

①

24. 1. 87

# LOOP

Uwe Koch  
 Frankenstraße 25  
 5880 LÜDENSCHIED  
 Tel. (023 51) 2 61 92

# 12

2. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter

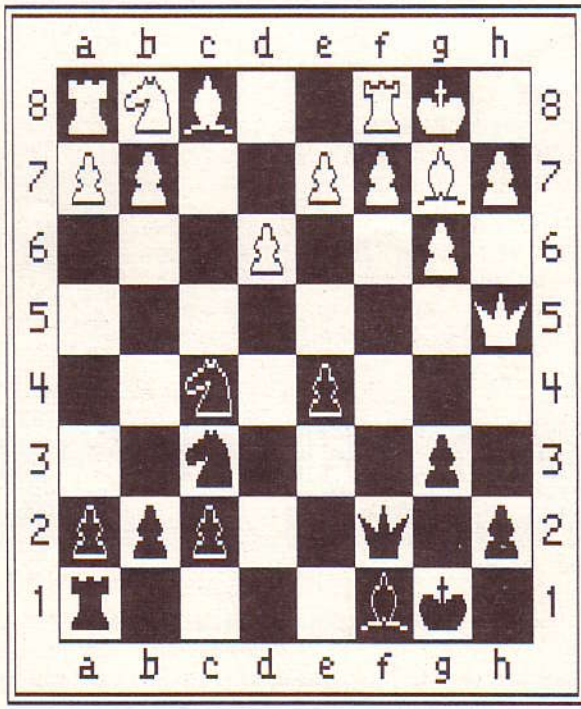
DM 3,-

## Schach (mit) dem NDR-Computer

SCHACH 2.6 - SCHACH 2.6

- Level 1.4
- 1. Be2-e4 Bd7-d6
- 2. Bd2-d4 Sg8-f6
- 3. Sb1-c3 Bg7-g6
- 4. Bf2-f4 Lf8-g7
- 5. Sg1-f3 0-0
- 6. Lf1-e2 Bc7-c5
- 7. Bd4: c5 Dd8-a5
- 8. 0-0 Da5: c5+
- 9. Tf1-f2 Sf6-g4
- 10. Le1-e3 Dc5: e3
- 11. Dd1-e1 Sg4: f2
- 12. De1: f2 Dc3: f4
- 13. Bg2-g3 Df4-g4
- 14. Sf3-e5 Dg4-h3
- 15. Le2-f1 Dh3-h5
- 16. Se5-c4

## SCHACH 2.6



den NDR-Computer mit den CPUs 680xx. Das Programm arbeitet nach der „Gewaltmethode“, d.h., es rechnet bis zu einer vorgegebenen Rechentiefe voraus. Diese Rechentiefe kann mit Angabe der Spielstufe eingestellt werden.

Die Ausgabe erfolgt graphisch auf dem Bildschirm und als Text, der auch auf einen Drucker umgeleitet werden kann. Des weiteren ist eine etwa 1600 Züge umfassende Eröffnungsbibliothek vorhanden, aus der der Spielzug des Rechners in der Eröffnungsphase zufällig ausgewählt wird.

Man kann Partien auf Diskette speichern und wieder laden, mit einem Stellungseditor beliebige Stellungen eingeben,

weiter auf Seite 12/2

### Aus dem Inhalt

**Titelgeschichte**  
 Schach (mit) dem NDR-Computer . . . . . 12/1

**Jetzt lieferbar**  
 ACRTC und CPU68020 nun auch als Leiterplatte . . . 12/4  
 CAD mit dem NDR-Computer und mc-Computer. . . 12/3  
 Disketteneditor für die 68000-Serie mit JADOS . . . 12/4  
 Update Modula 2 Compiler V 1.10 . . . . . 12/4  
 Video-Verwaltung . . . . . 12/4

**Für Einsteiger bis Z80/SBC**  
 Welchen Code hat die Taste? . . . . . 12/5

**Z80-Vollausbau bis ZEAT**  
 Arithmetic-Routinen in UPN-Notierung 32 bit . . . . 12/7  
 Verschlusszeiten elektronisch gemessen . . . . . 12/6

**Z80 bis CP/M 2.2**  
 Autostart . . . . . 12/9  
 Flimmerfreie Bilder d. Seitenumschaltung im FLOMON 12/8

**PASCAL und BASIC**  
 Erweiterungen zur Disc-Doc-Tool-Diskette . . . . . 12/10  
 Tips und Tricks bei HEBAS, Nr. 6 . . . . . 12/11

**Für 68000-Einsteiger**  
 Neues Schreibgefühl mit der GDP . . . . . 12/11  
 Nie mehr auf den Drucker warten . . . . . 12/13  
 Verbesserter und schnellerer Figur-Befehl mit X- und Y-Vergr. . . . . 12/13

**JOGIDOS, JADOS, RL-BASIC**  
 RL-Basic liest ASCII-Dateien. . . . . 12/17

**CP/M68K, C und MODULA**  
 Copydisk 68 . . . . . 12/20  
 Suchen im Editor . . . . . 12/19

**Der mc-CP/M-Computer**  
 TERM1-Programm-Quelle nun auf Diskette . . . . . 12/22

**Tips und Tricks**  
 JADOS und die Sound-Baugruppe. . . . . 12/23  
 Kann NDR-Computer steuerlich abgesetzt werden? 12/23  
 NDR 68000 ohne Waits - was noch zu beachten ist 12/22

Endlich: ein professionelles Schachprogramm für den NDR-Computer unter JADOS (Ausbau CPU 68008 bis 68020). Preis: DM 79,-!

**Die Möglichkeiten in Kurzform:**  
 Spiel Mensch gegen Computer  
 Spiel Computer gegen sich selbst  
 Eröffnungsbibliothek selbst erweiterbar(!)

Speichern und Laden von Partien und Zwischenständen auf Diskette  
 Stellungseditor  
 Anzeige graphisch (siehe Hardcopy)  
 Programm gibt Zugvorschläge  
 Rücknahme beliebig vieler Züge möglich  
 Variable Spielstufen  
 Läuft auf allen 680xx-CPU's unter JADOS  
 SCHACH 2.6 ist ein Schachprogramm für

sich Zugvorschläge geben lassen, beliebig viele Züge zurücknehmen oder den Rechner gegen sich selbst spielen lassen. Außerdem wird die Uhrenkarte, falls vorhanden, verwendet, um die verbrauchte Zeit anzuzeigen.

#### Speicheraufteilung

SCHACH 2.6 verwendet etwa 6 KByte RAM unterhalb des Stacks und hinter dem Programmtext so viel RAM, wie für Eröffnungsbibliothek und Partieprotokolle benötigt werden.

(Protokoll ca. 1 – 2 KByte, EROEFF.LIB ca. 4 KByte.)

Es läuft mit Grundprogrammversion ab 4.3 und JADOS ab Version 2.0. SCHACH 2.6 ist relocativ.

#### Die Bedienung des Programmes.

Beim Aufruf von JADOS aus kann ein Parameter eingegeben werden, nämlich der Name der Eröffnungsbibliothek. Das ermöglicht, zwischen mehreren Bibliotheken zu wählen. Durch Angabe von '-N' wird keine Eröffnungsbibliothek geladen; wird kein Parameter angegeben, so wird die Datei EROEFF.LIB als Eröffnungsbibliothek geladen.

Nach dem Programmstart von JADOS aus oder mit der Bibliotheksfunktion des Grundprogrammes erscheint ein Startmenue, in das der Name des Spielers, die gewünschte Farbe und die Spielstärke eingegeben werden muß. Außerdem kann optional eine Partie von der Diskette geladen werden (Datei vom Typ .PAR, die von SCHACH 2.6 erzeugt werden kann) oder mit Hilfe des Stellungseditors eine beliebige Stellung eingegeben werden.

Je nach gewählter Option erscheint nun das Lademenue oder das Schachbrett.

#### G = Grundstellung

Bei Angabe dieser Option (per Default voreingestellt) wird nach Verlassen des Startmenues durch 'Escape' das Schachbrett angezeigt und ein Eingabefeld erscheint. Durch Eingabe von 'H', gefolgt von CR, erhält man eine Übersicht über die möglichen Kommandos.

Unter dem Eingabefeld ist eine Statuszeile zu sehen. Ganz rechts steht hier anfangs „Analyse“. In diesem Fall können die Züge beider Parteien über die Tastatur eingegeben werden. Um zu spielen, wird durch Eingabe von 'S', gefolgt von CR, der Modus auf „Spiel“ umgestellt. Nun berechnet das Programm immer, wenn es am Zug ist, seinen Zug und führt ihn gleich aus.

#### E = Stellungseditor

Wird diese Option angegeben, so wird ebenfalls das Schachbrett angezeigt, mit einem blinkenden Cursor in der linken oberen Ecke. Durch Drücken der Taste 'H' erhält man eine Übersicht über mögliche Eingaben. Nach Betätigen der „Escape“-Taste geht es weiter wie bei der Option G = Grundstellung.

#### L = Partie laden

Gibt man diese Option an, gelangt man in das Lademenue. Alle Steuerfunktionen gleichen denen im Startmenue. Auch hier gibt es ein Optionefeld; gibt man hier 'E' an, gelangt man zurück in das Startmenue, mit 'D' erhält man ein Verzeichnis aller Dateien vom Typ .PAR auf dem aktuellen Laufwerk, mit 'L' wird die unter Dateiname angegebene Datei geladen.

Dabei wird überprüft, ob eine spezielle Kennung vorhanden ist, die die Datei als ein gültiges Partieprotokoll ausweist. Ist diese Kennung vorhanden, so geht es weiter wie bei der Option G = Grundstellung.

Der Partiname wird gespeichert und bei einer eventuellen späteren Partieabspeicherung als Default angegeben.

#### Kommandoeingabe

In das Eingabefeld wird der Zug des Spielers eingegeben. Dieser Zug wird nur ausgeführt, wenn er zulässig ist. Beispiele für Zueingabe:

Sf1-g3 oder g1-f3 oder g1f3

Die Rochade wird durch Angabe des Königszuges angegeben oder durch 0-0 (kurze Rochade) bzw. 0-0-0 (lange Rochade). En Passant muß nicht besonders gekennzeichnet werden, die Angabe des Bauernzuges reicht aus.

#### Die Spielstufen von SCHACH 2.6

Die Spielstufe, wie sie im Startmenue und nach Eingabe des L-Kommandos verlangt ist, hat die Form n.m.

Das ist so zu verstehen, daß SCHACH 2.6 bis zum (n+1)-Halbzug weit vorausrechnet und dann Schlagzüge und Schachgebote noch m-Halbzüge weiterverfolgt, um Schlagabtäusche möglichst vollständig zu erfassen.

Der Defaultwert von 1.4 ergibt Bedenkzeiten von einigen Sekunden bis zu einer Minute in komplizierteren Stellungen. Spielstufen oberhalb von etwa 3.4 sind außer für Schachprobleme wohl indiskutabel, da hier die Bedenkzeit schon bis zu 40 min. gehen kann.

tabel, da hier die Bedenkzeit schon bis zu 40 min. gehen kann.

Spielstufe	Antwortzeit	Spielstärke
0.2	sofort	findet einzügige Matts
1.2	1...3 sec.	verhindert einzügige Matts
1.4	einige sec.	wie 1.2, Schlagabtäusche werden besser behandelt
2.2	1...2 min.	findet zweizügige Matts
2.4	einige min.	wie 2.2, Schlagabtäusche werden besser behandelt (Turnierstufe)
3.2	bis ½ h	verhindert zweizügige Matts

... usw.

(Die angegebenen Zeiten beziehen sich auf die CPU 68008 mit 8 MHz)

Für Schachprobleme sind die Spielstufen x.2 geeignet, für Spiele die Spielstufen x.4. Selbstverständlich sind auch alle anderen Kombinationen erlaubt; es ist aber zu bedenken, daß die Bedenkzeit bei höheren Spielstufen enorm ansteigt. Das Schachprogramm wurde von unserem Software-Partner Klaus Rumrich in Maintal geschrieben. Unsere Anerkennung für diese Leistung!

Bestell-Nr.	Bezeichnung	DM
10874	Schachprogramm, 5¼" 80 Spuren	79,-
10873	Schachprogramm, 3½" 80 Spuren	79,-
10292	JADOS 5¼" 80 Spuren	140,-
10690	JADOS 3½" 80 Spuren	140,-

Das Schachprogramm benötigt das Diskettensystem JADOS! JADOS ist im Farbkatalog auf Seite 115 ausführlich beschrieben.

## In eigener Sache

Zunächst einmal: Alles Gute zum neuen Jahr! Wir wünschen Ihnen, daß alle Ihre Wünsche und Hoffnungen an das neue Jahr erfüllt werden! Wir wünschen uns, daß Sie weiterhin so treue Leser und Kunden bleiben und diese Zeitschrift auf dem Level halten, den sie jetzt hat.

Unsere guten Vorsätze aus LOOP 11 (jeden zweiten Monat) haben wir erfüllt, wenngleich wegen der Feiertage diese LOOP nicht in den ersten Tagen des ungeraden Monats erscheinen konnte. Und dies, obwohl der angekündigte zweite Redakteur A. Granel es vorgezo-

gen hat, seinen Wirkungskreis wieder nördlich der Donau zu suchen und nicht mehr für uns arbeiten mag!

Zum TCM-Artikel aus LOOP 11 fand sich reges Interesse, nur vergaß der Autor zu erklären, was die Abkürzung TCM bedeutet: „Thermal Conducted Module“, das bedeutet etwa „thermisch verbundenes Modul“.

Über Ihre Kritik – ob positiv oder negativ – über jede LOOP-Ausgabe freuen wir uns. Schreiben Sie uns doch eine formlose Postkarte, welche Artikel Ihnen besonders gut und welche Ihnen weniger gefal-

len haben! Ihre Antworten, wie auch die auf den LOOP-Verlängerungsscheinen, interessieren uns sehr.

Neu für LOOP-Leser: Ab sofort (1. 1. 87) können Sie alle Waren bei uns bequem per Bankeinzug bestellen. Wir können so schneller und problemloser liefern. Die-

ses Angebot gilt vorerst nur für LOOP-Leser! Bitte geben Sie uns bei Ihrer Bestellung Ihr Bankkonto, die Bezeichnung der Bank und die Bankleitzahl an.

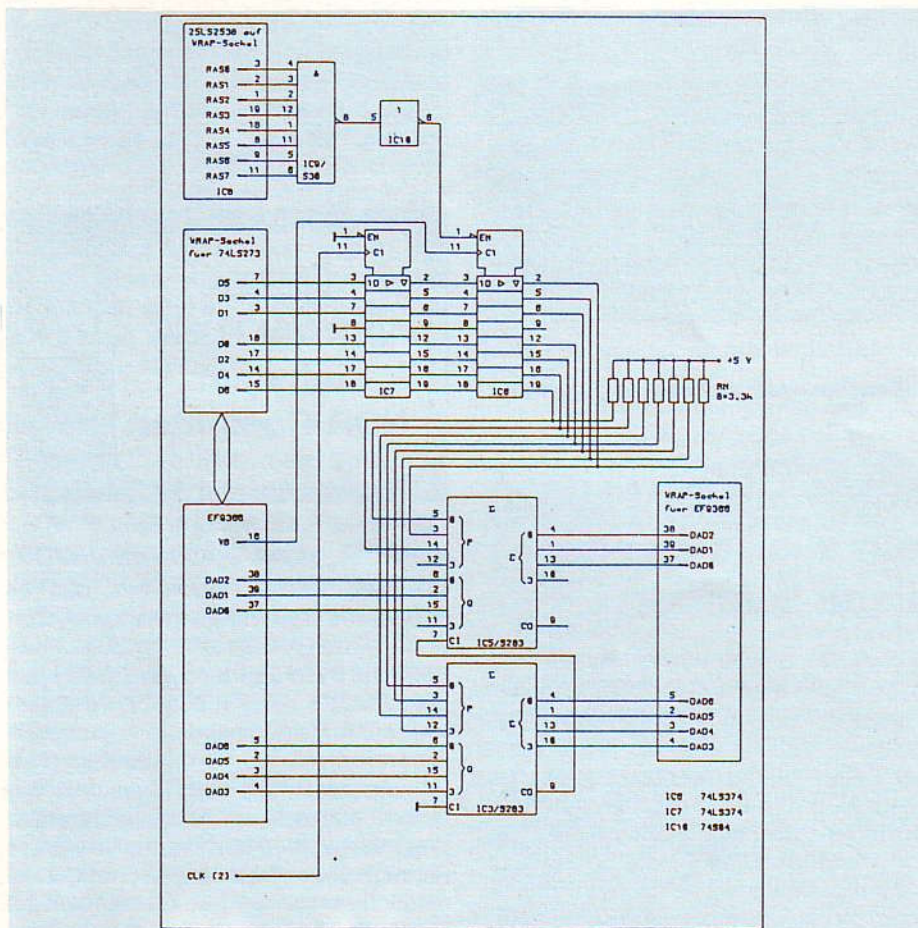
Ein Aufruf an alle Lehrkräfte, die mit dem NDR-Computer arbeiten: Teilen Sie uns Ihre Erfahrungen mit! Vielleicht haben Sie schon Unterrichtsmaterial erstellt? Her-

damit – wir verteilen über die LOOP.

Als Hinweis: Wir würden uns freuen, Sie, lieber LOOP-Leser, während der Hannoverer CeBit 87 (4. bis 11. März 1987) an unserem Stand (Halle 6, F21) zu begrüßen! Nutzen Sie die Gelegenheit, um direkt mit uns zu sprechen: Es hilft beiden!

## Jetzt lieferbar - Jetzt lieferbar

# CAD mit dem NDR-Computer und dem mc CP/M-Computer



Das vorliegende CAD-Programm bildet ein preiswertes Zeichenprogramm, das die hardwaremäßigen Möglichkeiten des mc-CP/M-Computers oder des NDR-Klein-Computers optimal ausnutzt. Zum Zeichnen stehen die Grundelemente Linie, Kreis, Kreissegment und Text zur Verfügung. Die Platzierung dieser Elemente erfolgt mit Hilfe einer Maus, dem im Augenblick effektivsten Eingabemittel bei graphischen Arbeiten. Zur Erleichterung der Arbeit und zur Verkürzung der Einarbeitungszeit arbeitet das Programm vollständig menügesteuert. Natürlich ist ein umfangreiches und mit Beispielen

versehenes Handbuch im Lieferumfang inbegriffen.

Zum Lieferumfang gehören mehrere Bibliotheken (800 KByte) mit den 400 gängigsten TTL-Symbolen (neue IEC-Norm). Bibliotheken können aber auch selbst erstellt, erweitert oder verändert werden. Ein solches Bibliothekssymbol läßt sich dann in Sekundenschnelle in die Zeichnung übernehmen. Das Symbol kann vor der endgültigen Platzierung verschoben, gedreht, vergrößert oder verkleinert werden.

Das Programm arbeitet auf den beiden genannten Rechnern mit der Z80-CPU

unter dem Betriebssystem CP/M 2.2. Für die Grafik wird eine TERM1 bzw. GDP-Baugruppe und für die Maussteuerung eine HCOPY/Maus-Baugruppe mit zugehöriger Maus benötigt. Empfohlen wird weiterhin die Verwendung von 780 KByte-Laufwerken und einer RAM-Floppy.

### Die wesentlichen Leistungsmerkmale sind:

- voll menügesteuert
- komfortable Eingabe mittels Maus
- praktisch jeder Plotter ist mittels Installationsprogramm anpaßbar
- je nach Auflösung steht eine Zeichenfläche von 32 cm x 32 cm (1/100 mm) bis 6.4 m x 6.4 m (2/10 mm) zur Verfügung
- auf Geschwindigkeit optimierte Bildschirmgrafik
- gleichzeitiger Zugriff auf maximal 5 Bibliotheken mit maximal 1000 Symbolen möglich
- Wechsel der Bibliotheken jederzeit möglich
- komfortables Plotprogramm zur Ausgabe der Zeichen
- schnelle Platzierung eines Symbols an die gewünschte Stelle
- ausführliches Handbuch mit einem Einführungsbeispiel
- knapp 300seitiges Bibliotheks-Handbuch mit allen Symbolen
- alle Zeichnungselemente können auf dem Plotter in 8 Farben und verschiedenen Strichstärken dargestellt werden
- flimmerfreies Fadenkreuz
- ganze Zeichnungsbereiche können kopiert, verschoben oder gelöscht werden
- Betrachtungsausschnitt am Bildschirm kann in mehreren Stufen vergrößert und verkleinert werden
- Funktion zum punktgenauen Ansetzen an bereits vorhandene Linien

Das CAD-Programm ist ab Lager lieferbar!

Bestell-Nr.	Bezeichnung	DM
10724	5 1/4" 80 Spuren	495,-
10725	3 1/2" 80 Spuren	535,-
10454	5 1/4" 40 Spuren	535,-
	8" SSSD	535,-

# Disketteneditor für die 6800-Serie mit JADOS

von  
Ralph Dombrowski

In der „LOOP 10“ wurde bereits ein Disketteneditor (Diskdoc) für den Z80 mit CP/M unter Turbo-Pascal vorgestellt. Jetzt ist auch ein Editor für die 68000-Serie unter Jados erhältlich. Mit diesem Programm ist es möglich, veränderte Bits oder Bytes in einem leistungsfähigen Hex/Ascii-Editor zu ändern. Es können aber auch Zahlen addiert, subtrahiert oder logisch verknüpft werden. Geänderte Bytes werden auf Wunsch gekennzeichnet. Änderungen von Zeichen können rückgängig gemacht werden. Es ist möglich, Bytes oder halbe Bytes in einem Sektor einzufügen oder zu löschen. Außerdem verfügt der Editor über Such- und Ersetzroutinen für Hex- und Ascii-Zeichen. Es können beliebige Sektoren auf frei wählbare Adressen geladen werden, oder Zeichenfolgen können abgespeichert werden.

Der Disketteneditor verfügt über eine Routine, die es ermöglicht, Hex- oder Ascii-Folgen, die irgendwo auf der Diskette liegen, zu suchen und durch andere

Zeichen zu ersetzen. Es kann auch überlappend von einem Sektor zum anderen gesucht und ersetzt werden. Sektoren oder Sektorenfolgen können auf einem Drucker ausgegeben werden. Die Druckersteuerung verfügt über umfangreiche Funktionen wie Doppeldruck, Fettdruck, Kursivschrift, Schmalschrift, Breitschrift und andere Befehle. Das ganze Programm ist durch eine Menüsteuerung sehr leicht zu bedienen.

Für den Betrieb des Disketteneditors sind ein Jados-System und mindestens 24 KByte Speicherplatz nötig.

Der RD-Disk-Editor ist ab Lager lieferbar und kostet DM 39,-.

Bestell-Nr.		DM
10855	5 1/4" 80 Sp.	39,-
10854	3 1/2" 80 Sp.	39,-
Andere Formate nicht lieferbar. Dazu benötigt: JADOS		
10292	JADOS 5 1/4" 80 Sp.	140,-
10690	JADOS 3 1/2" 80 Sp.	140,-

## Update Modula-2 Compiler V1.10

Die Version 1.10 des Modula-2 Compilers für den NDR-Klein-Computer unter CP/M68K ist nun erhältlich. Der Update enthält im wesentlichen die folgenden Änderungen gegenüber der Version 1.02:

- Das Modul MathLib mit folgenden mathematischen Funktionen: Sqrt, Exp, Ln, Sin, Cos, Arctan, Entier und Power. Geeignet für Systeme mit oder ohne FPU 68881.
- Das Modul InOut zur Kompatibilität mit Programmen, die auf der von Wirth vorgeschlagenen Library beruhen.
- Beseitigung aller bekannten Fehler im Compiler und in der Bibliothek
- Neues, überarbeitetes Handbuch

Der Update kann für Besitzer des Compilers (einschließlich PEDIT) zum Preis von DM 40,- bestellt werden. Für Besitzer des Compilers ohne PEDIT kostet der Update (einschließlich PEDIT) DM 120,-.

Bestell-Nr.		DM
10878	Update	40,-
10879	Update u. Text-Editor	120,-

## ACRT und CPU 68020 nun auch als Leiterplatte lieferbar!

Dem Wunsch vieler unserer Kunden folgend, haben wir uns entschlossen, diese Leiterplatten nun auch einzeln zu verkaufen.

