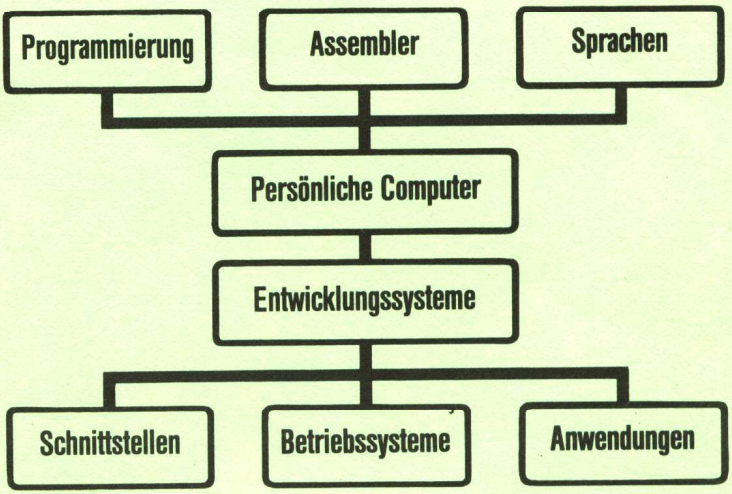
MICRO MAG

DM 9,50 Nr. 43 Juni 1985



Inhaltsverzeichnis

te-wi aktuell...

Fordern Sie unser komplettes Programm über C-64 an!

C-64 ENZYKLOPÄDIE

DER AUTOR RAETO WEST verwendete 1 Jahr der Analyse und Dokumentation auf den C-64! Ergebnis seiner völlig unzeitgemäßen Geduld: Das einzige enzyklopädische 64er-Buch, das neben Ihrem Computer liegen bleibt.

Alle Erklärungen, auch komplexer System- und Programmfragen, umfassen bei Ray West stets beides: Kompetenz durch Einsicht und solides Faktenwissen. Beispielhaft: Musiktheorie und SID-Chip in Kapitel 13!

EIN REFERENZBUCH für professionelle Hard/Software-Entwickler auf dem US-Standard des Buchs PROGRAMMING THE PET/CBM des gleichen Autors; **EIN LEHRBUCH** zu Aufbau und Anwendung von Mikrocomputern am Beispiel des C-64 für alle Autodidakten und Einsteiger; **EIN ANWENDUNGS-HANDBUCH** zum C-64/SX-64 mit über 300 Programmierungen aller 64er-Funktionen – auch der schwierigen, seltenen und meistgemiedenen.

über 600 Seiten, Softcover, DM 66,–

te-wi

te-wi Verlag GmbH
technisch wissenschaftliche Elektronik-Literatur
Theo-Prosel-Weg 1 8000 München 40

tm 4304

Roland Löhr

Orientierung zu 'C'

Die Sprache 'C' findet mit den 16 Bit-Mikrocomputern zunehmende Verbreitung. Für die Linie der IBM-PC's und der kompatiblen kann man diese Sprache als Option beziehen. Systeme mit dem Betriebssystem UNIX und dessen Derivaten haben auch 'C', denn diese Sprache wurde u.a. geschaffen, um UNIX schreiben zu können - Bei Computern mit dem Betriebssystem CP/M-68K von Digital Research für 68xxx-CPU gehört 'C' zum Lieferumfang. - 'C' wird sogar für den 8 Bit-Rechner C-64 von Commodore durch eine düsseldorfer Firma angeboten. Erfahrungsberichte dazu liegen beim Herausgeber noch nicht vor, es wäre aber interessant, ggfs. von den Lesern dazu zu hören. - 'C' dürfte auch für den jetzt kommenden Atari 520ST zur Verfügung stehen.

In diesem aktuellen Umfeld soll hier keine Einführung in die Sprache gegeben werden. Die findet man zuverlässiger in der in diesem Heft auch aufgeführten Literatur zu 'C'. Der Leser soll sich jedoch ein Bild machen können, was mit 'C' zu leisten ist, so daß er sich schon etwas vorbereitet ggfs. um weitere Informationen bemühen kann.

Was spricht für 'C'?

Man wird fragen, ob denn nicht die bisher verbreiteten Sprachen mit ihren jeweiligen Stärken und Schwächen für die Programmierung ausreichend seien. Dazu ist zu bemerken, daß die Entwicklung in der Technik und am Markt rasant ist. Neue CPU-Typen treten hinzu und auch Mikrocomputer mit Ihnen. Computer haben oft nur eine kurze Lebenszeit am Markt. Es war schon öfters zu erleben, daß gute Software für einen Computer erst dann zur Verfügung stand, als dessen Produktion schon eingestellt war. Software hat bei sinkenden Computerpreisen einen besonderen Wert, der möglichst über die Computergenerationen hinweg erhalten werden sollte.

In Assembler geschriebene Programme haben den Nachteil geringer Portabilität. Sie können bestenfalls als Vorlage dienen, wenn für eine andere CPU programmiert werden soll. Der Übergang zu einer anderen CPU bedeutet aber auch, daß der Programmierer eine neue Assemblersprache lernen muß. In den Betrieben ist deutlich eine Scheu vor solchen Umstellungen zu bemerken.

Das in der Software und in der Erfahrung der Mitarbeiter steckende Kapital läßt sich bewahren, wenn eine leistungsfähige Sprache zur Verfügung steht, die von Rechner zu Rechner portabel ist und die möglichst schnell ausführenden Assemblercode erzeugt. Beim Einsatz einer bis hin zum Maschinencode compilierenden Sprache braucht man nicht einmal die verschiedenen Assembler zu lernen. Eine Sprache dieser Art ist 'C'. Sie liegt insbesondere für die Systemprogrammierung derzeit im Trend, ebenso wie das Betriebssystem UNIX, und zwar für Personal Computer wie auch für große Maschinen. Das die Sprache beschreibende Buch von Kernighan und Ritchie soll weltweit 300.000mal verkauft worden sein, und die Zahl der 'C'-Programmierer wird wird derzeit auf eine halbe Million mit zunehmender Tendenz geschätzt.

Man sollte also sorgfältig prüfen, ob 'C' für die jeweiligen Zwecke das gedankliche Vehikel sein kann, mit dem man in Software erarbeitete Lösugen beständiger machen kann.

Die wichtigsten Merkmale

'C' ist eine Compilersprache: Der in 'C' formulierte Quelltext wird über verschiedene Zwischen-Files in ein ladbares und ausführbares maschinensprachliches Modul übergeführt. Die Bearbeitungen bis zu diesem Punkt werden durch den Preprozessor, den Parser, den Code-Generator, den Assembler und schließlich den Lader/Linker durchgeführt. Diese Bearbeitungen löst man im allgemeinen durch ein Kommandofile mit Namen cc.sub zum automatischen Ablauf aus.

Das Formulieren des Quelltextes setzt neben diesen Dienstleistungsprogrammen einen Text-Editor voraus, der in komfortablerer Form als Full Screen Editor ausgeführt sein sollte.

'C' kommt je nach Implementierung mit 28 reservierten Befehlsworten aus. Es sind dies:

```
int         extern      else
char        register    for
float       typedef     do
double      static      while
struct      goto        switch
union       return      case
long        sizeof      default
short       break       entry
unsigned    continue
auto        if
```

Befehlsworte, die der Ein- und Ausgabe dienen, wird man in der vorstehenden Liste vergebens suchen. Sie werden im Paket der Implementierung in der clib (library) mitgeliefert, die das runtime package für Programme in Maschinensprache enthält, das natürlich maschinenabhängig ist. Beim Autor hat die clib für CP/M-68K einen Umfang von 68 KB. Hinzu kommen für 'C'-Implementierungen verschiedene Hilfsfiles mit Standard-Definitionen, so z.B. stdio.h und portab.h für die Ein- und Ausgabe und für Parameter der Portabilität.

Wie in Betriebssystemen üblich, die auf CP/M fußen oder dessen 'Benutzeroberfläche' zur Verfügung stellen, arbeitet man file-orientiert: Alle Dienste werden vom Massenspeicher (Diskette) in die Maschine und zur Ausführung gebracht. Das bedingt viele Zugriffe zwischen dem Editieren, Compilieren bis hin zum Laden und Ausführen. Das Arbeiten erfolgt schneller, wenn möglichst viele oder alle Hilfsprogramme und der Quelltext in eine RAM-Disk geladen werden können. Beim Autor sind dann schon mehr als 400 KB im RAM mit den Systemprogrammen belegt. Man achte also darauf, daß die Systemausstattung ein zügiges Arbeiten ermöglicht, denn erfahrungsgemäß muß man auch für ein kleines Programm mehrmals den Editor laden, den Quelltext zur Neubearbeitung hereinziehen, abspeichern und dann die Compilerstufen durchlaufen.

Wie die vorstehende Liste reservierter Worte erkennen läßt, ist 'C' eine block-strukturierte Sprache. Wir finden dafür die typischen Formulierungen wie ('expression' ist hier mit 'expr' abgekürzt):

```
if (expr) statement
if (expr) statement else statement
while (expr) statement
do statement while (expr)
for (expr1; expr2; expr3) statement
```

Die Mehrwege-Fallverzweigung wird typisch wie folgt geschrieben:

```
switch (expr) {
  case Konstante: statement
  break ;
  ...
  default: statement
  }
```

Hier ist gefordert, daß der Ausdruck expr einen ganzzahligen Wert hat. Man erkennt damit ein Konstrukt, daß Ähnlichkeiten mit einem Parser nebst ON GOTO-Verzweigung hat. break dient dabei dem Verlassen von Konstrukten in die nächst höhere Strukturebene. - Im Gegensatz zu anderen höheren Sprachen erlaubt 'C' ein goto Label, womit man gelegentlich Konstrukte übersichtlicher auflösen kann.

Der Begriff 'statement' steht in 'C' gleichsinnig für 'compound statement': Überall, wo eine Anweisung möglich ist, kann man auch eine Anweisungsfolge einsetzen, die intern durch Semikolons getrennt wird und nach außen hin durch geschweifte Klammern (Akkoladen). Akkoladenpaare bilden damit Blöcke. - Es gibt auch das leere Statement, das durch ein einfaches Semikolon erzeugt wird, z.B. if (expr) ; else statement.

Datentypen

'C' kennt von Haus aus nur wenige Datentypen, nämlich char (Character), int (Integer) mit der möglichen Ergänzung short oder long bzw. unsigned short oder long. sowie float und double für einfach oder doppelt genaue Gleitkommazahlen. Die beiden letzten Typen werden jedoch nicht von jedem Compiler unterstützt. - Mit den Typangaben ist keine Vorschrift verbunden, in wieviel Bits die Datentypen abzubilden sind. Das ist von der Implementierung abhängig und ggfs. auch von der Registerbreite der CPU, die ausgenutzt werden kann. Man darf aber grob davon ausgehen, daß bei den kleineren Computern ASCII-Zeichen (char) 8 Bits beanspruchen, int oder short int 16 Bits umfassen und long int mindestens 32 Bits.

