ATMOSPHÄRE

FACHZEITSCHRIFT FÜR ORIC-1 UND ATMOS COMPUTER

Nr.3

NOVEMBER/DEZEMBER 86

Oric	Reihe (a)	Nr.	Reihe (c)	Oric
34	G N D	1	G N D	34
	+ 12V	2	+ 12V	
	− 12V	3	− 12V	
	28V	4	28V	
		5		
	Decoder #320	6	Decoder #330	
	Decoder #340	7	Decoder #350	
	Decoder #360	8	Decoder #370	
	Decoder #380	9	Decoder #390	
	Decoder #3A0	10	Decoder #3B0	
	Decoder #3C0	11	Decoder #3D0	
	Decoder #3E0	12	Decoder #3F0	
	Read	13	Write	
	* ROMEN	14	DATEN Freigabe	
	ROMDIS IN	15	ROMDIS OUT	
1	MAP	16	ROMDIS	2
3	φ 2	17	RESET	4
5	I / O	18	I / O Control	6
7	R / W	19	IRQ	8
9	D 2	20	D 0	10
11	A 3	21	D 1	12
13	A 0	22	D 6	14
15	A 1	23	D 3	16
17	A 2	24	D 4	18
19	D 5	25	A 4	20
21	A 5	26	D 7	22
23	A 6	27	A 15	24
25	A 7	28	A 14	26
27	A 8	29	A 13	28
29	A 9	30	A 12	30
31	A 10	31	A 11	32
33	+ 5V	32	+ 5V	

* ROMEN = Adresse #C000 − #FFFF

Okt.1986

ORIC − Busbelegung
fuer 64 pol Stecker

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

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

I N H A L T :

I M P R E S S U M :

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

VORWORT

Liebe Leser und Leserinnen,
Wir investieren viel Zeit und Arbeit um die ATMOSPHÄE heraus zu bringen. Und mancheiner von Ihnen hat sein letztes Taschengeld geopfert, um ein Jahresabonnement zu bezahlen. Es wäre daher fatal, wenn wir an Ihren Interessen vorbei schreiben würden.
Bisher haben wir nur einen "Hilferuf" erhalten (s. LESER-FORUM). Uns stellt sich nun die Frage "Ist dies ein Einzelfall oder ist dies der einige, der den Mut gehabt hat zu sagen, daß er nichts versteht?" Es kann ja sein, daß die Redakteure Dinge voraussetzen,die nicht vorhanden sind. Doch dann müssen Sie uns darauf aufmerksam machen und gezielte Fragen stellen.
Wir schreiben die Zeitung um Ihnen zu helfen und nicht, weil wir nichts besseres zu tun haben und Sie haben die ATMOSPHÄRE nicht bestellt, weil Sie das Geld übrig hatten.
Allerdings möchten wir uns auch nicht Ihren Kopf zerbrechen.Das heißt wir wollen uns nicht tagelang hinsetzen, irgendwelche Programme schreiben und Ihnen die Listings vorsetzen, die Sie dann stundenlang eintippen. Der Lern- und Erfahrungswert wäre gleich Null.
Natürlich werden wir auch komplette Programme veröffendlichen, das ist dann die sogenannte "PUBLIC DOMAIN"-Software, die für jedermann interessant sein kann.(Da kann auch ruhig mal ein kleines Spiel dabei sein.) Doch sollen dies Ausnahmen bleiben.
Das berühmte "AHA"-Erlebnis kommt eben nur beim Programmieren, daß heißt wenn man sich mit einem "Problem" auseinandersetzt.Dann kann es auch nicht passieren, seit Jahren nicht aus der 1.Klasse zu kommen.
Denken Sie mal daran, daß die Redakteure dieser Zeitschrift,zu Anfang teilweise weniger Informationen hatten als Sie. Wer weis,ob sie heute solche Experten wären, wenn dies nicht so gewesen wäre. Heute können sich diese Leute an jeden x-beliebigen Computer setzen und wissen in 5 Minuten mehr als nur wo der Knopf zum ein- und ausschalten ist.
Natürlich haben sie das nicht allein geschaft, denn auch "Profis" sehen manchmal den Wald vor lauter Bäumen nicht. Sie haben dann mit anderen ORIC-Usern gesprochen und diese um Rat gefragt. Da ist keinem eine Zacke aus der Krone gefallen.
Und Sie haben ein ganzes Heer von Ratgebern,denn Sie haben die ATMOSPHÄRE mit den Redakteuren und den LESERN.
Doch was Sie am dringensten brauchen sind "Probleme". Dann werden Sie sich z.B. auf solche Leckerbissen wie die MCP-40-TOOLS von Hr.Schmidt stürzen, weil Sie gerade mit dieser kleinen Routine mit IHREM "Problem" ein Stück weiterkommen.
Also, wer keine "Probleme" hat, der macht sich welche. Sinn oder Unsinn eines Programms sind dabei völlig uninteressant. Nur die Lösung des "Problems" zählt.
Ich kann mich noch gut an die müde lächelnden Gesichter meiner Bekannten erinnern, wenn ich es geschafft hatte meinen kleinen "lächerlichen" ATMOS zu bewegen irgendwelche Männchen über den Bildschirm hüpfen zu lassen.
Heute habe ich nur ein müdes Lächeln, wenn Sie voller Panik erzählen, daß im nächsten Monat ihr Betrieb auf Computer umgestellt wird und ihr Chef so einen komischen Blick bekam, als sie ihm gestehen mußten, daß sie nur wissen wie COMPUTER geschrieben wird.

Also wenn Sie in Ihrem Programm etwas in die Statuszeile schreiben oder Speicherbereiche hin und her schieben möchten und nicht wissen wie das geht, dann fragen Sie. Entweder wir oder ein Leser wird Ihnen bestimmt weiterhelfen. Natürlich sind wir auch an Ihren kleinen Lösungs-Routinen interessiert. Vieleicht haben Sie ein Problem gelöst, an dem sich selbst die "Profis" die Zähne ausgebissen haben.

K.D.B.

Teil 2

von Ekkehard Otto

4.) Zu zweit geht es besser.

In der ersten Folge haben wir die Rechenroutinen mit zwei Argumenten kennen gelernt. Wenn man sie benutzen will, muß man natürlich auch den FAC2 ansprechen können. Dazu braucht man noch einige Routinen, die Transportaufgaben zwischen den FACs und zwischen dem RAM und den FACs übernehmen.

Transportroutinen

Name	ORIC1	ATMOS	Beschreibung
LDFA1	DE73	DE7B	FAC1=RAM(A,Y) setzt $DØ
LDFA2	DD4D	DD51	FAC2=RAM(A,Y) setzt $DØ und $DE
STFA1	DEA5	DEAD	RAM(X,Y)=FAC1
MOV12	DEDD	DEE5	FAC2=FAC1 setzt $DØ
MOV21	DECD	DED5	FAC1=FAC2 setzt $DØ
GIVAY	D3ED	D499	FAC1=Y,A 16bit Zahl mit Vorzeichen
GIV2	D8D5	DF4Ø	FAC1=Y,A (Ylow,Ahigh)ohne Vorzeichen
GIVA	DF15	DF24	FAC1=A 8bit Integer mit Vorzeichen
FLINT	D871	D92C	$D3/4=INT(FAC1)
FOUT	EØD1	EØD5	Der Inhalt von FAC1 wird nach $1ØØ ff als ASCII-Zeichen geschrieben.
PI	D8EE	DE77	FAC1=PI
STFAB	DE98	DEAØ	($CB ff)=FAC1 setzt $DØ
STFA6	DE9B	DEA3	($C6 ff)=FAC1 setzt $DØ

Im ROM sind auch einige benötigte Konstanten gespeichert, die mit

```
LDA #Adresse low
LDY #Adresse high
JSR LDFA1 bzw. LDFA2
```

in die FACs geladen werden können. Die Konstanten sind in dem gleichen Format abgespeichert, wie sie von den Routinen STFA* erzeugt werden, auch BASIC-Variable stehen in dem selben Format im Speicher, wenn man also die Adresse einer BASIC-Variable kennt, kann man sie auch mit LDFA* in die FACs laden. (Über das Format der Variablen und wie man die Adresse im Maschinenprogramm finden kann, sprechen wir später.) Für ganz Ungeduldige schon 'mal ein Tip: im BASIC-Programm findet man mit X=DEEK(#B6) die Adresse der Variablen X und kann dann mit PEEK die folgenden fünf Bytes untersuchen.

ORIC1	ATMOS	Wert
DC46	DC77	LN(1Ø)
DC4B	DC81	1
DC9B	DC65	SQR(.5)
DC6A	DCAØ	SQR(2)
DC6F	DCA5	-1/2
DC74	DCAA	LN(2)
EØA7	EØAB	99999999.9
EØAC	EØBØ	999999999
EØB1	EØB5	1ØØØØØØØØØ
E2Ø1	E2Ø5	1/2
E4Ø3	E4Ø7	PI/2
E4Ø8	E4ØC	2*PI
D8E9	DC7C	PI
DDBA	DDBE	1Ø
E278	E27C	1/LN(2)
E4ØD	E411	1/4

Als Anwendung dieser Routinen und Konstanten hier zwei Routinen in einem Programm. Als erstes die Berechnung der dritten Wurzel einer Zahl nach dem gleichen Verfahren, nach dem auch der Interpreter die zweite Wurzel einer Zahl berechnet. Es gilt ja: $\sqrt[3]{x}=x^{(1/3)}$ und $x^{(1/3)}$=EXP(LN(X)/3). Deshalb wird zunächst der ln(x) ausgerechnet, durch 3 geteilt und dann EXP davon berechnet. Der 2-er-Logarithmus einer Zahl wird berechnet, indem man den natürlichen Logarithmus der Zahl berechnet und durch den natürlichen Logarithmus von 2, der ja als Konstante vorliegt, teilt. (Wem das zuviel Mathematik ist, kann die Programme ja auch einfach nur so verwenden.)

```
00110 B400                      ;**********************************
00120 B400                      ;* BEISPIEL 2                     *
00130 B400                      ;* DRITTE WURZEL EXP(LN(X)/3)     *
00140 B400                      ;* 2-ER LOGARITHMUS LN(X)/LN(2)   *
00150 B400                      ;**********************************
00210 B400 MOV12=$DEDD
00220 B400 FDIV=$DDE3
00250 B400 LN=$DC79
00260 B400 EXP=$E2A6
00270 B400 GIVA=$DF15
00275 B400 FMULM=$DCB7
00300 B400 20 79 DC W3          JSR LN      ;"3. Wurzel
00310 B403 20 DD DE             JSR MOV12   ;"sichern
00320 B406 A9 03                LDA #3
00330 B408 20 15 DF             JSR GIVA    ;"3 nach FAC1
00340 B40B 20 E3 DD             JSR FDIV
00350 B40E 20 A6 E2             JSR EXP
00360 B411 60                   RTS
00370 B412 20 79 DC L2          JSR LN      ;"2er-Logarithmus
00380 B415 A9 78                LDA #$78
00390 B417 A0 E2                LDY #$E2
00400 B419 20 B7 DC             JSR FMULM
00420 B41C 60                   RTS
```

Da in dem Maschinenprogramm zwei Funktionen definiert werden, bietet sich natürlich die Verwendung von USR und & an. Das dazugehörige BASIC-Programm könnte etwa so aussehen.

```
100 DEF USR=#B400
110 DOKE#2FC,#B412
120 INPUT X
130 PRINT USR(X)
140 PRINT &(X)
```