Der Entschluß fiel uns nicht leicht. Es handelt sich bei den drei Leiterplatten um Multilayer-(Mehrlagen) Leiterplatten, die normalerweise nur in einem Schwallbad ordentlich zu löten sind. Bei sehr großer Löterfahrung und gutem Werkzeug (temperatureregelter LötKolben) ist es jedoch möglich, diese Platinen auch selbst zu löten.

**Wir übernehmen jedoch keinerlei Garantie für die Funktionsfähigkeit eines so hergestellten Gerätes und lehnen jegliche Reparatur oder Inbetriebnahme ab.**

Diese Einschränkung müssen wir ganz klar treffen, da mehrlagige, durch einen falschen Lötvorgang zerstörte Platinen nicht mehr zu reparieren sind. Solche Leiterplatten sind nichts für Anfänger. Sie eignen sich eigentlich nur für Anwender, die die Möglichkeit haben, auf ein

Schwallbad zuzugreifen oder extrem sauber löten können. Daß dies funktioniert, haben einige unserer Softwarepartner bereits bewiesen.

Einen Bausatz für diese Geräte wird es aus den oben erwähnten Gründen sicher nie geben, da wir bei einem Bausatz für die Funktionsfähigkeit des Gerätes geradestehen müssen. Also, diese Platine nur für absolute Köpfer auf eigene Gefahr!

Bestell-Nr.		DM
50047	Platine CPU68020	398,-
50033	ACRTC-Basis	348,-
50034	ACRTC-Color	358,-
60026	CPU XC-68020-12MHz	445,-
60020	HD 63484 ACRTC	215,-
10772	68881XC-RC 12 MHz	698,-
10239	68881 16 MHz	875,-
60743	68020 RC16B 16 MHz	1405,-

*Hinweis:* Andere Bauelemente für diese Systeme können von uns einzeln nicht geliefert werden; Schaltpläne und Bestückungspläne werden mitgeliefert.



### Hardware: 680xx-System mit JADOS

RD-Video ist ein Programm, das es ermöglicht, 350 Videokassetten zu verwalten. Pro Kassette sind für Seite 1 und Seite 2 je 8 Einträge möglich. Die Titel auf der Kassette können beliebig verändert werden. Außerdem wird die Anfangszeit, die Endzeit, die Länge, der Freiraum hinter und zwischen den Titeln und das Datum eingegeben oder ausgewertet. Die Daten werden vom Programm auf ihre Richtigkeit geprüft oder errechnet. Es ist möglich, einzelne Titel, Freiräume oder Filmlängen zu suchen. Die Ausgabe kann auf einen Epson-kompatiblen Drucker umgeschaltet werden. Zum Betrieb des Programms ist ein JADOS-Betriebssystem ab Vers. 2.0 nötig. RD-Video ist relokativ und kann daher von JADOS an jeder beliebigen Stelle im Speicher angelegt werden. Das Programm selbst benötigt einen Speicher von 15 KByte. Es ist auf allen Prozessoren der 68000-Serie lauffähig.

Bestell-Nr.		DM
10852	5 1/4" 80 Spuren	49,-
10853	3 1/2" 80 Spuren	49,-
Dazu benötigt: JADOS!		

# Für Einsteiger Z80 SBC2

## Welchen Code hat die Taste? *Einsteigerpaket HEXIO*

von Martin Husemann,  
4800 Bielefeld 18

Mit dem Programm TASTE können Sie auf einfache Art und Weise sehr schnell den Code einer Taste ermitteln. Wenn Sie zum Beispiel wissen wollen, welchen Code die Taste START erzeugt, starten Sie das Programm TASTE, darauf erscheint: tAStE auf der Anzeige. Drücken Sie nun die START-Taste, so erscheint der Tastencode der START-Taste, nämlich 4b (sedezimal). Nun können Sie weitere Tastencodes ermitteln, drücken Sie zum Beispiel CR, so erscheint 5 7; das ist der sedezimale Code der Taste CR. (Vielleicht fällt Ihnen schon eine Regelmäßigkeit auf, probieren Sie doch mal BEF und SPE, dann bekommen Sie 4 7 und 5b. Schauen Sie sich nun die Anordnung der Tasten an und versuchen Sie einmal zu erraten, welchen Code die 0- oder MVE-Taste hat!) Um wieder einen HEXMON-Befehl eingeben zu können, müssen Sie die RESET-Taste drücken. Dann erscheint wieder das HALLO-1.1.

### Wie macht das Programm das?

Im Programm werden viele Unterprogramme des HEXMON benutzt:

*HoleTaste* fragt solange die Tastatur ab, bis eine Taste gedrückt wurde. Während des Wartens ist die Anzeige sichtbar.

*Print* kopiert einen Text (der hier Tabelle genannt wird) in die Anzeige. Dieser Text besteht aus acht Anzeigecodes, die man dem HEXMON-Handbuch entnehmen kann.

*PrtAc* bedeutet print accumulator, es wandelt den Inhalt des Registers A (Akkumulator) in eine zweistellige Sedezimalzahl um und kopiert die Anzeigecodes dieser Zahl an die Adresse, die in den Registern IX angegeben wird. Hier wird als Adresse der Speicherbereich für die Anzeige angegeben, so daß die Sedezimalzahl ganz links in der Anzeige erscheint. Macht man aus dieser Adresse Anzeige+6, bzw. aus dem Befehl 21 06 80 statt 21 00 80, und läßt man den Befehl call Clear weg (oder ersetzt ihn durch NOP-Befehle, CD 33 00 wird zu 00 00 00), dann bleibt die Schrift tAStE in der Anzeige stehen und der Tastencode erscheint rechts daneben.

*Clear* löscht den Speicher für die Anzeige.

Außerdem wird noch die Adresse Anzeige (8000 sedezimal) verwendet, dort speichert HEXMON die Anzeige.

```
PAGE 1

; Programm TASTE      1.0
;
; Kleines Utility, ermöglicht das ermitteln von Tastencodes
; und zeigt, wie man HEXMON Unterprogramme nutzen kann

.Z80

HoleTaste EQU 000CH ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
Print EQU 0015H ; Kopiert einen Text in die Anzeige
PrtAc EQU 0018H ; Schreibt den Akkumulator-Inhalt Sedezimal in die Anzeige
Clear EQU 0033H ; Löscht die Anzeige
Anzeige EQU 8000H ; Adresse des Anzeige-Speichers

ORG 01150H

1150 21 65 11 TASTE: ld hl,tabelle ; Adresse des Textes in Register HL
1153 CD 15 00 ; Text ausgeben
1156 CD 0C 00 loop: call HoleTaste ; Auf Tastendruck warten, Tastencode steht im Accum.
1159 CD 33 00 call Clear ; Anzeigefeld löschen
115C DD 21 00 80 ld ix,Anzeige+0 ; Position der Ausgabe in der Anzeige ganz links
1160 CD 18 00 call PrtAc ; Tastencode sedezimal ausgeben
1163 18 F1 jr loop ; wiederholen

tabelle:
1165 87 88 92 87 86 FF FF FF db 87H, 88H, 92H, 87H, 86H, 0FFH, 0FFH, 0FFH
; hier steht t A S t E

end
```

(Das ist notwendig, da aus technischen Gründen – Multiplexbetrieb – immer nur eine der acht Anzeigeneinheiten arbeitet. Um nun eine 8stellige Anzeige zu bekommen, muß HEXMON regelmäßig alle Anzeigen nacheinander einschalten. Dafür braucht er aber wiederum die Information, was angezeigt werden soll. Diese Information steht im Speicher von 8000 bis 8007 sedezimal. Drücken Sie die RESET-Taste, dann erscheint evtl. eine der Anzeigen heller, während alle anderen dunkel werden. Das liegt daran, daß HEXMON nun nicht mehr alle Anzeigen nacheinander aktiviert, so daß nur die zuletzt eingeschaltete leuchtet.)

Das eigentliche Programm ist dann recht einfach: Mit Hilfe des Unterprogrammes Print wird der Text tAStE in die Anzeige geschrieben, dann wird mit Hilfe des Unterprogrammes HoleTaste ein Tastendruck erwartet. Der Tastencode dieser Taste steht anschließend im Akkumulator. Nun wird dieser Code mit Hilfe des Unterprogramms PrtAc in die Anzeige geschrieben, nachdem vorher mit dem Unterprogramm Clear die Anzeige gelöscht wurde. Das war's eigentlich. Um weitere Tastencodes ermitteln zu können, folgt noch ein Sprung (jump) zur Stelle loop (Schleife), wo dann wieder auf einen Tastendruck gewartet wird.

### Verwendete Befehle:

*call*  
ruft ein Unterprogramm auf.

### *ld load*

lädt ein Register oder ein Registerpaar (z.B. HL) mit einem Wert, zum Beispiel der Adresse „tabelle“.

### *jr jump relative*

springt zu einer anderen Stelle im Programm, wobei die Stelle nicht als absolute (sedezimal vierstellige) Adresse angegeben wird, sondern als relative Adresse (Displacement), die nur zwei Sedezimalstellen benötigt (1 Byte). Diese Adresse wird zur Adresse im Register PC (dem Befehlszähler) addiert. Dort steht die Adresse, an der der jr-Befehl im Programm steht. Das Ziel des Sprunges bezieht sich also auf die Adresse des Sprungbefehls, daher der Name relative Adresse.

**Wichtiger Hinweis:** Das Programmlisting ist ab Adresse 1150H übersetzt (Befehl ORG 1150). Falls Sie dieses Programm in's RAM eingeben wollen, müssen Sie die Startadresse zu der ersten freien RAM-Zelle (z.B. 8100H) ändern. Ab dieser Zelle geben Sie dann den Maschinencode ein.

Also: Auf Adresse 8100H: 21H eingeben  
8101H: 65H  
8102H: 11H  
usw.

# Z 80-Vollausbau bis ZEAT

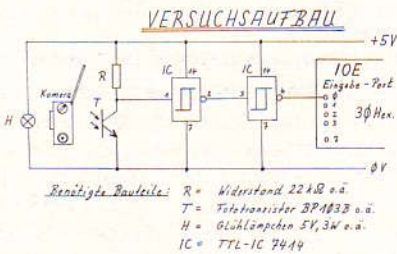
## Verschlußzeitenmessung von Kameras

Eine Anwendung mit dem Z80-Minimal-system: SBC2, Bildschirm und BASIC-Interpreter

von Horst Joachim Fischer, Herrrenteich 11, 4520 Melle 1 und Gerd Graf

Diese Anwendung gefällt uns aus folgenden Gründen besonders gut:

- Sie zeigt das Z80-Grundpaket im praktischen Einsatz: Justieren von Kameraschlußzeiten
- Sie zeigt die Mischung von BASIC-Interpreter und Assembler



Das Prinzip ist einfach: Der NDR-Computer mißt die Zeit, während der eine Fotozelle beleuchtet wird. Der Versuchsaufbau zeigt: Das Licht der Birne H fällt durch das Objektiv (oder die geöffnete Kamera) auf den Phototransistor T. Über zwei Schmitt-Trigger (IC 7414 = 6fach Schmitt-Trigger) wird aus „Hell“ eine „0“ und aus „Dunkel“ eine „1“.

Das Maschinenprogramm (siehe Ablaufplan) prüft nun dauernd den Wert des Bits 0, des Ports 30H der IOE; solange hier „0“ liegt, wird ein Zähler erhöht, der auf Speicherplatz 8FFE und 8FFF gerettet wird.

Das BASIC-Programm lädt nun das unten gezeigte Maschinenprogramm in den Speicher. Achtung, BASIC verwendet dezimale Ziffern in der DATA-Anweisung; so steht in Zeile 150: DATA 33, 0, 0, 219 ..., was hexadezimal 21H, 00H, 00H, DBH bedeutet. Man hätte natürlich auch die DATA-Werte mit dem BASIC-Befehl HEX("nn") wandeln können.

Abhängig vom gewählten Verschlußzeitbereich wird nun in Zeile 310 - 330 ein „schnelleres“ oder „langsames“ Zählprogramm geladen. Herr Fischer hat in Bild 2 bereits die Zykluszeiten daneben geschrieben; das BASIC-Programm muß also nur noch den Zählerwert lesen (Zeile 350 - 370), mit der Schleifenzeit multiplizieren (380 - 400), durch 10<sup>6</sup> teilen (400) und so in 1/Sekunden ausgeben.

Eine hervorragende Anwendung, die natürlich für viele andere Anwendungsfälle zu verwenden ist! Man kann zwei Lichtschranken an die IOE anschließen und Geschwindigkeiten messen - dazu muß nur der Abstand zwischen beiden Lichtschranken bekannt sein. Jeder Teilnehmer im Straßenverkehr kennt solche Geräte ...

Messen Sie doch auch „mal“!

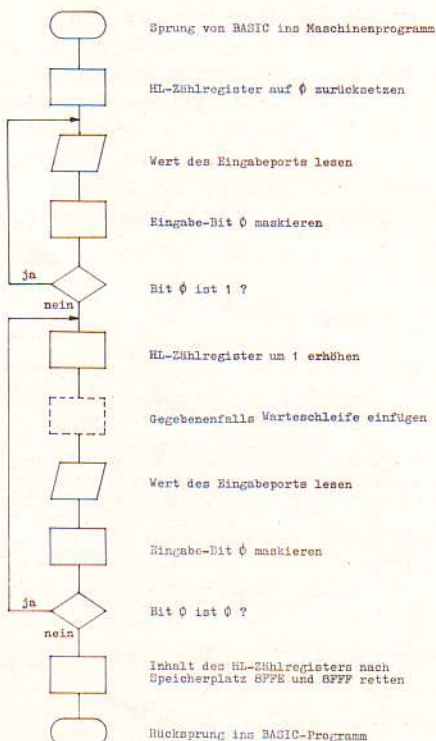
### BASIC-Programm

```

100 REM *** Maschinenprogramm einlesen ***
110 FOR A=HEX("8FA0") TO HEX("8FD1")
120 READ MP
130 POKE A, MP
140 NEXT A
150 DATA 33, 0, 0, 219, 48, 230, 1, 194, 163, 143
160 DATA 35, 219, 48, 230, 1, 202, 170, 193, 34, 254, 143, 204
170 DATA 33, 0, 0, 219, 48, 230, 1, 194, 163, 143
180 DATA 35, 19, 150, 13, 194, 163, 219, 48, 230, 1, 202,
    152, 143, 34, 254, 143, 204

200 REM *** Verschlußzeiten ermittlung ***
210 CLRS
220 OUT 115, 34
230 GOSUB 700
240 PRINT "Bitte wählen Sie den Messbereich: "
250 PRINT "a) = Verschlußzeiten < 1 sec."
260 PRINT "b) = Verschlußzeiten >= 1 sec."
270 A$ = GET$: IF A$ = "a" THEN 270
280 IF A$ <> "a" AND A$ <> "b" THEN 270
290 PRINT A$
300 GOSUB 700
310 PRINT "Bitte betätigen Sie den Kameraauslöser!"
320 IF A$ = "a" THEN CALL HEX("8FA0")
330 IF A$ = "b" THEN CALL HEX("8FB6")
340 CLRS
350 B = PEEK(HEX("8FFE"))
360 C = PEEK(HEX("8FFF"))
370 D = 256 * C + B
380 IF A$ = "a" THEN T = 8.5
390 IF A$ = "b" THEN T = 535.25
400 E = (D * T) / 1000000
410 F = 1/E
500 REM *** Verschlußzeiten ausgabe ***
510 GOSUB 700
520 OUT 114, 4
530 IF A$ = "a" THEN PRINT "Die Verschlußzeit beträgt 1/"; F; " sec."
540 IF A$ = "b" THEN PRINT "Die Verschlußzeit beträgt 1/"; E; " sec."
550 OUT 114, 1
560 OUT 115, 17
570 FOR J=1 TO 15: PRINT: NEXT J
580 PRINT "Wollen Sie erneut ein Verschlußzeit messen, so drücken Sie eine Taste"
590 B$ = GET$
600 IF B$ = "n" THEN 530
610 GOTO 200
700 FOR J=1 TO 4: PRINT: NEXT J
710 RETURN
    
```

Ablaufplan der Maschinenprogramme  
"Verschlußzeitenmessung von Fotokameras"



Maschinenprogramm zum Listing "Verschlußzeitenmessung von Fotok."

AD	OP-Code	MNE	Operand	Bemerkungen
8FA0	21 00 00	LD HL	0000H	
8FA3	DB 30	IN A,	(030H)	
8FA5	E6 01	AND	01H	
8FA7	C2 A3 8F	JPNZ,	8FA3H	
8FAA	23	INC	HL	
8FAB	DB 30	IN A,	(030H)	
8FAD	E6 01	AND	01H	
8FAF	CA AA 8F	JP Z,	8FAAH	
8FB2	22 FE 8F	LD	(8FFEH), HL	
8FB5	C9	RET		
8FB6	21 00 00	LD HL,	0000H	
8FB9	DB 30	IN A,	(030H)	
8FBB	E6 01	AND	01H	
8FBD	C2 B9 8F	JPNZ,	8FB9H	
8FC0	23	INC	HL	
8FC1	0E 96	LDC,	96H	
8FC3	0D	DEC	C	
8FC4	C2 C3 8F	JPNZ,	8FC3H	
8FC7	DB 30	IN A,	(030H)	
8FC9	E6 01	AND	01H	
8FCB	CA C0 8F	JP Z,	8FC0H	
8FCE	22 FE 8F	LD	(8FCEH), HL	
8FD1	C9	RET		

**Taktzeiten:**

$6T$   
 $11T = 34T \cdot 0,25\mu s$   
 $7T = 8,5\mu s$

**Taktzeiten:**

$6T$   
 $7T$   
 $4T \cdot 150 = 2144T \cdot 0,25\mu s$   
 $10T \cdot 150 = 535,25\mu s$

## Arithmetikroutinen in UPN-Notierung mit 32 bit Breite

von Jens Decker

Da meine Fraktalprogramme in Grenzfällen wegen unzureichend genauer Arithmetik unbrauchbar wurden, habe ich mich entschieden, die Rechnerei mit 32 bit Breite durchzuführen. Da sicherlich auch andere Z80-Programmierer daran brüten, nicht zuletzt die vielen Computereulinge, wäre es sicherlich nützlich, das derzeitige Zwischenergebnis (% Grundrechenarten, Module zum Handling der auf dem Stack liegenden Daten) in LOOP zu veröffentlichen, zumal das Berechnen der Winkelfunktionen durch die Ablage der Daten auf dem Stack wesentlich vereinfacht wird (keine Zwischenspeicherung mehr nötig) und so das Verfahren mühelos auf sämtliche Berechnungen ausgebaut werden kann.

Arithmetik in Maschinensprache ist ein heißes Thema. 16 bit-Versionen für die Grundrechenarten hat wohl jeder in der Schublade, pardon auf der Utility-Kassette. Bei 32 bit-Routinen, die auch für längere Rechnungen brauchbar sind, sieht es jedoch meistens düster aus.

Dem will dieser Artikel abhelfen. Um ohne Zwischenspeicherung und ähnlichem Zeugs auszukommen, sollen die Zahlen auf dem Stack abgelegt werden; es wird also wie bei Forth in umgekehrt polnischer Notation gearbeitet.

### Was heißt dies nun und was bringt's?

Statt zu schreiben und zu rechnen  $(5 + 30) \times (54 + 2)$  macht man  $5;30;+;54;2;+;*$ !!! Zunächst legt man die Zahlen

5 und 30 ab, dann wird addiert, das Ergebnis liegt sodann statt der Summanden auf dem Stack, sodann wird 54 bzw. 2 auf den Stack gegeben und addiert. Nun liegen zwei Zahlen auf dem Stack; 35 von der ersten Addition und 56 von der zweiten. Und nun kommt der Trick, diese beiden Zahlen werden einfach multipliziert und das Ergebnis vom Stack genommen, ohne Klammern und ohne Zwischenspeicherung. Dieses Verfahren läßt sich natürlich auch bei beliebig langen Termen anwenden und ist nicht auf die Grundrechenarten beschränkt.

### Wie wird dies nun verwirklicht?

Jede der vorliegenden Routinen besteht aus drei Teilen:

1. Datenholen vom Stack zu einem allen Routinen gemeinsamen Block
2. Arithmetikroutine mit 32 bit Breite
3. Datenablage des Ergebnisses auf den Stack

Die Routinen 1 und 3 sind für alle Routinen gleich (lediglich bei der Division wurde 3 etwas abgeändert, da sonst der Rest auf den Stack käme) und vollkommen relokativ (= verschiebbar). Das komplette Arithmetikunterprogramm z.B. für die Addition setzt man sich einfach durch Aneinanderhängen der Routinen zusammen. Der Datenblock ist nur einmal nötig, da die Daten ja auf dem Stack liegen und – der Z80 ist leider kein Multitaskingprozessor – nur jeweils eine Routine auf die Kopie zugreift.

Beim Aufruf werden einfach die Faktoren auf den Stack gegeben, das Arithmetikunterprogramm aufgerufen (CD xxxx) und am Ende der Rechnerei das Ergebnis vom Stack genommen, wodurch die darunterliegenden Adressen wieder frei sind.

### Datenformat im Block:

Langwort eins  
llsb mlsb lmsb mmsb  
Langwort zwei  
llsb mlsb lmsb mmsb

Resultat  
llsb mlsb lmsb mmsb  
(mlsb entspricht most less signifikant byte)

### Datenformat auf dem Stack

9000 FFFF  
8FFE lmsb  
8FFE mmsb

als erstes abgelegt,  
der Stack wächst nach  
unten!!!

8FFD mlsb  
8FFC llsb

8FFF bis 8FFC ent-  
spricht Langwort 2

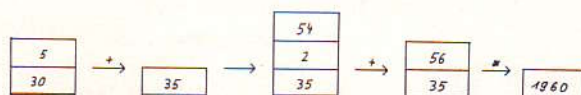
8FFB es folgt Langwort 1 in gleicher Art  
wie Langwort 2

### Was läßt sich nun damit anfangen?

Zunächst einmal lassen sich alle Berechnungen durchführen, die auf den vier Grundrechenarten beruhen. Sind Bruchzahlen vorhanden, so muß eben Bit 16 als 1 angesehen werden. Für die Berechnung von sin, cos oder e-Funktionen lassen sich Unterprogramme entwickeln, die auf den hier abgedruckten Unterprogrammen aufbauen. Versehen mit einer Datenhol- und einer Datenablageroutine, wie sie auch für die Grundrechenarten verwendet wird, passen sie sich nahtlos in das System ein, da die Daten ja auf dem Stack liegen und somit eine Reihenentwicklung zum Kinderspiel wird.

