 Ø2 ØØ 9D 2Ø 2Ø 2A ;....   *
A13Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A138 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A14Ø 2Ø 2Ø 2Ø 2A ØØ 58 Ø5 Ø3     *.X..
A148 ØØ 9D 2Ø 2Ø 2A 2Ø 41 55 ..   * AU
A15Ø 54 4F 53 54 41 52 54 2Ø TOSTART
A158 53 54 4F 5Ø 5Ø 45 4E 2Ø STOPPEN
A16Ø 2A ØØ 75 Ø5 Ø4 ØØ 9D 2Ø *.u.....
A168 2Ø 2A 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø  *
A17Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
A178 2Ø 2Ø 2Ø 2Ø 2Ø 2A ØØ 92      *..
A18Ø Ø5 Ø5 ØØ 9D 2Ø 2Ø 2A 2A .....  **
A188 2A 2A 2A 2A 2A 2A 2A 2A ********
A19Ø 2A 2A 2A 2A 2A 2A 2A 2A ********
A198 2A 2A 2A ØØ 98 Ø5 Ø6 ØØ ***.....
A1AØ 9D ØØ 9E Ø5 Ø7 ØØ 9D ØØ ........
A1A8 A4 Ø5 Ø6 ØØ 9D ØØ AA Ø5 ........
A1BØ Ø9 ØØ 9D ØØ BØ Ø5 ØA ØØ ........
A1B8 9A ØØ C7 Ø5 ØB ØØ 99 2Ø .......
A1CØ E6 28 23 43 3Ø 37 36 29 .(#C076)
A1C8 D5 D3 3Ø 2Ø C9 2Ø 31 35 ..Ø . .5
A1DØ ØØ DB Ø5 ØC ØØ 6D 2Ø 4B ......m K
A1D6 D4 23 32 35 37 2Ø C3 2Ø .#257 C
A1EØ 23 32 35 46 ØØ EA Ø5 ØD #25F.....
A1E8 ØØ 95 2Ø 41 3A 2Ø B9 2Ø .. A: .
A1FØ 4B 2Ø 41 ØØ Ø5 Ø6 ØE ØØ K,A.....
A1F8 9Ø 2Ø 4B 3A 2Ø 8A 2Ø 23 . K: . #
```

PROGRAMMSEKTOREN

Als erstes File ist in der Directory das Programm"HALT.COM" eingetragen.Es ist zwei Sektoren lang, beginnt in SpurØ/Sektor7,endet in SpurØ/Sektor #ØA und verfügt über einen Schreibschutz.
Abb.3 ist eine Screen-Kopie des ersten Programmsektors. Die Bytes ØØ bis #ØA enthalten die für das DOS notwendigen Informationen über das Programm. Die ersten beiden Bytes weisen auf Spur und Sektor des folgenden File-Sektors hin. Haben beide den Wert ØØ, heißt das, daß der letzte Programmsektor vorliegt.Enthalten die Bytes Ø2 und Ø3 die Werte #ff und ØØ, erkent das DOS, daß es sich um den ersten Sektor eines Files, oder bei mit ",M"ergezusammengehängten Programmen den ersten Sektor eines Teilprogramms handelt. In

Spur:ØØ Sektor:1Ø Abb.4

```
A1ØØ ØØ ØØ C3 32 32 39 2Ø 23 ...229 #
A1Ø8 32 35 37 3A 2Ø 97 2Ø 31 257: . 1
A11Ø 39 ØØ 1D Ø6 ØF ØØ 8D 2Ø 9.....
A118 4B D4 31 2Ø C3 2Ø 39 3A K.1 . 9:
A12Ø 2Ø 95 2Ø 41 3A 2Ø 9Ø 2Ø  . A:
A128 4B ØØ 31 Ø4 1Ø ØØ 8D 2Ø K.1...
A13Ø 4B D4 23 32 32 31 2Ø C3 K.#221 .
A138 2Ø 23 32 32 41 ØØ 4Ø Ø6  #22A. .
A14Ø 11 ØØ 95 2Ø 41 3A 2Ø B9 ... A: .
A148 2Ø 4B 2C 41 ØØ 55 Ø6 12  K,A.U .
A15Ø ØØ 9Ø 2Ø 4B 3A 2Ø 8A 2Ø .. K:
A158 23 32 34 35 2C 23 32 32 #245,#22
A16Ø 31 ØØ 52 Ø6 13 ØØ C1 3A 1.^... :
A168 2Ø 8Ø ØØ 88 Ø6 14 ØØ 91  ..... .
A17Ø 2Ø 23 34 38 2C 23 41 39  #48,#A9
A178 2C 23 3Ø 3Ø 2C 23 38 35 ,#ØØ,#85
A18Ø 2C 23 36 33 2C 23 36 38 ,#63,#68
A188 2C 23 34 43 2C 23 3Ø 33 ,#4C,#03
A19Ø 2C 23 35 43 ØØ B6 Ø6 15 ,#EC.. .
A198 ØØ 91 2Ø 23 34 38 2C 23 .. #48,#
A1AØ 41 39 2C 23 3Ø 3Ø 2C 23 A9,#ØØ,#
A1A8 38 44 4C 23 41 44 4C 23 8D,#AD,#
A1BØ 3Ø 32 2C 23 36 38 2C 23 Ø2,#68,#
A1B8 34 43 2C 23 32 32 2C 23 4C,#22,#
A1CØ 45 45 ØØ ØØ ØØ ØØ ØØ ØØ EE.....
A1C8 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ........
```

diesem Fall enthalten die nächsten beiden Bytes die Anfangsadresse und die folgenden zwei die Endadresse des Programms im RAM.
Die beiden Bytes,die dann folgen, geben Hinweise auf den Programmtyp und Autostartparameter.
Bytes Ø8 und Ø9:
- Ø1 ØØ = BASIC ohne Autostart
- Ø2 ØØ = BASIC mit Autostart
- ØØ ØØ = Maschinenprogramm ohne Autostart
- andere Werte = Maschinenprogramm mit Autostart bei der in den Bytes Ø8 und Ø9 angegebenen Adresse.

In Byte #ØA schließlich ist die Anzahl der Programmbytes dieses Sektors, die bei #ØB beginnen, eingetragen.
Bei dem hier untersuchten Programm "HALT.COM" handelt es sich also um ein selbststartendes BASIC-Programm, das den RAM-Bereich von #Ø5Ø1 bis #Ø6B8 belegt. Von diesen 439 Bytes sind im aktuellen Sektor 7, Spur Ø 245 Bytes abgelegt. Die Fortsetzung des Programms ist auf Spur Ø im Sektor #ØA (1Ø) gespeichert.
In Abb.4 ist dieser Sektor als HEX-Dump abgebildet.
Die ersten zwei Bytes dieses Sektors enthalten den Wert ØØ, was bedeutet, daß es sich hier um den letzten Programmsektor handelt.
In Byte 3 ist die Anzahl der Programmbytes dieses Sektors (+1) festgehalten, die mit dem folgenden Byte beginnen.
Für den aktuellen Sektor heißt dies, daß hier noch #C3 -1 = 194 Bytes des Programms "HALT.COM" gespeichert sind.
Wie aus dem bisher Ausgeführten zu erkennen ist, speichert das ORIC-DOS Programme nicht in aufeinanderfolgenden Sektoren ab, sondern überspringt immer zwei Sektoren. Dieses Verfahren erhöht zwar die Datensicherheit und die Geschwindigkeit,weil zwischen dem Lesen oder Schreiben von zwei Sektoren Zeit für die Positonierung des Schreib-Lesekopfes und andere Disk-Operationen bleibt,es hat jedoch auch einen Nachteil.
Wie man zum Teil beim zweiten Sektor des Programms "HALT.COM" sehen kann, ist der Rest dieses Sektors mit 'ØØ' aufgefüllt, d.h. es wird freier Disk-speicherplatz nicht mehr genutzt. Im Extremfall kann ein Sektor somit nur ein Programmbyte enthalten, wird aber vom DOS als voll belegt angesehen Dieser "Verschwendung" von Diskettenkapazität stehen jedoch sehr viele positive Seiten, wie unter anderem auch eine sehr hohe Schreib- und Lesesicherheit gegenüber,deren Wert auf jeden Fall viel höher einzuschätzen ist.

Klaus Grigat

Das Programm "DISC-MONITOR" folgt in der nächsten ATMOSPHÄRE.

GEÄNDERTER 'MAKER' von Frank Klein

Das Programm DATA-MAKER von E.Otto aus der ATMOSPHÄRE Nr.1, hat mir gefallen. Es hat aber einen Fehler: beim Einlesen von Datenfeldern kann es Passieren,daß der ORIC diese als Token identifiziert und beim LISTEN als Befehle ausdruckt.
Auf der nächsten Seite folgt das verbesserte Programm (n.für ORIC-1). Durch die gesetzten Komma-Zeichen wird auch das Abtippen erleichtert. Außerdem habe ich die Löschroutine so verändert, daß man beliebige Zeilen löschen kann; die Eingabe ist:

Anfangszeile, Endzeile, Abstand zwischen den Zeilen

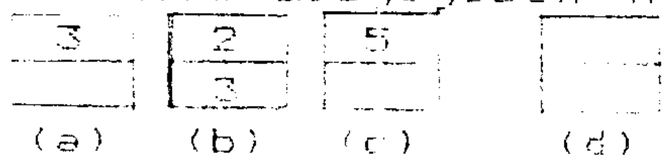
Das Einlesen der Daten habe ich durch Ausschalten der Tastatur beschleunigt. In den Zeilen 2ØØØ-2Ø25 sind die Daten für einen neuen Zeichensatz, der in der Zeile 22 eingelesen und in Zeile 22 gedruckt wird.