Es können auch mehrdimensionale Datenfelder zu den Datentypen angelegt werden. Man schreibt dann die Ausdehnung in eckige Klammern hinter den Namen des Feldes. So sind insbesondere Strings kein Grundtyp, sondern Arrays, die z.B. wie folgt erklärt werden: name[24].

Wie in anderen Hochsprachen, so sind auch in 'C' Datentyperklärungen erforderlich, und zwar vor dem erstmaligen Gebrauch von Namen zu den benutzten Konstanten, Variablen und Funktionen. Sofern Tyerklärungen innerhalb geschweifter Klammern stehen, sind sie für den Block intern, d.h. lokal. Ihre Werte gehen beim Verlassen des Blockes verloren. Das hat mit der üblichen Reservierung von Platz auf dem Datenstack zu tun. Man kann Variablen aber als static erklären, so daß sie bei einem zweiten Funktionsaufruf mit ihrem alten Wert zur Verfügung stehen.

Variablen können unter Beachtung der syntaktischen Regeln auch als global oder extern erklärt werden. Hierarchisch niedrigere Konstrukte können im allgemeinen nicht die Variablenwerte der höheren Konstrukte verändern, damit unbeabsichtigte Nebenwirkungen vermieden werden. Bei einem Funktionsaufruf mit Parameterübergabe werden der Funktion also Kopien der Variablenwerte mitgeteilt (Call by Reference). Gleichwohl können Funktionen Werte zurückliefern, die man in der aufrufenden Ebene einer Variablen zuweist, z.B' c = funct();.

Variable können als Pointertyp erklärt werden, indem man vor ihren Namen einen Stern setzt, wie z.B. *p. p ist damit nach einer Wertzuweisung nicht das zu bearbeitende Objekt, sondern ein Adreßzeiger auf das zu bearbeitende Objekt. In 6502-Assemblersprache ist das vergleichbar mit (p),Y, stellt also eine indirekte Adressierung dar.

Man muß hinzufügen, daß die Benutzung von Arrays nicht mit einer Kopierung ihres Inhaltes verbunden ist, sondern mit der Übergabe einer Pointeradresse auf das erste Element des Arrays. Insofern sind Arrays global erreichbar und veränderbar. Und mit Pointervariablen kann man ggfs. auch die sonst vorherrschende Lokalität von Variablen durchbrechen oder auch Interfaces adressieren. Pointer sind auch bei der File-Bearbeitung als Zeiger auf die Puffer in Benutzung.

'C' erlaubt auch die Definition von benutzer-eigenen Datentypen und Datensätzen. Datentypen werden mit typedef erklärt, z.B. typedef int zentimeter, meter, km;. Datensätze werden mit struct beschrieben. Innerhalb der Struktur werden die Datenfelder (unterschiedlichen Typs) im einzelnen beschrieben. Jedes Feld ist mit den Mitteln der Sprache einfach zu erreichen. Strukturen können auch überlagert werden, um ggfs. Speicherplatz zu sparen. Dem dient die Anweisung union. Ein Bereich kann so abwechselnd z.B. Datensätze verschiedener Struktur aufnehmen, wobei sich die Platzreservierung nach der größten in union herangezogenen Struktur richtet. In einigen Fällen wird man mit enum auch von Aufzählungstypen Gebrauch machen.

Funktionen

Wie auch in anderen Sprachen sind Funktionen Dienstleistungsblöcke. Sie haben einen Namen, gefolgt von einem Paar runder Klammern, in denen Argumente übergeben werden können. Sofern zu einer Funktion Argumente gehören, so ist im Kopf der Definition zunächst deren Datentyp zu definieren. Es folgt dann der ausführende Teil, 'body' oder Funktionskörper genannt in Akkoladen. Funktionen können per return(expr); einen Wert an den Aufrufer zurückliefern. Funktionen können rekursiv benutzt werden, d.h. sie können sich selbst aufrufen, denn mit jedem Aufruf wird ein eigener Variablenbereich angelegt. - Bei der Zurückgabe von Ergebnissen muß dem Aufrufer der Datentyp des Ergebnisses bekannt sein.

Funktionen sind in 'C'-Programmen untereinander gleichrangig und von überall erreichbar. Ledig-

lich das Hauptprogramm main(), das selber eine Funktion ist, steht hierarchisch über allen anderen Funktionen. main() darf in einem Programm nur einmal benutzt werden. Auch an die Funktion main() können Parameter übergeben werden, z.B. die Namen standardmäßig zu benutzender Files. Dateinamen können natürlich auch interaktiv im Programmlauf eingegeben werden.

Operatoren

'C' zeichnet sich durch die Vielzahl seiner Operatoren aus, mit denen prägnant knapper Code geschrieben werden kann.

Bei den Operatoren für die vier Grundrechenarten ist lediglich zu erwähnen, daß für Division auch der Modulus (Rest) zur Verfügung steht. - Auch bei den Vergleichsoperatoren haben wir den üblichen Satz gleich, ungleich, größer, kleiner, größer/gleich, kleiner/gleich.

Interessant wird es bei den Zuweisungsoperatoren. Hier läßt sich z.B. knapp formulieren a += 2, a *= 2 usw., was in anderen Sprachen lauten wüde a = a + 2 bzw. a = a * 2. Die Abarbeitung ist hier 'rechts nach links', es wird also a + 2 gebildet und das Ergebnis a zugewiesen. Gleichartige kombinierte Operationen können mit den übrigen arithmetischen und auch mit den logischen Operatoren vorgenommen werden. - Knapp sind auch Zuweisungsmöglichkeiten wie x=y=sum=0. Auch hier wird von rechts her abgearbeitet, wobei alle Variablen auf 0 gesetzt werden.

In den Assemblersprachen kennen wir das Pre-Inkrement und das Post-Dekrement. In 'C' haben wir gleichartige Operatoren, nämlich ++ und --, Z.B. in ++i, --i, i++, i--. Der Inhalt der Variablen i wird dabei um 1 verändert. Wenn ein solcher Operator in Ausdrücken vorkommt, dann stellt sich die Frage, wann das i bewertet wird. Bei vorangestelltem Operator wird zunächst verändert und erst dann bewertet. Bei nachgestelltem Operator ist es umgekehrt.

Es gibt weitere Zuweisungsmöglichkeiten mit einer sehr kurzen Schreibweise, z.B.

```
while ((a=getchar()) != '\n' ++i)
    s[i] = c;
```

Diese Zeilen sind wie folgt zu lesen: '\n' ist die Schreibweise für das übliche Carriage Return und getchar() ist eine Bibliotheksfunktion, die ein Zeichen von der aktiven Eingabe holt. Also: Hole ein Zeichen und weise es c zu. das ist der Teil c=getchar(). Diese Tätigkeit ist (while) solange auszuführen, wie das Zeichen ungleich (!=) CR ist. In jedem Durchgang ist i zunächst um 1 zu erhöhen und es ist das empfangene Zeichen c an die durch i bezeichnete Stelle in das Array s einzustellen. Es wird also ein String bis zum Auftreten eines 'CR' eingelesen. - Kombinierte Zuweisungen und Abfragen erlauben eine sehr prägnante Schreibweise.

In 'C' stehen auch die üblichen logischen Operatoren zur Verfügung, die 'wahr' oder 'nicht wahr' abliefern. Erwähnenswert sind die Operatoren für bitweises Arbeiten für die logischen Operationen AND, OR und EXOR. Daneben gibt es Operatoren für das Rechts- und Linksverschieben von Operanden um eine anzugebende Zahl Bitstellen. Damit bietet die Sprache die Möglichkeit, mit Bitlisten und mit Interfaces bequem zu arbeiten. - Interessant ist auch der Bedingungsoperator '?', der wie if .. else eine Zweiwege-Entscheidung bei Zuweisungen erlaubt.

Bibliotheksfunktionen und Hilfsfiles

Es wurde schon erwähnt, daß 'C' in der Beschreibung von Kernighan und Ritchie 28 Schlüsselworte hat, unter denen sich keine für die Ein- und Ausgabe befinden. Gleichwohl ist 'C' eine reich ausgestattete Sprache, und zwar durch die Programmbibliothek mit dem Namen clib, die je nach Implementierung ca. einhundert Funktionen zur Verfügung stellt, darunter etliche für die Ein- und Ausgabe, auch in formatierter Form mit scanf() und printf(). Es sind alle notwendigen Funktionen für das Arbeiten mit Files vorhanden, für die Zeichenumwandlung, für die Stringbearbeitung, für die Klassifizierung von ASCII-Zeichen, für die Speicherreservierung und -freigabe usw. Bei der Ausgabe können Zahlen dezimal, hexadezimal oder auch oktal dargestellt werden.

Das 'C'-Paket wird durch zahlreiche Standard-Hilfsdateien unterstützt, mit denen man Makros schreiben kann, die zu einer Textersetzung im Quelltext führen. Ebenso können mit #include beliebige eigene Hilfsdateien mit Definitionen oder Formulierungen in die Compilierung einbezogen werden. Die Compilierung kann auch bedingt erfolgen mit #ifdef und #ifndef, wenn etwas schon oder noch nicht definiert worden ist.

Zusammenfassung

'C' ist mit der zunehmenden Verbreitung moderner 16 Bit-Personal Computer im Kommen und leichter erreichbar. Es ist eine Sprache, die auf längere Zeit Portabilität von Software verspricht, die auf verschiedenen Computer-Jahrgängen und -Modellen zum Laufen gebracht werden kann. Die Sprache hat ein sehr maschinennahes Interface, das es insbesondere über die indirekte Pointer-Adressierung erlaubt, jedes Objekt zu erreichen oder zu manipulieren. Mit der Möglichkeit, Standardfunktionen aus der Bibliothek zu entnehmen und beliebige Hilfsdateien auch bedingt in die Compilierung einzubeziehen, steht dem Programmierer ein sehr wirksames Instrument zur Codierung seiner Lösungen zur Verfügung. In manchem erinnert 'C' an eine gut strukturierte Assemblerprogrammierung. Die Sprache dürfte vor allem für die Systemprogrammierung benutzt werden, auch für die Anlage und Verwaltung von Datenbanken, weil Strukturen leicht erreicht werden können.

Wie alle höheren Sprachen so bedingt auch 'C' mehr Code, als eine reine Assemblerprogrammierung. So hat ein leeres Programm, das nur aus main() und dem Akkoladenpaar besteht, immerhin schon 6K Code. In Assembler stünde da nur JSR main und main bestünde nur aus RTS. Aber auch andere kurze Programme laufen schnell zu 10 K Code auf. Das verlangt nicht nur ausreichende Ausstattung des Computers mit Speicher, sondern auch die Beobachtung des Zeitverhaltens von Programmen.