### Reihenentwicklung:

1.  $\sin(x) = x/1! - x^3/3! + x^5/5! - \dots$
2.  $\cos(x) = 1 - x^2/2! + x^4/4! - \dots$
3.  $e^x = 1 + x/1! + x^2/2! + x^3/3! + \dots$
4.  $\ln(1+x) = x - x^2/2 + x^3/3 - \dots$



### Datenholen für 32bit Arithmetik in UPN-Abfolge

```
Langwort1 := 8800
Langwort1+2 := 8802
Langwort2 := 8804
Langwort2+2 := 8806
Resultat := 8808
Resultat+2 := 880A
Datenholen := 880C (für die erste Routine)
```

880C:	FD E1	PDP IY	Rücksprungadresse retten
	E1	PDP HL	Der Reihe nach werden
	22 Langwort1	LD (Langwort1),HL	die Faktoren vom Stack
	E1	PDP HL	genommen und abgelegt
	22 Langwort1+2	LD (Langwort1+2),HL	
	E1	PDP HL	
	22 Langwort2	LD (Langwort2),HL	
	E1	PDP HL	
	22 Langwort2+2	LD (Langwort2+2),HL	
	AF	XDR AF	Das A-Register wird ge-
	21 Resultat	LD HL, Resultat	löscht und der Wert (00)

```
77 LD (HL),A      In die Speicherzellen
23 INC HL        für das Ergebnis ge-
77 LD (HL),A    geschrieben
23 INC HL
77 LD (HL),A    Übertragen
23 INC HL        und Zeiger incrementieren
882B 77 LD (HL),A
```

### Additionsteil

Im Anschluß an das jeweilige Datenholen:

2A Langwort1	LD HL,(Langwort1)	HL und DE mit dem low-Teil
ED 5B Langwort2	LD DE,(Langwort2)	der Summanden laden
19	ADD HL,DE	Ohne Übertrag addieren
22 Resultat	LD (Resultat),HL	Ablegen
2A Langwort1+2	LD HL,(Langwort1+2)	HL und DE mit dem high-Teil
ED 5B Langwort2+2	LD DE,(Langwort2+2)	der Summanden laden
ED 5A	ADC HL,DE	Mit Übertrag addieren
22 Resultat+2	LD (Resultat+2),HL	Ablegen

### Multiplikationsteil

D6 20	LD B, 32	Reg. B als Schleifenzähler
2A Resultat	LD HL,(Resultat)	Linksschieben des gesamten
29	ADD HL,HL	Resultats durch Verdopplung
22 Resultat	LD (Resultat),HL	(low+low, high+high+Carry)
2A Resultat+2	LD HL,(Resultat+2)	
ED 6A	ADC HL,HL	
22 Resultat+2	LD (Resultat+2),HL	
2A Langwort1	LD HL,(Langwort1)	Linksschieben des Multi-

```

29      ADD HL,HL
22 Langwort1  LD (Langwort1),HL
2A Langwort1+2 LD HL,(Langwort1+2)
ED 6A      ADC HL,HL
22 Langwort1+2 LD (Langwort1+2),HL
30.17     JRN C,+23
2A Resultat  LD HL,(Resultat)
ED 5B Langwort2 LD DE,(Langwort2)
19        ADD HL,DE
22 Resultat  LD (Langwort2),HL
2A Resultat+2 LD HL,(Resultat+2)
ED 5B Langwort2+2 LD DE,(Langwort2+2)
ED 5A      ADC HL,DE
22 Resultat+2 LD (Resultat+2),HL
1D C7     DJNZ,-57

```

Ende der Schleife

#### Subtraktionsteil

```

2A Langwort1  LD HL,(Langwort1)
ED 5B Langwort2 LD DE,(Langwort2)
AF          XOR A
ED 52        SBC HL,DE
22 Resultat  LD (Resultat),HL
2A Langwort1+2 LD HL,(Langwort1+2)
ED 5B Langwort2+2 LD DE,(Langwort2+2)
ED 52        SBC HL,DE
22 Resultat+2 LD (Resultat+2)

```

HL und DE mit den low-Teilen laden  
Carry löschen  
Abziehen  
Ablegen  
HL und DE mit den high-Teilen laden  
Abziehen (mit Carry von low); Ablegen

#### Divisionsteil mit modifizierter Datenablage

Da sich bei der Division der Rest in Resultat ablegt, und sich das Ergebnis in Langwort1, LU1+2 befindet, muß das erste Langwort auf den Stack gegeben werden.

```

06 20     LD B,32
2A Langwort1  LD HL,(Langwort1)
ED 6A      ADC HL,HL
22 Langwort1  LD (Langwort1),HL
2A Langwort1+2 LD HL,(Langwort1+2)
LD 6A     ADC HL,HL
22 Langwort1+2 LD (Langwort1+2),HL
2A Resultat  LD HL,(Resultat)
LD 6A     ADC HL,HL
22 Resultat  LD (Resultat),HL
2A Resultat+2 LD HL,(Resultat+2)
ED 6A      ADC HL,HL
22 Resultat+2 LD (Resultat+2),HL
AF        XOR AF
2A Resultat  LD HL,(Resultat)
ED 5B Langwort2 LD DE,(Langwort2)

```

Reg. B als Schleifenzähler  
Dividend (auch Ergebnis) nach links schieben und Carry (Ergebnisbit) einschieben  
Bit 31 des Dividenden wird nach Resultat eingeschoben (davon wird der Divisor abgezogen; späterer Rest)  
Carry löschen, da kein SUB HL,DE existiert  
Vom Dividendenüberlauf

```

ED 52      SBC HL,DE
22 Resultat LD (Resultat),HL
2A Resultat+2 LD HL,(Resultat+2)
ED 5B Langwort2+2 LD DE,(Langwort2+2)
ED 52      SBC HL,DE
22 Resultat+2 LD (Resultat+2),HL
38 02     JR C,+2
18 18     JR Z,+24
2A Resultat  LD HL,(Resultat)
ED 5B Langwort2 LD DE,(Langwort2)
19        ADD HL,DE
22 Resultat  LD (Resultat),HL
2A Resultat+2 LD HL,(Resultat+2)
ED 5B Langwort2+2 LD DE,(Langwort2+2)
ED 5A      ADC HL,DE
22 Resultat+2 LD (Resultat+2),HL
37        SCF
3F        CCF
10 AB     DJNZ,-88
2A Langwort1  LD HL,(Langwort1)
ED 6A      ADC HL,HL
22 Langwort1  LD (Langwort1),HL
2A Langwort1+2 LD HL,(Langwort1+2)
ED 6A      ADC HL,DE
E5        PUSH HL
2A Langwort1  LD HL,(Langwort1)
E5        PUSH HL
FD E9     JP (Y)

```

wird der Divisor abgezogen  
Verfahren wie beim Subtraktionsteil  
falle Divisor größer bei Gleichheit Umgehung der Addition, die nötig ist, falls der Divisor größer ist als der Dividendenüberlauf, also zu unrecht abgezogen wurde  
Verfahren wie beim Additionsteil  
Carry setzen (zum löschen)  
Carryflag invertieren  
Ende der Schleife  
letztes Ergebnisbit einschieben (wie oben)  
high-Teil gleich auf Stack  
low-Teil holen  
und auf Stack legen  
Rücksprung

#### Datenablage für 32bit Arithmetik in UPN-Ablage

Im Anschluß an die Arithmetikroutine (bei Division bereits angehängt!):

```

2A Resultat+2 LD HL,(Resultat+2)
E5          PUSH HL
2A Resultat  LD HL,(Resultat)
E5          PUSH HL
FD E9     JP (Y)

```

high-Teil holen  
und auf Stack legen  
low-Teil holen  
und auf Stack legen  
Rücksprung

#### Routine zum Umliegen der beiden obersten Stackelemente

```

FD E1     POP IY
E1        POP BC
D1        POP DE
E1        POP HL
F1        POP AF
D5        PUSH DE
C5        PUSH BC
F5        PUSH AF
E5        PUSH HL
FD E9     JP (Y)

```

Rücksprungadresse retten  
runter vom Stack  
und wieder drauf  
Rücksprung

# Z80-CP/M2.2

## Flimmerfreie Bilder durch Seitenumschaltung im FLOMON

Rolf-Dieter Klein

Da bisher leider nur sehr wenig mit den unsichtbaren Bildseiten gearbeitet wurde, möchte ich sie an dieser Stelle einmal ausführlich erklären.

Sowohl bei FLOMON als auch beim Grundprogramm gibt es die Möglichkeit, mehrere Bildseiten der GDP zu verwenden. Auf der GDP64 befindet sich ein Speicher, um vier vollständige Bilder zu speichern. Der Inhalt eines Speicherbereichs wird permanent angezeigt, in die anderen drei Seiten kann man wahlweise unsichtbar schon neue Bilder einschreiben.

Die Grafik-Routinen des Flomons lassen sich über sogenannte ESCAPE-Sequen-

zen erreichen. Das heißt, man sendet spezielle Steuersignale und dann Befehle. Senden bedeutet einfach eine Ausgabe auf den Bildschirm mit den normalen Ausgabebefehlen, wie PRINT in BASIC oder WRITELN in PASCAL. Dazu ein paar Beispiele. Zeichnen einer Linie:

```

1. PRINT CHR$(27);CHR$(;7;"G";
oder
1. WRITE(CHR(27),CHR(27),'G');
```

damit schaltet man bei Flomon den Graphik-Mode ein. Alle weiteren Zeichen haben nun eine besondere Bedeutung und werden nicht mehr auf dem Bildschirm als Buchstaben abgebildet. Achtung: Tritt hier ein Fehler auf, z.B. in BASIC

"SyntaxERROR" so ergeben sich unter Umständen wilde Bilder.

```

2. PRINT "MO 0 D 511 255"
oder
```

```

2. WRITELN('MO 0 D 511 255');
```

zeichnet die Linie. Eine vollständige Zusammenstellung aller Befehle ist übrigens im CP/M-Sonderheft 81 der Zeitschrift MC abgedruckt. Die Linie erscheint nun auf dem Bildschirm. Achtung: diesen Befehl kann man nicht im Direkt-Modus von BASIC aus aufrufen.

Damit man wieder in den Text-Mode zurückkommt gibt man noch den Befehl:

```

3. PRINT "A";
oder 3.WRITE('A')
```

Danach blinkt die dargestellte Linie, da wieder der Zweiseitenmode aktiviert wird.

Für die Seitensteuerung gibt es nun eine Reihe von wichtigen Befehlen, die arbeiten, wenn man wieder den Graphik-Mode anwählt:



a)  
PRINT"X";n  
oder WRITELN('X',n);

Wenn n einen Wert ungleich Null besitzt, werden die Bildseiten 0, 1, 2 und 3 zyklisch angezeigt. Der Faktor gibt dabei an, in welchem Zeitabstand die Umschaltung erfolgt, eine Einheit steht dabei für 20ms. Wenn man den Wert Null eingibt, so wird die Umschaltung gestoppt.

b)  
PRINT"Y";n  
oder WRITELN('Y',n);

Wie bei den Befehlen X, es werden jedoch nur zwei Seiten miteinander gewechselt: Entweder 0 mit 1 oder 2 mit 3, je nachdem, welche die letzte angezeigte Seite war.

Dieser Befehl ist normalerweise nach dem Start von Flomon aktiv, denn der Cursor wird auf diese Weise am Blinken gehalten. Wenn man den Wert 0 für n eingibt, so wird der Mode ausgeschaltet. Die angezeigte Linie blinkt dann nicht mehr.

c)  
PRINT"S",n  
oder WRITELN('S',n);

Die Schreib- und Lesebildseite wird ausgewählt. Flomon hat die Möglichkeit, eine Seite auf dem Bildschirm darzustellen und die andere Seite beschreiben zu lassen.

n berechnet sich wie folgt:

$n = \text{schreibseite} * 4 + \text{leseseite}$

Soll also Seite 1 angezeigt und Seite 3 beschrieben werden, so gibt man für n den Wert  $3 * 4 + 1 = 13$  ein.

Der Befehl S hat außerdem die Eigenschaft, die Bildseite synchron zum nächsten Bildwechsel umzuschalten. Damit wird zusätzliches Flimmern vermieden.

d)  
PRINT"P",n  
oder WRITELN('P',n);

Wie bei S, jedoch erfolgt hier ein unmittelbares Umschalten der Bildseiten, ohne zu warten.

Mit dieser Seitenumschaltung lassen sich verschiedene Effekte erzielen:

1. Bewegungen von Gegenständen können auf dem Bildschirm ausgegeben werden, ohne daß man den Löschkvorgang sieht.

2. Mit der automatischen Seitenumschaltung können Vier-Phasen-Animationen durchgeführt werden, bei der sich sogar sehr komplexe Gegenstände auf bestimmte Weise bewegen können.

zu 1.):

Dazu setzt man z.B. die Schreibseite auf 1 und die Leseseite auf 0. Nun zeichnet man eine Figur. Dann wird die Schreibseite auf 0 gesetzt und die Leseseite auf 1. Nun kann man die Figur ein Stück versetzt ausgeben. Man schaltet wieder um, und nun muß man zuerst die allererste Figur löschen. Dies geschieht entweder

mit dem C-Befehl, der allerdings langsam ist, oder dadurch, daß man die Figur durch überschreiben löscht. Dazu muß man den Eraser-Mode selektieren. Das geschieht mit dem G-Befehl (siehe Beschreibung des Grafik-Prozessors EF9366, Eraser = G 0 1, Löschstift oder Eraser). Mit dem G-Befehl kann man direkt in die Register des GDPs schreiben. Nach dem Löschen muß man den Schreibstift wieder aktivieren und das geschieht mit dem Befehl G 0 0 (also PRINT"G 00" oder WRITELN('G 00')). Man schaltet die Seiten wieder um, usw.

zu Methode 2:

Alle vier Phasen werden in die Seiten 0 bis 3 geschrieben. Dann gibt man den X-Befehl (nicht vergessen mit Y 0 die normale Seitenumschaltung auszuschalten) und das Bild bewegt sich.

Zur Berechnung der vier Phasen:

Bei einem Zahnrad zum Beispiel kann man eine Drehung dadurch erreichen, daß man das Zahnrad immer um einen kleinen Winkel dreht. Bei der vier Phasen Animation muß man es gerade um soviel drehen, daß es nach vier mal drehen wieder identisch mit dem ersten Bild ist. So lassen sich auch Pfeilbewegungen darstellen, wenn man auf einer Linie genügend viele Pfeile anreicht.

Wir würden uns freuen, wenn wir dazu ein paar ausführliche Programmbeispiele an die LOOP-Redaktion gesendet bekämen.

## Autostart unter CP/M2.2

Es wäre schon ganz schön, wenn nach dem Start des Betriebssystems auch ein Programm automatisch geladen werden könnte. Dies ist im Prinzip möglich, wenn in den Befehlsbuffer des CCP ein Programmname eingetragen und dann auf die Diskette geschrieben wird. Doch dieses Verfahren hat einen Hacken, bei jedem Warmstart wird der CCP nachgeladen und damit das eingetragene Programm neu gestartet, dieser Effekt ist jedoch fast immer unerwünscht. Es muß also eine Möglichkeit gefunden werden, diesen Nebeneffekt zu umgehen. Beim näheren Betrachten des CCP fällt auf, daß dieser mit zwei Sprungbefehlen beginnt. Der erste Einsprung auf X400 ist der normale CCP-Aufruf, dieser prüft, ob ein Eintrag im Befehlsbuffer vorliegt, und führt diesen Befehl aus. Der zweite Einsprung auf X403 löscht einen Eintrag im CCP-Puffer, indem er die Längeninformation auf Null setzt, und springt dann zum CCP-Aufruf. Könnte man nun beim Kaltstart den normalen CCP-Aufruf und beim Warmstart den CCP-CLEAR verwenden, dann wäre das eingangs erwähnte Problem gelöst. Der Aufruf des CCP erfolgt aus dem BIOS, also müssen wir uns dessen Listing näher ansehen. Im Sonderheft 2 auf Seite 93 ist es zu finden. Betrachtet man

nun die Programmteile ab Seite 94, Spalte 2 für Kalt- und Warmstart, so wird man feststellen, daß der Programmteil "GOCPM" von beiden Routinen benutzt wird. Dieser Teil endet mit dem Sprung in den CCP. Hier ist also die Stelle an der man ändern muß.

Bevor man mit der Änderung des BIOS beginnen kann, müssen folgende Programme auf der Diskette verfügbar sein:

CP/M, Wordstar, DDT, SYSGEN80, BIOS80.ASM, MAC.COM + DISKDEF.LIB oder ASM.COM.

Ändern des BIOS:

WS laden und mit der Funktion N BIOS80.ASM aufrufen. Nach der Zeile CPMB EQU CCP ; START CP/M-BOOT wird CCPCLR EQU CCP+3 eingefügt. Vor dem Label WBOOT: wird der Befehl JMP GOCPM ersetzt durch den Befehl CALL GOCPM daran anschließend wird JMP CPMB eingefügt.

Vor dem Label GOCPM: werden folgende zwei Befehle eingefügt: CALL GOCPM und JMP CCPCLR. Am Ende der Routine GOCPM: wird der Befehl JMP CPMB ersetzt durch RET.

Wer nun in Besitz des MAC.COM ist, für den ist die Arbeit schon beendet. Er kann die geänderte Datei mit CTRL KD abspeichern und mit dem MAC neu übersetzen. ALLE anderen müssen alle Makrodefinitionen aus der Quelle entfernen und die Makroaufrufe durch Abtippen der Tabellen aus dem Sonderheft ersetzen. Ist dies geschehen so kann das neue BIOS auch mit dem ASM.COM übersetzt werden. Beide Assembler erzeugen die Dateien BIOS80.PRN und BIOS80.HEX. Für die weitere Arbeit wird die Datei mit dem Namen BIOS80.HEX benötigt.

Erzeugen eines neuen Betriebssystems:

Zuerst mit SYSGEN80 und SAVE die Datei CPM60.SYS erzeugen. Anschließend mit DDT die Datei CPM60.SYS laden (siehe LOOP7, 420K RAMFLOPPY). Wenn dies erfolgt ist, mit dem I-Befehl des DDT die neue Datei festlegen, IBIOS80.HEX. Nun mit dem R-Befehl die Datei mit Versatz einlesen. Der Versatz errechnet sich aus der Startadresse des BIOS 0EA00H und der Adresse auf die eingelesen werden soll, hier ist dies die Adresse 01F80H, somit beträgt der Versatz 3580H. Also lautet die komplette Eingabe R3580. Beim Einlesen von HEX-Dateien werden diese automatisch vom DDT in Maschienen-Dateien umgewandelt. Nun wird mit CTRL C der DDT verlassen, dann SYSGEN80 aufgerufen und das neue Betriebssystem auf die Diskette geschrieben. Dabei für Source-Drive nur Return betätigen, da sonst das alte System von der Diskette gelesen wird.

Eintragen der Datei in den CCP-Befehlsbuffer:

CPM60.SYS mit SYSGEN80 und SAVE neu abspeichern. Die neue Datei mit DDT laden. Der CCP beginnt auf Adresse 980H. Wir wollen nun das Programm SPEED.COM nach jedem Kaltstart aufrufen. Dazu müssen die Buchstaben SPEED in Großschreibweise in den Befehlsbuffer eingetragen werden. Das Byte auf der Adresse 987H enthält die Information über die Länge des Befehls. Hier müssen wir 5 eintragen, da der Name aus 5 Buchstaben besteht. Die nachfolgenden Bytes enthalten den Namen des Programms, die entsprechenden Werte holen wir uns aus der ASCII-Tabelle. Nun die Eingaben in ihrer Reihenfolge:

S987CR, 05CR, 53CR, 50CR, 45CR, 45CR, 44CR, .CR  
 Nachdem Verlassen des S-Befehls kann mit D980 überprüft werden ob alles richtig ist. Nach Verlassen des DDT's kann nun das Betriebssystem mit SYSGEN80 auf die Bootspur gebracht werden. Nach einem Kaltstart muß nun ohne weitere Eingabe das Programm SPEED aufgerufen werden. Jetzt kann zwar nach dem Kaltstart ein Programm aufgerufen werden, aber was tun, wenn eine ganze Programmfolge automatisch nach einen Kaltstart abgearbeitet werden soll. Dazu ein Beispiel. Wir wollen nach dem Kalt-

start Speed aufrufen, die Ramfloppy initialisieren und alle Dateien, die mit .COM enden in die RAMFLOPPY kopieren. In den Befehlspeicher tragen wir SUBMIT COPY ein. ALLE Zeichen zählen, auch das Leerzeichen dazwischen, also als Länge 0B eintragen. Mit Wordstar eine Datei mit dem Namen COPY.SUB erstellen. Dann werden alle Programme über diese Datei aufgerufen. Befehle in der Datei COPY.SUB:  
 SPEED  
 INITRAM  
 PIP E:=A:\*.COM

Diese Version hat den Vorteil, daß

sie von Diskette zu Diskette unterschiedliche Aufgaben ausführen kann, da nur die SUBMIT-Datei geändert werden muß. Diese Änderung des BIOS funktioniert sowohl beim NDR- als auch beim MC-CP/M-Computer.

PS: Wer nicht in der Lage ist, die Änderungen des BIOS selbst durchzuführen, kann über GES eine Diskette beziehen, auf der sich die Quellen und die HEX-Dateien des geänderten BIOS befinden. Die Quell-Dateien sind bereits für eine Übersetzung mit dem ASM.COM geeignet. Im NDR-BIOS ist ein 420K-RAMFLOPPY installiert, im MC-BIOS eine 128K-RAMFLOPPY.

# PASCAL und BASIC

## Erweiterungen zur Disk Doc-TOOL-Diskette

von Peter Porbadnig,  
 Finkenweg 23, 2070 Ahrensburg,  
 Telefon: (04102) 57051

Ich habe mir vor einigen Tagen in Ihrer Filiale in Hamburg den Disk-Doc für CP/M bzw. Z80 gekauft. Das Programm gefällt mir sehr, zumal ich damit eine aus Versehen gelöschte Datei zurückholen konnte. Was mir sehr gefällt ist, daß der Pascal-Quellcode mit auf der Diskette ist.

Wie der Autor des Programms im Text empfiehlt, habe ich mich auch gleich an die Arbeit gemacht und zwei neue Prozeduren für den DiskDoc geschrieben, die ich für sehr wichtig halte, bzw. die die Arbeit noch komfortabler gestalten. Als erstes habe ich, da ich zwei Laufwerke besitze, die zu bearbeitende Disk in Drive B gelegt, damit nicht immer die System Disk gewechselt werden muß.

Als zweites habe ich eine kleine Prozedur geschrieben, die aus der im Directory angegebenen Blocknummer Spur und Sektor berechnet, um auf der Disk gezieht auf Files zugreifen zu können. Die zweite Prozedur liest aus dem RAM der

BANKBOOT-Karte den Bildschirm-RAM aus und kopiert den Inhalt nach \$9000 in den Arbeitsspeicher, damit Turbo-Pascal darauf zugreifen kann. Das ermöglicht einen schnellen Ausdruck des Bildschirm-Inhaltes, was manchmal sehr nützlich sein kann. Als Anhang sende ich Ihnen die Ausdrucke zu meinen Änderungen.

*Hinweis:* Die TOOL-Diskette kostet (nur) DM 39,—

Bestell-Nr.	Bezeichnung	DM
10550	5 1/4" 80 Spuren	39,—
10549	3 1/2" 80 Spuren	39,—

Änderung zum Bearbeiten von Drive B:

```
IF SEITE=0 THEN ISEITE:=#D2 ELSE ISEITE:= #D3;
vorher :          DO          D1
```

Die Prozeduren für Blockrechnen und Hardcopy:

```
PROCEDURE block; (* Eingabe als HEX-Zahl *)
```

```
var block,seite,track,sector,b : integer;
```

```
begin
  write (' Block : ');
  readln (block);
  seite := 1; track := 2; sector := 2; (*default*)
  for b := 4 to block do
    begin
      sector := sector + 2;
      if sector > 5 then
        begin
          sector := sector - 5;
          if seite = 0 then
            seite := 1
          else
            begin
              track := track + 1;
              seite := 0;
            end;
          end;
        end;
      writeln;
      writeln (' Block : ',block); writeln;
      writeln (' Seite : ',seite,' Track : ',track,' Sektor : ',sector);
    end;
end;
```

```
PROCEDURE Print_Screen;
```

```
var wert : char;
    zeile,spalte : byte;
```

```
begin
```

```
inline
```

```
(
  (* cseg *)
  (* init: *)
  $21/++16/ (* ld hl,start *)
  $11/$00/$FB/ (* ld de,0f800h *)
  $01/$14/$00/ (* ld bc,ende-start *)
  $ED/$B0/ (* ldir *)
  $CD/$00/$FB/ (* call 0f800h *)
  $C3/++22/ (* jp weiter *)
)
```

```
(* start: *)
(* ab hier wird das Programm nach *)
(* F800 verschoben *)
$3E/$00/ (* ld a,00h ; bankboot *)
$D3/$C8/ (* out (0c8h),a *)
$21/$B8/$5B/ (* ld hl,5B8Bh ; screen *)
$11/$00/$90/ (* ld de,9000h ; RAM fuer SCREEN *)
$01/$80/$07/ (* ld bc,24*80 ; SCREEN-laenge *)
$ED/$B0/ (* ldir *)
$3E/$B0/ (* ld a,B0h ; BANK 0 (CP/M) *)
$D3/$C8/ (* out (0c8h),a *)
$C9/ (* ret *)
(* ende: *)
(* weiter: *)
$00/ (* nop *)
$00/;
```

```
writeln (LST); writeln (LST);
for zeile := 0 to 23 do
  begin
    for spalte := 0 to 79 do
      begin
        wert := chr(memA $9000 + zeile*80 + spalte 0);
        (* A u = eckige Klammer auf und zu *)
        write (LST,wert);
      end;
    writeln (LST); write(LST,chr(12),chr(7));
  end;
```

Einbinden der Prozeduren in die Tastenabfrage:

```
*B : begin
  GOTOXY(1,1);
  FOR I:=1 TO 7 DO
    WRITELN(' ');
  gotoxy (1,1);
  block;
  while not keypressed do;
  info;
  BCURS(INDEX,'>','<');
  end;
```

```
*Y : Print_Screen;
```

Änderungen des Menues :

```
PROCEDURE INFO;
```

```
BEGIN
  GOTOXY(CPOSX4,CPOSY4);
  WRITE('! = info ^D = ENDE ^B = Block ^Y = HCopy');
```

## Tips und Tricks bei HEBAS, Nr. 6

von Dr. Hans Hehl

### 1. HEBAS und CP/M Plus (Vers. 3)

HEBAS gibt es nun in der überarbeiteten Version 3.1 für den NDR-Rechner, die auch unter dem Betriebssystem CP/M PLUS (Vers. 3) von Digital Research läuft (Austausch alter Versionen über den UPDATE-Service, siehe LOOP 8/9, S. 22).