So, bevor nun jeder die Funktion schreibt, die er für seine Anwendung gebrauchen kann, (ATMOSPHÄRE ist bereit, Beispiele aus dem Leserkreis zu veröffentlichen) hier noch ein Beispiel, das etwas umfangreicher ist und auch eine Verzweigung enthält, die Berechnung von ARCSIN(x) nach den Gleichungen:

```
IF ABS(X)=1 THEN Y=SGN(X)*PI/2
 ELSE
Y=ATN(X/SQR(-X*X+1)
```

```
00110 B400                ;**********************************
00120 B400                ;* BEISPIEL 3                     *
00130 B400                ;* ARCSIN(X)=ATN(X/SQR(-X*X+1))*
00140 B400                ;* MIT ABFRAGE X=1 ODER X=-1      *
00150 B400                ;**********************************
00210 B400 MOV12=$DEDD
00220 B400 FDIV=$DDE3
00230 B400 SQR=$E22A
00240 B400 CHS=$E26D
00250 B400 FADDA=$E072
00260 B400 STFA1=$DEA5
00270 B400 LDFA2=$DD4D
00275 B400 LDFA1=$DE73
00280 B400 ATN=$E43B
00290 B400 FMUL=$DCBA
00291 B400 FMULM=$DCB7
00295 B400 SIGN=$DF04
00296 B400 SGN=$DF12
00300 B400 20 DD DE       JSR MOV12   ;"X NACH FAC2
00305 B403 A9 00          LDA #0
00306 B405 A0 00          LDY #0
00310 B407 20 A5 DE       JSR STFA1   ;"X NACH $0/1
00311 B40A A5 D5          LDA $D5     ;"VORBEREITEN
00312 B40C 45 DD          EOR $DD     ;"DER FLAGS
00313 B40E 85 DE          STA $DE     ;"FUER DIE
00315 B410 A5 D0          LDA $D0     ;"RECHENROUTINEN
00320 B412 20 BA DC       JSR FMUL    ;"X*X
00330 B415 20 6D E2       JSR CHS     ;"-X*X
00340 B418 A9 01          LDA #1
00350 B41A 20 72 E0       JSR FADDA   ;"-X*X+1
00351 B41D 20 04 DF       JSR SIGN    ;"FAC =0
00352 B420 D0 12          BNE NN      ;"NEIN, WEITER
00353 B422 A9 00          LDA #0
00354 B424 A0 00          LDY #0
00355 B426 20 73 DE       JSR LDFA1   ;"X AUS $0/1 NACH FAC2
00356 B429 20 12 DF       JSR SGN     ;"SGN(X)
00357 B42C A0 E4          LDY #$E4
00358 B42E A9 03          LDA #$03
00359 B430 20 B7 DC       JSR FMULM   ;"* PI/2
00359 B433 60             RTS
00360 B434 20 2A E2 NN    JSR SQR     ;"SQR(-X*X+1)
00370 B437 A9 00          LDA #0
00380 B439 A0 00          LDY #0
00390 B43B 20 4D DD       JSR LDFA2   ;"X AUS $0/1 NACH FAC2
00400 B43E 20 E3 DD       JSR FDIV    ;"X/SQR(...)
00410 B441 20 3B E4       JSR ATN
00420 B444 60             RTS
```

Für dieses Mal soll es nun reichen (nicht vergessen: ATMOSPHÄRE wartet auf Programme). In der nächsten Folge geht es praktische Beispiele.

um Funktionen mit Strings als Argumenten und um Funktionen mit mehr als einem Argument. Natürlich gibt's auch wieder

ORIC-1 MCP-40-TOOLS von Felix Schmidt

Das Programm 'MCP Output' macht dem lästigen hantieren mit Strings, bei der Befehlseingabe ein Ende.
FORMAT: !<BEFEHL> <n1>,<n2>,...
Z.B.: !M 240,-10
Man kann sowohl Ziffern als auch Variablen verwenden.Dabei ist zu beachten, daß nach dem Befehl mindestens ein Wert folgt, die bei einstelligen Befehlen jedoch keinerlei Einfluß auf die nachfolgenden Befehle hat. Disk-Besitzer müssen darauf achten, daß die Einsprungadresse des '!' geändert wird. Nach dem Gebrauch der Routine muß in #2F5 wieder der Wert #4C4 geschrieben werden.
Anfang:#9300 / Ende:#935F

'MCP HARDCOPY', ist wie der Name schon sagt eine Hardcopyroutine, die jedoch nicht nach dem Punkt-für-Punkt-Verfahren arbeitet, sondern ganze Linien zieht. Eine Besonderheit ist, daß man das Bild sowohl in X- als auch in Y-Achse strecken kann. Zero Page 4 bestimmt den X-Zerrfaktor, Adresse 5 den den Y-Zerrfaktor.
Anfang: #9500 / Ende> #9630

Im dritten Programm, 'MCP CIRCLE', wird im Prinziep die gleiche Routine wie im ORIC-1 Betriebssystem verwendet, es wurden nur einige Unterprogramme verändert. Aus diesem Grund sind auch nur die Radien von 1ß127 erlaubt. Der Radius wird mit POKE #2E1,<WERT> angegeben.
Es gibt allerdings eine Kompromißlösung,um den Radius zu verdoppeln
POKE#91AF,50:POKE#91BA,50:
POKE#91C0,50:POKE#91CB,50
Nun sind jedoch wesendlich deutlichere Abstufungen zu erkennen.
Anfang: #9100 / Ende: #9204

Da alle Programme hintereinander im Speicher stehen, können sie gleichzeitig verwendet werden.

ANMERKUNG-Redaktion:
Wer schreibt diese Plotter-Routinen so, daß sie auf ORIC-1 und ATMOS gleichzeitig laufen.

DISK - MONITOR von Klaus Grigat

Wie in der letzen Ausgabe der ATMOSPHÄRE angekündigt, folgt auf den nächsten Seiten das Listing zu meinem 'Disk-Monitor'. Zu dem Programm selbst ist nicht viel zu sagen, da es Menü-Gesteuert ist und sich dadurch selbst erklärt. Zu erwähnen ist noch, daß Sie mit der Tastenkombination <CTRL F>, die von einem Programm belegten Sektoren direkt verfolgen können. Alle Zahlenwerte müssen dezimal eingegeben werden. Das Programm läuft ohne Änderung auf ORIC-1 und ATMOS. Doch nun viel Spaß beim Eintippen und der Erforschung Ihrer Disketten.

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

Die Kontaktsuche aus der letzten Ausgabe war ein voller Erfolg!!!
Ein neuer ORIC-Club ist entstanden und kann schon 7 Mitglieder aufweisen.