Es bleibt, noch kurz auf die Erlernbarkeit von 'C' einzugehen: Die syntaktischen Regeln der Sprache sind einfach. Man gewöhnt sich schnell daran, wo Klammern der verschiedenen Arten paarweise zu benutzen sind, daß Erklärungen der Benutzung von Namen vorausgehen müssen und auch daran, wo ein Semikolon als Begrenzer eines Statements zu setzen ist. Die innere Logik der Sprache ist einfach, und wir haben ja nur wenige Schlüsselworte. Etwas mehr Aufmerksamkeit verlangen die Bibliotheksfunktionen. Für sie sollte eine gute Dokumentation zur Verfügung stehen, die zeigt, welche Parameter von welchem Typ beim Aufruf übergeben werden müssen, was sie leisten und was sie als Ergebnis mit welchem Datentyp abliefern. Mit diesen Unterlagen und einem guten Lehrbuch neben dem Computer ist man nach einigen Tagen Übung in der Lage, schon einfache Programme in 'C' zu formulieren, ohne daß der Compiler allzuoft Anstoß nimmt.

□○□

Roland Löhr

Literatur zu 'C'

Wenn man sich in eine Sprache einarbeiten will, dann sollte man nicht nur eine gute Implementierung auf dem Computer zur Verfügung haben, sondern auch eine ausreichende Dokumentation zur Implementierung und geeignete Literatur. Der Autor bespricht zur Orientierung der Leser die Quellen, die ihm zur Verfügung standen:

Brian W. Kernighan und Dennis M. Ritchie: The C Programming Language. Verlag Prentice Hall, Englewood Cliffs 1978, 228 S., ISBN 0-13-110163-3.

Jack Purdum: C Programming Guide. Que-Verlag, Indianapolis 1983, 250 S., ISBN 0-88022-022-8.

Jack Purdum: Einführung in C. Verlag Markt und Technik, Haar 1983, 304 S., ISBN 3-922120-49-0. Deutsche Übersetzung vorstehenden Titels durch Peter Lüke.

Zdenek Stanka und Stefan Lösch: Die C-Sprache. te-wi-Verlag, München 1984, 288 S., ISBN 3-921 803-28-4.

Al Kelley und Ira Pohl: A Book on C. Verlag: Benjamin/Cummings Publishing Co., Menlo Park 1984, 362 S., ISBN 0-8053-6840-4.

Wie die nachfolgende Besprechung noch zeigt, kann man dem Leser die Bücher in folgender Reihenfolge für die gründliche Einarbeitung empfehlen: A Book on C, dann den Pudum im englischen Original und schließlich Kernighan und Ritchie als Handbuch mit der Beschreibung der Sprache.

Ritchie gilt mit seinen Arbeiten an den Bell-Laboratorien als Schöpfer der Sprache. Sie wurde zuerst auf PDP-Maschinen implementiert. Das Buch von Kernighan und Ritchie wird im Fachjargon

'K&R' genannt. In dieser Zeitschrift wurde es bereits kurz in Heft 34 besprochen. Es ist klar geschrieben und entsprechend zu lesen. Es ist aber nicht unbedingt ein Lehrbuch, sondern mehr ein 'Reference Guide', ein Handbuch. Wenn man sich mit diesem Buch einarbeiten will, dann vermißt man schnell die Beispiele, auch die einfachen, die ja immer sehr zur Übung beitragen.

Im 'K&R' findet man nach einer Übersicht folgende Kapitel: Typen, Operatoren und Ausdrücke, Ablaufkontrolle, Funktionen und Programmstrukturen, Pointer und Arrays, Strukturen, Ein- und Ausgabe, Das Interface zu UNIX und schließlich das C Reference Manual. Vor allem dieser letzte Abschnitt gibt dem schon eingearbeiteten Benutzer eine schnelle Informationsquelle in Zweifelsfragen.

Jack Purdum ist Chef eines Software-Hauses. Auch sein Buch ist klar geschrieben und wohl weniger 'akademisch' abgehalten. Es ist mit vielen Beispielen und einfachen Grafiken unterlegt und daher für die einübende Benutzung geeignet. Nach der Einführung finden wir dort folgende Kapitel: Operatoren, Variable und Schleifen, Zahlensysteme, Wie man eigene Funktionen schreibt, Die Benutzung von Pointern, Ein- und Ausgabe, Andere Datentypen, Strukturen und Unions, Das Arbeiten mit Files auf Disketten. Der Anhang mit der Zusammenfassung der Syntax ist dagegen weniger prägnant herausgearbeitet.

Die deutsche Übersetzung des Purdum kann nicht empfohlen werden. Man soll sich bei einem Arbeitsbuch zwar nicht darüber aufregen, daß der Text im Schreibmaschinensatz gesetzt ist und die vielen hilfreichen druck-grafischen Hervorhebungen der amerikanischen Vorlage vermissen läßt, aber aufregen muß man sich über die liederliche Übersetzung mit undeutlichen Formulierungen und Fortlassungen, wo der Übersetzer mangels Verständnis offensichtlich sprachliche Schwierigkeiten hatte.

Das Buch von Stanka und Lösch stammt von deutsch-sprachigen Autoren. Es ist klar formuliert. Schwerpunkt der Darstellungen ist die Beschreibung der Logik der C-Sprache. Zu diesem Zweck werden kürzere Programmsequenzen in vielen Variationen ausgearbeitet und mit Programm-Ablaufplänen jeweils unterlegt und in den wichtigen Formulierungen zeilenweise besprochen. Eine erschöpfende Darstellung der Sprache ist nicht beabsichtigt, die Logik kommt aber gut heraus.

Dem Rezensenten hat das Buch 'A Book on C' von Kelley und Pohl am besten gefallen und am meisten geholfen, es ist mit 362 sorgfältig aufgemachten und gegliederten Textseiten das umfangreicheste der besprochenen Bücher, es enthält damit mehr Information, viele kurze Programmsequenzen und dutzende vollständiger Programme. In guter didaktischer Darstellung decken diese verständliche ernsthafte und spielerische Aufgabestellungen ab, um die Einsatzfähigkeit der Sprache vorzuführen. Auch hier wird die Wirkung der einzelnen Statements besprochen, auch in Alternativen. Die Syntax der Sprache wird bei den Anwendungen in den einzelnen Kapiteln erläutert, nicht in einem Anhang. So findet man die Information beieinander. Übersichtstabellen und Grafiken verbessern den Extrakt. An die einzelnen Kapitel sind insgesamt über 200 Übungen angeschlossen, die zum eigenen Experimentieren anregen.

Nach einem zusammenfassenden Überblick finden wir bei Kelley und Pohl folgende Kapitel: Zur Syntax, Deklarationen, Ausdrücke, Zuweisungen und Datentypen, Ablaufkontrolle, Funktionen, Verzweigungen, bitweise Verarbeitung und Aufzählungstypen, Pointer, Arrays und Strings, Rekursionen, Funktionen als Argumente, Der Preprozessor, Strukturen, Unions und Typerklärungen, Strukturen und Listenverarbeitung, Ein- und Ausgabe und die UNIX-Umgebung. - Es ist sehr zu wünschen, daß dieses Buch bald in deutscher Übersetzung erscheint.

Weitere Literaturhinweise: Mit dem Betriebssystem CP/M-68K liefert Digital Research auch eine umfangreiche Dokumentation von schätzungsweise 600 Seiten. Sie enthält auf etwa 90 Seiten den Abschnitt 'The C Programming Guide for CP/M-68K'. Dieser Abschnitt belegt vor allem die in der CLIB (library) mitgelieferten Funktionen mit den Ein- und Ausgangsbedinungen und ihrer Wirkung. Diese Dokumentation wurde in Heft 39 ab Seite 20 besprochen. Dem Rezensenten ist nicht bekannt, auf welchem Weg diese Dokumentation in der Distribution ist. Man wird sie jedoch benötigen, wenn man CP/M-68K und 'C' auf einem der in den großen Publikumszeitschriften veröffentlichten Nachbaukomputer oder z.B. auf dem neuen Atari 520ST mit Gewinn betreiben will.

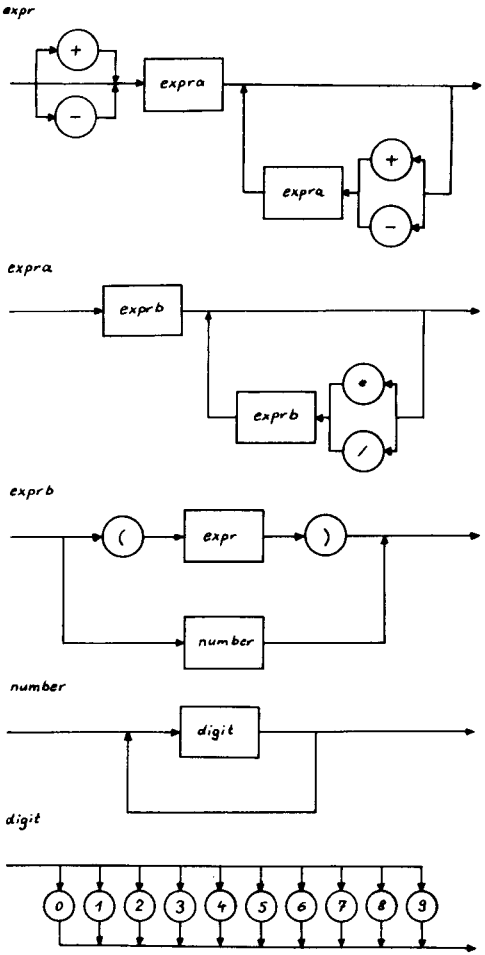
Eine gute Übersicht in Deutsch erschien mit dem Titel 'Die Sprache C' in den Heften 2/83, 3/83 und 4/83 der Elektronik, und zwar aus der Feder von Bernd Pohl.

Ewald Scherer, 1000 Berlin 61

Einfacher Formelinterpreter

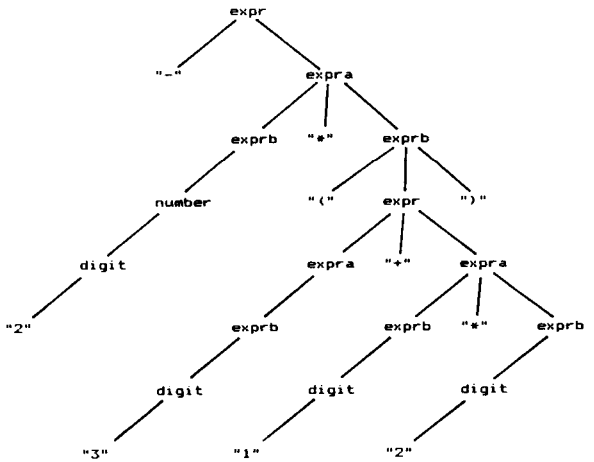
für die 68000-Familie

Die Bezeichnung 'Interpreter für arithmetische Ausdrücke' würde das hier vorgestellte Programm zwar zutreffender benennen, der Einfachheit halber wird der Name 'Formelinterpreter' benutzt. Dieser Formelinterpreter wurde vor seiner Implementierung in 68000-Assembler in Form eines Syntax-Diagramms dargestellt (s. Abb.). - Syntaxdiagramme werden benutzt, um moderne Programmiersprachen wie PASCAL und C darzustellen. Ein Bild sagt da mehr als tausend Worte. Syntaxdiagramme kommen der menschlichen Auffassungsgabe sehr entgegen. Eine knappe Beschreibung von Syntaxdiagrammen findet sich in dem Büchlein 'Compilerbau' von Prof. N. Wirth.