### HEBAS und illegale Z80-Opcodes

Für die HEBAS-Anwender, die Interpretänderungen selber machen können,

erscheint im Januarheft der mc, Franzis-Verlag, ein Artikel über HEBAS, indem beschrieben wird, wie die illegalen Z80-Opcodes entfernt werden. Damit läßt sich HEBAS auch leichter tracen, da Debugger in der Regel die illegalen Z80-Opcodes nicht verstehen. HEBAS läuft mittels CP/M 2.2-Emulator nun auch auf dem 68000-NDR-System, so daß Aufsteiger ihre alten BASIC-Programme weiter verwenden können. Demnächst wird HEBAS auch mit der CPU HD64180 funktionieren.

### 3. HEBAS und ein neuer Befehl: USER

Außerdem wird im Januarheft der mc ein

neuer BASIC-Befehl, nämlich USER, beschrieben. Damit können unter HEBAS die User-Ebenen 0 – 9 direkt angesprochen werden. So kann z.B. von User-Ebene 8 ein Programm geladen und problemlos auf die User-Ebene 5 abgespeichert werden, was mit PIP.COM direkt nicht geht.

### 4. Ein HEBAS-Trick:

Drückt man während des Ablaufs eines BASIC-Programmes die Tastenkombination „Control-T“, wird die gerade bearbeitete Zeilennummer ausgegeben. Zu Testzwecken kann dies im Programm mit dem Befehl CALL 1003 durchgeführt werden.

# FÜR 68000-EINSTEIGER

## Neues Schreibgefühl mit der GDP

*Unterstreichen, Blinken, Invers, Geneigt und Text*

von Dietmar Arnds,  
Auf der Heide 76, 5804 Herdecke

Diese Routinen dienen dazu, der GDP Unterstreichen, Blinken, Inversdarstellung, rechtsgeneigte Schrift und Textausgabe im Kasten beizubringen. Die Unterprogramme benötigen keinerlei RAM und es wird auch nichts auf den Stack geschrieben. Ebenfalls bleiben die Register D0, D1, D2 und D3 unverändert. Benutzt werden ausschließlich die Register D4, D5, D6 und natürlich D7 (Trap #1). Den Kritikern, denen der 68008 zu wenig Register hat, möchte ich hiermit zeigen, daß 8–32 Bit Register auch 16–16 Bit Register sein können.

### Nun aber zum Programm selbst:

Die erweiterten Funktionen werden über Register D3 gesteuert. Die Funktionen sind dem Programm und den Beispielen zu entnehmen. Es sollte aber darauf geachtet werden, daß die Anzahl des Blinkens mittels z.B. MOVE #64,D3 vor dem Setzen einzelner Bits geschieht, da dieser Befehl die unteren 4 Bits auf Null setzt!!!!

Aufregendes passiert im Programm nicht. Als erstes wird die Textbreite und die Texthöhe aus D0 berechnet. Danach noch die Länge des Textes (Anzahl Buchstaben). Ausgehend davon, daß die kleinste Textbreite 6 Punkte (mit Zwischenraum) und die kleinste Texthöhe 8 Punkte benötigen, werden jeweils die Delta Rechts und Hoch in Punkten ermittelt.

Dann werden alle Optionen durchgegangen, ob Bits gesetzt sind. Um eine Inversdarstellung zu erreichen wird erst eine gefüllte Box an die Stelle des Textes gezeichnet und dann löschend in diese Box geschrieben.

Die Anzahl Blinken wird mit den zweiten 4 Bits von D3 in D16-er Schritten festgelegt (siehe Programm).

Die Blinkfrequenz liegt auf etwa einer Sekunde (50 x 20ms), kann aber beliebig geändert werden.

Gibt man in D1 einen Wert über 511 ein (z.B. 600), dann wird der angegebene Text mittig gesetzt.

Im übrigen funktioniert die Routine wie bei der Writeroutine des Grundprogramms.

```
*****
* STEUERUNG DER GDP : UNTERSTREICHEN,KASTEN,INVERS UND BLINKEN *
* RECHTSGENEIGT UND MITTIG SETZEN *
* BY DIETMAR ARNDS NOVEMBER 1986 *
*****
:
: IN REGISTER D3 WERDEN DIE INFORMATIONEN UEBERGEHEN
: BSET #0,D3 UNTERSTREICHEN AN
: BSET #1,D3 KASTEN AN
: BSET #2,D3 INVERS AN
: BSET #3,D3 RECHTSGENEIGT AN
:
: UEBER DEZIMAL 16 WIRD BLINKEN GESTEUERT
: IN 16 ER SCHRITTEN
: ALSO DEZIMAL 16 ENTSpricht 1 X BLINKEN
: DEZIMAL 32 ENTSpricht 2 X BLINKEN
: DEZIMAL 64 ENTSpricht 4 X BLINKEN
: USW.
: WENN IM D1 REGISTER EIN WERT UEBER 511 STEHT,WIRD DER
: TEXT MITTIG AUSGEGEBEN
TEXT:DC,B 'HIER NUN EINE AUSWAHL',0
GDP EQU $FFFFF70; BASISADRESSE GDP
:STARTADRESSE IST R
R:
CLR.L D0; ALLES AUF NULL
CLR.L D1
CLR.L D2
CLR.L D3
CLR.L D4
CLR.L D5
CLR.L D6
CLR.L D7
:
: HIER NUN EINIGE BEISPIELE
:
: 1.
MOVE.B # $11,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
MOVE #220,D2; POSITION HOCHWERT
LEA TEXT,A0
```

```
BSR WRITE
2.
CLR.L D3
MOVE.B # $22,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
MOVE #200,D2; POSITION HOCHWERT
BSET #0,D3; UNTERSTREICHEN AN
LEA TEXT,A0
BSR WRITE
3.
CLR.L D3
MOVE.B # $11,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
MOVE #180,D2; POSITION HOCHWERT
BSET #1,D3; KASTEN AN
LEA TEXT,A0
BSR WRITE
4.
CLR.L D3
MOVE.B # $22,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
MOVE #160,D2; POSITION HOCHWERT
BSET #2,D3; INVERS AN
LEA TEXT,A0
BSR WRITE
5.
CLR.L D3
MOVE.B # $22,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
MOVE #130,D2; POSITION HOCHWERT
BSET #2,D3; INVERS AN
BSET #3,D3; RECHTSGENEIGT
LEA TEXT,A0
BSR WRITE
6.
CLR.L D3
MOVE.B # $11,D0; TEXTHOEHE TEXTBREITE
MOVE #010,D1; POSITION RECHTSWERT
```

```

MOVE #100,D2;
MOVE #32,D3;
BSET #2,D3;
LEA TEXT,A0
BSR WRITE
;
7.
CLR.L D3
MOVE.B #S11,D0;
MOVE #600,D1;
MOVE #070,D2;
LEA TEXT,A0
BSR WRITE
RTS
; BERECHNUNG DER AUSGANGSDATEN *****
PARA:
LAENGE:
CLR.L D4;
CLR.L D5;
CLR.L D6
FOR1:
MOVE.B (A0)+,D4;
CMPI.B #0,D4;
BEQ.S NEXT1
ADDQ #1,D6;
BRA FOR1
NEXT1:
SUBQ #1,A0;
SUBA D6,A0;
BREITE:
CLR.L D4
MOVE.B D0,D4;
DIVU.B #S10,D4;
MOVE.B D4,D5;
MOVE.B D0,D4
HOEHE:
CLR D7
MOVE.B #S10,D7
MULU D5,D7;
SUB.B D7,D4;
MITTIG:
CMPI #S12,D1;
BMI ENDPARA;
CLR.L D7
MOVE.B D6,D7
MULU #6,D7;
MULU D5,D7;
MOVE #S11,D1;
SUB D7,D1
DIVU #2,D1;
ENDPARA:
RTS
SCHRIFTART:
BTST #3,D3;
BEQ ENDART
MOVE.B #4,GDP+2;
ENDART:
RTS
GERADE:
MOVE.B #0,GDP+2;
RTS
WARTE:
CLR.L D7
MOVE.B GDP,D7;
BTST #2,D7
BEQ.S WARTE
RTS
;
SEKUNDE:
SWAP D6;
CLR D6;
SWAP D0;
CLR D0;
MOVE.B #50,D6;
FOR5:
MOVE #1SYNC,D7;
TRAP #1
CMPI.B #0,D0
BEQ.S FOR5
DBRA D6,FOR5;
CLR D6;
SWAP D6;
CLR D0;
SWAP D0;
RTS
;
UNTERSTREICHEN *****
UNTER:
BTST #0,D3;
BEQ KASTEN;
BSR AUSGABE;
MOVE #1MOVETO,D7;
TRAP #1
CLR.L D7
MOVE.B D6,D7;
MULU.B #6,D7;
SUBQ #1,D7;
MULU D5,D7;
SWAP D6;
MOVE D7,D6;
ADD D6,D1;
MOVE #1DRAWTO,D7;
TRAP #1
SUB D6,D1;
CLR D6;
SWAP D6;
BRA ENDWRITE
;
KASTEN:
BTST #1,D3;
BEQ INVERS;
BSR AUSGABE;
CLR.L D7
MOVE.B D6,D7;
MULU #6,D7
MULU D5,D7
BTST #3,D3;
BEQ.S NEXT7;
MULU #8,D4;
ADD D4,D7;
DIVU #8,D4;
NEXT7:
ADDQ #2,D7;
SWAP D6
CLR D6
MOVE D7,D6
SUBQ #3,D1;
MOVE #1MOVETO,D7;
POSITION HOCHWERT
BLINKEN
INVERS AN
TEXTHOEHE TEXTBREITE
POSITION RECHTSWERT UEBER 511
POSITION HOCHWERT
ES WERDEN NUR REGISTER D4,D5 UND
D6 VERWENDET.
TEXT IN A0
WENN NULL DANN ENDE TEXT
IN D6 TEXTLAENGE
AUF LETZTEN BUCHSTABEN
WIEDER AUF ANFANGSWERT
TEXTBREITE TEXTHOEHE NACH D4
INTEGERDIVISION DURCH H10=TEXTBREITE
TEXTBREITE IN D5
D0-(TEXTBREITE*H10)=
TEXTHOEHE IN D4
IST D1 >511 DANN MITTIG SETZEN
SONST ENDE
D6(LAENGE TEXT)*6(BREITE EINES ZEICHENS)
DAS GANZE * TEXTBREITE
511 - LAENGE TEXT=REST
DIVIDIERT DURCH 2=RECHTSWERT TEXT
WENN GESETZT DANN RECHTSGENEIGT
BEFEHL RECHTSGENEIGT NACH GDP
BEFEHL GERADE NACH GDP
TESTEN OB GDP BESCHAEFIGT
ZEITSCHLEIFE ETWA 1 SEKUNDE
SICHERN D6
16 BIT LOESCHEN
SICHERN D0
16 BIT LOESCHEN
50 * 20 MS = 1 SEKUNDE
SYNCHRONIMPULS ALLE 20 MS
RUECKWAERTS ZAEHLEN BIS NULL
16 BIT LOESCHEN
UND AUSGANGSWERT HERSTELLEN
16 BIT LOESCHEN
UND AUSGANGSWERT HERSTELLEN
UNTERSTREICHEN GESETZT
NEIN DANN WEITER
TEXT AUSGEBEN
UNTEN LINKS
LAENGE TEXT
* ZEICHENLAENGE MIT ZWISCHENRAUM
- LETZTEN ZWISCHENRAUM
MULTIPLIZIERE MIT TEXTBREITE
SICHERN
DELTA Y IN D6
AUSGANGSRECHTSWERT DAZU
UND STRICH ZEICHNEN
AUSGANGSKOORDINATE D1 HERSTELLEN
16 BIT LOESCHEN
UND AUSGANGSWERT HERSTELLEN
KASTEN GESETZT
WENN NEIN DANN WEITER
TEXT AUSGEBEN
ERSTMAL BREITE DES TEXTES ERMITTELN
WENN RECHTSGENEIGT MUSS IN DER
BREITE NOCH ETWAS DAZU SONST WEITER
TEXTBREITE * B
IN D7 JETZT DELTA RECHTSWERT
D4 WIEDER AUF AUSGANGSWERT
EIN WENIG ABSTAND ZUM TEXT
AUCH HIER EIN WENIG ABSTAND
UNTEN LINKS

```

```

TRAP #1
ADD D6,D1
MOVE #1DRAWTO,D7;
TRAP #1
SWAP D5;
CLR D5;
CLR.L D7
MOVE.B D4,D7;
MULU #8,D7;
ADDQ #1,D7;
MOVE D7,D5
ADD D5,D2
MOVE #1DRAWTO,D7;
TRAP #1
SUB D6,D1;
MOVE #1DRAWTO,D7;
TRAP #1
SUB D5,D2;
MOVE #1DRAWTO,D7;
TRAP #1
ADDQ #3,D1;
CLR D5
CLR D6
SWAP D5
SWAP D6
BRA ENDWRITE
; INVERSARSTELLUNG *****
INVERS:
BTST #2,D3;
BEQ AUSGABE;
CLR.L D7
MOVE.B D6,D7;
MULU #6,D7
MULU D5,D7
BTST #3,D3;
BEQ.S NEXT8;
MULU #8,D4
ADD D4,D7
DIVU #8,D4
NEXT8:SUBQ #1,D1
ADD D1,D7
SWAP D1
CLR D1
MOVE D7,D1;
SWAP D6
CLR D6
CLR.L D7
MOVE.B D4,D7;
MULU #8,D7
ADDQ #1,D7
MOVE D7,D6;
SWAP D4
MOVE D2,D4;
SWAP D1
MOVE #1MOVETO,D7;
TRAP #1;
SWAP D1;
MOVE #1DRAWTO,D7
TRAP #1
ADDQ #1,D2
SUBQ #1,D6;
BEQ.S NEXT4
BRA FOR4
NEXT4:CLR D1
SWAP D1
ADDQ #1,D1;
MOVE D4,D2;
CLR D4
SWAP D4
CLR D6
SWAP D6
MOVE #1ERAPEN,D7;
TRAP #1;
BSR AUSGABE
CMPI.B #15,D3;
BGT ENDWRITE;
MOVE #1SETPEN,D7
TRAP #1
RTS
; AUSGABE EINES TEXTES AN GDP *****
AUSGABE:
SWAP D4
BSR WARTE;
MOVE #1MOVETO,D7
TRAP #1
MOVE.B D0,GDP+3;
FOR2:
MOVE.B (A0)+,D4;
CMPI.B #0,D4
BEQ NEXT2
BSR WARTE
MOVE.B D4,GDP
BRA FOR2
NEXT2:
CLR D4
SWAP D4
SUBQ #1,A0;
SUBA D6,A0
ENDWRITE:
RTS
; HAUPTPROGRAMM *****
WRITE:
BSR PARA
BSR SCHRIFTART
FOR6: BSR UNTER;
CMPI.B #15,D3;
BLE ENDE;
BSR SEKUNDE;
MOVE #1ERAPEN,D7;
TRAP #1
BSR UNTER;
BSR SEKUNDE;
MOVE #1SETPEN,D7;
TRAP #1
SUB #S10,D3;
BRA FOR6;
ENDE:
BSR GERADE
RTS
LINIE NACH UNTEN RECHTS
SICHERN
16 BIT LOESCHEN
TEXTHOEHE NACH D7
* 8 PUNKTE
EIN WENIG ABSTAND DAZU
LINIE NACH OBEN RECHTS
AUSGANGSKOORDINATE IN D1
LINIE NACH OBEN LINKS
AUSGANGSKOORDINATE IN D2
LINIE ZUM AUSGANGSPUNKT
ORIGINALKOORDINATE HERSTELLEN
INVERS GESETZT
WENN NEIN DANN WEITER
TEXTBREITE ERMITTELN
WENN RECHTSGENEIGT DANN ETWAS
ADDIEREN SONST WEITER
EFFEKTIVKOORDINATE ENDWERT AUCH IN D1
TEXTHOEHE ERMITTELN
DELTA X IN D6
STARTKOORDINATE D2 SICHERN
ES WERDEN NUN LINIEN GEZOGEN
BEIDE KOORDINATEN RECHTSWERT IN D1
HOEHE IN D6
D6 RUNTERZAEHLEN BIS NULL
WIEDER AUF STARTWERT
WIEDER AUF STARTWERT
IN DIESE BOX NUN SCHREIBEN
WOBEI PIN AUF LOESCHEN STEHT
BLINKEN GESETZT
WENN JA DANN PIN AUF LOESCHEN
GDP BEREIT
TEXTGROESSE AN GDP
TEXT IN SCHLEIFE AUSGEBEN
A0 AUF AUSGANGSWERT
D3 AUSWERTEN UND TEXT AUSGEBEN
BLINKEN GESETZT
WENN UEBER 15 DANN GESETZT SONST ENDE DER AUSGABE
EINE SEKUNDE WARTEN
TEXT LOESCHEN
TEXT LOESCHEND AUSGEBEN
EINE SEKUNDE WARTEN
TEXT SCHREIBEN
H10 ABZIEHEN EINE SEKUNDE WENIGER
UND TEXT SCHREIBEND AUSGEBEN

```

# Nie mehr auf den Drucker warten

Druckerausgabe interruptgesteuert für 68008-Grundprogramm

von Norbert Gatz,  
Langenfelde 2, 2300 Ottendorf

## Programmbeschreibung:

Da man in der Softwareentwicklung nicht umhin kommt, öfter etwas auszudrucken, fand ich es sehr lästig, mit der Arbeit zu warten, bis der Drucker alle Daten übernommen hat. Der Preis für einen Druckpuffer war mir zu hoch, so daß ich mir eine andere Möglichkeit überlegte: man mußte die Druckerausgabe doch auch interruptgesteuert durchführen können. Nach einigen Überlegungen machte ich mich also ans Werk. Der 68000/68008-Prozessor hat insgesamt 3 Interruptleitungen, die durch das Statusregister maskiert werden können. Somit sind 7 verschiedene Interrupts möglich, denn die Bitkombination 111 in den Interruptregistern bedeutet keinen Interrupt.

Ich entschloß mich, die INT-Leitung zu benutzen, da diese auch auf der IOE-Karte, die ja gleichzeitig die Centronics-Schnittstelle darstellt, vorhanden ist. Für die Arbeit mit dem Grundprogramm genügt dann eine einfache Verbindung der INT-Leitung mit Pin 18 des 74LS245, der mit der BUSY-Leitung des Druckers verbunden ist. Soweit die Hardwareseite.

Das Programm selbst ist relokativ geschrieben, läßt sich also beliebig im Speicher positionieren. Bevor die eigentliche Interruptroutine (DRUCKINTERRUPT) ausgelöst wird, ist die Routine (DRUCK) aufzurufen. In ihr wird zunächst die Startadresse der Interruptroutine auf den dafür im Grundprogramm vorgesehenen Speicherplatz (Adr. 0006h-000Ch nach Ende des Grundprogramms) geschrieben. Zunächst wird auf den ersten 2 Bytes der Maschinencode für den JMP-Befehl eingetragen und auf den anschließenden Bytes die Startadresse relativ zum Programmzähler. Dann wird der aktuelle Textstart in die Adresse MERKADR geschrieben, die zum Retter der aktuellen Druckposition verwendet wird. Jetzt muß nur noch der Interrupt aktiviert werden, was dadurch geschieht, daß in der Interruptmaske ein Bit auf 0 gesetzt wird. Sobald der Drucker nun ein READY-Signal liefert, d.h., sobald die INT-Leitung auf logisch 0 liegt, wird der Interrupt ausgelöst und die Routine (DRUCKINTERRUPT) wird durchlaufen. Hier ist es wichtig, vorher alle betroffenen Register zu retten, da nach dem Verlassen der Interruptroutine der gleiche Zustand wie vor dem Eintreten herrschen muß. Das Adreßregister A0 wird dann mit der aktuellen Druckposition geladen und das

nächste Zeichen nach D0 gebracht, wobei A0 gleichzeitig um 1 inkrementiert wird. Sollte es sich bei dem transportierten Byte nicht um das Nullbyte handeln (Texte werden mit diesem Byte abgeschlossen), so wird die aktuelle Druckposition wieder gerettet und das in D0 stehende Zeichen an den Drucker gesendet. Die geretteten Register werden zurückgeladen und die Interruptroutine verlassen, wobei durch den RTE-Befehl auch das Statusregister zurückgeschrieben wird. Sollte ein Textende erkannt sein, so wird durch den Befehl ORI#0700,SR der Interrupt gesperrt, d.h. selbst wenn jetzt die INT-Leitung auf logisch 0 liegt, wird der Interrupt nicht durchgelassen. Man muß jetzt nur noch dafür sorgen, daß der neue Status auch außerhalb der Interruptroutine gültig ist – denn der RTE-Befehl stellt ja den Status vor dem Eintritt in die Interruptroutine

wieder her –, deshalb muß der neue Status auf den Inhalt des Stacks A7 geschrieben werden, denn der Stack zeigt hier gerade auf das gerettete Statusregister.

Will man das Programm in der so beschriebenen Form verwenden, so darf man natürlich während des Ausdrucks nicht den selben Text editieren. Möchte man auch dies, so muß man im Programm (DRUCK) noch eine Verschieberoutine einbauen, die den aktuellen Text in einen freien Speicherbereich legt, und in MERKADR muß dann der Beginn dieses freien Speicherbereichs liegen. Da aber nicht jeder so viel Speicherraum frei hat, ist die oben angegebene Möglichkeit die günstigste, um „parallel“ zur Druckerausgabe weiterarbeiten zu können. Assemblieren und Programmausführung sind auf jeden Fall möglich.

```
30.08.86 Drucker interruptgesteuert
```

```
DRG $0  
OFFSET $1A000
```

```
*****  
*                                     *  
*          D R U C K E R A U S G A B E   *  
*          interruptgesteuert         *  
*          für 68008                   *  
*                                     *  
*          (C) 1986 Norbert Gatz       *  
*          Langenfelde 2               *  
*          2300 Ottendorf              *  
*                                     *  
*****
```

```
DRUCK:  
MOVEM.L D0/D7/A0,-(A7)      * betroffene Register retten  
MOVE #4EF9,6(A5)           * Maschinencode für JMP  
LEA DRUCKINTERRUPT(PC),A0  * Umlenkung des INT - TRAPS  
MOVE.L A0,8(A5)            * auf DRUCKINTERRUPT  
MOVE #1,GETSTX,D7          * aktuellen Textstart holen  
JSR #420-$B000(A5)         * entspricht TRAP #1  
LEA MERKADR(PC),A0         * Adresse zur Rettung von A0  
MOVE.L D0,(A0)            * aktueller Textstart jetzt in MERKADR  
ANDI #$2FF,SR             * Interrupt maskieren  
MOVEM.L (A7)+,D0/D7/A0    * betroffene Register zurückholen  
RTS
```

```
DRUCKINTERRUPT:  
MOVEM.L D0-D7/A0/A1/A6,-(A7) * betroffene Register retten  
MOVEA.L MERKADR(PC),A0      * A0 auf aktuelle Textposition  
MOVE.B (A0)+,D0            * in D0 aktuelles Zeichen  
BEQ.S DRUCKENDE            * Textende erreicht?  
LEA MERKADR(PC),A1         * aktuelle Textposition retten  
MOVE.L A0,(A1)             * Ausgabe eines Zeichens auf Drucker  
MOVE #1,D7                 * entspricht TRAP #1  
JSR #420-$B000(A5)
```

```
DRUCKENDE:  
MOVEM.L (A7)+,D0-D7/A0/A1/A6 * betroffene Register zurückholen  
BNE.S KEINSTATUSWECHSEL    * Sprung, falls noch kein Textende  
ORI #0700,SR               * Interrupt sperren  
MOVE SR,(A7)              * an alte SR - Position schreiben  
KEINSTATUSWECHSEL:  
RTE                         * Rücksprung aus Exception
```

```
MERKADR:  
DS.L 1  
END
```

## Verbesserter und schnellerer Figurbefehl mit X- und Y-Vergrößerung

von Ralph Dombrowski,  
Große Deichstr. 33, 2208 Glückstadt,  
Telefon (04124) 2520

Sehr geehrter Leser, wer schon einmal mit dem Figurbefehl des 680xx-Grundprogramms gearbeitet hat, wird sicherlich gemerkt haben, daß dieser nicht besonders komfortabel ist. Die Vergrößerung kann nur gemeinsam für die X- und Y-Vergrößerung geändert werden, außerdem nimmt die Geschwindigkeit der Figurausgabe bei Vergrößerungen, die größer als 3 sind, extrem schnell ab. Mein Figurbefehl arbeitet deshalb nicht mit

den Kurzvektoren, die beim alten Figurbefehl verwendet werden, sondern mit dem Vektorenbefehl der GDP-Karte, der auch für die Drawtoroutine des Grundprogramms verwendet wird. Mein Befehl wird genauso aufgerufen wie der alte Figurbefehl. Der Unterschied besteht nur im Register D0. Dort wird die Vergröße-

rung für die X-Richtung in den unteren 8 Bits abgelegt, die Y-Komponente wird in den Bits 8 - 15 abgelegt. Es ist jeweils eine Vergrößerung von 0 bis 255 möglich.