ORIC-CLUB-DUISBURG

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

Wir hoffen,daß das gute Beispiel schule macht und sich weitere Personen als erste Kontaktperson zur Verfügung stellen.

```
10 REM *** MCP-OUTPUT ***
20 GOSUB 1000:POKE#2F5,#9300:END
1000 READV,D
1010 REPEAT:SU=0:READD$,CS$:FORI=1TOLEN(D$)STEP2
1020 X=VAL("#"+MID$(D$,I,2)):SU=SU+X:POKEV,X:V=V+1:NEXT
1030 IFSU<>VAL("#"+CS$)THENPRINT"DATA ERROR IN LINE";DEEK(#AE):STOP
1040 UNTILV>D:RETURN
2000 DATA#9300,#935F
2001 DATAA000990094EA984820E200208BCE20D1E068A8A2FFC8E8BD0001F00AC92090,F75
2002 DATAF6990094D0F0EA984820E800C92CD00C68A8A92C9900949848D0CEEA68A8A9,10B9
2003 DATA009900942001EDA0009848B90094207BF568A8C8B90094D0F0A90D207BF520,DE3
2004 DATAC7EC60,213
```

```
10 REM *** MCP-HARDCOPY ***
20 GOSUB 1000: END
1000 READV,D
1010 REPEAT:SU=0:READD$,CS$:FORI=1TOLEN(D$)STEP2
1020 X=VAL("#"+MID$(D$,I,2)):SU=SU+X:POKEV,X:V=V+1:NEXT
1030 IFSU<>VAL("#"+CS$)THENPRINT"DATA ERROR IN LINE";DEEK(#AE):STOP
1040 UNTILV>D:RETURN
2000 DATA#9500,#9630
2001 DATA2001EDA9008510A9A08511A9208D1502A92C8DE4BFA92D8DE5BFA90D8DE9BF,E8F
2002 DATAA9C88501A9308DE6BF8DE7BF8DE8BFA5058507A51048A51148AD150248A900,E4F
2003 DATA8502A9F08500A9308DE1BF8DE2BF8DE3BF20BB952004F0208895C600D0F320,1072
2004 DATAF59520A595C607F00D688D150268851168851O4C3295686868C601D0B4A948,D3C
2005 DATA207BF5A90D207BF520C7EC60A5048506A202FEE1BFBDE1BFC93A9008A9309D,FED
2006 DATAE1BFCA10EEC606D0E860A202FEE6BFBDE6BFC93AB00160A9309DE6BFCA10ED,12EB
2007 DATA60A000B1104C2096F004A901D002A900C502D00160A502F004A944D002A94D,C24
2008 DATA8DE0BFEAA0008403B9E0BF207BF5A403C8C00A90F1A5024901850260A502F0,F4E
2009 DATA07EAEAEAEA20CF95A952207BF5A930207BF5A92C207BF5A92D207BF5A93120,FEC
2010 DATA7BF5A90D207BF56055297FC92090A22D15024CC295C295555555,B6B
```

```
10 REM *** MCP-CIRCLE ***
20 GOSUB 1000:END
1000 READV,D
1010 REPEAT:SU=0:READD$,CS$:FORI=1TOLEN(D$)STEP2
1020 X=VAL("#"+MID$(D$,I,2)):SU=SU+X:POKEV,X:V=V+1:NEXT
1030 IFSU<>VAL("#"+CS$)THENPRINT"DATA ERROR IN LINE";DEEK(#AE):STOP
1040 UNTILV>D:RETURN
2000 DATA#9100,#9204
2001 DATA20F091AD1A0238EDE102A8AE190220A6EFADE102850F20F0F3A9808D1B028D,E1F
2002 DATA1D02A9008D1C02ADE1028D1E02A900850F204E91207E91A50FF00320D091AD,AF0
2003 DATA1C02D0EAAD1E02CDE102D0E220C7EC60AD1D02AE1E0220DFF3A50C186D1B02,D19
2004 DATA8D1B02AD1C02850C650D8D1C02C50CF00FB00620AE914C799120B491A90185,AF2
2005 DATA0F60AD1B02AE1C0220DFF338AD1D02E50C8D1D02AD1E02850CE50D8D1E02C5,A5A
2006 DATA0CF00FB00620BF914CA99120C591A901850F60A9318DE2BF60A92D8DE1BFA9,EDF
2007 DATA318DE2BF60A9318DE5BF60A92D8DE4BFA9318DE5BF60A0008400B9E0BF207B,10B2
2008 DATAF5A400C8C00790F1A9308DE1BF8DE2BF8DE4BF8DE5BF602001EDA94A8DE0BF,12CB
2009 DATAA92C8DE3BFA90D8DE6BF4CE191,7AA
```

```
0 REM ************************
1 REM *                      *
2 REM *  ORIC Disk-Monitor   *
3 REM *                      *
4 REM *  ' K. Grigat  4/85   *
5 REM *                      *
6 REM *  Stand: 01.01.1986   *
7 REM *                      *
8 REM ************************
9 REM
10 HIMEM#7FFE: CLS: PAPER 0: INK 2: CLEAR
11 POKE48035,0: POKE618,11: GOSUB 1000: IF PEEK(#C076)<>0 THEN POKE#98E6,#D9
12 JOB$(1)="Sektor lesen ......": JOB$(2)="Sektor schreiben .."
13 JOB$(3)="Monitor ...........": JOB$(4)="Ende ............."
14 LW=#A000: T=#A001: S=#A002: A1=#A100: A2=#A200: RD=#A003: WR=#A006
15 GOSUB 300: GOSUB 600: GOSUB 400
16 GOSUB 400
17 FOR K=1 TO 4: PRINT SPC(5) JOB$(K); K: PRINT: NEXT K
18 PRINT: PRINT: PRINT: PRINT SPC(5) "Bitte waehlen Sie ! ";
19 REPEAT: K$=KEY$: UNTIL K$="1" OR K$="2" OR K$="3" OR K$="4": J=VAL(K$)
20 IF J=3 THEN 701
21 IF J=4 THEN POKE LW,D0: !CLOSE R: !LOAD"BOOTUP.COM"
22 REM
23 REM * Spur u. Sektor waehlen *
24 REM
25 GOSUB 400
26 IF J=1 THEN J$=" -  lesen  - " ELSE J$="- schreiben -"
27 J$=CHR$(17)+"  "+J$+"   "+CHR$(16): PLOT 9,7,J$: PRINT: PRINT
28 PRINT "Bitte Spur eingeben      0 - "TR$" ";
29 B%=LEN(TR$): U=0: O=VAL(TR$)
30 GOSUB 50: SP$=RIGHT$("000"+FR$,B%)
31 SX=VAL(SP$): PRINT: PRINT
32 PRINT "Bitte Sektor eingeben    1 - 16 ";
33 B%=2: U=1: O=16
34 GOSUB 50: SE$=RIGHT$("00"+FR$,2)
35 SE=VAL(SE$): PRINT: PRINT
36 GOSUB 206
37 GOSUB 400
38 IF J=2 THEN GOSUB 150: GOTO 16
39 GOSUB 100: GOTO 16
50 FR$=""
51 GET U$: IF U$=CHR$(13) AND VAL(FR$)>=U THEN RETURN
52 IF U$=CHR$(127) THEN 58
53 IF U$<CHR$(48) OR U$>CHR$(57) THEN PING: GOTO 51
54 IF LEN(FR$)>=B% THEN PING: GOTO 51
55 FR$=FR$+U$: PRINT U$;
56 IF VAL(FR$)<=O THEN 51
57 U$=CHR$(127): PING
58 IF LEN(FR$)=0 THEN 51
59 PRINT U$;
60 FR$=LEFT$(FR$,LEN(FR$)-1)
61 GOTO 51
100 REM
101 REM * Sektor lesen *
102 REM
103 POKE LW,DR: POKE T,SK: POKE S,SE: CALL RD
104 POKE LW,D0: RETURN
150 REM
151 REM * Sektor schreiben *
152 REM
153 POKE1277,1: POKE LW,DR: POKE T,SK: POKE S,SE
154 PLOT 3,7,"Sektor wird zurueckgeschrieben !": CALL WR: POKE1277,0
```

```
155 PR=PEEK(1279): IF PR<>6 AND PR<>26 THEN 159
156 PING: PLOT 3,7,"Schreibschutz !     --> Leertaste": POKE 1279,0
157 REPEAT: K$=KEY$: UNTIL K$=" "
158 GOTO 153
159 PLOT 3,7,"                                          ": POKE LW,D0: RETURN
200 REM
201 REM * Spur und Sektor anzeigen *
202 REM
203 SP$=STR$(SX): LP=LEN(SP$): SE$=STR$(SE): LE=LEN(SE$)
204 SP$=RIGHT$(SP$,LP-1): SE$=RIGHT$(SE$,LE-1)
205 SP$=RIGHT$("00"+SP$,L-1): SE$=RIGHT$("0"+SE$,2)
206 FOR K=1 TO L-1: POKE48049+K,ASC(MID$(SP$,K,1)): NEXT K
207 FOR K=1 TO 2: POKE48073+K,ASC(MID$(SE$,K,1)): NEXT K
208 SK=SX: IF SK>(T1-1) THEN SK=SK+TD
209 RETURN
300 REM
301 REM * Bildschirmkopf *
302 REM
303 PRINT "  Spur:                        Sektor:"
304 PRINT: PRINT "  * * * ORIC  Disk - Monitor * * *"
305 PRINT: PRINT "  Diskettenname:"
306 DOKE48040,1043: DOKE48200,1043: POKE48121,1
307 PRINT: PRINT: PRINT: PRINT
308 DOKE49040,787: DOKE 49080,787
309 PLOT 3,25,"Sektor  + -         Spur     < >"
310 PLOT 3,26,"Schreiben --> S     Ende     <-'"
311 RETURN
400 REM
401 REM * Bildschirm loeschen *
402 REM
403 POKE618,10: CALL#9873: POKE618,11
404 RETURN
500 REM
501 REM * Spur und Sektor weiter *
502 REM
503 IF K$="." AND SX=TR THEN SX=-1
504 IF K$="," AND SX=0 THEN SX=TR+1
505 IF K$="=" AND SX<TR AND SE=16 THEN SX=SX+1: SE=0
506 IF K$="=" AND SE=16 THEN SE=0
507 IF K$="-" AND SX>0 AND SE=1 THEN SX=SX-1: SE=17
508 IF K$="-" AND SE=1 THEN SE=17
509 IF K$="." THEN SX=SX+1: GOSUB 200: GOSUB 100
510 IF K$="," THEN SX=SX-1: GOSUB 200: GOSUB 100
511 IF K$="=" THEN SE=SE+1: GOSUB 200: GOSUB 100
512 IF K$="-" THEN SE=SE-1: GOSUB 200: GOSUB 100
513 IF K$="S" THEN GOSUB 150: GOTO 711
514 IF K$=CHR$(16) THEN GOSUB 550: GOTO 711
515 IF K$=CHR$(6) THEN GOSUB 520: GOSUB 200: GOSUB 100
516 IF K$="." OR K$="," OR K$="=" OR K$="-" OR K$=CHR$(6) THEN 701
517 GOTO 718
520 IF SE=1 AND SX=0 THEN RETURN
521 IF PEEK(#A101)=0 THEN RETURN
522 SE=PEEK(#A101): SX=PEEK(#A100)
523 IF SX>#7F THEN SX=#50+SX AND #7F
524 RETURN
550 REM
551 REM * Schirmkopie *
552 REM
553 POKE618,10: WAIT 50: PR$="": K1=PEEK(0)
554 GOSUB 575
555 IF PEEK(#C076)=0 THEN CALL#ED01
556 FOR K=0 TO 8: PR$=PR$+CHR$(PEEK(48219+K)): NEXT K
```

```
557 LPRINT "      Diskette :    ";CHR$(14);PR$;CHR$(20): LPRINT
558 LPRINT "      Spur : "; SP$; "    Sektor : "; SE$: LPRINT: LPRINT
559 FORK0=0TO1: POKE0,(K0*128): CALL#9800: FOR K=0 TO 15
560 KA=48400+K*40: PR$="      "
561 FOR KK=2 TO 39
562 PR$=PR$+CHR$(PEEK(KA+KK))
563 NEXT KK
564 LPRINT PR$
565 NEXT K: NEXT K0: POKE0,K1: CALL#9800: GOSUB 800
566 LPRINT CHR$(12);
567 POKE618,11: IF PEEK(#C076)=0 THEN CALL#ECC7
568 RETURN
575 REM
576 REM * Drucker ein ? *
577 REM
578 POKE#301,13: POKE#300,PEEK(#300) AND 239: POKE#300,PEEK(#300) OR 16
579 IF PEEK(#30D) THEN 583
580 PLOT8,7,CHR$(12)+CHR$(1)+"Drucker einschalten !":WAIT1: ZAP
581 IF PEEK(520)=169 THEN POP: GOTO 583
582 GOTO 578
583 PLOT9,7,"                                ": RETURN
600 REM
601 REM * Laufwerkswahl *
602 REM
603 PRINT "Bitte geben Sie das gewuenschte": PRINT
604 PRINT "Laufwerk an   0 ... 3 !  ";
605 GET DR$: IF DR$<"0" OR DR$>"3" THEN 605
606 PRINT DR$: DR=VAL(DR$)
607 S1=A1: S2=S1+4
608 POKET,0: POKES,1: POKE#A019,192: POKE#A01C,160: CALL RD: D0=PEEK(#A000)
609 POKET,0: POKES,1:  KE#A019,160: POKE#A01C,192: POKELW,D0: CALL RD
610 IF PEEK(S1+DR)<>0 THEN GOSUB 400: POKELW,DR: GOTO 613
611 PING:GOSUB 400: PRINT "Laufwerk "DR$" nicht angemeldet !"
612 WAIT 300: GOTO 624
613 PRINT "Legen Sie bitte die Diskette ein,"
614 PRINT "die untersucht werden soll !"
615 PRINT: PRINT: PRINT SPC(10) "Leertaste  ";
616 REPEAT: K$=KEY$: UNTIL K$=" "
617 POKE LW,DR: CALL 9
618 FOR K=0 TO 8
619 POKE(48219+K),PEEK(41240+K)
620 NEXT K
621 T1=PEEK(#A100+DR): T2=PEEK(#A104+DR): TR=T1+T2-1: TD=128-T2
622 TR$=STR$(TR): L=LEN(TR$): TR$=RIGHT$(TR$,L-1)
623 RETURN
624 GOSUB 400
625 PRINT "Ist das Laufwerk "DR$" angeschlossen ?": PRINT
626 PRINT "   J / N    ";
627 REPEAT: K$=KEY$: UNTIL K$="J" OR K$="N": PRINT K$: WAIT 100
628 IF K$="J" THEN PRINT "Laufwerk bitte mit !SET anmelden": J=4: GOTO 21
629 CLS: RUN 12
700 REM
701 REM * Monitor *
702 REM
703 VP=10: HP=2: POKE49041,4:POKE49081,4
704 A=#A100
705 SP=A: EP=A+120
706 POKE 618,PEEK(618) AND 254
707 A3=A-#A100: POKE0,A3: CALL#9800
708 POKE618,PEEK(618) OR 1
709 A=SP: VP=10: HP=2
```

```
710 X=(HP-2)*3+7: Y=VP: GOSUB 800
711 GOSUB 875
712 IF K=13 THEN 775
713 IF K=8 THEN 724
714 IF K=9 THEN 734
715 IF K=10 THEN 736
716 IF K=11 THEN 726
717 GOTO 500
718 GOSUB 750
719 V=PEEK(A+HP-2): IF V>127 THEN V=V-128
720 IF V<32 OR V=127 THEN V=46
721 POKE48000+40*VP+29+HP,V
722 IF K<0 THEN ZAP
723 GOTO 710
724 HP=HP-1: IF HP>=2 THEN 710
725 HP=9
726 A=A-8: VP=VP-1
727 IF A>=SP AND A<=EP THEN 710
728 IF A<#A100 AND K=8 THEN HP=2
729 IF A<#A100 THEN PING: A=#A100: VP=VP+1: GOTO 727
730 IF A<#A180 THEN A=#A100: VP=10: HP=2: GOSUB 400: GOTO 705
731 PRINT CHR$(11);
732 VP=VP+1: SP=SP-8
733 EP=EP-8: GOTO 710
734 HP=HP+1: IF HP<10 THEN 710
735 HP=2
736 A=A+8: VP=VP+1
737 IF A<EP+8 THEN 710
738 IF EP<#A180 THEN A=#A180: VP=10: HP=2: GOSUB 400: GOTO 705
739 IF A>#A1F8 THEN PING: A=#A1F8: VP=VP-1: GOTO 737
740 PRINT CHR$(10) CHR$(10);
741 VP=VP-1: EP=EP+8
742 SP=SP+8: GOTO 710
750 GOSUB 850: IF K<0 THEN RETURN
751 V=K
752 GOSUB 875
753 GOSUB 850: IF K<0 THEN RETURN
754 V=V*16+K: POKE A+HP-2,V
755 PRINT RIGHT$("00"+MID$(HEX$(PEEK(A+HP-2)),2),2);
756 RETURN
775 POKE618,PEEK(618) OR 1
776 POKE49041,3: POKE49081,3: GOTO 16
800 A1=PEEK(616)*40+PEEK(617)+48000
801 POKE616,Y
802 DOKE18,Y*40+48000: POKE617,X
803 POKE A1,PEEK(A1) AND 127
804 RETURN
850 K=K-48: IF K<10 THEN 853
851 K=K-17: IF K< 0 THEN 853
852 K=K+10: IF K>15 THEN K=-K
853 RETURN
875 K$=KEY$: IF LEN(K$)=0 THEN 875
876 K=ASC(K$)
877 RETURN
1000 REM
1001 REM * MC - Laderoutine *
1002 REM
1003 RESTORE
1004 FOR K=#A003 TO #A041
1005 READ A: POKE K,A
1006 NEXT K
```

```
1010 DATA #4C,#3B,#A0,#A9,#21,#A0,#D4,#8D,#85,#04,#8C,#86,#04,#A9,#00,#20
1011 DATA #87,#04,#A2,#02,#BD,#00,#A0,#9D,#00,#C0,#CA,#10,#F7,#A9,#00,#A0
1012 DATA #A1,#8C,#04,#C0,#8D,#03,#C0,#20,#34,#A0,#A9,#06,#20,#87,#04,#58
1013 DATA #60,#BA #8E,#06,#C0,#4C,#90,#04,#A9,#24,#A0,#D4,#4C,#0A,#A0
1100 REM
1101 REM * MC-Monitor Laderoutine *
1102 REM
1103 FOR K=#9800 TO #98F2
1104 READ A: POKE K,A
1105 NEXT K
1106 RETURN
1110 DATA #AD,#76,#C0,#F0,#05,#A9,#D9,#8D,#E6,#98,#20,#73,#98,#20,#A7,#98
1111 DATA #A6,#00,#8E,#00,#A2,#AD,#00,#A2,#20,#CF,#98,#8D,#03,#A2,#AD,#00
1112 DATA #A2,#20,#D3,#98,#8D,#04,#A2,#A9,#08,#85,#01,#A9,#06,#85,#02,#A9
1113 DATA #00,#85,#03,#BD,#00,#A1,#20,#CF,#98,#A4,#02,#99,#00,#A2,#E6,#02
1114 DATA #C8,#BD,#00,#A1,#20,#D3,#98,#99,#00,#A2,#E6,#02,#E6,#02,#BD,#00
1115 DATA #A1,#20,#E8,#98,#A4,#03,#99,#1E,#A2,#E6,#03,#E8,#C6,#01,#D0,#D3
1116 DATA #A0,#00,#B9,#01,#A2,#F0,#07,#20,#E5,#98,#C8,#4C,#62,#98,#C6,#04
1117 DATA #D0,#A0,#60,#A2,#00,#BD,#DE,#98,#F0,#07,#20,#E5,#98,#E8,#4C,#75
1118 DATA #98,#A2,#13,#A9,#0A,#20,#E5,#98,#A9,#0E,#20,#E5,#98,#CA,#D0,#F3
1119 DATA #A2,#00,#BD,#DE,#98,#F0,#0A,#20,#E5,#98,#20,#E5,#98,#E8,#4C,#92
1120 DATA #98,#A9,#0B,#20,#E5,#98,#60,#A2,#26,#A9,#20,#9D,#00,#A2,#CA,#D0
1121 DATA #F8,#A9,#0D,#8D,#26,#A2,#A9,#0A,#8D,#27,#A2,#A9,#00,#8D,#28,#A2
1122 DATA #A9,#41,#8D,#01,#A2,#A9,#31,#8D,#02,#A2,#A9,#10,#85,#04,#60,#6A
1123 DATA #6A,#6A,#6A,#29,#0F,#C9,#0A,#30,#02,#69,#06,#69,#30,#60,#1E,#0A
1124 DATA #0A,#0A,#0A,#0A,#00,#4C,#12,#CC,#C9,#20,#30,#04,#C9,#7E,#30,#02
1125 DATA #A9,#2E,#60
```