Syntax-Diagramm

Ableitung für den Ausdruck -2*(3+1*2)



" " terminale Zeichen

Unser Formelinterpreter beherrscht Punkt- und Strichrechnung in der bekannten mathematischen Hierarchie sowie die Klammerung von Ausdrücken. Die mögliche Schachtelungstiefe für die Klammerung ist nur durch die Größe des Stacks der konkreten Maschine begrenzt, auf der der Interpreter läuft.

Für die mathematischen Operationen Addition, Subtraktion, Multiplikation und Division werden in dem hier vorgestellten Programm die entsprechenden Maschinenbefehle des MC 68000 benutzt. Der Wertebereich ist daher auf 32 Bit Integer beschränkt, wobei für die Multiplikation und Division weitere Einschränkungen gelten (siehe MC 68000 User's Manual). Ein Überlauf des Wertebereiches wird bei Addition und Subtraktion, sowie eingeschränkt bei der Division, durch Testen des V-Bits im Condition Code Register des MC 68000 erkannt. Anstelle des verwendeten Befehls BVS könnte auch ein TRAPV stehen. Dann müßte allerdings der entsprechende Exception Vector auf eine Fehlerbehandlungsroutine zeigen.

Eine Erweiterung des Formelinterpreters auf Floating Point und triginometrische Funktionen ist ohne großen Aufwand möglich, wobei das 'Fast Floating Point Package' von Motorola gute Dienste leisten kann. Dieses FFFP arbeitet mit 32 Bit Floating Point-Zahlen mit folgender Darstellung:

```
-------------------------------------------------------------
M M M M M M M M.M M M M M M M M.M M M M M M M M.S E E E E E E E
-------------------------------------------------------------
31              23              15              7             0

   M - 24 bit mantissa
   S - sign of the number
   E -  7 bit exponent in excess 64 notation
```

Hier noch einige Ausführungszeiten (8 MHz MC68000, keine Wartezyklen):

Addition	: 5,25 - 54,38	Mikrosekunden
Subtraktion	: 4,25 - 55,63	"
Multiplikation	: 3,75 - 51,75	"
"	: 3,75 - 61,5	"
Division	: 5,25 - 85,0	"
Sinus	: 413	"
Cosinus	: 409	"
Tangens	: 501	"

Der Formelinterpreter arbeitet beim Auftreten von Klammerungen rekursiv, die bereits berechneten Werte der aktuellen Inkarnation werden dabei auf den Stack gerettet.

Um einen 'Nachbau' des Interpreters durch einen möglichst großen Leserkreis zu ermöglichen, folgt ein vollständiges Listing in der Assembler-Syntax des MC 68000. Das Syntaxdiagramm ist in der Struktur des Assemblerprogrammes leicht wiederzuerkennen. — In einer späteren Ausgabe des MICRO MAG wird der Autor diesen Formelinterpreter mit allerdings erheblich erweitertem Leistungsumfang, vollständig in der Sprache 'C' geschrieben, vorstellen.

Literaturhinweise:

MC 68000 16 Bit Microprocessor User's Manual; Motorola Inc.
Compilerbau, N. Wirth, Teubner Studienbücher
pascal User's manual and report, K. Jensen, N. Wirth; Springer Verlag

```
C P / M   6 8 0 0 0   A s s e m b l e r          Revision 02.01          Page   1
Source File: m:formel.s

  1
  2   ************************************************************
  3   *
  4   *       "Formel" - Interpreter fuer 68 000 Familie
  5   *
  6   *
  7   *       Verwendung, auch auszugsweise , fuer gewerbliche Zwecke
  8   *       nur mit schriftlicher Genemigung des Autors gestattet.
  9   *
 10   *       5.4.85
 11   *       erhard scherer , zimmerstr.12 , 1000 berlin 61
 12   *       030/ 392 30 11        ;        030 / 251 39 35
 13   *
 14   ************************************************************
 15
 16
 17
 18 0000 23CF00000064      move.l  sp,save   sp fuer errorroutine retten
 19 0006 41F900000000      lea     buffer,a0
 20 000C 61000148          bsr     getstr  zeichenkette einlesen
 21 0010 611C              bsr     expr    arith. ausdruck berechnen
 22 0012 41F900000000      lea     buffer,a0
 23 0018 610001AA          bsr     putdezi binaer nach BCD ASCII
 24 001C 703D              moveq   #'=,d0 geichheitszeichen
 25 001E 61000226          bsr     outchar
 26 0022 7020              moveq   #' ,d0
 27 0024 61000220          bsr     outchar
 28 0028 610001D0          bsr     outstr  zeichenkette ausgeben
 29 002C 4E75              rts
 30
 31   ************************************************************
 32   *
 33   *       expr
 34   *
 35   ************************************************************
 36
 37 002E 0C10002D expr     cmpi.b  #'-',(a0) negative zahl ?
 38 0032 6608              bne     expr1
 39 0034 5288              addq.l  #1,a0   zeiger auf naechstes zeichen
 40 0036 613E              bsr     expra   wert von expra
```

```
 41 0038 4487              neg.l   d7      einerkomplement
 42 003A 600A              bra     expr3
 43
 44 003C 0C10002B expr1    cmpi.b  #'+',(a0) positive zahl ?
 45 0040 6602              bne     expr2
 46 0042 5288              addq.l  #1,a0
 47
 48 0044 6130      expr2   bsr     expra   wert fuer pos. zahl von expra
 49
 50 0046 0C10002D expr3    cmpi.b  #'-',(a0) subtraieren ?
 51 004A 6612              bne     expr4
 52 004C 5288              addq.l  #1,a0
 53 004E 2F07              move.l  d7,-(sp)        wert kellern
 54 0050 6124              bsr     expra
 55 0052 2C1F              move.l  (sp)+,d6        wert aus keller
 56 0054 CD47              exg     d6,d7
 57 0056 9E86              sub.l   d6,d7 subtraieren der werte
 58 0058 69000086          bvs     error3  ueberlauf
 59 005C 60E8              bra     expr3   weitere subtraktion/addition
 60
 61 005E 0C10002B expr4    cmpi.b  #'+',(a0) addieren ?
 62 0062 6610              bne     exprrt
 63 0064 5288              addq.l  #1,a0
 64 0066 2F07              move.l  d7,-(sp) wert akt. inkarnation kellern
 65 0068 610C              bsr     expra
 66 006A 2C1F              move.l  (sp)+,d6 wert aus keller
 67 006C DE86              add.l   d6,d7   addieren der werte
 68 006E 69000070          bvs     error3  ueberlauf
 69 0072 60D2              bra     expr3
 70
 71 0074 4E75      exprrt  rts
 72
 73   ***************************************************************
 74   *
 75   *       expra
 76   *
 77   ***************************************************************
 78
 79 0076 612A      expra   bsr     exprb   wert von expra
 80
 81 0078 0C10002A expra1   cmpi.b  #'*',(a0) multiplikation ?
 82 007C 660C              bne     expra2
 83 007E 5288              addq.l  #1,a0
 84 0080 2F07              move.l  d7,-(sp) wert der akt. inkarnation kellern
 85 0082 611E              bsr     exprb
 86 0084 2C1F              move.l  (sp)+,d6 wert aus keller
 87 0086 CFC6              muls    d6,d7
 88 0088 60EE              bra     expra1
 89
 90 008A 0C10002F expra2   cmpi.b  #'/',(a0) division ?
 91 008E 6610              bne     exprarts
 92 0090 5288              addq.l  #1,a0
 93 0092 2F07              move.l  d7,-(sp) wert der akt. inkarnation retten
 94 0094 610C              bsr     exprb
 95 0096 2C1F              move.l  (sp)+,d6 wert aus keller
 96 0098 CD47              exg     d6,d7
 97 009A 8FC6              divs    d6,d7
 98 009C 6942              bvs     error3  ueberlauf
 99 009E 60D8              bra     expra1
100
101 00A0 4E75      exprarts        rts
102
103   ***************************************************************
104   *
105   *       exprb
106   *
107   ***************************************************************
108
109 00A2 0C100028 exprb   cmpi.b  #'(',(a0) klammerausdruck ?
110 00A6 660E              bne     exprb1
```

```
111 00A8 5288              addq.l  #1,a0
112 00AA 6182              bsr     expr
113 00AC 0C100029          cmpi.b  #')',(a0)
114 00B0 660E              bne     error1
115 00B2 5288              addq.l  #1,a0
116 00B4 4E75              rts
117
118 00B6 610000D2 exprb1   bsr     getdezi hole wert number
119 00BA 4A87              tst.l   d7      fehler ?
120 00BC 6B12              bmi     error2
121 00BE 4E75              rts
122
123   ************************************************************
124   *
125   *       fehlerbehandlung
126   *
127   ************************************************************
128
129 00C0 2E7900000064          error1  move.l  savesp,sp
130 00C6 43F900000118      lea     errmsg1,a1
131 00CC 6122              bsr     errout
132 00CE 4E75              rts
133
134 00D0 2E7900000064          error2  move.l  savesp,sp
135 00D6 43F900000134      lea     errmsg2,a1
136 00DC 6112              bsr     errout
137 00DE 4E75              rts
138
139 00E0 2E7900000064          error3  move.l  savesp,sp
140 00E6 43F900000149      lea     errmsg3,a1
141 00EC 6102              bsr     errout
142 00EE 4E75              rts
143
144 00F0 2E08      errout  move.l  a0,d7   zeiger auf fehler
145 00F2 2C3C00000000      move.l  #buffer,d6
146 00F8 9E86              sub.l   d6,d7 errcount
147 00FA 6710              beq     erromsg
148 00FC 7020              moveq   #$20,d0
149 00FE 61000146 ermsl    bsr     outchar stelle auf schirm anfahren
150 0102 5387              subq.l  #1,d7
151 0104 66F8              bne     ermsl
152 0106 705E              moveq   #$5e,d0
153 0108 6100013C          bsr     outchar stelle markieren
154 010C 1019      erromsg move.b  (a1)+,d0
155 010E 6706              beq     errort
156 0110 61000134          bsr     outchar
157 0114 60F6              bra     erromsg
158 0116 4E75      errort  rts
159
160   ************************************************************
161   *
162   *       fehlermeldungen
163   *
164   ************************************************************
165
166 0118 207363686C696573      errmsg1 dc.b    " schliessende klammer fehlt",0
166 0120 73656E6465206B6C
166 0128 616D6D6572206665
166 0130 686C7400
167 0134 207A61686C207A75      errmsg2 dc.b    " zahl zu gross/klein",0
167 013C 2067726F73732F6B
167 0144 6C65696E00
168 0149 2075656265726C61      errmsg3 dc.b    " ueberlauf",0
168 0151 756600
169 0154 0000              dc.w    0
170
171   ************************************************************
172   *
173   *       getstr
174   *
175   *       liest einen ASCII String von inoutch
176   *       zeichenweise in den buffer
```

```
178   *       in  : a0 zeiger auf buffer
179   *
180   ************************************************************
181
182
183 0156 48E78080 getstr  movem.l d0/a0,-(sp)
184 015A 610000DE gs1     bsr     inoutch hole zeichen
185 015E 0C000008         cmpi.b  #8,d0   korrektur ?
186 0162 6716             beq     gskorr
187 0164 10C0             move.b  d0,(a0)+        zeichen nach buffer
188 0166 0C00000D         cmpi.b  #$d,d0  return ?
189 016A 66EE             bne     gs1
190 016C 700A             moveq   #$a,d0
191 016E 610000D6         bsr     outchar LF ausgeben
192 0172 4210     gsrt    clr.b   (a0)
193 0174 4CDF0101         movem.l (sp)+,d0/a0
194 0178 4E75             rts
195 017A 7020     gskorr  moveq   #$20,d0
196 017C 610000C8         bsr     outchar
197 0180 7008             moveq   #8,d0
198 0182 610000C2         bsr     outchar
199 0186 5388             subq.l  #1,a0
200 0188 60D0             bra     gs1
201
202   ************************************************************
203   *
204   *       getdezi
205   *
206   *       wandelt einen ASCII String in eine binaere Zahl
207   *
208   *       in  : zeiger auf string in A0
209   *       out : wert in D7.L
210   *       error : -1 in D7.L
211   *
212   ************************************************************
213
215 018A 48E77000 getdezi movem.l d1-d3,-(sp) register retten
216 018E 4287             clr.l   d7      anfangswert
217 0190 4282             clr.l   d2
218 0192 7209             moveq   #9,d1
219 0194 1418     gdl     move.b  (a0)+,d2 zeichen von string
220 0196 04020030         subi.b  #$30,d2 ASCII nach BCD
221 019A 651A             bcs     gdret   kein dezi wert
222 019C 0C02000A         cmpi.b  #10,d2
223 01A0 6414             bcc     gdret   kein BCD wert
224 01A2 2607             move.l  d7,d3   alten wert sichern
225 01A4 DE87             add.l   d7,d7   wert mal 2
226 01A6 E78B             lsl.l   #3,d3   wert mal 8
227 01A8 DE83             add.l   d3,d7   mal8 + mal2 = mal10
228 01AA DE82             add.l   d2,d7   neues digit addieren
229 01AC 5301             subq.b  #1,d1   stellenzaehler
230 01AE 6AE4             bpl     gdl
231 01B0 4287     gderr   clr.l   d7      zu viele stellen
232 01B2 5387             subq.l  #1,d7 -1 errorcode
233 01B4 6008             bra     gdrts
234 01B6 5388     gdret   subq.l  #1,a0 zeiger auf letztes zeichen
235 01B8 0C010009         cmpi.b  #9,d1   keine zahl ?
236 01BC 67F2             beq     gderr
237 01BE 4CDF000E gdrts   movem.l (sp)+,d1-d3
238 01C2 4E75             rts
239
240   ************************************************************
241   *
242   *       putdezi
243   *
244   *       wandelt eine binaerzahl nach BCD ASCII
245   *
246   *       in  : d7 wert
247   *             a0 zeiger auf buffer
248   *
249   ************************************************************
```

```
252 01C4 48E7F180 putdezi movem.l d0-d3/d7/a0,-(sp) werte retten
253 01C8 4280             clr.l   d0      vorzeichenmerker
254 01CA 7609             moveq   #9,d3
255 01CC 720A             moveq   #10,d1  divisor
256 01CE 4A87             tst.l   d7      negative zahl ?
257 01D0 6A04             bpl     pdl
258 01D2 4487             neg.l   d7
259 01D4 702D             moveq   #$2d,d0 vorzeichen
260 01D6 4A87     pdl     tst.l   d7      fertig ?
261 01D8 670C             beq     pdrt0
262 01DA 6144             bsr     divu32
263 01DC 06020030         addi.b  #$30,d2 zu ASCII wandeln
264 01E0 10C2             move.b  d2,(a0)+ zeichen nach buffer
265 01E2 5383             subq.l  #1,d3
266 01E4 60F0             bra     pdl
267 01E6 4A80     pdrt0   tst.l   d0      neg.vorzeichen ?
268 01E8 6704             beq     pdrt
269 01EA 10C0             move.b  d0,(a0)+
270 01EC 5383             subq.l  #1,d3
271 01EE 4218     pdrt    clr.b   (a0)+
272 01F0 5383             subq.l  #1,d3
273 01F2 6AFA             bpl     pdrt
274 01F4 4CDF018F         movem.l (sp)+,d0-d3/d7/a0
275 01F8 4E75             rts
276
277   ************************************************************
278   *
279   *       outstr
280   *
281   *       gibt einen ASCII string aus
282   *
283   *       in  : zeiger auf string in a0
284   *
285   ************************************************************
286
287 01FA 48E7C080 outstr  movem.l d0-d1/a0,-(sp) register retten
288 01FE 7209             moveq   #9,d1
289 0200 10301800 osl     move.b  (a0,d1.l),d0
290 0204 660E             bne     oso
291 0206 5381             subq.l  #1,d1
292 0208 6AF6             bpl     osl
293 020A 7030             moveq   #$30,d0
294 020C 6138             bsr     outchar zahl ist gleich 0
295 020E 4CDF0103 osrt    movem.l (sp)+,d0-d1/a0
296 0212 4E75             rts
297 0214 6130     oso     bsr     outchar
298 0216 5381             subq.l  #1,d1
299 0218 6BF4             bmi     osrt
300 021A 10301800         move.b  (a0,d1.l),d0
301 021E 60F4             bra     oso
302
303   ************************************************************
304   *
305   *       divu32
306   *
307   *       32 bit division ohne vorzeichen
308   *
309   *       in  : divisor d1
310   *             divident d7
311   *       out : ergebnis  d7
312   *             rest      d2
313   *
314   ************************************************************
315
316
317 0220 2F03     divu32  move.l  d3,-(sp)
318 0222 7620             moveq   #32,d3
319 0224 4282             clr.l   d2
320 0226 E387     divul   asl.l   #1,d7
321 0228 E392             roxl.l  #1,d2
322 022A B481             cmp.l   d1,d2
```

```
022C 6504              bcs      divuchk
022E 9481              sub.l    d1,d2
0230 5287              addq.l   #1,d7
0232 5383     divuchk  subq.l   #1,d3
0234 66F0              bne      divul
0236 261F              move.l   (sp)+,d3
0238 4E75              rts


0000                   data


0000          buffer   ds.b     100      buffer fuer ein/ausgabe - string
0064          savesp   ds.l     1        stack pointer retten
```

extern: inoutch, outchar

Klaus-Dieter Kossow, 2074 Mollhagen

RENUMBER und FIND

in BASIC-Programmen

Dieses für den CBM 710/720 geschriebene Programm liegt in Form einer Befehlserweiterung in der Systembank 15 im Bereich $0400-07FC. Es wird durch SYS(1024) erstmalig initialisiert.

find,suchstring durchsucht ein BASIC-Programm und gibt die gefundenen Zeilen, die diesen String enthalten, auf den Bildschirm aus. Zu beachten ist hierbei, daß die Eingabe des Suchstings die Token-Umwandlungsroutine durchläuft, so daß hier Einschränkungen in Kauf genommen werden müssen. Probleme können z.B. entstehen bei #, $ oder Klammer. INPUT# und PRINT# werden als ein Byte abgespeichert, deshalb werden diese Befehle durch INPUT oder PRINT allein nicht gefunden. - Stringfunktionen können auch durch $ nicht gefunden werden, ebenso wie TAB(und SPC(durch den Suchstring TAB oder SPC bzw. (allein nicht gefunden werden.

Mit dem Befehl RENUMBER können Teile eines BASIC-Programmes oder ganze Programme andere Zeilennummern erhalten. Sofern nötig, wird der neunumerierte Zeilenbereich an eine andere Stelle des Programmes transportiert. Die Eingabe erfolgt in folgendem Format:

renumber (oder renU) Quellbereich, erste Zielzeile, Zeilen-Inkrement

Der Quellbereich wird wie bei LIST angegeben, kann also z.B. auch ganz entfallen. Dann wird das gesamte Programm umnumeriert. - Die erste Zielzeile gibt an, ab welcher Zeilennummer die Zeilen des Quellbereiches abgelegt werden. Wenn diese Angabe fehlt, dann tritt ein Syntax-Error ein. Das Zeileninkrement bestimmt den Abstand der neuen Zeilennummern. Wenn diese Angabe fehlt, dann wird 10 angenommen. off schaltet das Programm aus.

Zum Programm-Ablauf: Unter renum (Zeilen ab 1000 inder nachstehenden Liste) wird zunächst der Quellzeilenbereich eingelesen und die erste Zeile dieses Bereiches gesucht. Diese Zeile wird in den Eingabepuffer kopiert (ab Zeile 5000). Es wird geprüft, ob die neue Zeilennummer bereits belegt ist (dann overflow error). Danach wird die alte Zeile im Programm gelöscht und die neue Zeile an der richtigen Stelle eingefügt. Die alte und die neue Zeilennummer werden auf den Bildschirm ausgegeben.

Sodann wird das gesamte Programm durchsucht, ob die alte Zeilenummer in irgendweiner Weise als Sprungziel auftaucht. Das kann z.B. der Fall sein bei den Befehlen goto, gosub, then, else, trap, restore, resume und run. Zeilen mit solchen Bezügen werden entsprechend geändert. Nachdem die letzte Zeile des Quellbereiches verarbeitet, die Link-Zeiger neu berechnet und nachdem die BASIC-Zeiger neu gesetzt wurden, erfolgt der Rücksprung über READY in den Eingabe-Modus.