Beispiel:

```
move.w #1020,d0
X-Vergrößerungen = $20=32
```

Y-Vergrößerung = \$10=16  
Um zu demonstrieren, daß der neue Figurbefehl wirklich viel schneller ist, habe ich noch ein kleines Beispielprogramm beigefügt, das recht gut den Geschwindigkeitsunterschied zeigt. Mit diesem Befehl kann man jetzt auch große Bilder über den Bildschirm bewegen.

```
***** RD-Figurbefehl *****
cpu equ 1          * 68008 CPU (Bei 68000 und 68020 ändern)

gdp equ $ff70     * Adresse GDP-Port

oldsize: dc.w 0   * Größe der alten Figur
oldadr: dc.l 0    * Adresse der letzten Figur
oldx: dc.w 0      * Alte X-Koordinate
oldy: dc.w 0      * Alte Y-Koordinate

setfig:          * Figur festsetzen
clr oldsize     * Keine alte Figur gültig
rts

figur:          * Figurbefehl (d7 wird zerstört)
movem.l d0-d2/a0,-(a7) * Register abspeichern
move oldsize,d0    * Alte Größe nach d0
beq.s figur1      * Wenn 0, dann ist keine alte zu löschende Figur da
moveq #!erapen,d7 * Nicht 0, dann alte Figur löschen (Löschmode ein)
trap #1
move oldx,d1      * X-Koordinate der alten Figur laden
move oldy,d2      * Y-Koordinate der alten Figur laden
movea.l oldadr,a0 * Adresse der alten Figur nach a0
bsr.s figset      * Figur löschen

figur1:
movem.l (a7)+,d0-d2/a0 * Register zurück
move d0,oldsize     * Neue Größe der Figur abspeichern
beq.s figur2        * Wenn Size=0, dann Figur nicht ausgeben
move d1,oldx       * Neue Figurdaten werden abgespeichert für nächstes Mal
move d2,oldy
move.l a0,oldadr   * Jetzt Schreibmode ein zum Zeichnen der neuen Figur
moveq #!setpen,d7 * Figur ausgeben
trap #1
bra.s figset       * Figur ausgeben (d7 wird zerstört)
figset:
movem.l d0/a0/a1,-(a7) * Register abspeichern
moveq #!moveto,d7    * GDP positionieren

trap #1
move.b d0,(gdp+5)*cpu.w * Vergrößerung X ins GDP-Register 5
asr.w #8,d0
move.b d0,(gdp+7)*cpu.w * Vergrößerung Y ins GDP-Register 7
lea figtab(pc),a1
figsch:
move.b (a0)+,d0      * Zeichen nach d0
ext.w d0
move.b 0(a1,d0.w),d0 * Befehl für GDP nach d0 holen
beq.s figsetf        * Wenn 0, dann Ende der Ausgabe
btst.b #2,gdp*cpu.w * Warten bis GDP fertig ist
beq.s *-6
move.b d0,gdp*cpu.w * Befehl an GDP ausgeben
bra.s figsch         * Schleife wiederholen
figsetf:
movem.l (a7)+,d0/a0/a1 * Register zurück
```

```
rts
figtab:
dc.b %00010000 * Befehlstabelle (Bit 4 gibt an,da0 es sich um einen
dc.b %00010001 * Vektor handelt / Die bits 0-3 geben die Richtung an)
dc.b %00010010 * rechts
dc.b %00010011 * rechts oben
dc.b %00010100 * oben
dc.b %00010101 * links oben
dc.b %00010110 * links
dc.b %00010111 * links unten
dc.b %00010100 * unten
dc.b %00010101 * rechts unten
dc.b %00010110 * neben
dc.b %00010111 * senken
dc.b %00010000 * Ende
ds 0

***** Demonstration für den neuen Figurbefehl *****
* Zuerst wird der neue Figurbefehl gezeigt, dann zum Vergleich der Figurbefehl
* aus dem Grundprogramm

quadrat:
dc.b 0,0,2,4,4,6,10 * Richtungen für die GDP-Karte
ds 0 * Codierung wie beim Grundprogramm-Figurbefehl

start:
lea quadrat,a0      * Adresse der Figur nach a0
lea figur,a2        * Zuerst RD-Figurbefehl
lea setfig,a3       * Und RD-Setfig
moveq #1,d6         * Zwei Durchgänge
loop0:
move #256,d5        * Anzahl der Schleifen
loop1:
move #256,d1        * Berechnung der Koordinaten für die Figur
sub d5,d1           * In d1 jetzt Nummer des Durchlaufs
neg d1              * Jetzt von 256 abziehen
add #256,d1
move d1,d2          * X-Koordinate = Y-Koordinate
lsl #1,d2           * Y-Koordinate durch 2 teilen wegen Bildschirmaufbau
jsr (a2)            * Figurbefehl aufrufen
jsr (a3)            * Setfig aufrufen
add #0101,d0        * Jetzt die Figur vergrößern
dbra d5,loop1       * Nächster Schleifendurchlauf
move #ffff,d0       * Größe der über den Bildschirm bewegten Figur
clr d2              * Y-Koordinate=0
move #1030,d6       * Anzahl der Schleifen
loop2:
move d6,d1          * X-Koordinate
sub #512,d1
jsr (a2)            * Figurbefehl aufrufen
dbra d6,loop2
lea $figur,a2       * Beim Zweiten Durchlauf Figurbefehl aus dem Grundprogr.
lea $setfig,a3      * zum Vergleich
jsr $clr            * Bildschirm löschen
dbra d6,loop0       * Zweiter Durchlauf
rts
end
```

# Makros für 680xx-Programmaufrufe

## Anleitung für Macro-Erweiterung

von Guido Scheil  
Immenburg 56, 2000 Hamburg 62

Vor einem Unterprogrammaufruf ist es meistens nötig, sämtliche Daten in die entsprechenden Register der CPU zu moven. Um Unterprogrammaufrufe einfacher zu machen und Tipparbeit einzusparen, schrieb ich dieses Programm.

Es ist damit möglich, beim Aufruf von RDK - und normalen Unterprogrammen - die move's wegzulassen. Ferner entfällt das jsr/bsr und das Definieren von Text- und Datenblöcken. Nach dem Start über die Bibliothek legt die Macro-Erweiterung einen Asstext hinter dem eingegebenen Text an und kopiert diesen in den Asstext. Dabei setzt die Erweiterung an den entsprechenden Stellen den 68008 Syntax ein. Anschließend wird der Asstext mit dem RDK-Assembler assembliert und

der Textstart auf den Anfangstext zurückgesetzt. Beim Kopieren werden Kommentare, die mit einem Semikolon beginnen, aus Speicherplatz- und Geschwindigkeitsgründen weggelassen. Kommentare hinter einem Stern bleiben.

Der Unterprogrammaufruf beginnt mit einem Pfeil nach oben. Anschließend folgt der Name der Unterroutine. Wenn der Anfangsbuchstabe groß oder ein @ ist, wird das Unterprogramm nachher mit jsr angesprungen, sonst mit bsr.

Falls keine Daten übergeben werden sollen, ist das alles.

Wenn Daten übergeben werden sollen, wird eine eckige Klammer auf [ direkt hinter den Namen gesetzt. Bei den Daten- und Adressregistern werden die folgen-

den Daten der Reihenfolge gemäß in die Register geladen. Also z.B. die erste Dateninformation nach d0, die zweite nach d1 usw. ... Da es aber durchaus vorkommt, daß ein Register überhaupt nicht geladen werden soll, wird dies bei Datenregistern einfach mit einem ~ und bei Adressregistern mit einem &~ übersprungen. Innerhalb einer Datenregisterübergabe sind fast alle move-Syntaxe erlaubt, die normal auf ein Datenregister angewandt werden dürfen (siehe Beispiel).

Zusätzlich sollte am Ende der Dateninformation der Adressmode vorhanden sein, sonst wird Information automatisch .w geladen. Bei Adressregistern muß am Anfang der Information ein & stehen, damit die Erweiterung weiß, daß es sich

um Adressregister handelt. Es ist so möglich, eine Adresse z.B. \$9C00 oder ein Label .l in ein Adressregister zu laden (siehe Beispiel).

Texte und Datenblöcke werden von der Ergänzung in ds Blöcke umgewandelt und übertragen. Wobei die Texte immer .b sind, während bei den Datenblöcken im Macro am Ende der Information immer

der Adressmode angegeben werden muß. Texte müssen mit einem " beginnen und aufhören. Daten beginnen mit einem ! und werden mit dem Adressmode beendet. Die Daten im Datenblock werden mit Kommas getrennt. Die Adresse der ds Zeile wird automatisch mit einem lea Befehl in das Adressregister geladen, das der Reihenfolge entspricht.

Das Macro wird mit einer eckigen Klammer zu ] abgeschlossen. In einem Macroaufruf können so viele Informationen vorkommen, wie der 68008 Register hat.

Die Art der Informationen spielt dabei keine Rolle. Alle Informationen in einem Macro-Aufruf werden durch Kommas getrennt.

Beispiele für Macroaufrufe:

```
x equ 100
y equ 100
print equ $f000

^@moveto[~,#0.w,#0.w]
^@drawto[~,#512,#256]
^@write[#21.b,x,y,"Beispiele"]
^@getflop[~,~,~,#1]
^@floppy[#a000,~,#,2,#0]
^print[%,! 27,64,10,10,13,27,'A',1.b]

end
```

Und so sieht es assembliert aus:

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

```
= 00000064 X EQU 100
= 00000064 Y EQU 100
= 0000F000 PRINT EQU $F000
009C00
009C00 323C 0000 MOVE.W #0,D1
009C04 343C 0000 MOVE.W #0,D2
009C08 4EB9 00000EE2 JSR $MOVETO
009C0E 323C 0200 MOVE.W #512,D1
009C12 343C 0100 MOVE.W #256,D2
009C16 4EB9 00000F4C JSR $DRAWTO
009C1C 103C 0021 MOVE.B #21,D0
009C20 3239 00000064 MOVE.W X,D1
009C26 3439 00000064 MOVE.W Y,D2
009C2C 600A BRA.S LAB0
009C2E 42656973706965 DATEN0: DC.B 'Beispiele',0
009C35 6C6500
009C38 DS 0
009C3B LAB0:
009C3B 41FA FFF4 LEA DATEN0(PC),A0
009C3C 4EB9 00001386 JSR $WRITE
009C42 383C 0001 MOVE.W #1,D4
009C46 4EB9 000049C6 JSR $GETFLOP
009C4C 41F9 0000A000 LEA #A000,A0
009C52 323C 0001 MOVE.W #1,D1
009C56 343C 0002 MOVE.W #2,D2
009C5A 363C 0000 MOVE.W #0,D3
009C5E 4EB9 000049BC JSR $FLOPPY
009C64 600B BRA.S LAB1
009C66 1B 40 0A 0A 0D DATEN1: DC.B 27,64,10,10,13,27,'A',1
009C6B 1B 4101
009C6E DS 0
009C6E LAB1:
009C6E 43FA FFF6 LEA DATEN1(PC),A1
009C72 6100 53BC BSR PRINT
009C76 END
009000 Ende-Symboltabelle
```

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 1

```
009C00 *****
009C00 *** MACRO - ERGÄNZUNG VERSION 3.0 ***
009C00 *** FÜR RDK - ASSEMBLER ***
009C00 *** (C) 30.5.1986 GUIDO SCHEIL ***
009C00 *****
040000 HEAD: ORG $40000
040000 55AA0100 DC.L $55AA0100
040004 404143524F4153 DC.B 'MACROASS'
040008 53
04000C 000402B8 DC.L START
040010 00036952 DC.L ENDE-HEAD
040014 00000000 DC.L 0,0,0,0,0
040018 00000000
04001C 00000000
040020 00000000
040024 00000000
040028
04002E 0000003E TITENDMARKE EQU #003E ; ETITIT BEI GRUNDFRS. 4.2 AB ADR #0000
040028 MACRO EQU #5E ; ENTHALT DIE ENDADRESSE DES TEXTES)
= 0000005E MACRO EQU #5E ; PFEIL NACH OBEN
= 0000005B KLAMMAUF EQU #5B ; KLAMMER AUF
= 0000005D KLAMMUN EQU #5D ; KLAMMER ZU
= 00000027 HOCHKM EQU #27 ; HOCHKOMMA
= 0000007E LEERFEG EQU #7E ; REGISTER ÜBERSPRINGEN
04002E
```

```
040028 00000000 TITANFANG: DC.L 0 ; TEXTANFANG
04002C 00000000 ASSTITANF: DC.L 0 ; ASS TEXTANFANG
040030 0000 LABELC: DC.W 0 ; LABELZÄHLER
040032 0000 ADRBUF: DS.B 16 ; ADRESSBUFFER
040042 INHBUF: DS.B 64 ; INHALTS- / INFORMATIONSBUFFER
0400B2 LABBUF: DS.B 16 ; LABELNUMMERBUFFER
040092 6D6F76652E2020 MVE: DC.B 'move.',0
040099 00
04009A 2C6400 MVE2: DC.B ',d',0
04009B 6C65612000 MVA: DC.B 'lea',0
0400A2 2C6100 MVA2: DC.B ',a',0
0400A5 6273722000 SUB1: DC.B 'bsr',0
0400AA 6A73722000 SUB2: DC.B 'jsr',0
0400AF 6272612E73206C MARKE: DC.B 'bra.s lab',0
0400B6 616200
0400E9 3A2064632E68220 DATA: DC.B ': dc.b ',0
0400C0 00
0400C1 2C300A 00 6473 EOFDATA: DC.B ',0',10,13,'ds 0',10,13,'lab',0
0400C7 20300A 00 6C61
0400CD 6200
0400CF 3A0A 00 6C6561 EOFDATA2: DC.B ': ,10,13,'lea daten',0
0400D5 20466174656E00
0400DC 287063292C6100 EOFDATA3: DC.B '(pc),a',0
0400E3 646174656E00 LABEL: DC.B 'daten',0
0400E9 0A 0A 00 2020 INFO1: DC.B 10,10,13,' Textlänge : ',0
0400EE 202020202020205A
0400F5 6578746CFB6E67
0400FC 6529292020202020
04103 202020203A2020
0410A 202000
0410E 20427974657300 INFO2: DC.B ' Bytes',10,13,' Ass Textlänge : ',0
040114 00 2020202020
04011A 20202041737320
040121 546570746CFB6E
04012B 67652020202020
04012F 203A202020202000
040136 20427974657300 INFO3: DC.B ' Bytes',0
```

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 2

```
04013D
04013D 0A 0A 00 2A2A ERROR1: DC.B 10,10,13,'** Macro nicht beendet **',10,13,0
040142 204D6163726F20
040149 6E696368742062
040150 65656E64657420
040157 2A2A0A 00 00
04015C 0A 0A 00 2A2A ERROR2: DC.B 10,10,13,'** Es gibt nur 8 Daten- und Adrregister ! **',10
040161 20457320676962
040168 74206E75722038
04016F 20446174656E20
040176 20756E6420A164
04017B 72726567697374
040184 65722021202A2A
04018B 0A
04019C 00 00 DC.B 13,0
0401BE 0A 0A 00 2A2A ERROR3: DC.B 10,10,13,'** Syntax Fehler im Datenblock **',10,13,0
040193 2053796E746178
04019A 204665686C6572
0401A1 20696D20446174
0401A8 656E26C6F6368
0401AF 202A2A0A 00 00
0401B5 00 DS 0
0401B6
0401B6 41FA FF31 INFORM: LEA INFO1(PC),A0 ; GIBT TEXT- UND ASSTEXTLÄNGE AUS.
0401BA 6100 0054 BSR PRINT
0401BE 2039 0000003E MOVE.L TITENDMARKE,D0 ; BRECHUNG DER TEXTLÄNGE
0401C4 90B9 00040028 SUB.L TITANFANG,D0
0401CA 41FA FE86 LEA LABBUF(PC),A0
0401CE 4EB9 000015AC JSR $PRINTB
0401D4 41FA FEAC LEA LABBUF(PC),A0 ; AUSGABE TEXTLÄNGE
0401D8 6100 0036 BSR PRINT
0401DC 41FA FF2F LEA INFO2(PC),A0
0401E0 6100 002E BSR PRINT
0401E4 200C MOVE.L A4,D0 ; BERECHNUNG DER ASSTEXTLÄNGE
0401E6 90B9 0004002C SUB.L ASSTITANF,D0
0401EC 41FA FE94 LEA LABBUF(PC),A0
0401F0 4EB9 000015AC JSR $PRINTB
0401F6 41FA FE8A LEA LABBUF(PC),A0 ; AUSGABE ASSTEXTLÄNGE
0401FA 6100 0014 BSR PRINT
0401FE 41FA FF36 LEA INFO3(PC),A0
040202 6100 000C BSR PRINT
040206 4E75 RTS
040208
04020B 180B UEBERTRAGUNG: MOVE.B (A0)+(A4)+ ; ÜBERTRÄGT ZEICHENKETTEN IN DEN ASS
04020A 4A10 TST.B (A0) ; TEXT. A0 ADRESSE, 00 ENDEKENNUNG.
04020C 66FA BNE.S UEBERTRAGUNG
04020E 4E75 RTS
040210
040216 101B PRINT: MOVE.B (A0)+,D0 ; SCHREIBT EINEN ZEICHENKETTE
```

```

040212 6708      BEG.S PRINT1      ; AUF DEN BILDSCHIRM
040214 4EB9 0000AEO JSR 3C02          ; A0 = ADR.
04021A 60FA      BRA.S PRINT
04021C 4E75      PRINT1:          RTS
04021E
04021E 41FA FE22    GETINHALT: LEA INHBUF(PC),A0 ; ÜBERTRAGT DIE ZEICHEN BIS ZUM NÄCHSTEN
040222 0C16 0021    CMP1.B #'1',(A6) ; DATENBLOCK ? ( EINE INFORMATION )
040224 6724      BEG.S GETIN2
040228 100E      GETINI: MOVE.B (A6)+,(A0)+ ; KOMMA ODER MACRO ENDE IN DEN INHBUF
04022A 6700 0112    BEG AUSW1B      ; TEXTENDE ?
04022E 0C16 005D    CMP1.B #KLANZU,(A6) ; KLAMMERZU (MACRO ENDE) ?
040232 6710      BEG.S GETIN1B
040234 0C16 002C    CMP1.B #'',(A6) ; KOMMA ?
040238 66EE      BNE.S GETINI
04023A 529E      ADDU.L #1,A6
    
```

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 3

```

04023C 4210      CLR.B (A0)        ; JE NACH ABRUCH FLAG SETZEN.
04023E 1A3C 0001    MOVE.B #1,D5     ; BEI KOMMA
040242 4E75      RTS
040244 4210      GETIN1B: CLR.B (A0)        ; BEI KLAMMERZU
040246 529E      ADDU.L #1,A6
040248 4205      CLR.B D5
04024A 4E75      RTS
04024C 101E      GETIN2: MOVE.B (A6)+,D0 ; SONDERBEHANDLUNG DATENBLOCK
04024E 6700 00EE    BEG AUSW1B
040252 0C00 005D    CMP.B #KLANZU,D0 ; KLAMMERZU ?
040254 6728      BEG.S GETIN4
040258 0C00 000D    CMP.B #D,D0      ; LF ?
04025C 6722      BEG.S GETIN4
04025E 0C00 002E    CMP.B #'',(D0) ; ENDE DES BOLCKS ?
040262 6704      BEG.S GETIN3
040264 10C0      MOVE.B D0,(A0)+
040266 60E4      BRA.S GETIN2
040268 10C0      GETIN3: MOVE.B D0,(A0)+
04026A 100E      MOVE.B (A6)+,(A0)+
04026C 0C1E 005D    CMP.B #KLANZU,(A6)+
040270 6606      BNE.S GETIN3B
040272 4210      CLR.B (A0)
040274 4205      CLR.B D5
040276 4E75      RTS
040278 4210      GETIN3B: CLR.B (A0)
04027A 1A3C 0001    MOVE.B #1,D5
04027E 4E75      RTS
040280 41FA FFOC    GETIN4: LEA ERROR3(PC),A0 ; SYNTAX FEHLER AUFGETRETEN
040284 6100 FFB2    BSR UEBERTRAGUNG
040288 598F      ADDU.L #4,A7
04028A 6000 0070    BRA WDH2
04028E
04028E 5204      ADRPL: ADDU.B #1,D4 ; ERHOHT DEN ADREGISTERCOUNTER UM 1
040290 0C04 0009    CMP.B #9,D4      ; UND PRÜFT AUF ÜBERLAUF
040294 6616      BNE.S ADRPL2
040296 41FA FECA    ADRPL1: LEA ERROR2(PC),A0
04029A 6100 FF6C    BSR UEBERTRAGUNG
04029E 598F      ADDU.L #4,A7
0402A0 6000 005A    BRA WDH2
0402A4 5203      DATPL: ADDU.B #1,D3
0402A6 0C04 0009    CMP.B #9,D4
0402AA 67EA      BEG.S ADRPL1
0402AC 4E75      ADRPL2: RTS
0402AE
0402AE 18FC 000A    RETURN: MOVE.B #10,(A4)+ ; LF
0402B2 18FC 000D    MOVE.B #13,(A4)+
0402B6 4E75      RTS
0402B8
0402B8 4EB9 000030E2 START: JSR #GETSTX ; TEXTADR. ERMITTELN
0402BE 23C0 00040028 MOVE.L D0,TXTANFANG
0402C4 2C40      MOVEA.L D0,A6
0402C6 2879 0000803E MOVEA.L TITANMARKE,A4 ; TEXTENDADR. HOLEN
0402CC 09FC 00000100 ADDA.L #100,A4 ; ASS TEXTSTART BERECHNEN
0402D2 23CC 0004002C MOVE.L A4,ASSTXTANF
0402D8 4279 00040030 CLR LABELC ; LABELZÄHLER LÖSCHEN
0402DE 101E      WDH1: MOVE.B (A6)+,D0 ; ÜBERTRAGUNGSSCHLEIFE
0402E0 671A      BEG.S WDH2
0402E2 0C00 005E    CMP.B #MACRO,D0 ; MACRO ?
0402E6 673A      BEG.S AUSWERTUNG
0402EB 0C00 001B    CMP.B #'',(D0) ; SEMICOLON ?
0402EC 660A      BNE.S WDH1C
0402EE 0C1E 000D    WDH1B: CMP1.B #D,(A6)+ ; BEI SEMICOLON DEN REST DER ZEILE
0402F2 66FA      BNE.S WDH1B ; ÜBERSPRINGEN
0402F4 539E      SUBU.L #1,A6
    
```