```
1 REM ********************************
2 REM ***    MAKER VON F.KLEIN     ***
3 REM ***    MIT N. ZEICHENSATZ    ***
4 REM ***       FUER ORIC-1        ***
10 REM*******************************
20 GOSUB1000  :REM MC-MAKER EINLESEN
21 FORI=32TO96:PRINTCHR$(I),,:NEXT:PRINT
22 GOSUB1000  :REM ZEICHENSATZ EINLESEN
40 DATA#B400,#B450
41 DATA4C,08,B4,51,B4,4C,2E,B4,AD,04,B4,85,20,A9,00,8D,F8,02,AD,03,B4,P,809
42 DATA85,1F,20,C9,D9,A9,00,91,1F,85,00,78,A2,02,BD,05,B4,9D,30,02,CA,P,80F
43 DATA10,F7,58,60,48,AD,DF,02,10,02,68,40,98,48,A4,00,B9,51,B4,F0,0B,P,83C
44 DATA09,80,8D,DF,02,E6,00,68,A8,68,40,A9,40,8D,30,02,D0,F5,P,802
80 INPUT"VON,BIS,ZEILE>1040";V,B,Z:CALL#E6CA:DOKE3,V:DOKE5,B:DOKE7,Z+1
90 Z$=STR$(Z)+"DATA"+HEX$(V)+","+HEX$(B)+CHR$(13)+"RUN 100"+CHR$(13):GOTO150
100 V=DEEK(3):B=DEEK(5):Z=DEEK(7):IFV=B+1THEN170
110 SU=0:Z$=STR$(Z)+"DATA":IFB>V+21THENBB=V+20ELSEBB=B
120 FORI=VTOBB:DA=PEEK(I):SU=SU+DA
130 Z$=Z$+RIGHT$(HEX$(DA+256),2)+",":NEXT
140 DOKE7,Z+1:DOKE3,BB+1
150 Z$=Z$+"P,"+MID$(HEX$(SU),2)+CHR$(13)+"RUN100"+CHR$(13):CALL#E804
160 CALL#B400,Z$:END
170 PRINT"LOESCHEN?";:GETA$:IFA$<>"J"THENEND
180 PRINT:INPUT"VON ZEILE,BIS ZEILE,SCHRITTWEITE ";V,B,ST
190 CALL#E6CA:Z$="START LOESCHEN"+CHR$(13)
195 Z$=Z$+" RUN200"+CHR$(13):GOTO230
200 V=DEEK(3):B=DEEK(5):ST=DEEK(7)
210 Z$=STR$(V)+CHR$(13):V=V+ST
220 IFV<B+1THENZ$=Z$+"GOTO200"+CHR$(13)ELSEZ$=Z$+"ENDE LOESCHEN"+CHR$(13)
230 DOKE3,V:DOKE5,B:DOKE7,ST:CALL#E804
240 CALL#B400,Z$:END
1000 CALL#E6CA:READV,D
1010 SU=0:REPEAT:READ D$
1020 X=VAL("#"+D$):SU=SU+X:POKEV,X:V=V+1:UNTIL D$="P"
1030 READCS$:V=V-1:IFSU<>VAL("#"+CS$)THENPRINT"DATA ERROR IN ";DEEK(#AE):STOP
1040 IF V>D THEN CALL#E804:RETURN:ELSE 1010
2000 DATA#B500,#B700
2001 DATA00,00,00,00,00,00,00,00,3F,1E,0C,0C,00,0C,0C,00,14,14,14,00,00,F,0C9
2002 DATA00,00,00,14,14,3E,14,3E,14,14,00,08,1E,28,1C,0A,3C,08,00,30,32,F,1FA
2003 DATA04,08,10,26,06,00,10,28,28,10,2A,24,1A,00,08,08,08,00,00,00,00,F,038
2004 DATA00,08,10,20,20,20,10,08,00,08,04,02,02,02,04,08,00,08,2A,1C,08,F,004
2005 DATA1C,2A,08,00,00,08,08,3E,08,08,00,00,00,00,00,00,00,00,08,08,10,00,F,0C0
2006 DATA00,00,3E,00,00,00,00,00,00,00,00,00,00,04,00,00,00,02,04,08,10,20,F,800
2007 DATA00,00,1C,3E,36,3E,3A,3E,1C,00,0C,1C,3C,2C,0C,0C,1E,00,1C,3E,26,F,2A8
2008 DATA0C,18,3E,3E,00,3E,3E,0E,1C,0E,3E,3C,00,36,36,36,3E,3E,06,06,00,F,2F8
2009 DATA3E,3C,30,3C,0E,0E,3C,00,06,0C,18,3C,36,3E,1C,00,3E,3E,06,1C,1C,F,2EE
2010 DATA30,20,00,1C,3E,36,1C,36,3E,1C,00,1C,3E,36,1E,0C,18,30,00,00,00,F,2BE
2011 DATA08,00,00,08,00,00,00,00,08,00,00,08,08,10,04,08,10,20,10,08,04,F,90
2012 DATA00,00,00,3E,00,3E,00,00,00,10,08,04,02,04,08,10,00,1E,3F,33,06,F,14C
2013 DATA0C,00,0C,00,02,02,0A,12,3E,10,08,00,3E,26,26,3E,3E,26,26,00,3C,F,21C
2014 DATA3E,36,3C,36,3E,3C,00,1E,3A,30,20,30,3A,1E,00,3C,3E,2E,26,2E,3E,F,3DA
2015 DATA3C,00,3E,3C,30,3C,30,3C,3E,00,3E,3C,30,3C,38,30,30,00,1C,3E,30,F,3D4
2016 DATA36,26,26,1C,00,36,36,36,3E,3E,36,36,00,3E,3E,1C,1C,1C,3E,3E,00,F,374
2017 DATA3E,3E,36,06,16,1E,1C,00,32,36,3C,38,3C,36,32,00,30,30,30,30,30,F,378
2018 DATA3E,3E,00,22,36,3E,3E,36,36,36,00,22,32,3A,3E,36,36,36,00,1C,3E,F,3BA
2019 DATA36,36,36,3E,1C,00,3C,3E,36,3E,3C,30,30,00,1C,3E,36,36,3E,3C,1A,F,3E0
2020 DATA00,3C,3E,36,3E,38,3C,36,00,1C,3E,30,18,0C,3E,1C,00,3E,3E,2A,08,F,34E
2021 DATA08,08,1C,00,36,36,36,36,36,3E,1C,00,36,36,36,36,36,1C,08,00,36,F,2FC
2022 DATA36,36,3E,3E,36,22,00,22,36,1C,1C,1C,36,22,00,22,36,3E,1C,08,08,F,306
2023 DATA1C,00,3E,3E,0C,18,30,3E,3E,00,1E,10,10,10,10,10,1E,00,00,20,10,F,224
2024 DATA08,04,02,00,00,3C,04,04,04,04,04,3C,00,08,14,2A,08,08,08,08,00,F,100
2025 DATA0E,10,10,10,3C,10,3E,00,0C,P,D4
```

PROGRAMMIEREN IN FORTH TEIL 2

UMGEKEHRT POLNISCHE NOTATION

Als stapelorientierte Sprache arbeitet FORTH mit der umgekehrten polnischen Notation (UPN). Diese wird auch als Postfix - Notation bezeichnet.
Rechenzeichen wie '+' oder '/' werden nicht zwischen sondern erst nach den Operanden geschrieben. Man schreibt also "3 2 +" statt "3 + 2".
Geben Sie ein 3 2 + . >CR<
--> FORTH antwortet OK
Was geschieht im einzelnen? Im Direkt-Modus arbeitet FORTH diese Zeile von links nach rechts ab. Der Computer findet zunächst eine Zahl, die er sofort oben auf den Stapel legt (Abb. a).
Als nächstes findet er eine weitere Zahl. Auch sie wird auf den Stapel gelegt (Abb.b).
Der Computer tastet die Befehlsfolge weiter ab und findet das Zeichen '+'. Da sich '+' als reserviertes Wort im FORTH-Vokabular befindet, wird es sofort abgearbeitet.Die beiden o b e r e n Elemente werden vom Stapel genommen und addiert. Das Ergebnis wird wieder als oberstes Element auf den Stapel gelegt (Abb.c).
Erst jetzt, nachdem die Addition bereits durchgeführt ist,wird die Abtastung der Befehlsfolge fortgesetzt. Der Computer findet als nächstes das Zeichen '.'. Auch dieses wird im Vokabular gefunden und kann sofort abgearbeitet werden. Dazu wird das obere Element vom Stapel genommen und auf dem Bildschirm ausgegeben (Abb.d).



(a) (b) (c) (d)

Möchte man in BASIC auf dem Bildschirm das Ergebnis einer Addition darstellen,so benutzt man im Direkt-Modus die Schreibweise

PRINT 3 + 2 >CR<