Im Programmteil find wird der Suchstring zunächst durch die Token-Routine bearbeitet, danach wird der Suchbereich eingelesen. Dieser Teil konnte jedoch aus Platzmangel im RAM nicht ab-

schließend ausgebaut werden, so daß diese Angabe noch entfallen kann. Alle Zeilen, die kürzer als der Suchstring sind, werden ohne weitere Prüfung übergangen. Eine Zeile, die den Suchstring enthält wird in der Programm-Liste ab Zeile 10225 auf den Bildschirm ausgegeben.

```
 10  1 0000 0000                    ;ren/find 710 src
 20  1 0000 0000                    ;*********************************
 25  1 0000 0000                    ;* renumber/find f. 710           *
 30  1 0000 0000                    ;* initialisierung d.             *
 35  1 0000 0000                    ;* sys1024                        *
 36  1 0000 0000                    ;* bload"re/find710"onb15,p1024   *
 40  1 0000 0000                    ;* aufruf                         *
 45  1 0000 0000                    ;* renumber ba-be,a,in            *
 50  1 0000 0000                    ;*      ba = bereich-anf.         *
 55  1 0000 0000                    ;*      be = breich-ende          *
 56  1 0000 0000                    ;*      (wie bei list)            *
 60  1 0000 0000                    ;*      a  = neuer anfang         *
 65  1 0000 0000                    ;*      in = zeilen-inkrement     *
 70  1 0000 0000                    ;* find,suchstring                *
 71  1 0000 0000                    ;* off                           *
 72  1 0000 0000                    ;*      find und renumber aus     *
 75  1 0000 0000                    ;*********************************
100  1 0000 0000                    ;system-adr.
105  1 0000 0000          lowtr     =$6d
110  1 0000 0000          linnum    =$1b
115  1 0000 0000          fndlin    =$871f;basic-zeile suchen
120  1 0000 0000          index1    =$22
125  1 0000 0000          index2    =$25
130  1 0000 0000          ready     =$85c0
135  1 0000 0000          lnkprg    =$86a4;link-adressen berechnen
140  1 0000 0000          runc      =$8a40;basic-zeiger init. (clr)
145  1 0000 0000          txtend    =$2f
150  1 0000 0000          buffpt    =$88
155  1 0000 0000          count     =$0e
165  1 0000 0000          error     =$8552;fehlermeld. ausg.
170  1 0000 0000          chrgot    =$ba29;letztes zeichen nocheinmal
175  1 0000 0000          chrget    =$ba26;naechstes zeichen
180  1 0000 0000          linget    =$8d4e;zn umw. in adr.-form.
185  1 0000 0000          txttab    =$2d;anf. basic-text
190  1 0000 0000          mapusr    =$ba8c;umschalten bank 2
195  1 0000 0000          buf       =$200
200  1 0000 0000          addon     =$8ce2;textzeiger + yr
205  1 0000 0000          qnum      =$ba50;test numerisch
210  1 0000 0000          txtptr    =$85
240  1 0000 0000          ocrlf     =$8ec8;ausgabe cr/lf
250  1 0000 0000          linprt    =$a3b4;umwandlung hex/dez
255  1 0000 0000          curlin    =$42
260  1 0000 0000          ochr      =$b53a;ein zeichen ausgeben
265  1 0000 0000          ldily     =$ba64
270  1 0000 0000          dores     =$13
275  1 0000 0000          lstpnt    =$54
280  1 0000 0000          icrnch    =$0284;sprung zur token-routine
285  1 0000 0000          resel     =$895b
290  1 0000 0000          tstdir    =$9d57;test direkt-modus
295  1 0000 0000          ncnnch    =$88c2;token-routine
305  1 0000 0000          main      =$85ca;interpreter-schleife
310  1 0000 0000                    ;-------------------------------------
500  1 0000 0000                    *=$0400
     2 0400 0000                    &=$5450
501  1 0400 5450 4c 21 04 start     jmp init
502  1 0403 5453                    ;-------------------------------------
504  1 0403 5453                    ;variable
505  1 0403 5453          nxtlin    =$49;1. bzw. akt. zn (alt)
506  1 0403 5453          lstlin    =$4b;letzte zn (alt)
507  1 0403 5453          linadr    =$44;adresse akt./next ln (alt)
508  1 0403 5453          nxtadr    =$4d
509  1 0403 5453          actlin    =$46;aktuelle zn
525  1 0403 5453          lincnt    =$4f;zeilen-inkrement
```

```
535  1 0403 5453                basend     =$5b;indikator f. basic-ende
545  1 0403 5453                chgflg     =$57;flag f. zeilenkorrektur
546  1 0403 5453                strlen     =$5a;laenge suchstring
550  1 0403 5453                           ;befehlswort-tabelle
600  1 0403 5453 52 45 4e tab1             ."renumbe",$d2,"of",$c6,"fin",$c4,$00
       0406 5456 55 4d 42
       0409 5459 45 d2 4f
       040c 545c 46 c6 46
       040f 545f 49 4e c4
       0412 5462 00
601  1 0413 5463                           ;einsprung-adressen
602  1 0413 5463 57 04 f0 tab2             .renum,off,find
       0416 5466 07 d3 05
603  1 0419 5469                           ;token-tab.
605  1 0419 5469 89 8a 8c tab3             .$89,$8a,$8c,$8d,$a7,$e1,$e2,$e3
       041c 546c 8d a7 e1
       041f 546f e2 e3
610  1 0421 5471                           ;---------------------------------------
800  1 0421 5471 a2 04          init       ldx #>cmmnd
810  1 0423 5473 a9 2c                     lda #<cmmnd
820  1 0425 5475 8e 85 02                  stx icrnch+1
830  1 0428 5478 8d 84 02                  sta icrnch
840  1 042b 547b 60                        rts;zurueck ins bs
845  1 042c 547c                           ;---------------------------------------
850  1 042c 547c 20 57 9d cmmnd            jsr tstdir;test direkt-modus
855  1 042f 547f f0 03                     beq cmmnd2
860  1 0431 5481 4c c2 88 cmmnd1           jmp ncnnch;ins os
862  1 0434 5484 a5 85    cmmnd2           lda txtptr
863  1 0436 5486 c5 88                     cmp buffpt
864  1 0438 5488 d0 f7                     bne cmmnd1
866  1 043a 548a                           ;akt. text mit befehlswort-tab. vergl.
867  1 043a 548a                           ;einsprung-adr. nach index1
870  1 043a 548a a9 04                     lda #>tab1
875  1 043c 548c a0 03                     ldy #<tab1
880  1 043e 548e 20 5b 89                  jsr resel
885  1 0441 5491 90 ee                     bcc cmmnd1
895  1 0443 5493 20 e2 8c                  jsr addon;txtptr + yr
900  1 0446 5496 06 0e                     asl count
905  1 0448 5498 a6 0e                     ldx count
910  1 044a 549a bd 13 04                  lda tab2,x
915  1 044d 549d 85 22                     sta index1
920  1 044f 549f bd 14 04                  lda tab2+1,x
925  1 0452 54a2 85 23                     sta index1+1
930  1 0454 54a4 6c 22 00                  jmp (index1)
945  1 0457 54a7                           ;---------------------------------------
1000 1 0457 54a7 20 8c ba renum            jsr mapusr;bank 1
1010 1 045a 54aa 20 f1 04                  jsr renrng;bereich lesen
2009 1 045d 54ad a5 49     ren10           lda nxtlin
2010 1 045f 54af a4 4a                     ldy nxtlin+1
2011 1 0461 54b1 85 46                     sta actlin
2012 1 0463 54b3 84 47                     sty actlin+1
2040 1 0465 54b5 20 e2 07                  jsr findln;adr. akt. zeile suchen
2200 1 0468 54b8 20 57 07                  jsr find20
2210 1 046b 54bb 85 22                     sta index1
2220 1 046d 54bd 86 23                     stx index1+1
2230 1 046f 54bf a0 02                     ldy #$02
2270 1 0471 54c1 b1 22                     lda (index1),y
2280 1 0473 54c3 85 49                     sta nxtlin;naechste zn (alt)
2290 1 0475 54c5 c8                        iny
2300 1 0476 54c6 b1 22                     lda (index1),y
2310 1 0478 54c8 85 4a                     sta nxtlin+1
5000 1 047a 54ca                           ;akt. zeile in buffpt
5010 1 047a 54ca a0 00                     ldy #$00
5020 1 047c 54cc a9 01                     lda #$01
5030 1 047e 54ce 91 88                     sta (buffpt),y;ersatz f. link
5032 1 0480 54d0 b1 22                     lda (index1),y;adr. naechste zeile
5034 1 0482 54d2 85 5b                     sta basend
5040 1 0484 54d4 c8                        iny
5045 1 0485 54d5 a9 01                     lda #$01
```

```
5060  1 0487 54d7 91 88                sta (buffpt),y
5062  1 0489 54d9 b1 22                lda (index1),y
5064  1 048b 54db 85 5c                sta basend+1
5070  1 048d 54dd c8                   iny
5080  1 048e 54de a5 42                lda curlin
5090  1 0490 54e0 91 88                sta (buffpt),y
5100  1 0492 54e2 c8                   iny
5105  1 0493 54e3 a5 43                lda curlin+1
5110  1 0495 54e5 91 88                sta (buffpt),y
5120  1 0497 54e7 c8          ren50    iny
5130  1 0498 54e8 b1 6d                lda (lowtr),y
5140  1 049a 54ea 91 88                sta (buffpt),y
5150  1 049c 54ec d0 f9                bne ren50
5160  1 049e 54ee 84 0e                sty count;yr retten
5170  1 04a0 54f0                      ;test ob neue zn bereits belegt
5180  1 04a0 54f0 24 5a                bit strlen
5190  1 04a2 54f2 10 0b                bpl ren51
5191  1 04a4 54f4 a2 00                ldx #$00
5192  1 04a6 54f6 86 5a                stx strlen;flag loeschen
5200  1 04a8 54f8 20 40 8a             jsr runc
5210  1 04ab 54fb a2 32                ldx #$32
5220  1 04ad 54fd d0 3f                bne renerr