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 4

```

0402F6 60E6      BRA.S WDH1
0402F8 18C0      WDH1C: MOVE.B D0,(A4)+
0402FA 60E2      BRA.S WDH1
0402FC 4214      WDH2: CLR.B (A4)
0402FE 2039 0004002C MOVE.L ASSTXTANF,D0 ; ASS TEXTSTART SETZEN
040304 4EB9 000030EB JSR SPUTSTX
04030A 4EB9 0000653A JSR #ASSEMBLIEREN ; ASSEMBLIEREN
040310 2039 00040028 MOVE.L TITANFANG,D0 ; TITANFANG AUF ALTEN TEXT
040316 4EB9 000030EB JSR SPUTSTX
04031C 6100 FE90    BSR INFORM
040320 4E75      RTS
040322
040322 41FA FDOE    AUSWERTUNG: LEA ADRBUF(PC),A0 ; ÜBERTRAGUNG DES LABELS IN DEN
040326 101E      AUSW1: MOVE.B (A6)+,D0 ; ADRBUF
040328 0C00 005B    CMP.B #KLANHAUF,D0 ; ENDE DES LABELS :
04032C 671E      BEG.S AUSW2 ; KLAMMERZU ?
04032E 0C00 000A    CMP.B #A,D0 ; RETURN ?
040332 6742      BEG.S AUSW3B
040334 0C00 0020    CMP.B #'',(D0) ; SPACE ?
040338 673C      BEG.S AUSW3B
    
```

```

04033A 10C0      MOVE.B D0,(A0)+
04033C 66E8      BNE.S AUSW1
04033E 41FA FDFD    AUSW1B: LEA ERROR1(PC),A0 ; ERROR NEBEN TEXTENDE
040342 6100 FECA    BSR UEBERTRAGUNG
040346 598F      ADDU.L #4,A7
040348 6000 FFB2    BRA WDH2
04034C 4210      AUSW2: CLR.B (A0)
04034E 4203      CLR.B D3 ; DATENREGISTERZÄHLER
040350 4204      CLR.B D4 ; ADREGISTERZÄHLER
040352 6100 FECA    AUSW3: MOVE.B INHBUF,D0 ; LADEN DES INHALTS IN DEN INHBUF
040356 1039 00040042 CMP.B #'',(D0) ; ART DER INFORMATION:
04035C 0C00 0022    BEG.S TEXTAUS ; TEXT ?
040360 6742      CMP.B #'',(D0) ; DATEN ?
040362 0C00 0021    BEG DATAUS
040366 6700 0108    CMP.B #'',(D0) ; ADREGISTER ?
04036A 0C00 0026    BEG ADRAUS
04036E 6700 019A    BRA REGAUS ; DATENREGISTER
040372 6000 012C    CLR.B (A0)
040376 4210      AUSW3B: BTST #5,ADRBUF ; GROßBUCHST.5 = JSR
040378 0B39 0005    AUSW4:
04037C 00040032    BEG.S AUSW5 ; KLEINBUCHST = BSR
040380 670C      LEA SUB1(PC),A0 ; BSR
040382 41FA F021    JSR UEBERTRAGUNG
040386 4EB9 00040208 BRA.S AUSW6
04038C 600A      LEA SUB2(PC),A0 ; JSR
040392 4EB9 00040208 JSR UEBERTRAGUNG
040398 41FA FC98    LEA ADRBUF(PC),A0
04039C 6100 FE6A    BSR UEBERTRAGUNG ; UNTERPPG. NAME
0403A0 6000 FF3C    BRA WDH1
0403A4
0403A4 13FC 0062    TEXTAUS: MOVE.B #' ',DATA+5
0403AB 0004009E      MOVE.B #' ',E0FDATA
0403AC 13FC 002C      MOVE.B #' ',E0FDATA+1
0403B0 000400C1      MOVE.B #'0 ',E0FDATA+1
0403B4 13FC 0030
0403B8 000400C2
0403BC 41FA FCB4    LEA INHBUF(PC),A0 ; ERSETZEN VON * GEGEN *
0403C0 4A10      TEXT1: TST.B (A0) ; ENDE ?
0403C2 670E      BEG.S TEXT2
0403C4 0C18 0022    CMP1.B #'',(A0)+
0403CB 66F6      BNE.S TEXT1
0403CA 117C 0027 FFFF MOVE.B #HOCHK,-1(A0) ; ERSETZEN
0403D0 60EE      BRA.S TEXT1
    
```

Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 5

```

0403D2 41FA FCDB    TEXT2: LEA MARKE(PC),A0 ; *** TEXTDATA IN DEN ASS TEXT SCHREIBEN
0403D6 6100 FE30    BSR UEBERTRAGUNG ; SPRUNG HINTER DATA
0403DA 2039 00040030 MOVE LABELC,D0 ; LABELNUMMER FÜR ÜBERSPRUNGS LABEL LAB
0403DE 5279 00040030 ADDU #1,LABELC
0403E2 41F9 000400B2 LEA LABBUF,A0
0403E8 4EB9 000015EA JSR SPRINT4D ; NUMMER IN ASCII ZEICHEN
0403F2 41FA FCBE    LEA LABBUF(PC),A0
0403F6 6100 FE10    BSR UEBERTRAGUNG
0403FA 6100 FEB2    BSR RETURN
0403FE 41FA FCE3    LEA LABEL(PC),A0 ; TEXT LABEL TEXT
040402 6100 FE04    BSR UEBERTRAGUNG
040406 41FA FCTA    LEA LABBUF(PC),A0 ; NUMMER FÜR TEXT LABEL
04040A 6100 FDFC    BSR UEBERTRAGUNG
04040E 41FA FC99    LEA DATA(PC),A0 ; DC.B SETZEN
040412 6100 FDF4    BSR UEBERTRAGUNG
040416 41FA FC2A    LEA INHBUF(PC),A0 ; ZEICHENKETTE
04041A 6100 FDEC    BSR UEBERTRAGUNG
04041E 41F9 000400C1 LEA E0FDATA,A0 ; DATAENDE
040424 6100 FDE2    BSR UEBERTRAGUNG
040428 41FA FCS8    LEA LABBUF(PC),A0 ; ÜBERSPRUNGS LABEL LAB MIT NUMMER
04042C 6100 FDDA    BSR UEBERTRAGUNG
040430 41FA FC98    LEA E0FDATA2(PC),A0 ; LEA TEXT MIT NUMMER
040434 6100 FDD2    BSR UEBERTRAGUNG
040438 41FA FCAB    LEA LABBUF(PC),A0
04043C 6100 FDCA    BSR UEBERTRAGUNG
040440 41FA FC9A    LEA E0FDATA3(PC),A0 ; (PC),A
040444 6100 FDC2    BSR UEBERTRAGUNG
040448 4240      CLR D0
04044A 1004      MOVE.B D4,D0
04044C 6100 FE40    BSR ADRPL
040450 41FA FC30    LEA LABBUF(PC),A0
040454 4EB9 000015EA JSR SPRINT4D ; NUMMER IN ASCII ZEICHEN
04045A 41FA FC26    LEA LABBUF(PC),A0
04045E 6100 FDB8    BSR UEBERTRAGUNG
040462 6100 FE4A    BSR RETURN
040466 4A05      TEXT4: TST.B D5
040468 6700 FFOE    BEG AUSW4 ; KEINE WEITEREN INFORMATIONEN
04046C 6000 FEE4    BRA AUSW3 ; WEITERE INFORMATIONEN
040470
040470 41FA FB00    DATAUS: LEA INHBUF(PC),A0 ; DATENBLOCK
040474 13FC 0020    MOVE.B #' ',E0FDATA
040478 000400C1      MOVE.B #' ',E0FDATA+1
04047C 13FC 0020
040480 000400C2
040484 13FC 0020    MOVE.B #' ',INHBUF
040488 00040042
04048C 0C18 002E    DAT1: CMP1.B #'',(A0)+
040490 66FA      BNE.S DAT1 ; ADRESSIERUNG SUCHEN
040492 422B FFFF    CLR.B -1(A0)
040496 13D0 000400BE MOVE.B (A0),DATA+5 ; ADRESSIERUNG IN DATA SCHREIBEN
04049C 6000 FF34    BRA TEXT2
0404A0
0404A0 41FA FBA0    REGAUS: LEA INHBUF(PC),A0 ; DATENREGISTER:
0404A4 0C10 007E    CMP1.B #LEERREG,(A0) ; ÜBERSPRINGEN ?
0404A8 6758      BEG.S REB4
0404AA 4A10      REG1: TST.B (A0)
0404AC 674A      BEG.S REG2 ; ADRESSMODE SUCHEN
    
```



04044E 0C18 002E	CMP1.B #'', (A0)+				
0404B2 66F6	BNE.S REG1				
0404B4 422B FFFF	CLR.B -1(A01)				
0404B8 13D0 00040097	MOVE.B (A0),MVE+5	; ADRESSMODE SETZEN			
0404BE 41FA FB02	LEA MVE(PC),A0				
0404C2 6100 FD44	BSR UEBERTRAGUNG	; BEFEHLSUBERTRAGUNG			
0404C6 41FA FB7A	LEA INHBUF(PC),A0				
Rolf-D.Klein 68000/08 Assembler 4.2 (C) 1984, Seite 6					
0404CA 6100 FD3C	BSR UEBERTRAGUNG				
0404CE 41FA FB0A	LEA MVE2(PC),A0				
0404D2 6100 FD34	BSR UEBERTRAGUNG				
0404D6 4240	CLR D0				
0404D8 1003	MOVE.B D3,D0	; NUMMER FÜR DATENREGISTER			
0404DA 6100 FDC8	BSR DATPL				
0404DE 41FA FB42	LEA LABBUF(PC),A0				
0404E2 4E89 000015EA	JSR SPRINT4D				
0404E8 41FA FB98	LEA LABBUF(PC),A0				
0404EC 6100 FD1A	BSR UEBERTRAGUNG				
0404F0 6100 F08C	BSR RETURN				
0404F4 6000 FF70	BRA TEXTA4				
0404F8 13FC 0077	MOVE.B #'m',MVE+5	; ADRESSMODE NICHT GEFUNDEN, ALSO .M			
0404FC 00040097					
040500 60BC	BRA.S REG3				
040502 6100 FDA0	BSR DATPL				
040506 6000 FF5E					
04050A	BRA TEXTA4				
04050A 0C39 007E			ADRAUS:	CMP1.B #LEERREG,INHBUF+1	; ADRESSREGISTER ÜBERSPRINGEN ?
04050E 00040043					
040512 6608	BNE.S ADRI				
040514 6100 FD78	BSR ADRPL				
040518 6000 FF4C	BRA TEXTA4				
04051C 41FA FB7F	LEA MVA(PC),A0	; LEA	ADR1:		
040520 6100 FCE6	BSR UEBERTRAGUNG				
040524 41FA FB1D	LEA INHBUF+1(PC),A0	; ADRESSE			
040528 6100 FCDE	BSR UEBERTRAGUNG				
04052C 41FA FB74	LEA MVA2(PC),A0	; ADRESSREGISTER			
040530 6100 FC06	BSR UEBERTRAGUNG				
040534 4240	CLR D0				
040536 1004	MOVE.B D4,D0	; NUMMER FÜR ADRESSREGISTER			
040538 41FA FB4B	LEA LABBUF(PC),A0				
04053C 4E89 000015EA	JSR SPRINT4D				
040542 41FA FB3E	LEA LABBUF(PC),A0				
040546 6100 FCC0	BSR UEBERTRAGUNG				
04054A 6100 FD62	BSR RETURN				
04054E 6000 FF16	BRA TEXTA4				
040552			ENDE:		
040552	END				
009000	Ende-Symboltabelle				

## 68000 — YOGIDOS UND JADOS, RL-BASIC

### RL-BASIC liest ASCII-Dateien

von Dipl.-Ing. Klaus Eilts-Grimm,  
Bei der Johanniskirche 9,  
2120 Lüneburg

Sehr geehrte LOOP-Redaktion, ich übersende Ihnen anbei zwei kleine Programme, die sicherlich für diejenigen LOOP-Leser interessant sein dürften, die etwas professioneller in die Arbeit mit RL-BASIC einsteigen wollen. Diese Programme ermöglichen die Ein- bzw. Ausgabe von Basic-Quelltexten im ASCII-Format (RDK-Editor) und wurden für die EPROM-Version des RL-BASIC geschrieben. Damit können dann z.B. Programme, die auf anderen Interpretern erstellt wurden und ebenfalls im ASCII-Format vorliegen, direkt ins Basic eingelesen werden (bzw. nach vorheriger Anpassung im Text-Editor). Man spart sich also viel Mühe beim Eintippen! Außerdem hat man nun die Möglichkeit, Basic-Programme im Text-Editor zu bearbeiten, was bei größeren Anwendungen sehr übersichtlich ist.

#### ASCII-Dateien für RL-BASIC (Eprom-Version)

Das BASIC68K besitzt leider keine Möglichkeit, den Quelltext im ASCII-Code abzulegen. Das macht sich besonders dann bemerkbar, wenn man Programme von anderen Basic-Dialekten übertragen will, ohne sie völlig neu einzutippen, oder wenn man größere Programmänderungen im Texteditor bearbeiten will. Da mir leider kein Quelltext des RL-BASIC vorliegt, habe ich zwei kurze Assembler-Programme geschrieben, die vom Monitor

aus die Übertragung eines Basic-Quellprogramms aus dem Basic in den Editor und umgekehrt ermöglichen. Die Programme sind von der Bibliothek zu und starten völlig relokativ. Das RL-BASIC kann ebenfalls auf einer beliebigen Adresse liegen, da es automatisch gesucht wird.

Das Programm EDIT)BAS überträgt ein Programm vom Editor ins Basic. Dabei wird die Eingabe-Routine des Basic über C12 auf den RAM-Speicher (aktuelle Textstartadresse) umgelenkt. Nach dem Start des Programms wird das RL-BASIC kalt gestartet und der Basic-Quelltext erscheint dann als fortlaufende Eingabe auf dem Bildschirm. In der letzten Zeile des Quellprogramms muß der Befehl SYSTEM (Direktmode) stehen. Dieser wird dann vom RL-Basic als Befehl interpretiert und übergibt die Kontrolle wieder an den Monitor. Bei erneutem Warmstart ist das Programm dann im Basic vorhanden.

Das Programm BAS)EDIT führt zunächst einen Warmstart durch. Mit der Eingabe POKE \$X802A,5 : LIST : SYSTEM (in einer Zeile) erfolgt dann die Übertragung des aktuellen Basic-Programms in den RAM-Bereich (aktuelle Textstartadresse). Am Schluß des Quelltextes wird automatisch der Direktbefehl SYSTEM abgelegt. Wird BAS)EDIT nur durch SYSTEM verlassen, so wird der entsprechende RAM-Bereich gelöscht. Will man das Programm im Editor erhalten, so muß das RL-BASIC direkt warm gestartet werden.

Vor der Benutzung von EDIT)BAS muß der Speicherbereich des RL-BASIC definiert sein (Editor-Bereich aussparen!). Dabei nimmt der Quelltext im Editor wesentlich

mehr Speicherplatz ein, als das abgelegte Programm im Basic. Das sollte besonders bei langen Programmen berücksichtigt werden! Vor der Benutzung von EDIT)BAS muß mindestens ein Kaltstart erfolgt sein.

Die Übertragung von Programmen ins RL-BASIC dauert wegen der damit verbundenen Bildschirmausgabe leider etwas. Ich habe bis jetzt keine Möglichkeit gefunden, das Echo abzuschalten. Da die Basic-Programme jetzt im Editor bearbeitet werden können, besteht nun auch die Möglichkeit, den RENUMBER-Befehl, der diesem Basic leider fehlt, als Bibliotheksprogramm zu realisieren (folgt). Es bleibt zu hoffen, daß diese Befehle über kurz oder lang in das RL-BASIC aufgenommen werden oder daß der Quelltext des Listings zur Verfügung gestellt wird.

#### Impressum:

LOOP Zeitung für Computerbauer  
Herausgeber: Gerd Graf  
Redaktion: Rolf-Dieter Klein, Gerd Graf  
Gestaltung und Druck:  
Karl-Heinz Rieder, Kempten  
Herstellung und Anzeigenverwaltung:  
GES GmbH  
Magnusstraße 13, 8960 Kempten  
Anzeigenpreisliste 1/84

# 68 000 - YOGIDOS UND JADOS, RL-BASIC

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

```

04C000          ORG $4C000
04C000          KOPF1 $55AA0180          * FUER BIBLIOTHEK
04C000          DC.L $55AA0180
04C004          454449543E4241          DC.B 'EDIT>BAS'
04C008          53
04C00C          000001A8          DC.L EDITBAS-KOPF1
04C010          0000021B          DC.L ENDE-CARSET
04C014          01          DC.B 1
04C015          00 00 00          DC.B 0,0,0
04C018          00000000          DC.L 0,0
04C01C          00000000
04C020          KOPF2:
04C020          55AA0180          DC.L $55AA0180
04C024          4241533E454449          DC.B 'BAS>EDIT'
04C028          54
04C02C          000001A2          DC.L BASEDIT-KOPF2
04C030          0000021B          DC.L ENDE-CARSET
04C034          01          DC.B 1
04C035          00 00 00          DC.B 0,0,0
04C038          00000000          DC.L 0,0
04C03C          00000000
04C040
= 00000018          USERCI EQU 000000018
= 0000001E          USERCSTS EQU 00000001E
= 00000024          USERCO EQU 000000024
= 00000040          KALT EQU 000000400
= 00000042          WARM EQU 000000424
= 0000002A          IOSTAT EQU 00000002A
= 0000002B          IOSTATB EQU 00000002B
= 00000036          STXTXT EQU 000000036
= 0000003A          AKTTXT EQU 00000003A
= 0000003E          ETXTXT EQU 00000003E
= 00000042          EOTXTXT EQU 000000042
04C040
04C040          CARSET:
04C040          003C 0001          OR #1,CCR
04C044          4E75          RTS
04C046
04C046          CARRES:
04C046          023C 0000          AND #0,CCR
04C04A          4E75          RTS
04C04C
04C04C          BIBSUCH:          * UEBERNOMMEN AUS RDK-GR
UNDRPROGRAMM
04C04C          0C30 55AA0180          CMP.L #55AA0180,(A0)
04C052          6500 FFCC          BNE CARSET
04C056          45E8 0004          LEA (A0),A2
04C05A          2028 0010          MOVE.L 16(A0),D0
04C05E          2268 000C          MOVEA.L 12(A0),A1
04C062          0C28 0000 0014          CMP.B #0,20(A0)
04C066          6700 000E          BEQ BIBIS
04C06C          0C28 0001 0014          CMP.B #1,20(A0)
04C072          6600 FFCC          BNE CARSET
04C076          D3C8          ADDA.L A0,A1
04C078          BRA CARRES
04C07C          6000 FFCC          BRA CARRES
04C07C          KALTSTART:          * BASIC-KALTSTART AUSFUE
HREN          MOVEA.L #0,A0          * BIBLIOTHEK NACH BASIC6
04C07C          207C 00000000
8K ABSUCHEN
04C082          KALT1START:
04C082          2F08          MOVE.L A0,-(A7)
04C084          KALTJSTART:
04C084          6100 FFCC          BSR BIBSUCH
04C088          6500 0048          BCS KALT1START
    
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

```

04C08C          2F08          MOVE.L A0,-(A7)          * EINTRAG GEFUNDEN
04C08E          48E7 8060          MOVEA.L D0/A1/A2,-(A7)
04C092          47FA 018E          LEA TEXTBAS(PC),A3
04C096          303C 0007          MOVE #8-1,D0
04C09A          KALT4START:          * VERGLEICH
04C09A          B70A          CMPM.B (A2)+,(A3)+
04C09C          662E          BNE.S KALT5START
04C09E          51C8 FFFA          DBRA D0,KALT4START
04CA02          4CDF 0601          MOVEA.L (A7)+,D0/A1/A2          * BASIC68K GEFUNDEN
04CA06          205F          MOVEA.L (A7)+,A0
04CA08          205F          MOVEA.L (A7)+,A0
04CA0A          48E7 0100          MOVEA.L D7,-(A7)
04CA0E          3E3C 0010          MOVE #1CLR,D7
04CB02          4E41          TRAP #1
04CB04          4CDF 0080          MOVEA.L (A7)+,D7
04CB08          4E48 0400          JSR KALT(A0)          * KALTSTART
04CB0C          48E7 0100          MOVEA.L D7,-(A7)
04CB0E          3E3C 005B          MOVE #1SETA5,D7
04CC04          4E41          TRAP #1
04CC06          4CDF 0080          MOVEA.L (A7)+,D7
04CC0A          4E75          RTS
04CC0C          KALT5START:
04CC0C          4CDF 0601          MOVEA.L (A7)+,D0/A1/A2
04CCD0          205F          MOVEA.L (A7)+,A0          * WEITER SUCHEN
04CCD2          KALT2START:
04CCD2          205F          MOVEA.L (A7)+,A0
04CCD4          D1FC 00000400          ADDA.L #8400,A0
04CCDA          B1FC 000F0000          CMA.L #80F0000,A0
04CCE0          6600 FFA0          BNE KALT1START
04CE04          4E75          RTS
04CE06          WARMSTART:          * BASIC-WARMSTART AUSFUE
HREN          MOVEA.L #0,A0          * BIBLIOTHEK NACH BASIC6
04CE06          207C 00000000
8K ABSUCHEN
04CE0C          WARM1START:
04CE0C          2F08          MOVE.L A0,-(A7)
04CE0E          WARM3START:
04CE0E          6100 FF5C          BSR BIBSUCH
04CE12          6500 0048          BCS WARM2START
04CE16          2F08          MOVE.L A0,-(A7)          * EINTRAG GEFUNDEN
04CE18          48E7 8060          MOVEA.L D0/A1/A2,-(A7)
04CE1C          47FA 0154          LEA TEXTBAS(PC),A3
04CE20          303C 0007          MOVE #8-1,D0
04CE24          WARM4START:          * VERGLEICH
04CE24          B70A          CMPM.B (A2)+,(A3)+
04CE26          662E          BNE.S WARM5START
04CE28          51C8 FFFA          DBRA D0,WARM4START
04CE2C          4CDF 0601          MOVEA.L (A7)+,D0/A1/A2          * BASIC68K GEFUNDEN
04CE30          205F          MOVEA.L (A7)+,A0
04CE32          205F          MOVEA.L (A7)+,A0
04CE34          48E7 0100          MOVEA.L D7,-(A7)
04CE38          3E3C 0010          MOVE #1CLR,D7
04CE3C          4E41          TRAP #1
04CE3E          4CDF 0080          MOVEA.L (A7)+,D7
04CE42          4E48 0424          JSR WARM(A0)
04CE46          48E7 0100          MOVEA.L D7,-(A7)
04CE48          3E3C 005B          MOVE #1SETA5,D7
04CE4C          4E41          TRAP #1
04CE4E          4CDF 0080          MOVEA.L (A7)+,D7
04CE50          4E75          RTS
04CE52          WARM5START:
04CE52          4CDF 0601          MOVEA.L (A7)+,D0/A1/A2
04CE54          205F          MOVEA.L (A7)+,A0
04CE56          WARM2START:
04CE56          205F          MOVEA.L (A7)+,A0
    