DIE NEUE SENSATION IST DA !!!

SUPEREXTENDED BASIC V2.0 (ORIC-1) und SUPEREXTANDED BASIC V2.1 (ATMOS)

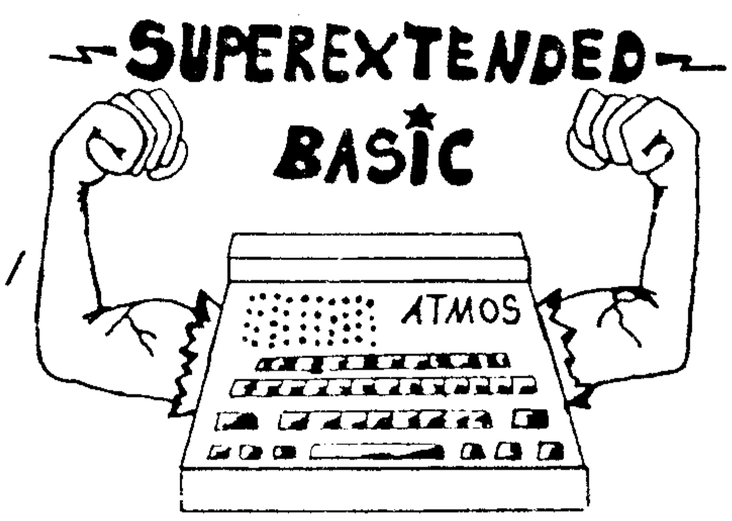
Autor: WOLFGANG KÜNZEL
Copyright: KDB - SOFTWARE

Dieses TOOLKIT läßt keine Wünsche mehr offen denn das ORIC-BASIC wird um 70 (SIEBZIG!!!) mächtige Befehle erweitert.

SUPEREXTENDED BASIC IST DISKETTEN FÄHIG!!!

Hier ein kleiner Auszug von den neuen Befehlen:

SPRINT/PRINT AT/HPRINT/
FLASH/BIG/CSET/INVERSE/
REPEAT/VARLIST/ZOOM/INIT/
ON ERROR/CHANGE/SCROL/
usw.usw.usw.....



DEFCHAR/SIZE/LORES 3/
HOME/RESTORE/WINDOW/
NAME/CLIST/VERIFY/PULL
ROTATE/SOUND x/CONVERT/

Zum Lieferumfang gehört ein ausführliches Handbuch in deutsch!
SUPEREXTENDED BASIC ist für 45,--DM (inkl.Versandkosten) bei
KDB - COMPUTER - VERSAND erhältlich. Greifen Sie zu!!!

ANZEIGE

SPEICHERKONZEPT

FORTH wurde zum Arbeiten mit Diskettenstationen entworfen. Dabei wird der Platz auf einer Diskette als eine Erweiterung des internen Speichers betrachtet. Die Daten werden dabei nicht als 'Files' sondern in Blöcken abgelegt. Die Größe dieser Blöcke ist in verschiedenen FORTH-Systemen unterschiedlich. Die Systemkonstante B/BUF (Bytes Per BUFFER) enthält die Anzahl der Bytes pro Block. Für Programme, die oft sehr umfangreich sind,ist eine bestimmte Menge von Datenblöcken zu einer größeren Einheit,den SCREENS, zusammengefaßt.
Die größe eines Screens ist in der Konstanten B/SCR (Block Per Screen) festgelegt. Durch diese Arbeitsweise ist stets die ganze Diskette "direkt" zugänglich und der Speicherplatz des Computers "erhöht" sich auf die Speicherkapazität einer Diskette.
Wollen Sie als Anwender z.B. mit dem 'Screen 3' arbeiten, so wird dieser von der Diskette in einen besonderen RAM-Speicher geladen; den sog. Diskettenpuffer. Enthält dieser Buffer schon einen anderen Screen, wird dieser erst auf die Diskette zurückgebracht und dann der neue Block eingelesen. Das Sichern eines Screens auf die Diskette geschieht mit dem Befehl FLUSH. Um den Buffer zu initialisieren, stellt FORTH den Befehl EMPTY-BUFFERS zur Verfügung.
Dieser Befehl m u s s daher nach jedem Kaltstart eingegeben werden.

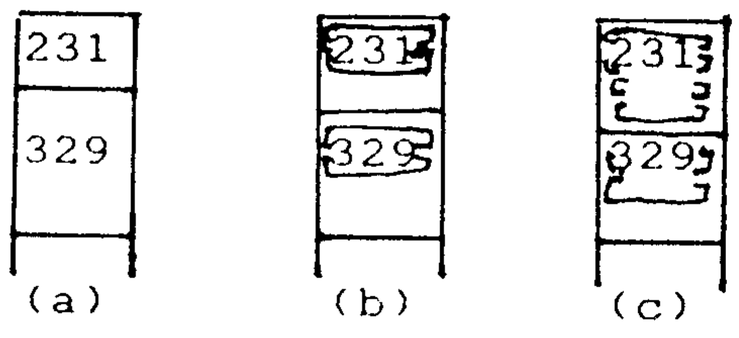
ORIC-Adaption

Um FORTH ohne eine Diskettenstation,sondern mit einem Kassettenrecorder benutzen zu können, wird beim 'ORIC-FORTH V.3' folgende Methode benutzt:Ein 7K-Byte-BlocK des RAM-Bereichs wird als "Micro-Disk" reserviert. Die FORTH-Routinen zum Laden und Speichern behandeln diesen Bereich als normale -(RAM)-Diskettenstation, mit 56 Blöcken d.h. 7 Screens. (Ein Screen besteht aus 8 Blöcken zu 128 Bytes = 1K-Byte)
Durch diese Vereinbarung ergibt sich für den Screen ein Format von 16 Zeilen zu je 64 Zeichen.
Um einen Screen vom Recorder zu laden (bzw.speichern) stellt ORIC FORTH die Befehle CLOAD od. CSAVE zur Verfügung. Diese Anweisungen erwarten zwei Parameter: z. B.
1 3 CLOAD <CR> läd Screen 1 bis 3
1 1 CSAVE <CR) speichert Screen 1 auf Kassette.
Diese Befehle können auch in ein Programm eingebaut werden.

Arithmetik in FORTH

Nachdem wir wissen, wie in FORTH gerechnet wird, wollen wir uns jetzt den einzelnen Rechenoperationen zuwenden,die uns von FORTH zur Verfügung gestellt werden.
Zuvor wollen wir uns aber mit den Zahlenbereichen in FORTH etwas genauer befassen.
In einem 8-Bit-Rechner besteht ein Speicherplatz aus einem Byte (= 8 Bit). Somit lassen sich in einem Byte Zahlen bis zu einer Größe von 255 abspeichern. (siehe Assembler-Kurs/ATMOSPHÄRE 2)
Da das aber in jedem Fall zu wenig ist, rechnet man in FORTH mit 16-Bit-Zahlen. (Single Precision Integers, einfache Genauigkeit). Eine Zahl -n- im Stapel nimmt also immer zwei Speicherplätze ein. Mit 16 Bits lassen sich somit vorzeichenlose Zahlen von 0 bis 65353 darstellen. (unsigned Integer)
Da man vereinbart hat, (und nicht nur in FORTH) daß das höchste Bit in einer Zahl immer das Vorzeichen darstellt,
(Bit 15 = 1 --> Zahl ist negativ
(Bit 15 = 0 --> Zahl ist positiv)
verschiebt sich der Zahlenbereich auf -32768 bis 32767.
Um dennoch mit größeren Zahlen rechnen zu können, bietet FORTH noch die Zahlen mit doppelter Genauigkeit. Diese Zahlen sind 32 Bit lang und umfassen den Zahlen-Bereich von -2.147.483.648 bis 2.147.483.647 .
Das Rechnen mit vorzeichenlosen Zahlen bringt einen erheblichen Speicherplatzgewinn.
Fassen wir zusammen:
FORTH rechnet grundsätzlich mit 16-Bit langen g a n z e n Zahlen oder wenn vorher vereinbart, mit

32-Bit langen Zahlen. Dezimal-*zahlen sind in Standart-FORTH* n i c h t bekannt.
Befinden sich z.B. die Zahlen 231 und 329 auf dem Stapel, so gibt die Abb.a den Stapelinhalt übersichtlicher wieder als Abb.b/c. Wir bleiben deshalb bei dieser Darstellungsart.