Der Computer wird dann '5' ausgeben. Was aber passiert beim Abarbeiten der Befehle? Ein Befehl in BASIC steht im Gegensatz als Zeichenkette, die mit einem RETURN Zeichen (ASCII 13) abgeschlossen wird. Die Interpretation des Befehls beginnt damit, daß der Computer das Zeilenende sucht.
Als nächstes wird die am Anfang stehende Zeichenkette (PRINT) mit dem resavierten Vokabular des Interpreters verglichen. Der Interpreter findet das Wort und ruft nun die dazugehörige Interpreterroutine auf.
Jetzt wird das Argument des Befehls (hier '3+2') interpretiert. Der Interpreter orientiert sich auch hier wieder an den reservierten Worten,d.h.er sucht einen eventuell vorhandenen Operator. In unserem Beispiel wird das '+' gefunden und die Additionsroutine aufgerufen.Diese Routine verlangt die Anwesenheit zweier Operanden, wie z.B. zweier Zahlen. Die erste wird von dem Interpreter vor, die zweite nach dem Pluszeichen gesucht.Werden die Zahlen gefunden, so wird die Addition durchgeführt und schließlich der PRINT-Befehl zur Darstellung des Ergebnisses ausgeführt.
Es fällt auf,daß die Verarbeitung eines BASIC-Befehls recht verschachtelt vor sich geht.Der Computer sucht das Zeilenende,findet es; er geht nun zurück um den ersten Befehl zu ermitteln; nun geht er wieder vorwärts in der Zeile, um den Operator ('+') zu suchen. Anschließend geht er wieder zurück in der Zeile, um den ersten Operanden (3) zu suchen und wieder vorwärst, um den zweiten Operanden (2) zu ermitteln. Erst jetzt kann die Addition und danach der PRINT-Befehl ausgeführt werden.
Man sieht, daß die Verarbeitung bei FORTH, vor allem durch die Verwendung von UPN, wesendlich ökonomischer ist als bei BASIC. UPN erlaubt eine streng sequentielle Verarbeitung, ohne die vielen Schleifen und das Vor- und Zurück einer BASIC-Interpretation In diesem grundlegenden Unterschied liegt ein Grund für die hohe Verarbeitungsgeschwindigkeit von FORTH.
Sehen wir uns ein Beispiel an

(3+5) * (4-2) -7 =

Mit UPN geben wir ein:
3 5 + 4 2 - * 7 - >CR<
--> FORTH meldet: ? OK
Die Abb.1 zeigt die Entwicklung des Stapels.

			TOS	INPUT
			3	3
		3	5	5
Zwischenergeb.			8	+
		8	4	4
	8	4	2	2
Zwischenergeb.		8	2	-
Zwischenergeb.			16	*
		16	7	7
Ergebnis			9	-
Ausgabe				.

Es fällt auf, daß das Rechnen mit UPN k e i n e Klammern benötigt. Sie sind in Forth auch nicht bekannt, da die Art der Operation nur davon abhängt, wie SIE die Rechenzeichen setzen.
Zudem ist der Gebrauch von Variablen um Zwischenergebnisse zu speichern nicht mehr nötig. (Die Zwischenergebnisse 8 und 2 werden im Stapel abgelegt und von den Rechenzeichen wieder aufgerufen.)

WÖRTER IN FORTH

Flexibilität ist ein weiteres Stichwort bei FORTH.
Ist es in Basic nur sehr schwer möglich, neue Befehle zu definieren, so ist das bei FORTH nicht nur sehr einfach, sondern sogar ein Grundprinziep, auf dem die Sprache aufgebaut ist.Das,was man beim Kauf der Kassette oder Diskette erhält,ist bei weitem nicht die ganze Sprache. Man kauft nur einen "Sprachkern".FORTH ist über diesen Kern hinaus beliebig erweiterbar. Warum ist das möglich? Grundsätzlich unterscheidet man zwischen Interpretern und Compilern.Ein Interpreter nimmt sich jede eingegebene Zeile des Programms vor, "schaut" nach, welche Befehle in dieser Zeile stehen, und arbeitet sie dann ab. Daraus resultiert, daß ein Interpreter vergleichsweise langsam arbeitet und daß er nur Kommandos verarbeiten kann, die ihm auch bekannt sind. Er ist damit auf eine feste Anzahl von Worten (Befehlen) begrenzt.
Ein Compiler nimmt sich das ganze Programm vor, übersetzt jeden Befehl in eine folge von Maschinenintruktionen, die dann mit der Geschwindigkeit des Prozessors ausgeführt werden.Das Compelieren eines Programms dauert zwar relativ lange, doch ist man damit in der Lage,neue Befehle zu kreieren FORTH ist nun beides.Es interpretiert bekannt Befehle und/oder compeliert neue Komandos,die dann interpretiert werden müssen.
FORTH erzeugt (ähnlich w. PASCAL) einen sog. Zwischencode (F-Code) und ist deshalb zwar nicht so schnell wie ein "echter Compiler" aber wesendlich schneller als ein Interpreter.
Ein "Wort" in FORTH besteht aus max. 31 ASCII-Zeichen.Ausgenommen sind vier Sonderzeichen:

Leerzeichen	(Space)	ASCII 32
CURSOR left		" 8
DEL	(DELETE)	" 127
RETURN	(>CR<):	" 13

Das Leerzeichen wird als einige Trennmarkierung zwischen FORTH-Wörtern verwendet,CORSOR-left und DEL dienen zur Korrektur und RETURN(>CR<) bedeutet für das System das Ende einer Eingabe.
Beispiele von FORTH-Worten:

DUP + NAME /MOD +LOOP . >CR<:

Diese Sammlung von Befehlen wird Wörterbuch (engl. Dictionary) genannt und kann mit dem Befehl VLIST auf den Bildschirm gelistet werden.

NEUE WORTE DEFINIEREN

Die Definition eines neuen Wortes beginnt mit einem Doppelpunkt (engl. Colon), gefolgt von einer Leerstelle und dem Namen des neuen Wortes. Danach werden die Befehle in der Reihenfolge eingegeben,wie sie später ausgeführt werden sollen. Das Ende der Definition wird durch ein Semicolon markiert.
Einige Beispiele:
Nehmen wir an, wir wollen zwei Werte auf dem Stack vertauschen und danach den zweiten Wert nach oben duplizieren.(Ein Vorgang der sehr oft gebraucht wird)

Bitte geben Sie ein:

: TEST SWAP OVER ; >CR<