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

```

04C13E          D1FC 00000400          ADDA.L #8400,A0
04C144          B1FC 000F0000          CMA.L #80F0000,A0
04C14A          6600 FFA0          BNE WARM1START
04C14E          4E75          RTS
04C150          INIT:          * SPRUNGADRESSEN FUER CI
2,C02
04C150          3B7C 4EF9 0024          MOVE.W #84E9,USERCO(A5)          * JMP
04C156          41FA 00E6          LEA SCHREIB(PC),A0
04C15A          2B48 0026          MOVE.L A0,USERCO+2(A5)          * ADRESSE
04C15E          3B7C 4EF9 0018          MOVE.W #84E9,USERCI(A5)          * JMP
04C164          41FA 00C8          LEA LESE(PC),A0
04C168          2B48 001A          MOVE.L A0,USERCI+2(A5)          * ADRESSE
04C16C          3B7C 4E75 001E          MOVE.W #84E5,USERCSTS(A5)          * RTS
04C172          48E7 0100          MOVEA.L D7,-(A7)
04C176          3E3C 001F          MOVE #1CIINIT2,D7
04C17A          4E41          TRAP #1
04C17C          4CDF 0001          MOVEA.L (A7)+,D0
04C180          4E75          RTS
04C182          SETEDIT:          * ENDE DES EDITOR-TEXTES
SUCHEN
04C182          206D 0036          MOVEA.L STXTXT(A5),A0
04C186          2B48 003A          MOVEA.L A0,AKTTXT(A5)
04C18A          SETEDIT1:
04C18A          4A18          TST.B (A0)+
04C18C          66FC          BNE.S SETEDIT1
04C18E          5388          SUBQ.L #1,A0
04C190          2B48 003E          MOVEA.L A0,ETXTXT(A5)
04C194          2B48 0042          MOVEA.L A0,EOTXTXT(A5)
04C198          48E7 0100          MOVEA.L D7,-(A7)
04C19C          3E3C 0010          MOVE #1CLR,D7
04C1A0          4E41          TRAP #1
04C1A2          4CDF 0001          MOVEA.L (A7)+,D0
04C1A6          4E75          RTS
04C1A8          EDITBAS:          * RAM INS BASIC
04C1A8          BSR INIT
04C1AC          1B7C 0006 002B          MOVE.B #6,IOSTATB(A5)
04C1B2          6100 FECC          BSR KALTSTART
04C1B6          1B7C 0002 002B          MOVE.B #2,IOSTATB(A5)
04C1BC          6100 FFC4          BSR SETEDIT
04C1C0          4E75          RTS
04C1C2          BASEDIT:          * BASIC INS RAM
04C1C2          BSR INIT
04C1C6          6100 FF1E          BSR WARMSTART
04C1CA          1B7C 0002 002A          MOVE.B #2,IOSTAT(A5)
04C1D0          206D 003A          MOVEA.L AKTTXT(A5),A0          * SYSTEM IST LETZTER BEF
EHL
04C1D4          10BC 0053          MOVE.B #'S',(A0)
04C1D8          D1FC 00000001          ADDA.L #1,A0
04C1DE          10BC 0059          MOVE.B #'',(A0)
04C1E2          D1FC 00000001          ADDA.L #1,A0
04C1E8          10BC 0053          MOVE.B #'S',(A0)
04C1EC          D1FC 00000001          ADDA.L #1,A0
04C1F2          10BC 0054          MOVE.B #'T',(A0)
04C1F6          D1FC 00000001          ADDA.L #1,A0
04C1FC          10BC 0045          MOVE.B #'E',(A0)
04C200          D1FC 00000001          ADDA.L #1,A0
04C206          10BC 004D          MOVE.B #'H',(A0)
04C20A          D1FC 00000001          ADDA.L #1,A0
04C210          10BC 000D          MOVE.B #00D,(A0)
04C214          D1FC 00000001          ADDA.L #1,A0
04C21A          10BC 000A          MOVE.B #00A,(A0)
04C21E          D1FC 00000001          ADDA.L #1,A0
04C224          10BC 0000          MOVE.B #000,(A0)
    
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 4

```

04C228          6100 FF58          BSR SETEDIT
04C22C          4E75          RTS
04C22E          LESE:          * EINGABE-ROUTINE
04C22E          48E7 0100          MOVEA.L D7,-(A7)
04C232          3E3C 0020          MOVE #1CI2,D7
04C236          4E41          TRAP #1
04C238          4CDF 0080          MOVEA.L (A7)+,D7
04C23C          4E75          RTS
04C23E          SCHREIB:          * AUSGABE-ROUTINE
04C23E          2F08          MOVE.L A0,-(A7)
04C240          206D 003A          MOVEA.L AKTTXT(A5),A0
04C244          1080          MOVE.B D0,(A0)
04C246          6706          BEQ.S SCHREIB1
04C248          51B8          ADDQ.L #1,A0
04C24A          2B48 003A          MOVEA.L A0,AKTTXT(A5)
04C24E          SCHREIB1:
04C24E          205F          MOVEA.L (A7)+,A0
04C250          4E75          RTS
04C252          TEXTBAS:
04C252          42415349433638          DC.B 'BASIC68K'
04C254          4B          DC.B 0
04C256          ENDE:
04C258          ENDE:
    
```

0E8D30 Ende-Symboltabelle



# CP/M 68K, C UND MODULA

## Suchen im Editor

von Jürgen Becker,  
Lipper Kamp 3, 4500 Osnabrück,  
Telefon (0541) 441222

Wer unter CP/M68K mit dem C-Compiler arbeitet und dafür den Editor des Grundprogrammes benutzt, soll hier angesprochen werden.

Da der Editor EDITRDK ohne Zeilennum-

mer arbeitet, wird man es schwer haben, in größeren Programmen Fehler zu finden, welche der Compiler aber mit Zeilennummer meldet. Das beigefügte C-Programm, welches ich SUCH.C nenne, soll da eine Hilfe sein.

Nachdem es zum SUCH.68K-Programm übersetzt wurde, kann es z.B. wie folgt gestartet werden: SUCH SUCH.C. Dem Aufruf muß der Name der Datei folgen, in

der der Fehler gesucht wird (in diesem Beispiel der Datei SUCH.C). Dann wird man aufgefordert, die gesuchte Zeilennummer einzugeben. Dies wird dann mit der Zeile vor und nach der Gesuchten auf der Console ausgegeben.

Man braucht nur noch eine markante Zeichenfolge über die Suchfunktion des Editors eingeben und hat die fehlerhafte Zeile gefunden.

```
/*
 * - Programm SUCH.C -           - Quelltext - Zeilensuchprogramm - *
 * 1986   Juergen Becker,       4500 Osnabrueck   861026   V1.0 *
 */
#include <stdio.h>
#define EOL 10

main(argc, argv)
int argc;
char **argv;
{
    /* BEGINN HAUPTPROGRAMM */

    char getch();
    char buff[20];
    int zeich, nummer, vergleich, drei, f;
    FILE *fopen();
    FILE *fp;

    if (argc != 2)
    {
        fprintf(stderr, "Bitte Dateinamen mit angeben ");
        exit(1);
    }

    do { /* noch einmal von vorne suchen */
        vergleich = 1;
        if ((fp = fopen(argv[1], "r")) == NULL)
        {
            printf("\n\tFehler : \n . . %s kann nicht ge|ffnet werden !", argv[1]);
            exit(2);
        }

        do { /* weiter suchen */
            do {
                printf(" Gib die Zeilen Nummer ein ! :");
                gets(buff); sscanf(buff, "%d", &nummer);
                f = nummer - 3;
            } while (nummer < 3 || f < vergleich);

            --nummer; /* Damit auch die Reihe vorher ausgegeben wird */

            do {
                do {
                    zeich = getch(fp);
                    } while (zeich != EOL && zeich != EOF);
                    /* Wenn Zeilenende kommt, aus Schleife raus */
                    ++vergleich;
                } while (vergleich != nummer && zeich != EOF);
                /* Wenn Zeilennummer gefunden aus Schleife raus */

                if (zeich != EOF)
                {
                    drei = 0;

                    while (drei != 3)
                    {
                        printf("\nZeile Nr. %d : \n", vergleich);
                        zeilout(fp);
                        ++drei;
                    }
                }
                else
                {
                    printf("\n\nFehler! Falsche Eingabe !\n Zeilen ");
                    printf("Nummern }ber > %d < gibt es nicht !\n", vergleich);
                }

                printf("\n Noch einmal von vorne suchen = 2 weiter suchen =");
                printf(" 1 Ende = 0\n");
                gets(buff); sscanf(buff, "%d", &nummer);
            } while (nummer == 1); /* Ende weiter suchen */

            fclose(fp);
        } while (nummer == 2); /* Ende noch mal von vorne suchen */

        printf(" \n\nEnde \n");
    } /* Ende main */

    zeilout(fp);
    FILE *fp;
    int ze;

    while ((ze = getch(fp)) != EOF && ze != EOL) putchar(ze);
}

```

```
zeich = getch(fp);
} while (zeich != EOL && zeich != EOF);
/* Wenn Zeilenende kommt, aus Schleife raus */
++vergleich;
} while (vergleich != nummer && zeich != EOF);
/* Wenn Zeilennummer gefunden aus Schleife raus */

if (zeich != EOF)
{
    drei = 0;

    while (drei != 3)
    {
        printf("\nZeile Nr. %d : \n", vergleich);
        zeilout(fp);
        ++drei;
    }
}
else
{
    printf("\n\nFehler! Falsche Eingabe !\n Zeilen ");
    printf("Nummern }ber > %d < gibt es nicht !\n", vergleich);
}

printf("\n Noch einmal von vorne suchen = 2 weiter suchen =");
printf(" 1 Ende = 0\n");
gets(buff); sscanf(buff, "%d", &nummer);
} while (nummer == 1); /* Ende weiter suchen */

fclose(fp);
} while (nummer == 2); /* Ende noch mal von vorne suchen */

printf(" \n\nEnde \n");
} /* Ende main */

zeilout(fp)
FILE *fp;
{
    int ze;

    while ((ze = getch(fp)) != EOF && ze != EOL) putchar(ze);
}

```

## Erzeugung der BOOT-Tracks beim CP/M68K

von Rüdiger Bäcker

Nachdem das Verfahren zur Änderung des BIOS beim CP/M68K bereits in der LOOP erklärt wurde, möchte ich mich heute einmal kurz mit der Erzeugung der BOOT-Tracks beschäftigen.

Das CP/M68K ist nicht wie das CP/M80 auf den Bootspuren enthalten, sondern steht als Datei CPM.SYS auf der Diskette. Auf den Bootspuren befindet sich ein sogenanntes Loader-BIOS, das eine Untermenge des eigentlichen BIOS darstellt. Dieses Loader-BIOS lädt das CP/M von der Diskette. Beim NDR-Klein-Computer wird dies durch das Programm NDRBOOT noch unterstützt. Das Programm NDRBOOT wird zunächst vom

Grundprogramm aus mit der Menue-Funktion "FLOPPY STARTEN" in den Speicher geladen und durch die Auto-start-Funktion (NDR zur Kennung) gestartet. Es sucht dann das Grundprogramm, da alle BIOS-Funktionen mit Ausnahme der Winchester-Routinen vom Grundprogramm erledigt werden. Ist das Grundprogramm gefunden, so wird das Loader-BIOS gestartet und das eigentliche CP/M geladen und gestartet.

Im NDRBOOT werden die systemspezifischen Teile des Systems initialisiert. Hier kann man z.B. die Routine CO2 auf Bildschirmausgabe zurückschalten. Das ist vorteilhaft, wenn man den Assembler des Grundprogramms benutzt hat und dann vor dem Booten vergißt, die Ausgabe wieder umzuschalten.

Da die Programme NDRLDRB (Loader-BIOS) und NDRBOOT ebenfalls als Source auf der Diskette vorhanden sind,

können hier Änderungen problemlos durchgeführt werden. In unserem Beispiel könnten wir im NDRBOOT unsere Umschaltung einfügen:

```
#####
# Boot Load- Routine fuer Reststart #
# durch Grundprogramm #
# dahinter befindet sich direkt das CP/M #
# Betriebssystem #
# Bei Boot von 5 1/4" Steprate = Fast #
# Bootmeldung, Umschaltung auf CRT #
# U 1.01 Ruediger Baecker 25.06.86 #
# Aenderungskennung = (--- #
# U 1.0 Rot-Dieter Klein #
# Boot 8" // 5 1/4" mit autom. Erkennung #
#####

tpa equ $#00 # Zieladresse, auch bei LO68 angeben.

start:
nop # Erkennung fuer Grundprogramm, das gueltiger Boot da

move #20,d7 # CLASCREEN
trap #1 # Bildschirm loeschen
move #19,d7 # CO2 Umschalten auf CRT (---
trap #1
move #11,d0 # 80 Zeichen / Zeile
move #25,d7 # Size

Textstart=003000 Fenster=03000 Tor=009797 amer CTRL-J=Mitre

```

Bild 1 - Änderung im NDRBOOT

Wie bekommen wir nun aber das geänderte Loader-BIOS auf die BOOT-Tracks? Dazu gibt es das Programm XPUTBOOT, welches das Loader-BIOS auf die Systemspur schreibt. Dazu müssen das Loader-BIOS und das Programm NDRBOOT in einer Datei (z.B. CPMLDR.SYS) stehen und das Programm XPUTBOOT folgendermaßen aufgerufen werden:

XPUTBOOT (filename) (Laufwerk)

Aber halt, das Loader-BIOS und das Programm NDRBOOT sind doch noch nicht übersetzt und zusammengebunden! Richtig, aber das erledigt das nachfolgende Submit-File, welches dann auch noch das erzeugte CPMLDR.SYS auf die Bootspuren der Diskette in Laufwerk A schreibt. Die Option -F beim Aufrufen des Assemblers und des Linkers bewirken, daß die Zwischendateien auf die Ram-

floppy geschrieben werden, dies geht schneller. Die erzeugten Listings werden ebenfalls auf die Ramfloppy geschrieben und können von dort, mit dem bereits in der LOOP beschriebenen Submit-File, auf dem Drucker ausgegeben werden.

Das Submit-File zum Übersetzen, Linken und zum Schreiben der Boot-Tracks:

```
as68 -f 8:-p ndrldr.s )g:ndrldr.lst
as68 -f g:-p ndrboot.s )g:ndrboot.lst
1068 -f g:-s -t400 -uldr -o cpmlldr.sys
ndrboot.o ldrlib ndrldr.o
xputboot cpmlldr.sys a
```

Die für die Übersetzung und das Linken erforderlichen Programme bzw. Dateien wurden schon im Artikel zur BIOS-Änderung beschrieben. Zusätzlich ist hier noch die Datei LDRLIB auf die Diskette zu kopieren. Darin ist ein Library mit Objektfiles zur Erzeugung der Bootspuren enthalten.

Für „Spezialisten“ noch die Anmerkung, daß das Programm XPUTBOOT die BDOS-Funktion "read (filename)" und die BIOS-Funktion "write (filename)" benutzt.

Die Anzahl der reservierten Spuren für die Bootroutinen holt sich das Programm aus dem Disk-Parameter-Block.

Dieser Beitrag soll nur als Anregung für eigene Experimente dienen, man sollte jedoch vor den ersten Versuchen sicherstellen, daß noch eine bootfähige Diskette übrigbleibt! Mögliche Erweiterungen sind z.B. noch die Ausgabe eines Boot-Textes oder die Initialisierung von Grundprogrammparametern, so daß ohne Umweg über das Grundprogramm gebootet werden kann, wie dies schon von Axel Granel einmal in der LOOP beschrieben wurde.

## COPYDISK 68

vom Rüdiger Bäcker

In der Reihe der Utilities für den 68008-NDR-Klein-Computer heute in Programm zur Erzeugung von Disketten-Sicherheitskopien.

Da die Mehrzahl der Benutzer des NDR-Computers nun sicher schon ihr System mit Disketten-Controller und Laufwerken ausgerüstet haben, stellt sich nun die Frage: Wie sicher sind die Daten auf der Diskette aufgehoben? Nun, über die Datensicherheit ist schon in vielen Veröffentlichungen etwas gesagt worden. In der Regel passiert mit qualitativ guten Disketten so schnell nichts, dennoch ist es besser, für den Fall der Fälle noch eine Kopie der Diskette zu haben. Dem wird sicher jeder zustimmen, der die leidige Situation der verlorenen Daten schon einmal erlebt hat!

Mit dem Programm COPYDISK steht nun eine einfache und schnelle Möglichkeit zur Verfügung von 5¼" und 3½" Disketten im NDR 80-Spur-Format Sicherheitskopien anzufertigen. Getestet wurde das Programm mit 5¼" Laufwerken und JOGIDOS sowie CP/M68K- und CP/M80-Disketten. Das Programm selbst ist relocativ und mit einem Bibliotheksvorspann versehen. Nach dem Aufruf meldet es sich wie in Bild 1 gezeigt. Man muß nun die zu sichernde Diskette in Laufwerk A einle-

### Copydisk 68 - V 1.1

(C) 1986 by Ruediger Baecker

```
Bitte Originaldisk in Laufwerk 1 einlegen
Leere Disk in Laufwerk 2 einlegen
ACHTUNG Disk in LW 2 wird ueberschrieben !
Start = CR
Bild 1
```

gen und eine formatierte Diskette in Laufwerk B. Durch Drücken von CR wird das Programm gestartet, jede andere Taste bricht das Programm ab. Zu beachten ist dabei, daß die Diskette in Laufwerk B nur dann nicht überschrieben wird, wenn der mechanische Schreibschutz gesetzt ist! Man sollte also schon überlegen, ob die Diskette tatsächlich überschrieben wer-

### Copydisk 68 - V 1.1

(C) 1986 by Ruediger Baecker

```
Bitte Originaldisk in Laufwerk 1 einlegen
Leere Disk in Laufwerk 2 einlegen
ACHTUNG Disk in LW 2 wird ueberschrieben !
Start = CR
```

Kopiere Seite 1 - Spur

Kopie fertig - keine Fehler

Bild 2

den darf. Kopiert wird dann Spur für Spur, die Seite und aktuelle Spur werden am Bildschirm ausgegeben (Bild 2). Als Puffer für die gelesene Spur wird ein Bereich hinter dem Grundprogramm benutzt, dieser Bereich wird dabei überschrieben. Beim Kopieren auftauchende Fehler werden bemerkt und das Programm mit einer Fehlermeldung unterbrochen (Bild 3).

### Copydisk 68 - V 1.1

(C) 1986 by Ruediger Baecker

```
Bitte Originaldisk in Laufwerk 1 einlegen
Leere Disk in Laufwerk 2 einlegen
ACHTUNG Disk in LW 2 wird ueberschrieben !
Start = CR
```

Kopiere Seite 0 - Spur 0

FEHLER beim schreiben !

F-Flip #Hence

Bild 3

Nach erfolgreicher Kopie gelangt man in das Grundprogramm zurück. Für Besitzer nur eines Laufwerkes sind im Programm einige Änderungen erforderlich. Es muß umgeschrieben werden, so daß dann nach jeder gelesenen Spur die Diskette gewechselt wird oder, daß je nach vorhandenem Speicher, mehrere Spuren eingelesen werden.

Rolf-D. Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

```
000400      ORG #400
000400
000400      *   C O P Y D I S K   6 8
000400      *
000400      *   V E R S I O N   1 . 1   -   28.06.86
000400      *
000400      *   P R O G R A M M   Z U M   K O P I E R E N   V O N   D I S K E T T E N   I M   N D R   8 0   S P U R - F O R M A T
000400      *
000400      *   C O P Y R I G H T   ( C )   1 9 8 6   B Y   R U E D I G E R   B A E C K E R   -   P O S T F A C H   4 1 1 1   -   5 8 2 0   B E V E L S B E R G
000400
000400      KOPF:
```

```
000400 55AA0180      DC.L #55AA0180
000404 434F5059444953 DC.B 'COPYDISK'
000408 4B
00040C 00000020      DC.L START-KOPF
000410 0000036C      DC.L ENDEALL-KOPF
000414 01           DC.B 1
000415 00 00 00      DC.B 0,0,0
000418 00000000      DC.L 0,0
00041C 00000000
000420
= 00001000      LESECODE      EQU #1000      * ZWISCHENSPEICHER FUER LESECODE, SCHREIBCODE
= 00001001      WRITECODE     EQU LESECODE+1    * UND PUFFER FUER EINE SPUR, ADRESSIERT WIRD
= 00001002      BUFFER       EQU LESECODE+2    * RELATIV ZU A5
000420
```

```

000420          START:
000420 1E3C 0011  MOVE.B #1CLPG,D7      * BILDSCHIRM LOESCHEN
000424 4E41      TRAP #1
000426 41FA 022A  LEA TXT1(PC),A0      * HEADERTEXTE AUSGEBEN
00042A 303C 0043  MOVE #43,D0      * SCHRIFTGROESSE
00042E 4281      CLR.L D1      * POSITION
000430 343C 00E6  MOVE #230,D2
000434 3E3C 000A  MOVE #!WRITE,D7      * UND AUSGEBEN
000438 4E41      TRAP #1
00043A
00043A 41FA 022A  LEA TXT2(PC),A0
00043E 0442 0023  SUB #35,D2
000442 103C 0032  MOVE.B #32,D0
000446 3E3C 000A  MOVE #!WRITE,D7
00044A 4E41      TRAP #1
00044C
00044C 41FA 0235  LEA TXT3(PC),A0
000450 0442 0023  SUB #35,D2
000454 103C 0021  MOVE.B #21,D0
000458 3E3C 000A  MOVE #!WRITE,D7
00045C 4E41      TRAP #1
00045E
00045E 41FA 024D  LEA TXT4(PC),A0
000462 0442 0014  SUB #20,D2
000466 103C 0021  MOVE.B #21,D0
00046A 3E3C 000A  MOVE #!WRITE,D7
00046E 4E41      TRAP #1
000470
000470
000470 41FA 025D  LEA TXT5(PC),A0
000474 0442 0014  SUB #20,D2
000478 103C 0021  MOVE.B #21,D0
00047C 3E3C 000A  MOVE #!WRITE,D7
000480 4E41      TRAP #1
000482
000482 41FA 0276  LEA TXT6(PC),A0
000486 0442 0014  SUB #20,D2
00048A 103C 0021  MOVE.B #21,D0

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

00048E 3E3C 000A  MOVE #!WRITE,D7
000492 4E41      TRAP #1
000494
000494 3E3C 000C  MOVE #!C1,D7      * EINGABE VON TASTATUR
000498 4E41      TRAP #1
00049A 0C00 000D  CMP.B #0D,D0      * CR = START,
00049E 6600 01B0  BNE ENDE      * SONST ABRUCH
0004A2
0004A2          COPYDISK:
0004A2 41FA 0261  LEA TXT7(PC),A0      * TEXT FUER KOPIEREN
0004A6 0442 0023  SUB #35,D2
0004AA 103C 0021  MOVE.B #21,D0
0004AE 3E3C 000A  MOVE #!WRITE,D7
0004B2 4E41      TRAP #1
0004B4
0004B4 41FA 0267  LEA TXT7A(PC),A0      * SEITE 0
0004B8 323C 00A8  MOVE #168,D1
0004BC 103C 0021  MOVE.B #21,D0
0004C0 3E3C 000A  MOVE #!WRITE,D7
0004C4 4E41      TRAP #1
0004C6
0004C6 4281      CLR.L D1      * STEPRADE SETZEN
0004CB 163C 00B0  MOVE.B #10000000,D3 * FAST
0004CC 3E3C 0048  MOVE #!FLOPPY,D7      * UND SETZEN
0004D0 4E41      TRAP #1
0004D2
0004D2 323C 0001  MOVE #1,D1      * LESEN
0004D6 343C 0001  MOVE #1,D2      * SEKTOR 1
0004DA 163C 0000  MOVE.B #0,D3      * SPUR 0
0004DE 187C 0021 1000 MOVE.B #200100001,LESECODE(A5) * PARAMETER
0004E4 187C 0022 1001 MOVE.B #100100010,WRITECODE(A5) * DTD, LM 2
0004EA 41ED 1002  LEA BUFFER(A5),A0      * ARLAGEADRESSE
0004EE 182D 1000  MOVE.B LESECODE(A5),D4 * PARAMETER HOLEN
0004F2 3E3C 004B  MOVE #!FLOPPY,D7
0004F6 4E41      TRAP #1
0004F8 0C40 FFFF  CMP #FFFF,D0      * FEHLER ?
0004FC 6700 00E2  BEQ RERROR
000500
000500          SCHLEIFE:
000500 0642 0001  ADD #1,D2      * NAECHSTER SEKTOR
000504 0C02 0006  CMP.B #6,D2      * 1...5 ERLAUBT
000508 6700 0018  BEQ NEXTSPUR      * JA, DANN NAECHSTE SPUR
00050C 00FC 0400  ADDA #400,A0      * NEIN, DANN NOCH EIN KBYTE
000510 3E3C 004B  MOVE #!FLOPPY,D7
000514 4E41      TRAP #1
000516 0C40 FFFF  CMP #FFFF,D0      * FEHLER ?
00051A 6700 00C4  BEQ RERROR
00051E
00051E 6000 FFE0  BRA SCHLEIFE
000522
000522          NEXTSPUR:
000522 1003  MOVE.B D3,D0      * GELESENE SPUR WIRD ABGESPEICHERT
000524 6100 0102  BSR AUSGABE      * FUER AUSGABE SPUR
000528 41ED 1002  LEA BUFFER(A5),A0      * AUSGEBEN
00052C 123C 0002  MOVE.B #2,D1      * ABLAGEADRESSE
000530 143C 0001  MOVE.B #1,D2      * SCHREIBEN
000534 182D 1001  MOVE.B WRITECODE(A5),D4 * SEKTOR 1
000538 3E3C 004B  MOVE #!FLOPPY,D7      * LAUFWERK 2
00053C 4E41      TRAP #1
00053E 0C40 FFFF  CMP #FFFF,D0      * FEHLER ?
000542 6700 0084  BEQ WERROR
000546
000546          NEXTSCHL:
000546 0642 0001  ADD #1,D2      * NAECHSTER SEKTOR
    