Seien Sie sich aber darüber im klaren, daß Abb.b den tatsächlichen Speicherinhalt im Stapel bei zwei 16-Bit-Zahlen wiedergibt.
Abb.c zeigt die Speicheraufteilung im Stapel unter der Voraussetzung, daß 231 eine 32-Bit und 329 eine 16-Bit-Zahl ist.

Mit diesem Wissen läßt sich über das Wort '.' (der Punkt) eine genauere Aussage machen; er nimmt i m m e r nur eine 16-Bit-Zahl (mit Vorzeichen) vom Stapel. Befindet sich eine vorzeichenlose oder eine 32-Bit-Zahl auf der Stapelspitze, so verursacht der der Punkt ein falsches Ergebnis.
Diese Zahlen werden mit dem Befehl 'D.' (doppel genau) abgerufen.

Der Stapel beinhaltet zwei dop peletgenaue Zahlen,die mit D. abgerufen werden.
Für die vier Grundrechenarten (mit 16-Bit-Zahlen und Vorzeichen) stellt FORTH die Worte '+-*/' zur Verfügung. Alle vier Rechenzeichen erwarten zwei einfach genaue Zahlen auf dem Stapel,verknüpft sie und legen ein 16-Bit-Ergebnis ab,daß mit dem Punkt abgerufen werden kann.

```
    10 30 + .    <CR>  40  OK
    20 20 * .    <CR> 400  OK
   144 12 # .    <CR>  12  OK
```

Folgendes Ergebnis ist falsch, da es die "Kapazität" einer 16-Bit-Zahl "übersteigt".

```
   400 400 * .  <CR> 28928 OK
```

Für die Multiplikation mit 32-Bit Zahlen gibt es die Befehle U* und M*.

U* --> Multiplitiert Vorzeichenlose 16-Bit-Zahlen
M* --> Multiplikation (32-Bit)

Beide Befehle hinterlassen 32-Bit Zahlen.

```
   40000 2 U* D.  <CR>  8000 OK
   400 400 M* D.  <CR> 16000 OK
```

Es versteht sich von selbst, daß der Befehl 'D.' bei 16-Bit-Zahlen nur Unsinn verursacht.
3 5 * D. <CR> 983040 (? D. EMPTY-STACK) OK.
Ebenso verursacht der Befehl '.' bei 32-Bit Zahlen einen Fehler,da er nur eine 16-Bit-Zahl erwartet.
Wie erwähnt,rechnet FORTH nur mit ganzen Zahlen. Das Ergebnis einer Division ist deshalb auch immer eine ganze Zahl:

		Zahlenbereich	Platzbedarf	Ausgabe
Zahlen mit einfacher Genauigkeit	mit Vorzeichen	-32768 bis 32767	16-Bit	.
	ohne Vorzeichen	0 bis 65353	16-Bit	D.
Zahlen mit doppelter Genauigkeit	mit Vorzeichen	-2.147.483.648 bis 2.147.483.647	32-Bit	D.

Eine 16-Bit-Zahl läßt sich mit dem Wort 'S->D' in eine 32-Bit-Zahl umwandeln. Z.B.

```
      5 S->D   <CR>
```

Die Zahl 5 nimmt jetzt einen Speicherplatz von 4 Bytes ein und muß mit 'D.' abgerufen werden.
Sie können eine 32-Bit-Zahl auch auf den Stapel legen, indem Sie direkt hinter der Zahl einen Punkt setzen. Z.B.

```
    12.  123.  <CR>
```

```
    150 12 / .  <CR> 12 OK
```

FORTH kennt deshalb beiden Befehle MOD und /MOD.

MOD --> Division; hinterläßt nur den R e s t
/MOD --> Division; hinterläßt Quotient und Rest

```
 150 12 MOD .  <CR>  6   OK
 150 12 /MOD . . <CR>  12 6  OK
```

Von sehr großer Bedeutung sind die Befehle '*/' und '*/MOD'.

*/ --> Multiplikation, danach Division ((n1 n2) / n3). 32-Bit-Zwischenergebnis, hinterläßt nur den Quotienten.

*/MOD --> wie oben, hinterläßt Quotient und Rest.

Beide Befehle liefern ein 16-Bit-Ergebnis.

```
300 200 10 */ . <CR>  6000 OK
300 200 10 */MOD . . <CR> 6000 0
```

Machen Sie sich mit den Eigenar- dieser Arithmetik vertraut und versuchen Sie eigene Rechnungen.

Die folgenden Befehle erklären sich von selbst und sollen hier nur kurz vorgestellt werden:

MIN --> hinterläßt die kleinere von zwei 16-Bit-Zahlen.

MAX --> hinterläßt die größere von zwei 16-Bit-Zahlen.

```
   3 4 MIN . <CR>  3  OK
 9 5 6 MAX . <CR>  6  OK
```

(Es werden nur die beiden Zahlen verglichen, die sich oben auf dem Stapel befinden.)

ABS --> hinterläßt den Absolutwert einer Zahl.

MINUS --> ändert das Vorzeichen einer Zahl.

```
-4  ABS   . <CR>  4  OK
 3 MINUS  . <CR> -3  OK
-6 MINUS  . <CR>  6  OK
```

'+-' Dieser Befehl benötigt zwei Zahlen auf dem Stack (n1 und n2). Das Ergebnis ist eine neue Zahl (n3), die den Wert von n1 hat und ein Vorzeichen, das sich nach folgendem Schema errechnet:

Vorzeichen von n1	Vorzeichen von n2	neues Vorzeichen von n3
+	+	+
-	-	+
+	-	-
-	+	-

Nach der Berechnung befindet sich nur ein Wert auf dem StacK (n3). n1 und n2 gehen verloren.

```
   -3  4 +- .  <CR>  -3 OK
   -2 -6 +- .  <CR>   2 OK
```

Die genannten drei Befehle verarbeiten und liefern 16-Bit-Zahlen. Für doppelt genaue Worte benutzen Sie die Befehle 'DABS', "DMINUS' und "D+-".

Vergleichsoperationen:

Wie BASIC, so kennt auch FORTH Worte zur logischen Verknüpfung. Es sind dies die Befehle 'AND' (UND), 'OR'(ODER) und 'XOR' (Exclusiv-Oder). Der Sysntax entspricht dem in BASIC (UPN beachten!). Deshalb soll hier nicht näher darauf eingegangen werden. Hinzu kommen folgende Befehle,die zwei Werte des Stapels vergleichen und eine "Boolesche-Zahl" (1=wahr ,0=falsch) auf den Stapel legen.(Die zu vergleichenden Zahlen gehen verloren):

```
<   --> Wahr, wenn n1 kleiner n2
>   --> Wahr, wenn n1 größer  n2
=   --> Wahr, wenn n1 gleich  n2
0<  --> Wahr, wenn TOS negativ
0=  --> Wahr, wenn TOS gleich Null
U<  --> Wahr, wenn u1 kleiner u2
        (32-Bit-Zahl)
    3 4 < . <CR>  1 OK
    4 5 0= . .  <CR> 0 4 OK
```

Zusammenfassung:

-Für Vergleichsoperationen müssen sich die zu testenden Werte auf dem Stack befinden.
-Getestet werden 16-Bit-Zahlen
-Das Ergebnis ist ein Boolescher Wahrheitswert. (1=Wahr/2=falsch)
-Werden die zu vergleichen Zahlen noch benötigt, müssen sie vorher dupliziert werden.
-Alle genannten arithmetischen Befehle sind FORTH-Wörter und k e i n e Rechenzeichen!!! Verändern Sie die Bedeutung des Zeichens '+':

```
: + - ;
```

Wenn Sie jetzt 4 1 + eingeben, wird eine '1' und keine '5' angezeigt.(Bitte das neue '+' mit FORGET gleich wieder löschen)
-FORTH kennt einige sehr nützliche Befehle (z.B.MIN oder MAX) Trotzdem kann man die arithmetischen Möglichkeiten eines FORTH-Systems eher bescheiden nenen, da sie keine Fließkommaarithmetik kennen. Hier arbeitet BASIC seh viel "bequemer".Natürlich kann man sich geeignete Befehle zur Decimaldarstellung definieren.

FORTH - PROGRAMMIERSPRACHE DER DER VIERTEN GENERATION
Elektor-Verl.(ISBN 3-921608-38-4)
Autor: Rüdiger Birkemeyer

Das BASIC-FORTH

von Rüdiger Birkemeyer

"FORTH- und kein Ende" werden einige Leser bei dieser Überschrift sicherlich denken. Aber bei dem abgedruckten Listing handelt es sich um BASIC.
Das Programm erzeugt einen "Forth Interpreter" und erlaubt die Eingabe von FORTH-Syntax.BASIC-FORTH ist für alle Leser, die noch keinen Compiler (Kassette od. Disk) besitzen und erst einmal nur "FORTH-Luft" schnuppern wollen, bevor sie an den Kauf eines Compilers denken.
Natürlich gehen dabei so ziemlich alle Vorteile von FORTH gegenüber BASIC verloren: Die Geschwindigkeit ist deutlich langsamer und es lassen sich keine neuen Befehle und Datenstrukturen erzeugen.
Das Programm will auch keinen Compiler ersetzen, sondern bietet die Möglichkeit, die in meiner Serie gebrachten Beispiele auch am Rechner nachzuvollziehen.
Selbst wenn man eben noch keinen Compiler besitzt.
Das Programm verarbeitet nur INTEGER-Zahlen und berücksichtigt die "post-fix-notation". Neben Stapelmanipulationen lassen sich mit den 58 Befehlen Entscheidungen, Verzweigungen sowie bedingte und unbedingte Wiederholungen nachbilden.

BEISPIELE:

```
1 2 3 . . .                          <CR>  - Legt drei Zahlen auf dem Stapel
                                             ab und gibt sie dann wieder aus.

1 2 .S                               <CR>  - Stapelanzeige

1 2 .S CLEAR .S                      <CR>  - CLEAR löscht den Stapelinhalt

1 2 SWAP DUP .S                      <CR>  - Stapelmanipulationen

7 EMIT od. 65 EMIT                   <CR>  - Ausgabe eines ASCII-Code's

." DIES IST FORTH "                  <CR>  - Textausgabe

10 0 DO I . LOOP                     <CR>  - Schleife

0 BEGIN 42 EMIT 1 + DUP
10 = UNTIL                           <CR>  - Schleife

3 4 = IF PING ELSE ZAP THEN          <CR>  - IF ... THEN ... ELSE

PON / POFF                           <CR>  - Drucker an / aus
```

Fallen Ihnen Weitere Inplementationen ein?

Ist die Einbindung der Befehle (fetch),!(store),KEY od. MOVE möglich?

Gibt es einen Algorithmus, mit dem man die Definitionsworte ":" sowie CONSTANT und VARIABLE programmieren kann?