6000  1 04af 54ff 20 cc 07 ren51       jsr outlin;alte zeile loeschen
6140  1 04b2 5502 20 7a 07             jsr inlin;neue zeile einfuegen
6550  1 04b5 5505 20 c8 8e             jsr ocrlf
6551  1 04b8 5508 a6 46                ldx actlin
      2 04ba 550a a5 47                lda actlin+1
      3 04bc 550c 20 b4 a3             jsr linprt
6555  1 04bf 550f a6 42                ldx curlin
6560  1 04c1 5511 a5 43                lda curlin+1
6565  1 04c3 5513 20 b4 a3             jsr linprt
6570  1 04c6 5516 20 74 06             jsr fnd100
6590  1 04c9 5519 20 60 07             jsr aut100
6610  1 04cc 551c a5 5b                lda basend
6611  1 04ce 551e 05 5c                ora basend+1
6619  1 04d0 5520 f0 11                beq end
6620  1 04d2 5522 a5 4c                lda lstlin+1;test letzte zeile
6621  1 04d4 5524 c5 4a                cmp nxtlin+1
6630  1 04d6 5526 90 0b                bcc end
6635  1 04d8 5528 d0 06                bne ren135
6640  1 04da 552a a5 4b                lda lstlin
6650  1 04dc 552c c5 49                cmp nxtlin
6660  1 04de 552e 90 03                bcc end
6665  1 04e0 5530 4c 5d 04 ren135      jmp ren10
6670  1 04e3 5533 20 40 8a end         jsr runc;basic-zeiger neu, clr
6675  1 04e6 5536 4c c0 85             jmp ready
8900  1 04e9 5539 20 40 8a synerr      jsr runc
8910  1 04ec 553c a2 2a                ldx #$2a;syntax-error
8930  1 04ee 553e 4c 52 85 renerr      jmp error;fehlermeldung ausg.
9000  1 04f1 5541                      ;bereich lesen
9030  1 04f1 5541 a0 00       renrng   ldy #$00
9031  1 04f3 5543 b1 2d                lda (txttab),y
9032  1 04f5 5545 85 0e                sta count
9034  1 04f7 5547 c8                   iny
9035  1 04f8 5548 b1 2d                lda (txttab),y
9036  1 04fa 554a 05 0e                ora count
9037  1 04fc 554c f0 e5                beq end
9039  1 04fe 554e 20 26 ba             jsr chrget;1. zeichen nocheinmal
9040  1 0501 5551 90 20                bcc rng100
9060  1 0503 5553 c9 2d                cmp #$2d;test "-"
9070  1 0505 5555 f0 0b                beq rng10
9080  1 0507 5557 c9 ab                cmp #$ab;test "-"
9090  1 0509 5559 f0 07                beq rng10
9100  1 050b 555b c9 2c                cmp #$2c;test komma
9110  1 050d 555d f0 03                beq rng10
9120  1 050f 555f 4c e9 04 rngerr      jmp synerr;syntax-error
9148  1 0512 5562 a0 03       rng10    ldy #$03
9149  1 0514 5564 b1 2d                lda (txttab),y;1. verfuegb. zn (alt)
9150  1 0516 5566 aa                   tax
```

```
9160  1 0517 5567 88                dey
9170  1 0518 5568 b1 2d             lda (txttab),y
9200  1 051a 556a 20 e9 07          jsr nxtgot;1. zeichen nocheinmal
9210  1 051d 556d c9 2c             cmp #$2c;test komma
9220  1 051f 556f f0 72             beq rng300
9230  1 0521 5571 d0 1a             bne rng110
9240  1 0523 5573 20 4e 8d rng100   jsr linget;zn in adr.-format umw.
9250  1 0526 5576 a5 1b             lda linnum
9260  1 0528 5578 a6 1c             ldx linnum+1
9290  1 052a 557a 20 e9 07          jsr nxtgot;1. zeichen nocheinmal
9300  1 052d 557d f0 e0             beq rngerr
9310  1 052f 557f c9 2d             cmp #$2d;test "-"
9320  1 0531 5581 f0 0a             beq rng110
9330  1 0533 5583 c9 ab             cmp #$ab;test "-"
9331  1 0535 5585 f0 06             beq rng110
9332  1 0537 5587 c9 2c             cmp #$2c
9333  1 0539 5589 d0 d4             bne rngerr
9334  1 053b 558b f0 13             beq rng115
9350  1 053d 558d 20 26 ba rng110   jsr chrget;naechstes zeichen
9360  1 0540 5590 c9 2c             cmp #$2c;test komma
9370  1 0542 5592 f0 4f             beq rng300
9380  1 0544 5594 18                clc
9390  1 0545 5595 20 4e 8d          jsr linget
9400  1 0548 5598 90 c5             bcc rngerr
9410  1 054a 559a a5 1b             lda linnum
9420  1 054c 559c 05 1c             ora linnum+1
9421  1 054e 559e f0 43             beq rng300
9422  1 0550 55a0 a5 1b    rng115   lda linnum
9423  1 0552 55a2 a6 1c             ldx linnum+1
9424  1 0554 55a4 85 4b             sta lstlin
9425  1 0556 55a6 86 4c             stx lstlin+1
9426  1 0558 55a8 20 29 ba rng120   jsr chrgot
9427  1 055b 55ab c9 2c             cmp #$2c;test komma
9428  1 055d 55ad d0 b0             bne rngerr;syntax error
9429  1 055f 55af 20 26 ba          jsr chrget;naechstes zeichen
9430  1 0562 55b2 b0 59             bcs rngfnd
      2 0564 55b4 20 4e 8d          jsr linget;umw. in adr.-format
9431  1 0567 55b7 90 a6             bcc rngerr;syntax error
9432  1 0569 55b9 a5 1b             lda linnum;1. zn (neu)
9433  1 056b 55bb a6 1c             ldx linnum+1
9435  1 056d 55bd 85 42             sta curlin
9436  1 056f 55bf 86 43             stx curlin+1
9437  1 0571 55c1 20 29 ba          jsr chrgot;1. zeichen nocheinmal
9438  1 0574 55c4 c9 2c             cmp #$2c;test komma
9439  1 0576 55c6 d0 0c             bne rng200
9440  1 0578 55c8 20 26 ba          jsr chrget;naechstes zeichen
9441  1 057b 55cb 20 4e 8d          jsr linget;umw. in adr.-format
9442  1 057e 55ce a5 1b             lda linnum
9443  1 0580 55d0 05 1c             ora linnum+1
9444  1 0582 55d2 d0 06             bne rng210
9445  1 0584 55d4 a9 0a    rng200   lda #$0a;keine angabe, dann 10
9446  1 0586 55d6 a2 00             ldx #$00
9447  1 0588 55d8 f0 04             beq rng220
9449  1 058a 55da a5 1b    rng210   lda linnum
9450  1 058c 55dc a6 1c             ldx linnum+1
9451  1 058e 55de 85 4f    rng220   sta lincnt
9452  1 0590 55e0 86 50             stx lincnt+1
9453  1 0592 55e2 60                rts
9454  1 0593 55e3                   ;letzte verfuegb. zn suchen
9455  1 0593 55e3 a5 2d    rng300   lda txttab
9460  1 0595 55e5 85 22             sta index1
9470  1 0597 55e7 a5 2e             lda txttab+1
9480  1 0599 55e9 85 23             sta index1+1
9490  1 059b 55eb a0 03    rng310   ldy #$03
9500  1 059d 55ed b1 22             lda (index1),y
9510  1 059f 55ef 85 4c             sta lstlin+1
9520  1 05a1 55f1 88                dey
9530  1 05a2 55f2 b1 22             lda (index1),y
9540  1 05a4 55f4 85 4b             sta lstlin
```

```
 9545  1 05a6 55f6 88                dey
 9550  1 05a7 55f7 b1 22             lda (index1),y
 9560  1 05a9 55f9 aa                tax
 9570  1 05aa 55fa 88                dey
 9580  1 05ab 55fb b1 22             lda (index1),y
 9590  1 05ad 55fd 85 22             sta index1
 9600  1 05af 55ff 86 23             stx index1+1
 9610  1 05b1 5601 c8                iny
 9620  1 05b2 5602 b1 22             lda (index1),y
 9630  1 05b4 5604 d0 e5             bne rng310
 9640  1 05b6 5606 88                dey
 9650  1 05b7 5607 b1 22             lda (index1),y
 9660  1 05b9 5609 d0 e0             bne rng310
 9670  1 05bb 560b f0 9b             beq rng120
 9770  1 05bd 560d a0 ff    rngfnd   ldy #$ff
 9775  1 05bf 560f c8       rng400   iny
 9780  1 05c0 5610 b1 85             lda (txtptr),y
 9785  1 05c2 5612 d0 fb             bne rng400
 9790  1 05c4 5614 88                dey
 9795  1 05c5 5615 84 5a             sty strlen;laenge d. suchstrings
 9800  1 05c7 5617 a2 00             ldx #$00
 9805  1 05c9 5619 b1 85    rng410   lda (txtptr),y;suchstr. revers
 9810  1 05cb 561b 9d 10 02          sta buf+16,x;in d. basic-puffer
 9815  1 05ce 561e e8                inx
 9820  1 05cf 561f 88                dey
 9825  1 05d0 5620 10 f7             bpl rng410
 9830  1 05d2 5622 60                rts
 9840  1 05d3 5623                   ;-------------------------------------
10000  1 05d3 5623 20 c2 88 find      jsr ncnnch
10005  1 05d6 5626 20 f1 04          jsr renrng;bereich einlesen
10010  1 05d9 5629 a5 2d             lda txttab;anf. basic-text
10020  1 05db 562b a6 2e             ldx txttab+1
10030  1 05dd 562d 20 24 07 fnd10     jsr find10
10040  1 05e0 5630 38                sec
10050  1 05e1 5631 a0 00             ldy #$00
10060  1 05e3 5633 b1 85             lda (txtptr),y
10070  1 05e5 5635 e5 44             sbc linadr;zeilenlaenge errechn.
10080  1 05e7 5637 e9 05             sbc #$05
10090  1 05e9 5639 c5 5a             cmp strlen;vergl. mit stringl.
10100  1 05eb 563b b0 06             bcs fnd20
10110  1 05ed 563d 20 57 07 fnd30     jsr find20;zeile zu kurz, weiter
10120  1 05f0 5640 4c dd 05          jmp fnd10
10130  1 05f3 5643 a0 03    fnd20    ldy #$03
10140  1 05f5 5645 a6 5a    fnd22    ldx strlen
10150  1 05f7 5647 c8       fnd25    iny
10160  1 05f8 5648 b1 85             lda (txtptr),y
10170  1 05fa 564a f0 f1             beq fnd30;zeilenende
10180  1 05fc 564c dd 10 02          cmp buf+16,x
10190  1 05ff 564f d0 f4             bne fnd22
10200  1 0601 5651 ca                dex
10210  1 0602 5652 10 f3             bpl fnd25
10212  1 0604 5654 b1 85             lda (txtptr),y
10214  1 0606 5656 20 50 ba          jsr qnum;test numerisch
10215  1 0609 5659 b0 0a             bcs fnd26
10216  1 060b 565b c8                iny
10217  1 060c 565c b1 85             lda (txtptr),y
10218  1 060e 565e f0 05             beq fnd26
10219  1 0610 5660 20 50 ba          jsr qnum
10220  1 0613 5663 90 e0             bcc fnd22
10225  1 0615 5665 20 c8 8e fnd26     jsr ocrlf;string gefunden
10230  1 0618 5668 a0 02             ldy #$02
10240  1 061a 566a b1 85             lda (txtptr),y