```
:  ----------> Start
TEST --------> Name
SWAP OVER --> Inhalt d.n. Wortes
; ----------> Ende d. Definition
```

Diese Art, Programmfragmente in FORTH zu defirieren heißt COLON-Definition. (Ein neues Wort wird ins Wörterbuch compiliert)
Um festzustellen, was unser neues Wort nun anstellt, geben wir zwei Zahlen auf den Stack: 1 2 >CR<
Anschließend rufen wir das neue Wort auf: TEST >CR<
FORTH meldet --> 2 1 2 OK
Es hat also funktioniert.Die beiden Werte wurden vertauscht und der zweite zusätzlich kopiert.
2. Beispiel:
Wir wollen ein Wort definieren, daß eine Kubikzahl berechnet.

```
: KUBUS DUP DUP * * . ;  >CR<
```

Probieren wir das neue Wort aus:
5 KUBUS >CR<
FORTH meldet --> 125 OK
Es hat wieder funktioniert.Zuerst wurde die Fünf auf den Stapel gelegt und danach das Wort KUBUS aufgerufen. Dieses Wort sorgt dafür, daß die oberste Zahl auf dem Stack zweimal dupliziert wird. (Stapelinhalt 5 5 5).
Anschließend werden die Zahlen multipliziert und das Ergebnis ausgegeben.
3. Beispiel:
Mit dem Wort >."< können Texte auf dem Bildschirm ausgegeben werden. Wir definieren:

```
: EINS ." 1 IST EINE EINS " ;>CR<
```

```
: ----------> Start Definition
EINS --------> Name
." ----------> Start Textausgabe
1 IST EINE EINS --> Text
" ----------> Ende Textausgabe
; ----------> Ende Definition
```

Definieren wir zwei weitere Worte
: ZWEI ." 2 IST EINE ZWEI " ;>CR<
: EINS ." 3 IST EINE DREI ;>CR<

Nach der letzten Definition warnt das System "EINS ISN't UNIQUE" (EINS ist nicht einmalig), da schon ein Wort mit dem Namen EINS definiert wurde. FORTH gibt hier k e i n e Fehlermeldung aus, sondern nur eine Warnung. Trotzdem wird das neue Wort erneut compiliert. Schauen wir uns das genauer an: Geben Sie VLIST ein.
Im Wörterbuch finden wir die Einträge: EINS ZWEI EINS KUBUS TEST
Rufen wir nun EINS auf, so erhalten wir den Text:
"3 IST EINE DREI" OK
Geben Sie nun ein: FORGET EINS CR
FORGET durchsucht das Wörterbuch nach dem l e t z t e n Eintrag für EINS und löscht ihn. Danach wird bei Aufruf von EINS der Text "1 IST EINE EINS" ausgegeben.
Wir geben jetzt noch einmal den Befehl FORGET EINS ein. Wenn wir nun versuchen ZWEI aufzurufen,erhalten wir die Fehlermeldung:
"ZWEI ? NOT FOUND"
Das Wort ZWEI steht nicht mehr im Wörterbuch.Es wurde mit dem zweiten FORGET EINS gelöscht!!!

```
FORGET xxx
löscht im Wörterbuch das Wort xxx
und a l l e Definitionen,die nach
xxx eingetragen wurden!!!
```

4. Beispiel:
Ein Wort kann ein anderes Wort aufrufen.Die Textausgabe kann mit anderen Worten kombiniert werden.

```
: ZEIT ." STUNDEN " ; >CR<
: TAG DUP CR . ." TAGE= " 24 .
  ZEIT ; >CR<
```

Aufruf: 3 TAG >CR<
FORTH meldet: 3 TAGE = 72 Stunden

Machen Sie sich mit den genannten Befehlen vertraut und definieren Sie eigene Worte.
Ein Kaltstart des Systems mit dem Befehl COLD, löscht alle Neueinträge.
Auf die Struktur des Wörterbuches wird noch ausführlich eingegangen

Rüdiger Birkemeyer

HINWEIS: Von dem Autor unserer FORTH-Serie ist beim ELEKTOR VERLAG ein Buch erschienen. "FORTH - PROGRAMMIERSPRACHE DER VIERTEN GENERATION" (ISBN 3-921608-38-4) --- In jedem Buchladen erhältlich!!!---

PHONETISCH SUCHEN

Bei vielen Anwendungen muß ein Programm, einen vom User eingegebenen Begriff aus einer Datei suchen. Zu 99% wird dazu der normale 'String'-Vergleich benutzt (IF A$ = B$ THEN...).
Doch was macht der Anwender, wenn er aus einer Namen-Datei einen Herrn Meier sucht und nicht mehr weis ob dieser mit 'ei','ai' oder 'ay' geschrieben wird?
Dieses Problem löst ein Algorithmus, der nach gleich klingenden Wörtern sucht. Die Begriffe werden also nicht nach ihren Buchstaben, sondern nach ihrem Klang untersucht. Mit anderen Worten nicht die Begriffe sondern deren 'phonetischen' Äquivalente werden verglichen.
Das folgende Programm zeigt Ihnen wie dies funktioniert.
Zuerst wird das Alphabet auf sieben Gruppen reduziert und alle anderen Zeichen unterdrückt:

A,E,I,U,Y,W,H,O	werden zu	A
B,F,P,V	werden zu	B
C,G,J,K,Q,S,X,Z	werden zu	C
D,T	werden zu	D
M,N	werden zu	M
L	bleibt	L
R	bleibt	R

Dieses 'phonetische' Alphabet wird in den Zeilen 380 - 400 dem Vergleichsfeld 'VG$' übergeben.
In den Zeilen 160 - 260 wird ein eingegebener Begriff 'phonetisch' zerlegt.
Von Zeile 270 -340 werden mehrfach auftretende Buchstaben durch einen einzigen ersetzt.
Zeile 90 gibt dann das in Zeile 70 eingegebene Original mit dessen 'Klang"-Wort aus.
In dem Variablen-Feld 'TE$' habe ich einige Begriffe vorgegeben, die von Zeile 100-130 mit Ihrer Eingabe verglichen werden.
Geben Sie mal "SCHAMPANJER" ein!

Dieses Verfahren eignet sich auch hervorragend für einen 'intelligenten' Kommando-Interpreter oder für selbst geschriebene Adventure bei denen man auch mal "neme" statt "nehme" eingeben kann.

K.D.BENKERT