Schreiben Sie uns Ihre Lösungen!!!

```
10 REM ************************
20 REM ***    BASIC - [illegible]   ***
30 REM *** BIRKEMEYER [illegible] ***
40 REM ************************
50 GOSUB 1100
60 M=0:N=0
70 K=1
80 PRINT : INPUT F$ : F$=F$+" "
90 L1=0
100 L(K)=L1:LO(K)=LEN(F$):L1=LO(K)
110 IF N<0 THEN 260
120 L(K) = L(K)+1
130 IF L(K) > LO(K) THEN 230
140 B$=MID$(F$,L(K),1)
150 IF L(K) > LO(K) THEN 230
160 IF B$=" " THEN 120
170 A$=B$
180 L(K) = L(K)+1
190 B$=MID$(F$,L(K),1)
200 IF B$=" " THEN 270
210 A$=A$+B$:GOTO180
220 GOTO270
230 IFK<2 THEN 70
240 K=K-1
250 F$=MID$(F$,1,LO(K)):L1=LO(K):GOTO120
260 PRINT"EMPTY-STACK " : GOTO 60
270 FOR Z=1 TO 58 : IF A$=W$(Z) THEN290
280 NEXT
290 ON Z GOTO 430,440,450,460,470,480,490,500,520,530,540,550
300 ON Z-12 GOTO 560,570,580,590,600,610,620,630,640,650,660
310 ON Z-23 GOTO 670,680,690,700,710,720,730,740,750,760,770
320 ON Z-34 GOTO 780,790,810,860,870,880,890,920,930,940,950
330 ON Z-45 GOTO 960,970,980,990,1000,1010,1030,1040,1050,1060,1070
340 ON Z-56 GOTO 1080,1090
350 NUM=1
360 FOR I=1 TO LEN(A$)
370 IF MID$(A$,I,1) <"0" OR MID$(A$,I,1) >"9" THEN NUM=0
380 IF I=1 AND MID$(A$,1,1) ="-" THEN NUM=1
390 NEXT I
400 IF NUM=0 THEN PRINT A$,"NOT DEFINED":GOTO 60
410 N=N+1:S(N)=VAL(A$):GOTO110
420 REM DICTIONARY
430 B$="DUP * ":F$=F$+B$:K=K+1:GOTO100' QUAD
440 B$="DO PI 10 / I * SIN . LOOP " :F$=F$+B$:K=K+1:GOTO100' TEST
450 N=N-1:S(N)=S(N)+S(N+1):GOTO110' +
460 N=N-1:S(N)=S(N)-S(N+1):GOTO110' -
470 N=N-1:S(N)=S(N)*S(N+1):GOTO110' *
480 N=N-1:S(N)=INT(S(N)/S(N+1)):GOTO110' /
490 N=N-1:S(N)=S(N)-(S(N+1)*INT(S(N)/S(N+1))):GOTO110' MOD
500 N=N-1:A=S(N)                                                    ' /MOD
510 S(N+1)=INT(A/S(N+1)):S(N)=A-(S(N+1)*INT(A/S(N+1))):N=N+1:GOTO110
520 S(N)=ABS(S(N)):GOTO110' ABS
530 S(N)=INT(S(N)):GOTO110' INT
540 S(N)=SQR(S(N)):GOTO110' SQR
550 S(N)=EXP(S(N)):GOTO110' EXP
560 S(N)=LOG(S(N)):GOTO110' LOG
570 S(N)=RND(-N):GOTO110' RND
580 S(N)=S(N)^S(N+1):GOTO110' ^
590 S(N)=SIN(S(N)):GOTO110' SIN
600 S(N)=COS(S(N)):GOTO110' COS
610 S(N)=TAN(S(N)):GOTO110' TAN
620 S(N)=ATN(S(N)):GOTO110' ATN
```

```
630 S(N)=SGN(S(N)):GOTO110' SGN
640 IFN<1THEN650ELSEFORI=1TON:PRINTS(N-I+1):NEXT:GOTO110' .S
650 IFN<1THENPRINT"EMPTY STACK?":GOTO60:ELSEPRINTS(N);:N=N-1:GOTO110' .
660 N=N+1:S(N)=3.14159:GOTO110' PI
670 N=N+1:S(N)=0:GOTO110' 0
680 N=N-1:IFS(N)=S(N+1)THENS(N)=1:GOTO110ELSES(N)=0:GOTO110' =
690 N=N-1:IFS(N)>S(N+1)THENS(N)=1:GOTO110ELSES(N)=0:GOTO110' >
700 N=N-1:IFS(N)<S(N+1)THENS(N)=1:GOTO110ELSES(N)=0:GOTO110' <
710 N=N+1:S(N)=S(N-1):GOTO110' DUP
720 N=N-1:GOTO110' DROP
730 S(N+1)=S(N-1):S(N-1)=S(N):S(N)=S(N+1):GOTO110' SWAP
740 N=N+1:S(N)=S(N-2):GOTO110' OVER
750 M=M+1:R(M)=S(N):N=N-1:GOTO110' >R
760 N=N+1:S(N)=R(M):M=M-1:GOTO110' R>
770 N=N+1:S(N)=R(M):GOTO110' I
780 M=M+1:R(M)=L(K):M=M+1:R(M)=S(N-1):M=M+1:R(M)=S(N):N=N-2:GOTO110' DO
790 R(M)=R(M)+1:IFR(M-1)>R(M)THENL(K)=R(M-2)ELSEM=M-3                ' LOOP
800 GOTO110
810 N=N-1:IFS(N+1)THEN110' IF
820 FOR I=L(K) TO (LO(K)) : B$=MID$(F$,I,4)
830 IFB$="ELSE"THENL(K)=I+4:GOTO110
840 IFB$="THEN"THENL(K)=I+4:GOTO110
850 NEXT:PRINT" IF? ":GOTO60
860 GOTO820' ELSE
870 GOTO110' THEN
880 M=M+1:R(M)=L(K):GOTO110' BEGIN
890 N=N-1                                                          ' UNTIL
900 IFS(N+1)THENM=M-1:GOTO110
910 L(K)=R(M):GOTO110
920 CLS:END                                                        ' ENDE
930 PING:GOTO110' PING
940 ZAP:GOTO110' ZAP
950 SHOOT:GOTO110' SHOOT
960 EXPLODE:GOTO110' EXPLODE
970 PRINT:GOTO110' CR
980 PRINTCHR$(32);:N=N-1:GOTO110' SPACE
990 PRINTSPC(S(N));:N=N-1:GOTO110' SPACES
1000 PRINTCHR$(S(N));:N=N-1:GOTO110' EMIT
1010 I=0:REPEAT:I=I+1:A=ASC(MID$(F$,L(K)+I,1)):PRINTCHR$(A);        ' ."
1020 UNTILASC(MID$(F$,L(K)+I+1,1))=34:L(K)=L(K)+I+1:GOTO110' "
1030 INK(S(N)):GOTO110' INK
1040 PAPER(S(N)):GOTO110' PAPER
1050 WAIT(S(N)):GOTO110' WAIT
1060 CLS : GOTO 110' CLS
1070 N=0 : GOTO 110' CLEAR
1080 POKE#2F1,#80 : GOTO 110' PON
1090 POKE#2F1,0   : GOTO 110' POFF
1100 REM   ***  INITIALISIERUNG  ***
1110 DIM S(100),R(100),L(100),LO(100),F$(800),B$(100),W$(58)
1120 CLS:PRINT:PRINT"BASIC - FORTH"
1130 FOR Z=1 TO 58 : READ W$(Z) : NEXT
1140 DATA QUAD,TEST,+,-,*,/,MOD,/MOD,ABS,INT,SQR,EXP,LOG,RND,^,SIN,COS,TAN
1150 DATA ATN,SGN,.S,.,PI,0,=,>,<,DUP,DROP,SWAP,OVER,>R,R>,I,DO,LOOP
1160 DATA IF,ELSE,THEN,BEGIN,UNTIL,STOP,PING,ZAP,SHOOT,EXPLODE,CR
1170 DATA SPACE,SPACES,EMIT,.",INK,PAPER,WAIT,CLS,CLEAR,PON,POFF
1180 RETURN
```

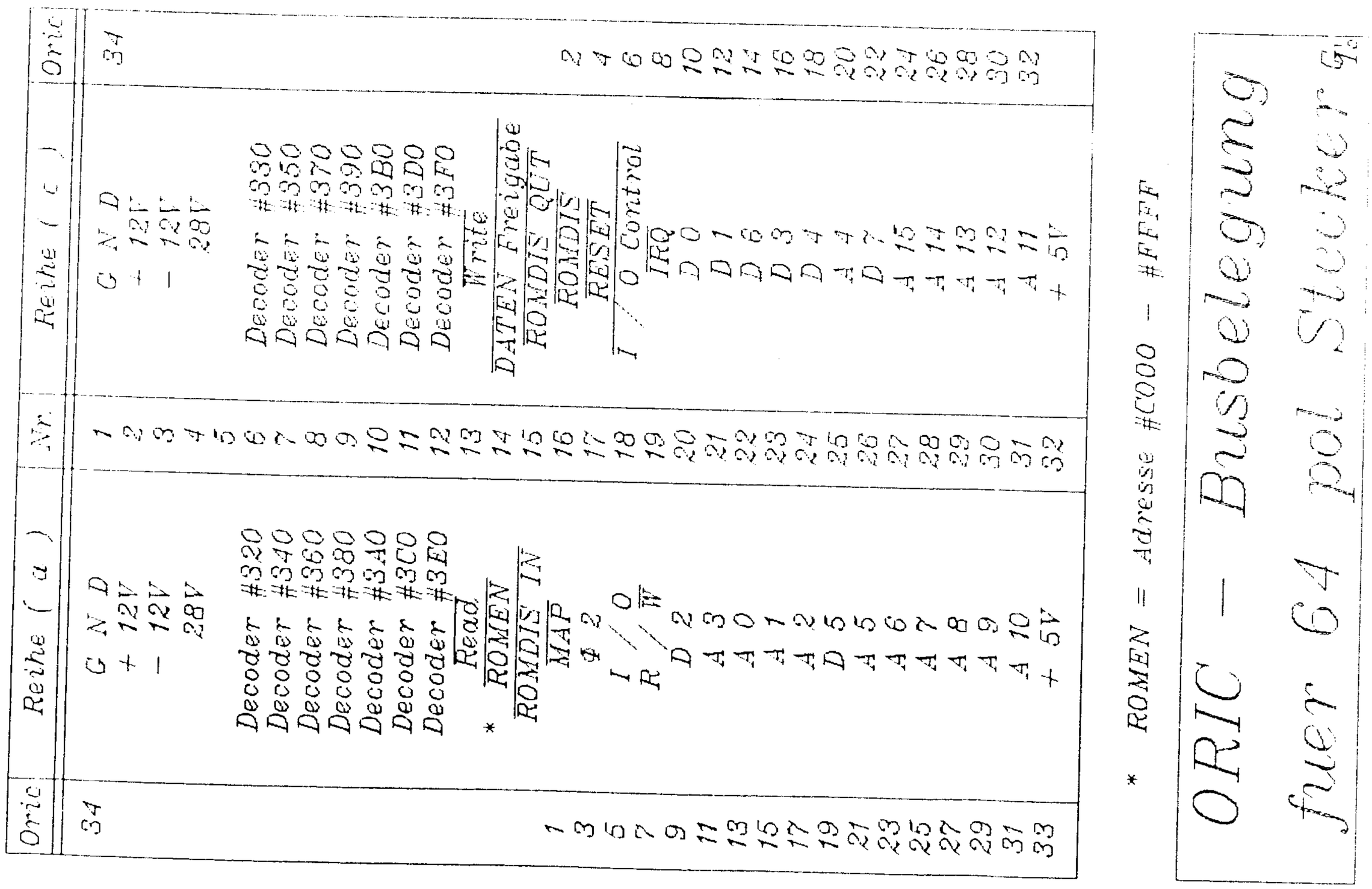
STANDART-BUS ORIC-1/ATMOS von Gerd Tetzlaf

Viele Köche verderben den Brei und viele unterschiedliche Schnittstellen sind der Graus eines Computer-Besitzers.
Deshalb möchten wir an dieser Stelle unseren ersten Standart vorstellen.Die unten im Bild gezeigte Busbelegung ist für eine 64pol.Steckeranordnung a/c Bestückung ausgelegt und wird allen kommenden Erweiterungen als Basis dienen. Wir möchten also alle Bastler, die eine Erweiterung für die ORIC's entwickeln möchten, bitten sich an die untere Belegung zu halten.Alle ORIC-User, die keine Ahnung von technischen Dingen haben, werden es Ihnen danken.