```

```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

00054A 0C02 0006  CMP.B #6,D2      * SEKTOR 6 ? 1...5 ERLAUBT
00054E 6700 0018  BEQ NEXTENDE      * JA, DANN ENDE ABSPEICHERN
000552 00FC 0400  ADDA #1024,A0      * SONST NOCH EIN KBYTE
000556 3E3C 004B  MOVE #!FLOPPY,D7
00055A 4E41      TRAP #1
00055C 0C40 FFFF  CMP #FFFF,D0      * FEHLER ?
000560 6700 0096  BEQ WERROR
000564
000564 6000 FFE0  BRA NEXTSCHL
000568
000568          NEXTENDE:
000568 0643 0001  ADD #1,D3      * NAECHSTE SPUR EINLESEN
00056C 0C03 0050  CMP.B #80,D3      * SPUR 80 ? 0...70 SIND ERLAUBT
000570 6700 0014  BEQ SEITE2      * JA, DANN SEITE 2 COPIEREN
000574 182D 1000  MOVE.B LESECODE(A5),D4 * PARAMETER HOLEN
000578 4242  CLR D2
00057A 41ED 1002  LEA BUFFER(A5),A0
00057E 323C 0001  MOVE #1,D1      * LESEN
000582 6000 FF7C  BRA SCHLEIFE
000586
000586          SEITE2:
000586 41FA 0197  LEA TXT7B(PC),A0      * PARAMETER FUER SEITE 2 SETZEN
00058A 323C 00A8  MOVE #168,D1      * SEITE 1
00058E 343C 0041  MOVE #65,D2
000592 103C 0021  MOVE.B #21,D0
000596 3E3C 000A  MOVE #!WRITE,D7
00059A 4E41      TRAP #1
00059C
00059C 41FA 0183  LEA TXT7C(PC),A0      * BLANKTXT
0005A0 303C 0021  MOVE #21,D0      * SCHRIFTGROESSE
0005A4 323C 010E  MOVE #270,D1      * KOORDINATEN
0005AB 343C 0041  MOVE #65,D2
0005AC 3E3C 000A  MOVE #!WRITE,D7      * AUSGEBEN
0005B0 4E41      TRAP #1
0005B2
0005B2 0C2D 00A1 1000 CMP.B #210100001,LESECODE(A5) * WAR SCHON SEITE 2 ?
0005B8 6700 0056  BEQ WERROR      * JA, DANN ENDE
0005BC 187C 00A1 1000 MOVE.B #10100001,LESECODE(A5) * CODE FUER SEITE 2
0005C2 187C 00A2 1001 MOVE.B #210100010,WRITECODE(A5) * DTD.
0005C8 182D 1000  MOVE.B LESECODE(A5),D4 * PARAMETER HOLEN
0005CC 4242  CLR D2
0005CE 4243  CLR D3
0005D0 323C 0001  MOVE #1,D1      * LESEN
0005D4 41ED 1002  LEA BUFFER(A5),A0
0005DB 42AD 1002  CLR.L BUFFER(A5)      * BUFFER LOESCHEN
0005DC 6000 FF22  BRA SCHLEIFE      * UND DANN SEITE 2
0005E0
0005E0          RERROR:
0005E0 41FA 015A  LEA TXT9(PC),A0
0005E4 303C 0021  MOVE #21,D0
0005E8 343C 0014  MOVE #20,D2
0005EC 4281  CLR.L D1
0005EE 3E3C 000A  MOVE #!WRITE,D7
0005F2 4E41      TRAP #1
0005F4 6000 005A  BRA ENDE
0005F8
0005F8          WERROR:
0005F8 41FA 012A  LEA TXTB(PC),A0
0005FC 303C 0021  MOVE #21,D0
000600 343C 0014  MOVE #20,D2
000604 4281  CLR.L D1
000606 3E3C 000A  MOVE #!WRITE,D7
00060A 4E41      TRAP #1
00060C 6000 0042  BRA ENDE
    
```

```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 4

000610
000610          RERROR:
000610 41FA 013E  LEA TXT10(PC),A0
000614 303C 0021  MOVE #21,D0
000618 343C 0014  MOVE #20,D2
00061C 4281  CLR.L D1
00061E 3E3C 000A  MOVE #!WRITE,D7
000622 4E41      TRAP #1
000624 6000 002A  BRA ENDE
000628
000628          AUSGABE:
000628 4BE7 FEB0  MOVEM.L D0-D6/A0,-(A7) * DIENT DER AUSGABE DER BEARBEITETEN SPUR AM BS
00062C 41ED 1002  LEA BUFFER(A5),A0      * RETTEN
000630 3E3C 002E  MOVE #!PRINTAD,D7      * ADRESSE BUFFER
000634 4E41      TRAP #1      * WERT IST SCHON IN HAUPTPRG. UEBERGEBEN WORDEN
000636 41ED 1002  LEA BUFFER(A5),A0
00063A 303C 0021  MOVE #21,D0      * SCHRIFTGROESSE
00063E 323C 010E  MOVE #270,D1      * KOORDINATEN
000642 343C 0041  MOVE #65,D2
000646 3E3C 000A  MOVE #!WRITE,D7      * AUSGEBEN
00064A 4E41      TRAP #1
00064C 4CDF 017F  MOVEM.L (A7)+,D0-D6/A0 * WIEDER ZURUECK
000650
000650          ENDE:
000650 4E75  RTS
000652
000652 436F7079646973  TXT1: DC.B 'Copydisk 68 - V.1.1',0
000659 68203638202D20
000660 5620312E3100
000666 28432920313938  TXT2: DC.B '(C) 1986 by Ruediger Baecker',0
00066D 36206279205275
000674 656469767657220
00067B 42616563686572
000682 00
000683 4269747465204F  TXT3: DC.B 'Bitte Originaldisk in Laufwerk 1 einlegen',0
    
```

```

00068A 72697696E616C
000691 6469736820696E
00069B 204C6175667765
00069F 72682031206569
0006A6 6E6C6567656E00
0006AD 4C65672652044  TXT4: DC.B 'Leere Disk in Laufwerk 2 einlegen',0
0006B4 69736820696E20
0006BB 4C617566776572
0006C2 6B20322065696E
0006C9 6C6567656E00
0006CF 41434954554E47  TXT5: DC.B 'ACHTUNG Disk in LW 2 wird ueberschrieben !',0
0006D6 20446973682069
0006DD 6E204C57203220
0006EA 77697264207565
0006EB 62657273636872
0006F2 696562656E2021
0006F9 00
0006FA 5374617274203D  TXT6: DC.B 'Start = CR',0
000701 20435200
000705 486F7069657265  TXT7: DC.B 'Kopiere Seite - Spur ',0
00070C 20536569746520
000713 20202020537075
00071A 722000
00071D 3000  TXT7A: DC.B '0',0
00071F 3100  TXT7B: DC.B '1',0
000721 202000  TXT7C: DC.B ',0
000724 46454B4C455220  TXT8: DC.B 'FEHLER beim schreiben !',0
00072B 6265696D207363
000732 6872656962656E
    
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 5

```

000739 202100
00073C 46454B4C455220  TXT9: DC.B 'FEHLER beim lesen !',0
000743 6265696D206C65
00074A 73656E202100
000750 486F7069652066  TXT10: DC.B 'Kopie fertig - keine Fehler',0
000757 6572746967202D
00075E 206865696E6520
000765 46454B4C455220
00076C
00076C ENDEALL: * MERKER FUER BIBLIOTHEKSEINTRAG
00076C
00076C END.
    
```

0E91C2 Ende-Symbolabelle

Fehler beim DISASS 2.1 von Rolf Lobreyer

Beim o.g. Assembler ist in der Version 2.1 noch ein Fehler enthalten, Operationen mit dem CCR des Prozessors werden falsch disassembliert:

Beispiel:

1. Source:

org \$35000

```

TEST:
and.b ##FE,CCR
or.b #1,CCR
rts
    
```

2. Ergebnis nach dem Assemblieren:

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

```

035000          ORG $35000
035000          TEST:
035000          AND.B ##FE,CCR
035000 023C 00FE  OR.B #1,CCR
035004 003C 0001  RTS
035008 4E75
03500A
    
```

3. Disassemblierte Routine:

Rolf Lobreyer 68000/68008 Disassembler 2.1 (c) 1984

```

035000 023C 00FE  ANDI.B  ##FE,##003C  ???
035004 003C      ORI.B    ##75,D1    ???
035006 0001 4E75  ORI.B    ##75,D1    ???
    
```

# Der mc-CP/M-COMPUTER

## TERM1 - Programm-Quelle jetzt auch auf Diskette

Wir sind dem Wunsch vieler Kunden nachgekommen und liefern nun das

TERM1-Quellisting auch auf Diskette aus! Somit können Anwender die Software des TERM noch besser modifizieren.

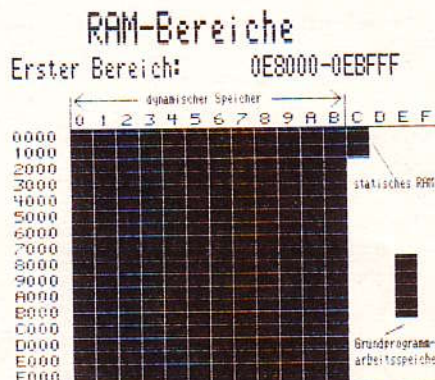
TERM ist eine Europakarte, die ein komplettes Terminal für Text und hochauflösende Graphik darstellt. Die Baugruppe ist ab Seite 136 unseres Kataloges detailliert beschrieben!

Bestell-Nr.	Bezeichnung	DM
	TERM-Software,	
10880	5 1/4" 80 Sp.	49,-
10881	3 1/2" 80 Sp.	49,-
10882	5 1/4" 40 Sp.	49,-

# TIPS + TRICKS - TIPS + TRICKS

## NDR 68000 ohne WAIT's - was noch zu beachten ist!

Mit der in der mc veröffentlichten Schaltung zur Verhinderung von unnötigen WAIT's bei der Baugruppe FLO2 und der Schaltungsrevision der RAM 64/256 sind nun alle Baugruppen so ausgelegt, daß nun noch WAIT's erzeugt werden, wenn man auf die Baugruppen zugreift. Nun gilt es jedoch noch einige Dinge zu beach-



ten, um wirklich ein optimales Ergebnis zu erzielen:

1. Das Grundprogramm legt den Stack immer an das Ende des größten, zusammenhängenden Speicherbereiches. Man sollte also dafür sorgen, daß hier ein statisches RAM sitzt, da sonst bei jedem Stackzugriff ein Wait erzeugt wird. Dies kann erreicht werden, wenn man auf den 1. Steckplatz einer hinter dem Rambereich sitzenden ROA ein statisches Ram setzt, wie im Bild 1 gezeigt. Bild 2 zeigt das Ergebnis, der SSp, USP und A7 sind mit

\$1CFF0 geladen, der Stack liegt also im statischen Ram.

2. Für den Betrieb mit CP/M68K sollte man möglichst auch den Bereich von \$0 bis \$3FF mit statischen Ram's bestücken und das CPM.SYS so relocieren, daß es in diesem Bereich liegt.

3. Wird das Grundprogramm aus anderen Gründen auf Adresse \$E0000 gelegt, so sollte trotzdem der untere Bereich mit statischen Ram's bestückt werden, da auch alle über TRAP's laufende Programmaufrufe sonst ein WAIT auslösen.

Rüdiger Bäcker

## JADOS und die SOUND-Baugruppe

von Uwe Koch

Alle JADOS-Anwender, die eine SOUND-Karte besitzen, und diese nicht auf Adresse \$FFFFFF50 betreiben wollen, weil sie z.B. die SOUND-Routine des Grundprogramms unter Assembler oder RL-Basic benutzen wollen, oder umschaltbar den Z80-Prozessor betreiben und hier die SOUND-Karte auf Port \$E0 brauchen, können mit einem einfachen Patch auf der Diskette die Adresse von \$FFFFFF50 auf \$FFFFFF40 oder \$FFFFFFE0 ändern. Probleme mit dem Drucker gibt es nicht mehr, wenn man die IOE-Karte, die als Centronics-Schnittstelle arbeitet, mit der erweiterten Adreßdekodierung aus einem meiner vorigen Tips ausstattet. Die Änderung im JADOS sollte natürlich nicht auf der Originaldiskette gemacht werden, sondern auf einer Arbeitskopie. Wenn man dann mit dieser Arbeitskopie die weiteren JADOS-Disketten mit SYS initialisiert wird die Änderung auch mit kopiert. Zum Ändern der Adresse lädt man die Systemspuren am

besten mit DOSEY, DOSEY II (von der SCC-Clubdiskette), dem Mini-DOS in J. Bastigkeits Grundprogramm oder JOGIDOS in das RAM.

Für DOSEY (II) und Bastigkeits Grundprogramm (das mit JADOS nicht funktioniert) kann man folgende Werte eingeben:

- Daten laden
- Speicherstart \$10000
- Länge (in K) 15
- Spur 0
- Sektor 1
- Daten lesen

Damit liegt das JADOS-System jetzt auf den Adressen \$10000 bis \$13BFF. Die nötigen Änderungen sind dann bei folgenden Adressen:

- \$1329A alt: 00 0F FF 50  
neu: 00 0F FF 40 oder: 00 0F FF E0
- \$132A6 alt: 00 0F FF 51  
neu: 00 0F FF 41 oder: 00 0F FF E1

Danach kann man den entsprechenden Bereich mit den gleichen Speicher-, Spur- und Sektorwerten auf die JADOS-Arbeitsdiskette zurückschreiben. Es müßten dann auch die Befehle SOUND, BELL und ERRNOISE sowie das Kommando BELL des JADOS funktionieren

## Kann der NDR-Klein-Computer von der Steuer abgesetzt werden?

In der LOOP Nr. 10 wurde in einem Artikel darauf hingewiesen, daß die Kosten für den NDR-Klein-Computer als Werbungskosten abzusetzen seien. Hierzu möchte ich Ihnen einiges aus meiner eigenen praktischen Erfahrung erzählen. Vielleicht sind meine Erfahrungen auch für Sie oder die Leser der LOOP interessant. Mit meiner Einkommensteuererklärung und einem dicken „Pack“ (mind. 2 cm) Rechnungen ging ich „persönlich“ zum Finanzamt. Auf dem Finanzamt wurde mir auf die Frage: „Kann ich meinen NDR-Klein-Computer absetzen?“ mit „nein“ geantwortet. Daraufhin verlangte ich eine schriftliche Begründung, warum dies so wäre. Es wurde mir geantwortet, daß meine Rechnungen von einem „EDV-Fachmann“ überprüft würden und daß ich, bei einer Ablehnung meines Antra-

ges, schriftlich Bescheid bekommen würde. Nach ca. zwei Monaten meldete sich das Finanzamt und fragte mich, wie ich den Computer nutze. Ich begründete meine Anschaffung wie folgt:

1. Durch den Selbstbau die Elektronikkenntnisse auffrischen und erweitern.
2. Die Kenntnisse und Fertigkeiten bei Sprachen wie z.B. Pascal, C und bei Standardsoftware wie z.B. dBASE II, Multiplan verbessern und erweitern.
3. Programme für die Firma zu Hause entwerfen und testen.
4. Ich erklärte, daß ich in der Firma mit einem Soft- und Hardwareprojekt beauftragt wurde und die mit dem NDR-Klein-Computer erstellte Software auf das Computersystem in der Firma übertragbar ist und auch übertragen wird. Es bestehen nämlich folgende Gemeinsamkeiten die das Übertragen erlaubten:
  - Aufwärtskompatibles Betriebssystem zu CP/M 2.2 (KOS der Fa. kontron).
  - Verwendung von PASCAL und der

Für das Taschenrechner-Programm, das ich geschrieben habe, brauche ich jetzt noch eine vernünftige Gleitkommaroutine. Können Sie mir ein Buch oder ähnliches empfehlen, nach dem man sich ein entsprechendes Programm schreiben kann?

Olaf Reinhold,  
Butterberg 1B, 3300 Braunschweig

Antwort LOOP:

Wer hilft Herrn Reinhold?

Suche Erfahrungsaustausch CPU 68008 bzw. CP/M68K. Raum Augsburg und München, bzw. nördlich von München.

H.-Dietrich Bade,  
Telefon: (089) 718011

**Zu verkaufen:**

NDR-68000/8: JADOS-Entwickler verk.:  
● REVERSI (starkes Strategiespiel!) 29,-  
● DISASS (komfort. Disassembler!) 49,-  
● INSPEC (Disketten-Editor / Drucker) 44,- zzgl. Porto, per NN; auf Disk 3 1/2" oder 5 1/4" unter JADOS! Reversi auch auf EPROM

K. Janßen  
Hanninxweg 74, 4150 Krefeld 1

**Verkaufe:**

NDR-Klein-Computer, betriebsbereit: SBC2, EBASIC, EGOSI, EGRUND, ESCOP, CPU68K, EASS03, EPASCAL, EDEMO, 7 x R8, 2 x ROA64, KEY, TAST1..., GDP64K, MON1, CAS, 2 x I/E, CENT, PROMMER, NE2, BUS2, POW 5V, GEHÄUSE, Literatur.

Carsten Schuldt  
Am Brunneck 10, 8012 Ottobrunn,  
Telefon: (089) 606521

**Zu verkaufen:**

NDR-CPM68K-Computer, 1 Laufwerk, 900K Ram, Hypy Promm.Ser. IO usw.; Bausatzpreis ca. 4.000,-, Verkaufspreis: 2.200,-. Außerdem SBC-Z80 (ass-basic-gosi). Verk. alle Teile auch einzeln!

Erhard Bauer  
Telefon (08323) 3597

**Verkaufe:**

NDR-Computer, betriebsber. CPU68K, 2 ROA64, GDP64, KEY, gr. Netzteil, gr. Gehäuse, Tastatur, Monitor, Sonderheft 1+2, RDK Mikrocomputer-Buch, VB DM 650,-

Frank Brumm  
Telefon (02238) 53594

Standardsoftware dBASE II auf beiden Systemen.

5. Zu Hause für die Firma Konzepte mittels Textverarbeitung ausarbeiten. Daraufhin wurde die Absetzung des Computers genehmigt. Ich bekam ca. 1/4 der Kosten zurück. Durch diesen Erfolg „beflügelt“, wurde sofort ein Drucker beschafft.  
Roland Welsch,  
Georg-Schäfer-Straße 54, 8603 Ebern,  
Telefon (09531) 81-358

# Theorie und Praxis rund um den NDR-Computer

## Mikroelektronik Einführung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

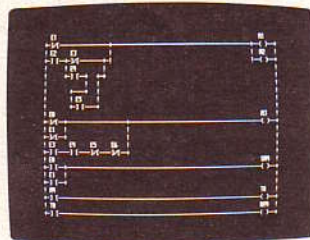
Der Kurs ist auf die HEXIO abgestimmt und ist für alle geeignet, die ihre ersten Schritte in Z 80-Maschinenprogrammierung machen.

Nach diesem Kurs sind Sie in der Lage, eigene Programme zu schreiben und die Arbeitsweise des Z 80 zu verstehen.

Der Kurs ist in verschiedene Fachgebiete aufgeteilt und bringt eine Menge Aufgaben, Beispielprogramme und Übungen.

**Aus dem Inhalt:** Was ist ein Mikroprozessor? \* Inbetriebnahme des Computers \* Planung von Programmen \* Aufbau der CPU \* Speicher und Adressen \* Datentransfer \* Lauflicht \* Breakpoints \* Hilfsfunktionen \* Logo-Elemente \* Strukturiertes Programmieren \* Label & Call.

## SPS-Programmierung



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Dieser Kurs zeigt Ihnen, wie SPS programmiert wird, die Normung, die Anwendungsmöglichkeiten und die verschiedenen Darstellungsarten.

Sie lernen spielend leicht, Relais- und Schützensteuerungen in SPS-Programme umzusetzen.

Beispielprogramme, Aufgaben und Übungen geben Ihnen die praktischen Erfahrungen und zeigen, wie SPS professionell eingesetzt wird. Nutzen Sie Ihren NDR-Computer für diese moderne Technik voll aus.

Der Kurs ist in folgende Fachgebiete gegliedert: Steuerungstechnik \* Digitaltechnik \* Methoden zur Beschreibung von Steuerungsaufgaben \* Programmierung \* Übungen und Tafeln.

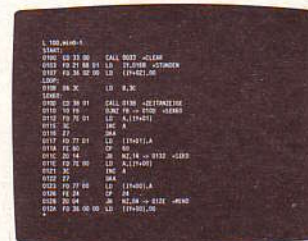
## ZEAT-Betriebssystem



Das Betriebssystem beinhaltet in drei EPROMs: Z 80-2-Pass-Assembler, Disassembler, Editor, Debugger, Telefonmodem-Programm, FLOMON 1.5, ausserdem eine ausführliche Dokumentation zum Preis von DM 198,-.

Das Betriebssystem ZEAT benötigt 64-K-RAM (dynamische RAM-Karte). Die EPROMs werden in die BANKBOOT-Karte eingesteckt und sind sofort betriebsbereit. Programmieren Sie Ihren NDR-Computer mit einem Profi-Assembler.

Das Textverarbeitungsprogramm hat volle Bildschirmmitteilung und kann neben der Programm- und Texteditierung auch zum Textschreiben eingesetzt werden.



## Z 80-Assembler-Programmierung

4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Der Kurs ist auf das ZEAT-Betriebssystem abgestimmt und zeigt Ihnen in leicht verständlicher Art, wie der NDR-Computer in Z 80-Assembler programmiert wird, bringt reichhaltig Übungsbeispiele und Anwendungen. Sie werden erstaunt sein, wie leicht diese Art der Programmerstellung ist. Und Sie lernen, wie man die serielle Schnittstelle bedient und Daten über Telefon übertragen kann.

Die Fachgebiete dieses Lehrgangs sind: Systembeschreibung \* Betriebssystem \* Programmierung \* Testen \* Modemprogramm \* Listings, Tafeln und Tabellen.

# Christiani

✂ Hier abtrennen und im Umschlag einsenden an: Dr.-Ing. P. Christiani GmbH, Techn. Lehrinstitut und Verlag, Postfach 35 69189, 7750 Konstanz

## Bestellcoupon

	Preis je Teil	Gesamtpreis
<input type="checkbox"/> Einführung mit dem NDR-Computer (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> Z 80-Assembler-Programmierung (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> SPS-Programmierung mit dem NDR-Computer (4 Kursteile)	DM 38,-	DM 152,-
<input type="checkbox"/> Kompakt-Kurs BASIC (angepasst an das RDK-BASIC)	DM 198,-	DM 198,-
<input type="checkbox"/> ZEAT-Betriebssystem (3 EPROMs mit Dokumentation)	DM 198,-	DM 198,-

Name, Vorname \_\_\_\_\_

Straße \_\_\_\_\_

PLZ, Ort \_\_\_\_\_

Datum \_\_\_\_\_ Unterschrift \_\_\_\_\_ 86189