```
10 REM ****************************
20 REM **** PHONETISCH SUCHEN ***
30 REM ****************************
40 REM
50 GOSUB 360
60 REM
70 INPUT"BEGRIFF: ";NA$
80 BB$="":GOSUB160
90 PRINT NA$;" --) ";PH$;" = ";
100 FOR I = 1 TO 10
110 :   IF PH$<>TE$(I,2) THEN 130
120 :   PRINTTE$(I,1):I=10
130 NEXT:PRINT:PRINT
140 GOTO70
150 REM
160 REM ****************************
170 REM * REDUKTION AUF 7 GRUPPEN  *
180 REM ****************************
190 L=LEN(NA$)
200 IF L=1 THEN PH$=NA$:RETURN
210 FOR I=1 TO L
220 :   AA$=MID$(NA$,I,1)
230 :   IF AA$<"A" OR AA$>"Z" THEN260
240 :   AA$=VG$(ASC(AA$)-64)
250 :   BB$=BB$+AA$
260 NEXT I
270 REM ************************************
280 REM * LOESCHEN 'A' UND DOPPELBUCHSTABEN *
290 REM ************************************
300 L=LEN(BB$):PH$=LEFT$(BB$,1):IF L=0 THEN RETUR
310 FOR I=2 TO L
320 :   AA$=MID$(BB$,I,1)
330 :   IFAA$<>MID$(BB$,I-1,1) AND AA$<>"A" THEN PH$=PH$+AA$
340 NEXT I
350 RETURN
360 REM ****************************
370 DIMVG$(26),TE$(10,2)
380 FOR I=1 TO 26
390 :   READ VG$(I)
400 NEXT I
410 TE$(1,1)="CHATEAU":TE$(1,2)="CD"
420 TE$(2,1)="SCHMITT" : TE$(2,2)="CMD"
430 TE$(3,1)="MEIER":TE$(3,2)="MR"
440 TE$(4,1)="PHONETISCH":TE$(4,2)="BMDC"
450 TE$(5,1)="CHAMPAGNER":TE$(5,2)="CMBMCR"
460 RETURN
470 DATA A,B,C,D,A,B,C,A,A,C,C,L,M
480 DATA M,A,B,C,R,C,D,A,B,A,C,A,C
```

ASSEMBLER - KURS TEIL 1

VORWORT

Zur Programmierung eines Computers bedient man sich im allgemeinen einer sogenannten höheren Programmiersprache, wie z.B. BASIC, FORTRAN, COBOL, PASCAL, usw. Diese Programmiersprachen wurden für spezielle Anwendungen entwickelt, weshalb man sie auch problemorientierte Programmiersprachen nennt.
Wenn man z.B. einen Heimcomputer in der Programmiersprache BASIC programmiert, wird man sehr schnell feststellen, daß die Rechengeschwindigkeit, mit der die Programme ausgeführt werden, nicht immer ausreichend ist. Der Grund hierfür ist, daß der Microprozessor (dies ist das Herz eines Computers) BASIC nicht versteht. Er arbeitet intern mit Dualzahlen. Die Umwandlung von BASIC in Dualzahlen, sowie das Testen auf Fehler, kostet sehr viel Zeit. Somit ist BASIC zur Programmierung zeitkritischer Probleme nicht gut geeignet. Man programmiert solche Aufgaben daher direkt in Maschinensprache oder zur Vereinfachung in Assembler.
Dieser Kurs, der das Ergebnis vieler erfolgreich durchgeführter Assemblerkurse ist,befaßt sich ausführlich mit der Programmierung des 6502 - Microprozessors in Assembler.
Der Autor ist davon ausgegangen, daß sich der Leser noch nie oder nur versuchsweise mit Assemblerprogrammierung beschäftigt hat.
Das verwendete Assemblerprogramm ist der ORION von LOTHLORIEN.
Wie die Erfahrung gezeigt hat, wird das Wissen durch selbsständiges Lösen von Übungsaufgaben vertieft. Aus diesem Grunde hat der Autor Aufgaben erstellt, die dem jeweils behandelten Stoff entsprechen. Die Lösungen zu diesen Tests, finden Sie immer am Ende eines Kursus-Teils.
Die ersten zwei Teile werden zum größten Teil den theoretischen Bereich behandeln. Wenn Sie in Assembler richtig programmieren wollen,müssen Sie sich dadurch beißen. Doch nun viel Erfolg!

---

1. KAPITEL Das Rechnen im Dualsystem

Ein Microprozessor arbeitet, wie bereits erwähnt, intern nach dem Dualsystem. Zum besseren Verständnis seiner Arbeitsweise ist es erforderlich, das duale Zahlensystem kennenzulernen.
Der 6502-Prozessor ist ein 8-Bit-Prozessor, d.h. alle zu verarbeitenden Informationen, also auch Zahlen, werden in Gruppen zu 8 Bit zusammengefaßt. Bit ist die Abkürzung von "binary digit" und stellt die kleinste logische Einheit dar. Ein Bit kann zwei Zustände annehmen,nämlich "0" und "1",was in der Digitalelektronik den Zuständen "AUS" und "EIN" entspricht.
Diese 8-Bit-Gruppen nennt man Byte, und mit einem Byte kann man 256 verschiedene Informationen darstellen, nämlich die Zahlen 0 bis 255.

1.1 Unterschiede zwischen Dezimal- und Dualsystem.

Das Dezimalsystem benutzt die arabischen Ziffern 0,1,2,3,...,9. Um die natürlichen Zahlen durch diese Ziffern auszudrücken,werden diese bekanntlich als Summe vom Vielfachen von Zehnerpotenzen dargestellt, wobei die Zehnerpotenzen und die Additionszeichen fortgelassen werden.

BEISPIEL:

$$370_{10} = 3*10^3 + 7*10^2 + 0*10^1 + 6*10^0$$

Der Wert, den eine Ziffer innerhalb einer Zahl vertritt, hängt also von Ihrer Stellung innerhalb der Ziffernfolge ab. Man nennt deshalb das Dezimalsystem im Gegensatz zum römischen Ziffern-

system, ein Positions- oder Stellenwertsystem.
Das Dualsystem ist ebenso wie das Dezimalsystem ein Stellenwert-system. Der Unterschied besteht in der Anzahl der Grundelemente (s.o.)
Zur Unterscheidung werden im Folgenden Dualzahlen mit % gekennzeichnet.

Stellenwerte bei Dualzahlen:

8	7	6	5	4	3	2	1	= Bit-Nummern
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
=128	64	32	16	8	4	2	1	
% Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	= Ø Dezimal
% 1	1	1	1	1	1	1	1	= 255 "
% Ø	Ø	Ø	Ø	1	Ø	1	1	= 11 "

In den folgenden Tabellen sind die zehn Grundelemente des Dezimalsystems den entsprechenden Dualzahlen gegenübergestellt.

dezimal	dual
Ø	%ØØØØ
1	%ØØØ1
2	%ØØ1Ø
3	%ØØ11
4	%Ø1ØØ

dezimal	dual
5	%Ø1Ø1
6	%Ø11Ø
7	%Ø111
8	%1ØØØ
9	%1ØØ1

ÜBUNG 1:
Wandeln Sie die Dualzahlen in Dezimalzahlen um.
%111Ø, %Ø1Ø1Ø und %111Ø1Ø11

1.2 Umwandlung einer Dezimal- in eine Dualzahl
Die Umwandlung von Dezimalzahlen in Dualzahlen unterliegt den gleichen mathematischen Gesetzmäßigkeiten, wie die Umwandlung von Dual- in Dezimalzahlen, nur verläuft hier alles umgekehrt.
Zunächst wird die größte Potenz von 2 bestimmt, die in der Dezimalzahl enthalten ist. Für den dabei entstehenden Rest wird dies wiederholt usw. Jede enthaltene Potenz von 2 bekommt eine "1" und jede nicht enthaltene Potenz eine "Ø".
Es gibt allerdings noch ein weiteres Verfahren, das einfacher als das oben beschriebene System handzuhaben ist. Dabei wird die umzuwandelnde Zahl durch 2 dividiert, der verbleibene Rest und der Quotient werden notiert. Dieser Quotient wird wieder durch 2 dividiert und wieder werden Quotient und Rest notiert. Dies geschieht so oft, bis der Quotient Ø erreicht ist. Die Reste, von unten nach oben gelesen, ergeben dann die gesuchte Dualzahl.

BEISPIEL: Die Dezimalzahl 55 soll in eine Dualzahl umgewandelt werden.

```
55 : 2 = 27 Rest 1
27 : 2 = 13 Rest 1
13 : 2 =  6 Rest 1
 6 : 2 =  3 Rest Ø
 3 : 2 =  1 Rest 1
 1 : 2 =  Ø Rest 1
```

Von unten nach oben gelesen ergeben die Reste die Dualzahl %11Ø111, was der dezimalen 55 entspricht.

ÜBUNG 2:
Berechnen Sie die duale Kodierung der Dezimalzahlen 11, 19, 257.

1.3 Rechnen mit Dualzahlen --> Addition
Die arithmethischen Regeln zum Umgang mit Dualzahlen sind sehr einfach zu erlernen. Für die Addition gilt dabei:

```
Ø + Ø = Ø
Ø + 1 = 1
1 + Ø = 1
1 + 1 = Ø  mit Übertrag 1 = (1)Ø
```

Hierbei ist zu beachten, daß "%1Ø" dem dezimalen Wert "2" entspricht. Die Addition von Dualzahlen geschieht unter Verwendung der oben aufgeführten Regeln genau wie bei Dezimalzahlen.

BEISPIEL:

```
  %11Ø1   1. Summand
+ %1Ø11   2. Summand
-----------------------
  1111    Überträge
-----------------------
 %11ØØØ   Summe
```

ÜBUNG 3:
Berechnen Sie 5 + 1Ø in binärer Form. Prüfen Sie nach, ob das Ergebnis wirklich 15 ist.

Subtraktion
Eine Subtraktion von Dualzahlen wird durch eine Addition der negativen Zahl durchgeführt.

Bisher haben wir nur mit positiven Zahlen gearbeitet. Um auch negative Zahlen darstellen zu können, brauchen wir eine andere Zahlendarstellungsart.
Vorzeichenbehaftete Dualzahlen (signed binary) können durch einen einfachen Trick dargestellt werden. Man verwendet das ganz links stehende Bit als Vorzeichenbit. Vereinbarungsgemäß entspricht "Ø" einem positiven und "1" einem negativen Vorzeichen.
Durch die Verwendung des links stehenden Bits als Vorzeichenbit verändert sich der darstellbare Zahlenbereich von Ø bis 255 nach -128 bis +127.

BEISPIEL:

```
+7 = %ØØØØØ111        +31 = %ØØØ11111
-7 = %1ØØØØ111        -31 = %1ØØ11111
```

Probieren wir einmal eine duale Addition der Zahlen "-5" und "+7"

```
  7   = %ØØØØØ111
(-5)  = %1ØØØØ1Ø1
-----------------
      = %1ØØØ1ØØ   = -12
```

Wir sehen, daß das Ergebnis falsch ist. Richtig wäre die Bitkombination %ØØØØØØ1Ø = 2 gewesen.
Eine einfache arithmetische Operation, wie z.B. die Addition, führt also nicht immer zum richtigen Ergebnis. Dieser Effekt ist natürlich nicht erwünscht.
Mathematiker haben nach einer einfachen Lösung für dieses Problem gesucht und diese in der sogenannten Zweierkomplementdarstellung gefunden. Um diese Darstellungsart verstehen zu können, soll zuerst das Einerkomplement eingeführt werden.

Einerkomplement

Hier werden die positiven ganzen Zahlen im gewohnten Binärformat dargestellt. "+3" z.B. wird als %ØØØØØØ11 wiedergegeben.
Das Komplement "-3" erhält man, indem man jedes Bit der Originaldarstellung komplementiert (umkehrt).
Das Komplement zu "Ø" ist "1" und das Komplement zu "1" ist"Ø".
"-3" wird somit im Einerkomplement als %111111ØØ dargestellt.
Probieren wir jetzt noch einmal die Addition von vorhin.

%ØØØØØ1Ø1 --> 5

%11111Ø1Ø --> -5 Einerkomplement

```
+    7   = %ØØØØØ111
+ (-5)   = %11111Ø1Ø
-----------------------
  (1) %ØØØØØØØ1 = 1 (der Übertrag wird ignoriert)
```

Auch dieses Ergebnis ist falsch. Gehen wir jetzt also direkt zur Zweierkomplementdarstellung über.

Zweierkomplement

Beim Zweierkomplement werden positive Zahlen wie beim Einerkomplement in der Signed-Binary-Form wiedergegeben.
Der Unterschied besteht in der Darstellung negativer Zahlen. Um eine negative Zahl im Zweierkomplement darzustellen, muß zunächst das Einerkomplement dieser Zahl ermittelt werden. Dann wird hierzu eine "1" addiert.

BEISPIEL 1:

%ØØØØØØ11 --> 3

```
  %11111ØØ --> -3 Einerkomplement
+        1
-----------------
  %11111Ø1 --> -3 Zweierkomplement
```

Und nun noch einmal die Addition von "-5" und "+7"

%ØØØØØ1Ø1 --> 5

```
  %11111Ø1Ø --> -5 Einerkomplement
+        1
-----------------
  %11111Ø11 --> -5 Zweierkomplement
```

```
+    7   = %ØØØØØ111
+ (-5)   = %11111Ø11
-----------------------
  (1) %ØØØØØØØ1Ø = 2 (der Übertrag wird ignoriert)
```

Wie Sie leicht sehen können, führt die Addition im Zweierkomplement zum richtigen Ergebnis.

BEISPIEL 2:
Addition der Zahlen "-3" und "-2".

```
    %ØØØØØØ11 -->  3

  %111111ØØ --> -3 Einerkomplement
+          1
-----------
  %111111Ø1 --> -3 Zweierkomplement


    %ØØØØØ1Ø --->  2

  %111111Ø1 --> -2 Einerkomplement
+          1
-----------
  %111111Ø --> -2 Zweierkomplement


   -3  = %111111Ø1
+ (-2) = %111111Ø
-----------------
     (1) %11111Ø11 = -5 (der Übertrag wird ignoriert)
```

PROBE
Als Probe ermittelt man das Zweierkomplement des Ergebnisses.

```
%11111Ø11 --> -5

 %ØØØØØ1ØØ --> Einerkomplement
+        1 --> Addition von "1"
-------------
 %ØØØØØ1Ø1 -->  5 (die Probe zeigt, daß das Ergebnis
                   richtig ist)
```

Allgemein und ohne mathematischen Beweis kann man sagen, daß die Zweierkomplementdarstellung immer richtig funktioniert.

Ralf Jesse

Lösungen zu den Übungsaufgaben:

ÜBUNG 1:

```
a)     %1110 = 14
b)   %Ø1Ø1Ø = 1Ø
c) %11101011 = 235
```

ÜBUNG 2:

```
a)   11 = %1Ø11
b)   19 = %1ØØ11
c)  257 = %1ØØØØØØØ1
```

ÜBUNG 3:

```
   5     %Ø1Ø1
+ 1Ø   + %1Ø1Ø
----   --------
= 15   = %1111
```

L E S E R - F O R U M

Als Abonnent der ATMOSPHÄRE habe ich einige Fragen zum von Ihnen angebotenen Floppy-Interface: Wird der Bus-Port durchgeschleift? Wie stark ist das mitgelieferte Netzteil, welche Spannungen hat es und ist es ein Steckernetzteil oder eingebaut, und sind die Spannungen für andere Erweiterungen verfügbar?
Wie gut ist das DOS intergriert, ist's so wie beim C-64 (nur pervers ansprechbar),wie bei CP/M Maschinen oder in der Mitte (wie beim Apple)?
Kann man in den ORIC-1 das von Ihnen angebotene ATMOS-EPROM stecken (bzw.wie in c't angegeben, zwischen beiden umschalten) und hat dann (softwaremäßig) einen ATMOS oder gibts Hardware-Inkompatibilitäten?
Können Sie in der nächsten Nummer Kurzbeschreibungen der aufgelisteten Software bringen (die Namen allein sagen mir nicht allzuviel)?
Mir hat das erste Heft recht gut gefallen,ich hoffe,Sie halten das Niveau.

Hans Kraus

Der Bus-Port wird durchgeschleift. Das eingebaute Netzteil versorgt nur den Controler und den Computer. Die Laufwerke benötigen eine seperate Stromversorgung. Wir können Laufwerke besorgen, bei denen ein Netzteil eingebaut ist.
Wenn Sie mit dem Gedanken spielen, sich zwei oder mehrere Laufwerke anzuschaffen, empfehlen wir Ihnen diese ohne Gehäuse und Netzteil zu kaufen. (Sie sparen ca. 100,- DM pro Laufwerk. Sie müßten sich dann nur ein geregeltes Netzteil mit +5V und +12V besorgen. Wir benutzen ein Netzteil mit +5V/5A und +12V/2.5A und können damit gleichzeitig 4 Laufwerke, den Computer und diverse Erweiterungen betreiben.
Das ORIC-DOS ist jederzeit über das '''+Befehl (direkt oder aus Programm) zu erreichen. Das neue DOS aus Frankreich (wir werden es bald anbieten) kann mit MS-DOS konkurieren und bietet einige Leckerbissen. (Auch das '!' wird nicht mehr gebraucht).
Bis auf die Fehlende FUNCT-Taste beim ORIC-1 gibt es keine Probleme beim Austausch der EPROMS und damit kann man leben.
Wenn wir zu jeder Software Kurzbeschreibungen drucken würden, würde dies etliche Seiten füllen.Wir hoffen immernoch auf Beschreibungen von Userr.

K.D.B.

Ich habe einige Fragen zu dem Programm DISK-MENU (aus ATMOSphere) und der Fortsetzung DISK-KARTEI (ATMOSPHÄRE 1). Ich besitze ORIC-1 und Disk-Menü steigt in Zeile 1590 aus (Bad subscript Error) Muß die Diskette leer sein? Meine Disketten haben schon einen Namen und einige Programme drauf.
DISK-KARTEI Teil 1 steigt nach dem Durchlauf in Zeile 150 aus (Overfow Error).Was muß für ORIC abgeändert werden?
Gibt es keinen Befehl,der das Directory ausdruckt?
Beim ORIC-DOS wird doch das BASIC ausgeblendet. Es sollen dann 64K vorhanden sein. Wie verhält es sich mit dem SUPEREXTENDED BASIC welches Disketten fähig ist?

M.Gebauer

Die Programme laufen ohne Änderungen auf beiden Geräten.Bitte überprüfen Sie in Zeile 30 und 1590 den Array-Namen.
30 DIM TITEL$(49)....
1590 ...LEFT$(TITEL$(Z),3)...
Noch ein paar Worte zur Anwendung:
Speichern Sie 'DISK-MENU' mit)!SAVE "BOOTUP.COM",AUTO(ab. Die Diskette muß nicht leer sein. Der spätere Ladevorgang geht aber etwas schneller,wenn das Programm auf einer neu formatierten Diskette direkt nach dem 'SYSTEM.DOS' gespeichert wird. Wenn das Programm nun mit dem Initialisierer gestartet wird, sucht es als erstes den File MENUE.DAT. Ist dieser nicht vorhanden, richtet das Programm ihn automatisch ein.Den Namen, den man dabei der Diskette gibt, erscheint nur im MENUE.DAT. Der Name den Sie beim Formatieren eingegeben haben wird nicht geändert.
Ihr Fehler bei DISK-KARTEI in Zeile 150 deutet zunächst darauf hin, daß Sie in den Zeilen 180-520 etwas falsch haben, denn diese Zeile wird erst aktiv, wenn (IF) die Kontrollsumme eine DATA-Zeile (SU) nicht stimmt. Ansonsten haben wir alles mögliche angestellt aber die Meldung 'Overflow Error' haben wir nicht erhalten.Bitte vergleichen Sie Ihr Listing nochmal mit unserem.
Das BASIC wird nicht ausgeblendet (s. Seite 7 - 101).Das DOS belegt nur keinen zusätzlichen Speicher (außer Seite 4).

)!PRINTER ON : !DIR(gibt das Directory auf dem Drucker aus.

K.D.B.

Hallo ORIC und ATMOS Freunde!
Kompakte Bauweise und günstiger Preis bewegten mich vor 2 Jahren zum Kauf des ORIC-1. Mit dem Verlangen umfangreiche Listen auf dem Bildschirm darzustellen,stellte sich heraus,daß eine 80 Z./Zeile-Darstellung von großer Bedeutung ist.Bin ich der einzige der so denkt? Meiner Meinung nach kann sich der ORIC/ATMOS nur behaupten,wenn eine 80Z.-Darstellung kostengünstig möglich wird. Fragt doch mal die ATMOSPHÄRE-Leser was sie mit Ihren Computern tun bzw. tun wollen.Vielleicht ergeben sich neue Anreize.
Meine Anwendungsvorstellung sieht so aus: Beruflich führe ich Lagerinventuren durch. Es müssen Differenzen berechnet werden (für den Rechner kein Problem), doch die mobile Datenerfassung fehlt.Ferner ist es notwendig ein Dutzend Formblätter formgerecht auszufüllen (für den Drucker problemlos), doch die zur Kontrolle notwendige Darstellung auf dem Monitor ist zu unübersichtlich. Vielleicht sind meine Ansprüche hoch gesteckt und ich sollte auf ein anderes System umsteigen und den ORIC nur als Zweitcomputer nutzen. Erfahrungsgemäß (ZX-81) staubt eine 'Zusatzkiste' nur ein.
Deshalb mein Fazit:
Die Spitzenmäßigen Redakteure der ATMOSPHÄRE sind ein wichtiger Überlebensimpuls - doch spitzenmäßige Erweiterungen (Ppeis / Leistung / Größe) wären mindestens genauso notwendig.

G. Gruszczynski

Vielen Dank im Namen der Redakteure.
Sie würden staunen, was man mit den ORIC's alles machen kann. Aus finanziellen Gründen (s.Vorwort) müssen wir allerdings Schritt für Schritt vorgehen. Die nächste Erweiterung wird in Kürze erscheinen (serielle Schnittstelle) und wenn wir bei unseren Soft- und Hardware-Spezialisten noch etwas Patriotismus wecken können, wird auch die 80-Zeichenkarte mit integrierter Textverarbeitung fertig.

K.D.B.

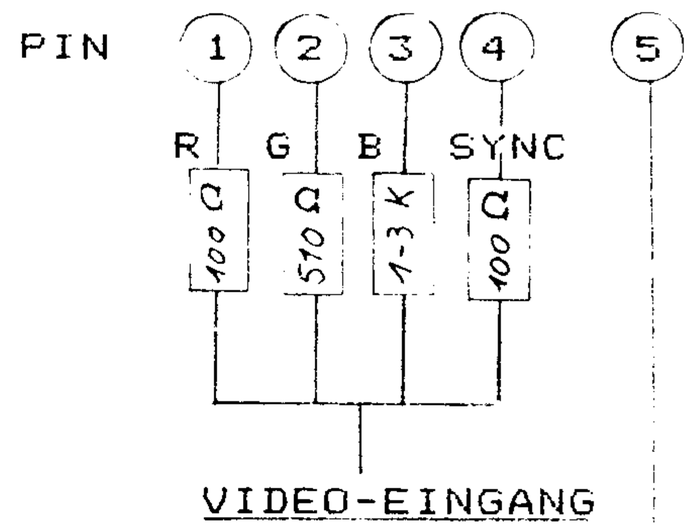
Ich besitze die beiden Assembler/ Disassembler "ORICMON" von PSS und TANSOFT. Wenn ich auf dem Drucker disassembliere, wird

1. bei jeder neuen Zeile das 1. Zeichen verschluckt.
2. weitere Zeichen (zufällig) verschluckt, dafür andere wiederholt.

Dies geschieht bei beiden Assemblern. Es scheint mir, als würde das Handshaking nicht ordentlich funktionieren. Ansonsten läuft der Drucker sowohl beim ORIC als auch beim ATMOS einwandfrei.
Der Drucker ist ein NEC-P2. Wer hatte schon mal das gleiche Problem und kann mir helfen?

E. Kusser

Ich habe mir einen Grün-Monitor an meinen ATMOS angeschlossen. Mit folgender Modifikation am Kabel, habe ich ein sehr gutes Bild erhalten:



Die 4 Wiederstände werden dazwischen gelötet und zu einem Anschluß zusammen gefaßt.

N.Stark

ORIC-CLUBS und KONTAKTADRESSEN

ORIC-HAUFEN-HEESEN

Peter Gaide
Jahnstr. 17
4700 Hamm 5
Tel. (02381) 3451

OriClub Gießen

Dietmar Belloff
Bergstr. 23
6305 Busek 1
Tel. (06408) 7351

Wenn einer dieser Clubs in Ihrer Nähe ist, melden Sie sich doch mal bei den Herren.

In Duisburg möchte ein ORIC-Fan andere ORIC-User kennenlernen!!! Es wäre schön, wenn wir in der nächsten Ausgabe einen neuen Club vorstellen könnten. Hier seine Anschrift:

Ralf Krupka

Emscherstr. 5
4100 Duisburg 18
Tel. (0203) 494150 (ab 18 Uhr)

Um eine starke Gemeinschaft zu werden, brauchen wir mehr solcher ORIC-User. Also fassen Sie sich ein Herz und stellen sich auch als erste Kontaktperson zur Verfügung.

TIPS und TRICKS von Wolfgang Künzel

Manchmal ist es notwendig zu erfahren, ob der CURSOR an- oder abgeschaltet, ESC oder CTRL+P gedrückt wurde.
Nur die wenigsten wissen jedoch, wie man diese Informationen am einfachsten erhält.
Das Prinzip ist ganz einfach:Man muß nur die einzelnen Bits der Speicherstelle 'é18' abfragen!
Von Maschinensprache aus gesehen kein Problem - aber von BASIC ???
Es ist ebenfalls überhaupt keine Schwierigkeit:

```
PRINT PEEK (é18) AND 1 --> Ø = CURSOR aus -- 1 = CURSOR an
                 AND 2 --> Ø = Video aus  -- 2 = Video an
                 AND 4 --> Ø = CTRL P aus -- 4 = CTRL P an*
                 AND 8 --> Ø = Klick an   -- 8 = Klick aus
                 AND16 --> Ø = ESC aus    --16 = ESC an
                 AND32 --> Ø = CTRL D aus --32 = CTRL D an**
                 AND64 --> Ø = 1.u.2. Spalte geschützt
```

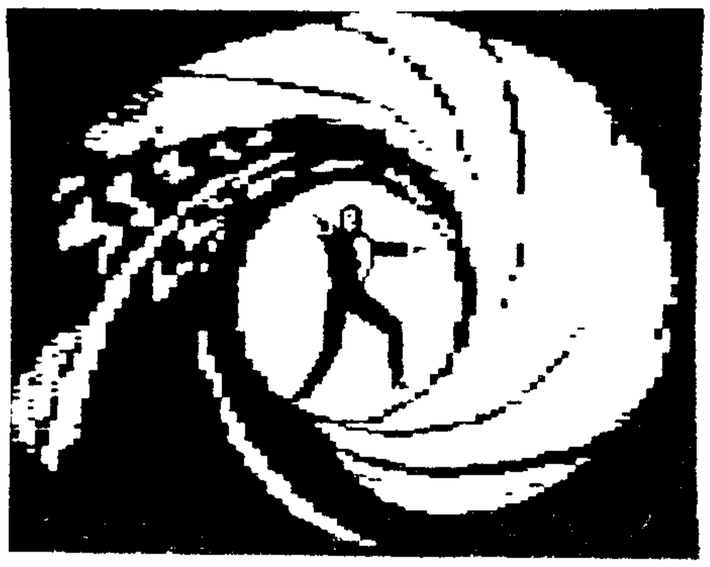
*CTRL P hat eine Flip Flop Funktion. D.h. wenn CTRL P gedrückt wird, geht das zugehörige Bit z.B. von Ø nach 1, nach nochmaligem drücken wird es wieder rückgesetzt.

**CTRL D = doppelte Zeichenhöhe

Mit dem ORIC-1 ist es nicht ohne weiteres möglich ein MC-Programm bzw einen Datenblock vom BASIC aus nachzuladen. Der Grund ist, daß ein Fehler im Betriebssystem den Zeiger 'BASICPROGRAMMENDE' auch beim laden von Maschinenprogrammen neu setzt. Der Erfolg ist ein "OUT OF MEMORY ERROR' oder ein spinnendes BASIC-Programm. Ebenso wird automatisch ein Warmstart bewirkt.
Das folgend kurze Programm behebt beide Fehler und stellt sogar noch die Möglichkeit einen etwaigen Autostart zu verhindern.
ACHTUNG!!! Mit dieser Routine dürfen nur Maschinenprogramme oder Datenblöcke, die mit >CSAVE"...",A...,E... usw. abgesaved wurden, geladen werden.

```
10 DOKE #223,DEEK(§2F5) : DOKE#2F5,#E725 : CALL#E6CA: !"..."(,§) :
   CALL #E4A8 : CALL #E8Ø4
11 DOKE #2F5,DEEK(#223) : IF PEEK(#63) = Ø THEN RETURN
12 PRINT "AUTOSTART! - (SPACE = ABBRUCH)" : GET A$ : IF A$ = " " THEN
   RETURN ELSE CALL DEEK(#5F)
```

Es handelt sich hierbei um ein Unterprogramm, das mit GOSUB 1Ø aufgerufen werden muß. Der CLOAD Befehl ist durch das '!' ersetzt. Der Vektor für das '!' wird jedoch vorher gerettet und nach dem laden wieder auf den alten Wert gesetzt.



ØØ7 JAMES BOND

A VIEW TO A KILL

Schlüpfen Sie in die Rolle des berühmten Kino-Helden.
Bewältigen Sie an seiner Stelle die Aufgabe die Welt vor dem Untergang zu retten!

A VIEW TO A KILL -> Ein Adventure mit bewegter Grafik, daß Sie auch ohne große englisch Kenntnisse bewältigen können.
Zwei Teile auf einer Kassette:
1. THE CITY HALL ESCAPE
2. THE SILICON VALLY MINE

KDB-COMPUTER-VERSAND Preisliste gültig ab 22.08.86
TYP: O = ORIC-1 48K / A = ATMOS 48K / OA = BEIDE
* = NEU IM PROGRAMM

NR.	TYP	NAME	HERSTELLER	PREIS
**********		ARCADE GAMES	**********	
OR001	OA	XENON-1	IJK	34.00 DM
OR002	OA	ZORGON'S REVENGE	IJK	34.00 DM
OR003	OA	XENON III	IJK	34.00 DM *
OR004	OA	DAMSEL IN DISTRESS	IJK	34.00 DM *
OR005	OA	DUMBUSTER (Flugsimulator)	IJK	34.00 DM
OR006	OA	PLAYGROUND 21	IJK	34.00 DM *
OR007	OA	DON'T PRESS THE LETTER Q	IJK	34.00 DM
OR008	OA	ZEBBIE	IJK	34.00 DM
OR009	OA	GUBBIE	IJK	34.00 DM
OR010	OA	ATTACK OF THE CYBERMEN	IJK	34.00 DM
OR011	OA	FRIGATE KOMMANDER (Simulator)	IJK	34.00 DM
OR014	OA	PROBE 3	IJK	24.00 DM
OR015	OA	INVADERS	IJK	24.00 DM
OR030	OA	THE HELLION	ORPHEUS	34.00 DM
OR031	OA	TROUBLE IN STORE	ORPHEUS	34.00 DM
OR032	OA	KRILLIS	ORPHEUS	34.00 DM
OR033	OA	MANIC MINER	PROJEKTS	32.00 DM *
OR063	O	GALAXIANS	SOFTEK	24.00 DM
OR070	O	JOGGER	SEVERN	24.00 DM
OR073	OA	GHOSTMAN	SEVERN	24.00 DM
OR080	OA	ULTIMA ZONE	TANSOFT	24.00 DM
OR082	OA	RAT SPLAT	TANSOFT	24.00 DM
OR083	OA	NOWOTNIK PUZZLE	TANSOFT	24.00 DM
OR084	O	SUPER BREAKOUT	ORIC	19.00 DM
OR090	O	STARSHIP	SECTOR 7	24.00 DM
OR092	O	PAINTER	A&F SOFTWARE	24.00 DM
OR093	OA	MINED OUT	QUICKSILVA	24.00 DM
**********		ADVENTURE SOFTWARE	**********	
OR100	OA	007 A VIEW TO A KILL	DOMARK	39.00 DM *
OR110	OA	THE HOBBIT	TANSOFT	53.00 DM
OR111	O	THE GRAIL	SEVERN SOFTW.	28.00 DM
**********		ANWENDER SOFTWARE	**********	
OR150	OA	O.G.D.S. (SUPER GRAPHIKPROGRAMM)	KDB-SOFTWARE	39.00 DM *
OR151	OA	ORITALK (Sprachsynthese)	KDB-SOFTWARE	39.00 DM *
OR152	O	SUPEREXTENDED-BASIC V2.0	KDB-SOFTWARE	45.00 DM *
OR153	A	SUPEREXTENDED-BASIC V2.1	KDB-SOFTWARE	45.00 DM *
OR154	OA	ORION ASSEMBLER/DISASSEMBLER/MONITOR	LOTHLORIEN	39.00 DM
OR156	OA	DBUG MONITOR/DEBUGGER	NO MANS LAND	34.00 DM
OR157	OA	AUTHOR (engl. Textverarbeitung)	TANSOFT	35.00 DM
**********		DISC-SOFTWARE FÜR ORIC- UND CUMANA-DOS	**********	
OR200	A	KARTEIBOX (SUPER Datenverwaltung)	KDB-SOFTWARE	39.00 DM *
OR201	OA	FORTH V.4 (3"-Diskette)		59.00 DM *
OR202	OA	FORTH V.4 (5,25"/40TRACK Diskette)		49.00 DM *
OR203	OA	FORTH V.4 (5,25"/80TRACK Diskette)		49.00 DM *
**********		HARDWARE	**********	
ER301	OA	EPROM mit ORIC-1 ROM		41.00 DM
ER302	OA	EPROM mit ATMOS ROM		41.00 DM
ER303	OA	DISK-CONTROLER kompl.Anschlußfertig		349.00 DM *
ER304	OA	ORIC 16K auf 48K kompl.mit Einbau		79.00 DM *
ER305	OA	EINBAU-UMSCHALTUNG ORIC-1/ATMOS		71.00 DM
ER306	OA	ATMOS-ROM V1.3		41.00 DM
ER307	OA	ORIC-1 ROM V1.2		41.00 DM
ER308	OA	5.25" DISK/DOPPELS. KOMPL. 40/80 TR.	UMSCHALTBAR	519.00 DM *
ER309	OA	5.25" 40/80 DOPPELS.UMSCHALTB. OHNE	NETZT.U.GEHÄU.	419.00 DM *

ALLE PREISE INKL. MWST UND VERSANDKOSTEN
VERSAND INS AUSLAND NUR PER VORKASSE !!!

KDB-COMPUTER-VERSAND / KORNSTR. 28 / D-5800 HAGEN 7
TELEFON: (02331) 40 06 01
BANKVERB.: PSCHA DORTMUND (BLZ 440 100 46) NR.:1117 71-469