Die Pinbelegung 16a/c bis 32a, sind mit dem ORIC-1/ATMOS 34pol.-Bus identisch.
Die GNG Leitung liegt hingegen auf Pin 1a/c

Die Pinbelegung 6a/c bis 15a/c sind Sonderbelegungen, die aus einer Treiberkarte gespeist werden.

Pin 2a/c bis 4a/c sind Spannungsversorgung.
Pin 2a/c und 3a/c für kommende serielle Schnittstelle RS232/V24.
Pin 4a/c für einen EPROM-Brenner.
Pin 5a/c ist frei.



ASSEMBLER-KURS (2) von Ralf Jesse

Hexadezimalzahlen

Um eine einfachere Darstellung der 8-Bit-Muster (Bytes) zu erreichen, teilt man diese in zwei Worte zu je vier Bit auf. Jedes dieser 4-Bit-Worte ermöglicht 16 Kombinationen.
Führt man für diese das Hexadezimalsystem ein, so kann man ein 8-Bit-Wort mit nur zwei Zeichen kodieren. Dies Hat den Vorteil, daß man sich zwei Zeichen besser merken kann als eine Reihe von Nullen und Einsen.
Hexadezimalziffern werden durch die Zeichen "0" bis "9" und "A" bis "F" wiedergegeben. Die folgende Tabelle zeigt die Dezimalzahlen "0" bis "15" und die entsprechenden dualen und hexadezimalen Kodierungen an.
Zur Unterscheidung werden im Folgenden Hex-Zahlen mit "#" gekennzeichnet.

dezimal	dual	hex
0	%0000	#0
1	%0001	#1
2	%0010	#2
3	%0011	#3
4	%0100	#4
5	%0101	#5
6	%0110	#6
7	%0111	#7
8	%1000	#8
9	%1001	#9
10	%1010	#A
11	%1011	#B
12	%1100	#C
13	%1101	#D
14	%1110	#E
15	%1111	#F

Auch das Hexadezimalsystem ist ein Stellenwertsystem. Allerdings gibt es hier 16 verschiedene Ziffern,d.h.die Basis dieses Systems ist die 16.

BEISPIEL:

$$1985 = \#7C1$$
$$= 7*16^2 + C*16^1 + 1*16^0$$
$$= 7*16^2 + 12*16^1 + 1*16^0$$

Das Hexadezimalsystem ist auch unter dem Namen Sedezimal- oder 16er-System bekannt.

Der Übertrag C (carry)

Bei den bisherigen Beispielen kam schon öfters der Begriff Übertrag vor (eingeklammerte Eins).
Der 6502-Prozessor ist ja ein 8-Bit-Prozessor, d. h. daß die zur Informationsdarstellung dienenden Register (Speicher) nur 8 Bit breit sind. Wenn man dort ein Ergebnis speichert, bleiben nur die Bits 0 bis 7 erhalten.Nun gibt es aber Zahlen,die mit 8 Bit nicht mehr darstellbar sind.

BEISPIEL:

```
  128 = %10000000
+ 129 = %10000001
-----------------
=     %(1)00000001 = 257
```

Der Übertrag stellt also ein neuntes Bit dar (Bit 8 des Ergebnisses). Es ist leicht einzusehen, daß dieser Übertrag erkannt werden muß.
Der 6502-Prozessor bietet die Möglichkeit,mit besonderen Befehlen einen eventuellen Übertrag zu erkennen und weiterzuverarbeiten. Diese Befehle lernen wir in einem späteren Kapitel kennen.

Der Überlauf V (overflow)

Ein Überlauf ist etwas ganz anderes als ein Übertrag. Während der Carry dazu dient, größere als 8 Bit darzustellende Zahlen zu verarbeiten,zeigt ein Überlauf an,daß das Vorzeichen des Ergebnisses einer Berechnung im Zweierkomplement "versehentlich" umgekehrt worden ist. Aus diesem Grund bezeichnet man den Überlauf auch als Zweierkomplement-Überlauf.
Zwei Beispiele sollen einen Überlauf verdeutlichen.

BEISPIEL 1:

```
  %01000000 -->   64
+ %01000001 -->   65
----------------------
= %10000001 --> -127
```

Richtig wäre +129 gewesen

BEISPIEL 2:

```
  %11000000 -->  -64
+ %10111111 -->  -65
----------------------
= %01111111 --> +129
```

Richtig wäre -129 gewesen

Mit Hilfe dieser beiden Beispiele kann man den Überlauf vollständig definieren.
Das Überlaufbit, eine speziell zu dessen Erkennung dienende Einrichtung im 'Prozessorstatuswort' des 6502 (die verschiedenen Regisster des 6502 werden im nächsten Kapitel erklärt), wird zu "1" gesetzt, wenn
1.) ein interner Übertrag von Bit 6 nach Bit 7, ohne externen Übertrag
2.) ein externer Übertrag ohne internen Übertrag von Bit 6 nach Bit 7
stattfindet.
In einem späteren Kapitel werden wir sehen, wie man Carry- und Overflow-Flaggen zur effektiven Programmierung einsetzt.

ÜBUNG 1:

Überprüfen Sie,ob die folgenden Additionen zum richtigen Ergebnis führen.Geben Sie die entsprechenden Dezimalzahlen in den Klammern an.Prüfen Sie,welchen Wert die V- bzw. C-Flagge nach der Addition haben und kreuzen Sie an,ob das Ergebnis richtig oder falsch ist.

```
  %10111111 (    )
+ %11000001 (    )
--------------------
=             (    )   V:   C:     Richtig ()  Falsch ()
```

```
  %00010000 (    )
+ %01000000 (    )
--------------------
=             (    )   V:   C:     Richtig ()  Falsch ()
```

```
   %01111110 (    )
 + %00101010 (    )
 ------------------
 =           (    )   V:   C:     Richtig ()  Falsch ()
```

Festformatdarstellung

Der mit 8 Bit in der Signed-Binary-Form darstellbare Zahlenbereich von -128 bis +127 ist für die meisten in der Praxis zu berechnenden Größen zu klein. Zur Ausweitung des Zahlenbereiches führt man deshalb die zwei-,drei- oder mehrfache Genauigkeit ein. Dabei heißt zweifache Genauigkeit, daß man zur Zahlendarstellung zwei Bytes, gleich 16 Bit, benutzt. Die dreifache Genauigkeit benutzt demnach drei Bytes usw.

BEISPIEL:

```
%00000000 %00000000 -->  "0"
%00000000 %00000001 -->  "1"
%11111111 %11111111 --> "-1"
```

Den Vorteil der Darstellbarkeit größerer Zahlenbereiche mit der mehrfachen Genauigkeit erkauft man sich mit drei grundlegenden Nachteilen.

1.) Arithmetische Operationen (Addieren, Subtrahieren, usw.) werden byteweise durchgeführt. Daraus ergibt sich eine langsamere Berechnung der gestellten Aufgabe.

2.) Man muß sich auf eine bestimmte Anzahl von Bytes festlegen. In der doppelten Genauigkeit wird dann jede Zahl mit 16 Bit dargestellt, auch wenn 8 Bit ausreichen würden.

3.) Wenn man sich auf eine bestimmte Anzahl von Bytes geeinigt hat,liegt diese unverrückbar fest.Führt eine Berechnung zu einem Ergebnis, das mit der festgelegten Anzahl Bits nicht mehr darstellbar ist, gehen automatisch einige verloren. Das Ergebnis wird also verfälscht.

ÜBUNG 2:

Welchen Zahlenbereich kann man mit der dreifachen Genauigkeit in der Signed-Binary-Form darstellen?

Die BCD-Darstellung

Zur Vereinfachung der Darstellung von Dualzahlen haben wir das Hexadezimalsystem kennengelernt.
Es gibt aber außer diesem und dem Dezimalsystem noch weitere Zahlensysteme. Eins davon ist das BCD-System.
BCD ist eine Abkürzung aus dem Englischen (binary coded decimal) und heißt auf deutsch "binär kodierte Dezimalziffern".
Diese Darstellungsart funktioniert folgendermaßen:
Jede der darzustellenden Dezimalziffern wird mit vier Bits binär kodiert. Mit vier Bits kann man 2 hoch 4 gleich 16 verschiedene Kombinationen erreichen, was zur Kodierung der zehn Dezimalziffern ausreichend ist. Drei Bits würden nicht alle Dezimalziffern erfassen, da man hiermit nur 2 hoch 3 gleich 8 verschiedene Kombinationen erhält.
Von den mit vier Bits darstellbaren 16 Kombinationen werden aber nur zehn benötigt. Die letzten sechs werden daher nicht benutzt.
Die ersten zehn werden Nutztetraden und die sechs nicht benutzten Pseudotetraden genannt.

Die nachfolgende Tabelle zeigt die möglichen Kombinationen.

BCD-Symbol	Kode	
0	%0000	
1	%0001	
2	%0010	
3	%0011	
4	%0100	
5	%0101	Nutztetraden (NT)
6	%0110	
7	%0111	
8	%1000	
9	%1001	
10	%1010	
11	%1011	
12	%1100	Pseudotetraden (PT)
13	%1101	
14	%1110	
15	%1111	

BEISPIEL:

```
                 2     3                      9     9
23 --> %0010 %0011        99 --> %1001 %100/1
```

ÜBUNG 3:
Wie lautet die BCD-Darstellung von "29" und "91"?

ÜBUNG 4:
Ist "%1010 %0000" eine gültige BCD-Darstellung?
Begründen Sie Ihr Ergebnis.

Der 6502-Prozessor kann BCD-kodierte Zahlen direkt verarbeiten. Dies ist ein großer Vorteil gegenüber anderen Prozessoren. Dieser Vorteil wird aber durch einige kleinere Nachteile geschmälert. Der Vorteil von BCD-Zahlen liegt darin, daß Sie vollkommen richtige Ergebnisse liefern. Ein Nachteil ist, daß zu ihrer Darstellung sehr viel Speicherplatz benötigt wird. Es muß ja jede einzelne Dezimalziffer getrennt kodiert werden.
Ein weiterer Nachteil ist, daß Rechenoperationen im BCD-Modus relativ langsam ablaufen. Dies ist in der industriellen Überwachung von technischen Prozessen natürlich nicht sinnvoll.
Aus diesem Grund findet diese Kodierung ihre Anwendung fast ausschließlich im kaufmännischen Bereich.

ÜBUNG 5:
Wieviel Bit benötigt man zur Darstellung der Zahl 9999 in der

a) BCD-Schreibweise
b) Zweierkomplementdarstellung ?

Gleitkommadarstellung

Es ist nun kein Problem mehr, positive und negative ganze Zahlen, die auch sehr groß sein können, darzustellen. Wie kann man aber gebrochene Dezimalzahlen wiedergeben? Hierbei muß berücksichtigt werden, daß man sich vor dem Programmieren auf eine feste Anzahl von Bits festlegen muß.So verschenkt man z.B.bei der Zahl 0.00023 drei Stellen hinter dem Komma.
Die Mathematik liefert uns aber mit der sogenannten Gleitkommaschreibweise eine Möglichkeit, diese Speicherplatz-Verschwendung zu umgehen.
Allgemein kann man sagen, daß sich jede Zahl 'n', egal ob positiv oder negativ, folgendermaßen darstellen läßt.

$n = m\ 10^E$ wobei $-1 < m < +1$ ist.

m ist die Mantisse, E ist der Exponent.

BEISPIELE:

$543.967 = 0.543967\ 10^3$
$22387.3 = 0.223873\ 10^5$
$0.00478 = 0.478\ 10^{-2}$

ÜBUNG 6:
Stellen Sie bitte folgende Zahlen im Gleitkommaformat dar:

1.346 =
233.378 =
0.99 =

ÜBUNG 7
Nennen Sie die größte und kleinste Zahl, die im Gleitkommaformat mit zwei Exponentenstellen darzustellen ist.

Darstellung alphanumerischer Zeichen

Für die Verschlüsselung alphanumerischer Zeichen haben sich 2 Kodes durchgesetzt. Der erste ist der von IBM entwickelte EBCDIC-Kode; dieser ist allerdings außerhalb des IBM-Bereichs kaum gebräuchlich.
In den meisten Fällen wird der ASCII-Kode verwendet. ASCII steht dabei für "American Standard Code for Information Interchange". In Deutsch heißt das soviel wie "Amerikanischer Standardkode zum Informationsaustausch". Der ASCII-Kode wird auch ISO-7-Bit-Kode genannt. Er umfaßt 26 Buchstaben in Groß- und Kleinschrift, 10 Ziffern und ca. 20 Sonderzeichen. Mit 7 Bit hat man 128 verschiedene Kombinationen zur Auswahl, was für unseren Zweck vollkommen ausreichend ist. Ein zusätzliches achtes Bit dient, falls es überhaupt benutzt wird, dazu, Fehler, die bei der Übertragung der Daten aufgetreten sind, zu erkennen. Hierzu bieten sich zwei Methoden an. Das Paritätsbit 8 wird so gesetzt, daß sich

a) eine gerade Anzahl von Einsen ergibt - Even Parity Check (EPC)
b) eine ungerade Anzahl Einsen ergibt - Odd Parity Check (OPC).

BEISPIEL
Zu ermitteln ist das Paritätsbit für gerade Parität für

%0010111 --> EPC = 0
%0000000 --> EPC = 0
%1010100 --> EPC = 1

ÜBUNG 8:
Die Buchstaben A, B, C sind im ASCII-Kode folgendermaßen kodiert:

A = 65 = %100000001
B = 66 = %100000010
C = 67 = %100000011

Ermitteln Sie das EPC und das OPC.

Lösungen zu den Übungsaufgaben:

Übung 1:

```
   10111111     (-65)
 + 11000001   + (-63)
------------  -------
(1)10000000    (-128)      Richtig   V:0   C:1

   00010000      (16)
 + 01000000   +  (64)
------------  -------
   01010000       80       Richtig   V:0   C:0

   01111110     (128)
 + 00101010   +  (42)
 ----------   -------
   10101000     (-88)      Richtig   V:1   C:0
```

Übung 2: 3-fache Genauigkeit
$-2^{23}...+2^{23} - 1 = -8388608... + 8388607$

Übung 3: 29 = %0010 %1001
91 = %1001 0001

Übung 4: %1010 %0000 ist keine gültige BCD-Barstellung, weil die ersten 4 Bit %1010" zu den Pseudotetraden gehören.

Übung 5: 9999
a) BCD = %1001 %1001 %1001 %1001 = 16 Bit
b) Zweierkomplement = %1001 %1100 %0011 %11 = 14 Bit

Übung 6: $1.346 = 0.1346\ 10^{1}$
$233.378 = 0233378\ 10^{3}$
$0.99 = 0.99\ 10^{0}$

Übung 7: Größtzahl = +99999999 +99
= $+99999999\ 10^{99}$

Kleinstzahl = -0.0000001 -99
= $+0.0000001\ 10^{-99}$

Übung 8:
A = 65 = %1000001 --> EPC = 0 OPC = 1
B = 66 = %1000010 --> EPC = 0 OPC = 1
C = 67 = %1000011 --> EPC = 1 OPC = 0

L E S E R - F O R U M

Seit langem habe ich keine bessere Mitteilung erhalten, wie Ihre über die ATMOSPHÄRE. Da ich mit dem ORIC-1, vom Gerät und der Programmierung her, sehr zufrieden bin, nehme ich die Gelegenheit besonders gerne an.
Ich wünsche mir und allen ORIC-Leuten sowie der Redaktion der ATMOSPHÄRE möglichst langen Bestand.

H. Klosek

Wo liegt beim ORIC-1 das Unterprogramm um Ein-Byte-Ausdrücke aus dem BASIC-Text zu holen? (Beim ATMOS #D8C5)

F. Schmidt

Das Unterprogramm liegt beim ORIC-1 ab der Adresse #D80A.

E.O.

STELLUNGNAHME ZUM VORWORT ATMOSPHÄRE 2

... Ersteinmal wollen wir klarstellen, daß die Prg. nicht sieben mal kopiert werden, sondern daß die Prg. höchstens 2mal kopiert werden und sonst das Original rumgereicht wird.....
.... Ferner erleiden Sie dadurch keinerlei Nachteile, sondern wir nur einen Vorteil, da es sich keiner von uns leisten kann, Ihre gesamte Programmpalette aufzukaufen. Und schließlich ist es ja egal, ob man für 60 DM zwei gleiche oder zwei verschiedene Prg. kauft. Die 60 DM repräsentieren in diesem Fall unser Erspartes von drei Monaten.
Außerdem gibt es Ausgaben, mit denen der "ORIC-Haufen" nicht belastet werden kann (z.B. Floppykauf). Das ist dann Sache der einzelnen 'Oricer' und die müssen ihr Geld auch irgendwie zusammen kriegen. Mit der ATMOSPHÄRE verhält es sich wie oben genannt, nur das diese nie kopiert wird. Allerdings müssen wir Ihnen hier einige Zugeständnisse machen. Betrachten Sie unsere Ansichten als freundlich gemeinte Rechtfertigung....
....Als kleines "Trostpflaster" haben wir einen kleinen Tip beigelegt, mit dem man den "ORICMON" von 'PSS' auf dem ATMOS lauffähig bekommt. Leider funktioniert die H-Funktion nicht richtig, aber die ist ja nicht ganz so wichtig.
Noch was: in was für einer Form wünschen Sie sich Softwareberichte und was soll getestet werden (Verhältnis: Spiel/Anwender)?
Auf weiterhin gute Zusammenarbeit...

ORIC-Haufen Heessen

Hier das Prg. für 'ATMOSMON':
Nach dem laden des 'ORICMON' ohne Autostart...RUN und CALL#A800.

```
10 POKE#A800,76:DOKE#A801,#B166:POKE
   #A814,75:POKE#A81A,76:DOKE#A802,
   #EE22
20 DOKE#AEC4,#EB78:POKE#AEDC,75:POKE
   #AEE1,76:DOKE#AEE6,#CCCE:POKE#AEEF,75
30 POKE#AEF4,76:DOKE#AF0B,#C4A8:DOKE
   #AF33,#CCB0:DOKE#AF9F,#E76A
40 DOKE#AFA8,#F5C1:DOKE#AFAB,#E93D:
   DOKE#AFB4,#CCD9
50 POKE#B166,169:POKE#B167,64:POKE
   #B168,141:DOKE#B169,#24A:POKE#B16B,32
60 DOKE#B16C,#B093:POKE#B16E,76:
   DOKE#B16F,#A803
```

Zum Thema 'Kopieren' möchte ich an dieser Stelle nicht viel sagen. Ich hoffe vielmehr auf weitere Stellungnahmen von unseren Lesern. Vielleicht finden wir dadurch einen Weg, der beide Seiten befriedigt. Wir sind für gute Vorschläge immer zu haben.
Ich möchte dazu unsere und dadurch auch indirekt Ihre Situation veranschaulichen:
Um Ihnen die Software zu einem einigermaßen günstigen Preis anbieten zu können, müssen wir bei den Herstellern 50 Kassetten pro Titel bestellen und cash (bar) bezahlen! (nebenbei bemerkt nicht alle Händler sind Milionäre) Aus diesem Grund haben wir auch noch Schwierigkeiten mit Frankreich, dort müßten wir 100 Kassetten pro Tilel bestellen.
An dieser Stelle beißt sich die Katze in den Schwanz. Ohne Käufer keine große Software-Auswahl und ohne große Auswahl laufen einem die letzten Käufer weg. Letztendlich hat dies auch unserem 'Altkonkurenten' C64 das Genick gebrochen. Lassen Sie sich vom einem C64-User aus seiner '1000 Stück'-Sammlung mal die Originale zeigen.

Software-Berichte sollten zunächst einmal Titeln gelten, die noch zu kaufen sind. Dann sollte berücksichtigt werden, daß Geschmäcker verschieden sind. Also nicht in der Form "...alles sche.."oder "alles prima". Es sollte auf jeden Fall eine sachliche Kurzbeschreibung des Programms dabei sein.
Danach sollten folgende Kriterien mit

'+' = gut
'-' = schlecht
'+/-' = mehr gut als schlecht
'-/+' = mehr schlecht als gut

beurteilt werden.

Spiele:	Grafik
	Sound
	Spielmotivation
	Bedienung des Spiels
	Preis/Leistung
	GESAMTURTEIL
Anwendungen:	Darstellung
	Bedienung
	Komfort
	Preis/Leistung
	GESAMTURTEIL

K.D.B.

.....Als erstes möchte ich Ihnen und Ihren Kollegen zu der Idee gratulieren eine Zeitschrift für die ORIC's auf den Markt zu bringen. Ich wünsche Ihnen daß sich die Sache auch lohnt und nicht alle so denken wie ich. Mir tut es jetzt schon leid Ihre Zeitung abonniert zu haben. Wie vielen mag es ebenso ergehen, die sich etwas leichtes für Anfänger gedacht haben und bekommen jetzt Chinesisch für Ihr Geld?
Können Sie nicht mal eine kleine Anfrage drucken, wie sich Ihre Leser zusammen setzen? Machen Sie meinetwegen kleine Testfragen, die beantwortet werden müssen und an hand dessen Sie Ihre Beiträge nach Schwierigkeitsgrad einteilen können. Dann haben auch die ORIC-Besitzer, die seit Jahren noch nicht aus dem 1. Schuljahr heraus sind, Freude an Ihren Beiträgen und können etwas lernen. Wenn man zu einem Computerhändler geht, hört man das gleiche wie nach Art Ihrer Zeitung.
Aber gerade die sollte doch für ALLE da sein und für jeden etwas bringen. Ich z.B. kann mit keiner Zeile etwas anfangen. Warum soll ich Dual in Dezimal umwandeln? Wofür benötige ich das? Ich habe Nr.1 nicht genommen, weis also nicht, was dabei herauskommt, wenn ich Seite 11 Nr.2 abtippe. Ich bin schon so oft hereingefallen, daß zum Schluß, nach stundenlanger "Arbeit", nur ein leerer Bildschirm mit einigen Figuren erschienen, die mit einem Knall (shoot) wieder verschwunden waren....

H.H.M

In Heft Nr.2/Seite 11, steht oben ganz deutlich: 'FUER ORIC-1'. Nach unseren Unterlagen besitzen Sie einen ORIC-ATMOS. Sie hätten sich viel Arbeit ersparen können, wenn Sie etwas genauer gelesen hätten. Alles weitere siehe Vorwort in diesem Heft.

K.D.B.

KDB-COMPUTER-VERSAND Preisliste gültig ab 01.11.86
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
OR070	O	JOGGER	SEVERN	24.00 DM
OR073	OA	GHOSTMAN	SEVERN	24.00 DM
OR080	OA	ULTIMA ZONE	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	OA	SUPEREXTENDED-BASIC V2.0/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
OR158	O	FORTH V3 (Kassette f. ORIC-1)		39.00 DM *
OR159	A	FORTH V3 (Kassette f. ATMOS)		39.00 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	**********	
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