10250  1 061c 566c aa                tax
10260  1 061d 566d c8                iny
10270  1 061e 566e b1 85             lda (txtptr),y
10275  1 0620 5670 84 54             sty lstpnt
10280  1 0622 5672 20 b4 a3          jsr linprt;print zn
10290  1 0625 5675 a9 20             lda #$20
10300  1 0627 5677 a4 54    fnd40    ldy lstpnt;adr1 zeiger letzter string
10310  1 0629 5679 29 7f             and #$7f
```

```
10320  1 062b 567b 20 3a b5 fnd41      jsr ochr
10330  1 062e 567e c9 22                cmp #$22
10340  1 0630 5680 d0 06                bne fnd42
10350  1 0632 5682 a5 13                lda dores
10360  1 0634 5684 49 ff                eor #$ff
10370  1 0636 5686 85 13                sta dores
10380  1 0638 5688 c8          fnd42    iny
10390  1 0639 5689 f0 b2                beq fnd30
10400  1 063b 568b b1 85                lda (txtptr),y
10410  1 063d 568d f0 ae                beq fnd30
10420  1 063f 568f 10 ea                bpl fnd41
10430  1 0641 5691 c9 ff                cmp #$ff
10440  1 0643 5693 f0 e6                beq fnd41
10450  1 0645 5695 24 13                bit dores
10460  1 0647 5697 30 e2                bmi fnd41
10470  1 0649 5699 aa                   tax
10480  1 064a 569a 84 54                sty lstpnt
10490  1 064c 569c a0 80                ldy #$80
10500  1 064e 569e 84 23                sty index1+1
10510  1 0650 56a0 a0 ef                ldy #$ef
10520  1 0652 56a2 84 22                sty index1
10530  1 0654 56a4 a0 00                ldy #$00
10540  1 0656 56a6 0a                   asl a
10550  1 0657 56a7 f0 10                beq fnd47
10560  1 0659 56a9 ca          fnd44    dex
10570  1 065a 56aa 10 0d                bpl fnd47
10580  1 065c 56ac e6 22       fnd45    inc index1
10590  1 065e 56ae d0 02                bne fnd46
10600  1 0660 56b0 e6 23                inc index1+1
10610  1 0662 56b2 20 64 ba fnd46       jsr ldi1y
10620  1 0665 56b5 10 f5                bpl fnd45
10630  1 0667 56b7 30 f0                bmi fnd44
10640  1 0669 56b9 c8          fnd47    iny
10650  1 066a 56ba 20 64 ba             jsr ldi1y
10660  1 066d 56bd 30 b8                bmi fnd40
10670  1 066f 56bf 20 3a b5             jsr ochr
10680  1 0672 56c2 d0 f5                bne fnd47
11000  1 0674 56c4 20 8c ba fnd100      jsr mapusr
11005  1 0677 56c7 a5 2d                lda txttab;anf. basic-text
11010  1 0679 56c9 a6 2e                ldx txttab+1
11020  1 067b 56cb 20 24 07 fnd110      jsr find10
11130  1 067e 56ce c8          fnd120   iny
11140  1 067f 56cf b1 85                lda (txtptr),y
11145  1 0681 56d1 91 25                sta (index2),y
11150  1 0683 56d3 f0 75                beq fnd500;zeilen-ende
11160  1 0685 56d5 10 f7       fnd125   bpl fnd120;kein token
11165  1 0687 56d7                      ;test goto,run,restore,gosub,then,else,t
                                                rap,resume
11166  1 0687 56d7                      ;
11170  1 0687 56d7 a2 08                ldx #$08
11180  1 0689 56d9 dd 19 04 fnd130      cmp tab3,x
11190  1 068c 56dc f0 05                beq fnd200;token gefunden
11200  1 068e 56de ca                   dex
11210  1 068f 56df 10 f8                bpl fnd130
11220  1 0691 56e1 30 eb                bmi fnd120
11500  1 0693 56e3 c8          fnd200   iny
11510  1 0694 56e4 b1 85       fnd210   lda (txtptr),y
11520  1 0696 56e6 91 25                sta (index2),y
11530  1 0698 56e8 f0 60                beq fnd500
11540  1 069a 56ea 20 50 ba             jsr qnum;test numerisch
11550  1 069d 56ed b0 46                bcs fnd400
11551  1 069f 56ef a5 85                lda txtptr
11552  1 06a1 56f1 85 4d                sta nxtadr;pntr retten
11553  1 06a3 56f3 a5 86                lda txtptr+1
11554  1 06a5 56f5 85 4e                sta nxtadr+1
11560  1 06a7 56f7 20 e2 8c             jsr addon;txtptr + yr
11562  1 06aa 56fa 98                   tya;save yr
11563  1 06ab 56fb 48                   pha
11569  1 06ac 56fc 20 29 ba             jsr chrgot
```

```
11570  1 06af 56ff 20 4e bd            jsr linget
11571  1 06b2 5702 68                  pla;yr vom stack
11572  1 06b3 5703 a8                  tay
11580  1 06b4 5704 a5 1b               lda linnum
11590  1 06b6 5706 c5 46               cmp actlin;akt. zn (alt)
11600  1 06b8 5708 d0 17               bne fnd245
11610  1 06ba 570a a5 1c               lda linnum+1
11620  1 06bc 570c c5 47               cmp actlin+1
11630  1 06be 570e d0 11               bne fnd245
11640  1 06c0 5710 a9 ff               lda #$ff
11650  1 06c2 5712 85 57               sta chgflg
11655  1 06c4 5714 a2 01               ldx #$01
11670  1 06c6 5716 bd 00 02 fnd235     lda buf,x
11675  1 06c9 5719 f0 21               beq fnd240
11680  1 06cb 571b 91 25               sta (index2),y
11690  1 06cd 571d c8                  iny
       2 06ce 571e e8                  inx
11700  1 06cf 571f d0 f5               bne fnd235
11701  1 06d1 5721 a5 4d     fnd245    lda nxtadr
11702  1 06d3 5723 85 85               sta txtptr
11703  1 06d5 5725 a5 4e               lda nxtadr+1
11704  1 06d7 5727 85 86               sta txtptr+1
11705  1 06d9 5729 c8        fnd246    iny
11706  1 06da 572a b1 85               lda (txtptr),y
11707  1 06dc 572c 91 25               sta (index2),y
11708  1 06de 572e f0 1a               beq fnd500
11709  1 06e0 5730 20 50 ba            jsr qnum
11710  1 06e3 5733 90 f4               bcc fnd246
11711  1 06e5 5735 c9 2c     fnd400    cmp #$2c;test komma
11712  1 06e7 5737 f0 aa               beq fnd200
11713  1 06e9 5739 aa                  tax
11714  1 06ea 573a d0 99               bne fnd125
11720  1 06ec 573c 98        fnd240    tya;index2 + yr
11730  1 06ed 573d 18                  clc
11740  1 06ee 573e 65 25               adc index2
11750  1 06f0 5740 85 25               sta index2
11760  1 06f2 5742 90 02               bcc fnd230
11770  1 06f4 5744 e6 26               inc index2+1
11900  1 06f6 5746 a0 00     fnd230    ldy #$00
11910  1 06f8 5748 f0 9a               beq fnd210
12050  1 06fa 574a 24 57     fnd500    bit chgflg
12051  1 06fc 574c 10 20               bpl fnd520
12052  1 06fe 574e e6 57               inc chgflg;flag loeschen
12059  1 0700 5750 20 57 07            jsr find20
12070  1 0703 5753 85 22               sta index1
12100  1 0705 5755 86 23               stx index1+1
12110  1 0707 5757 20 cc 07            jsr outlin;zeile (alt) loeschen
12140  1 070a 575a a0 03               ldy #$03
12150  1 070c 575c c8        fnd510    iny
12160  1 070d 575d b1 88               lda (buffpt),y
12170  1 070f 575f d0 fb               bne fnd510
12180  1 0711 5761 84 0e               sty count;zeilenlaenge
12181  1 0713 5763 a5 44               lda linadr
12182  1 0715 5765 a4 45               ldy linadr+1
12183  1 0717 5767 85 6d               sta lowtr
12184  1 0719 5769 84 6e               sty lowtr+1
12190  1 071b 576b 20 81 07            jsr inln10;korrig. zeile einf.
12220  1 071e 576e 20 57 07 fnd520     jsr find20;adr. n. zeile
12270  1 0721 5771 4c 7b 06            jmp fnd110
13000  1 0724 5774 85 85     find10    sta txtptr
13010  1 0726 5776 85 6d               sta lowtr
13020  1 0728 5778 85 44               sta linadr
13030  1 072a 577a 86 86               stx txtptr+1
13040  1 072c 577c 86 6e               stx lowtr+1
13050  1 072e 577e 86 45               stx linadr+1
13060  1 0730 5780 a5 88               lda buffpt
13070  1 0732 5782 a6 89               ldx buffpt+1
13080  1 0734 5784 85 25               sta index2
13090  1 0736 5786 86 26               stx index2+1
```

```
13100  1 0738 5788 a0 00              ldy ##00
13110  1 073a 578a b1 85              lda (txtptr),y
13120  1 073c 578c 91 25              sta (index2),y
13130  1 073e 578e 85 0e              sta count
13140  1 0740 5790 c8                 iny
13150  1 0741 5791 b1 85              lda (txtptr),y
13160  1 0743 5793 91 25              sta (index2),y
13170  1 0745 5795 05 0e              ora count
13180  1 0747 5797 d0 03              bne find15
13185  1 0749 5799 68                 pla;ende basic-text
13186  1 074a 579a 68                 pla
13187  ' 074b 579b 60                 rts
13190  1 074c 579c c8        find15   iny
13200  1 074d 579d b1 85              lda (txtptr),y
13210  1 074f 579f 91 25              sta (index2),y
13220  1 0751 57a1 c8                 iny
13230  1 0752 57a2 b1 85              lda (txtptr),y
13240  1 0754 57a4 91 25              sta (index2),y
13250  1 0756 57a6 60                 rts
13300  1 0757 57a7 a0 01     find20   ldy ##$01
13310  1 0759 57a9 b1 6d              lda (lowtr),y;adresse n. zeile
13320  1 075b 57ab aa                 tax
13330  1 075c 57ac 88                 dey
13340  1 075d 57ad b1 6d              lda (lowtr),y
13350  1 075f 57af 60                 rts
20000  1 0760 57b0 18        aut100   clc;naechste zn
20010  1 0761 57b1 a5 42              lda curlin
20020  1 0763 57b3 65 4f              adc lincnt
20036  1 0765 57b5 85 42              sta curlin
20037  1 0767 57b7 aa                 tax
20040  1 0768 57b8 a5 43              lda curlin+1
20050  1 076a 57ba 65 50              adc lincnt+1
20066  1 076c 57bc 85 43              sta curlin+1
20067  1 076e 57be a8                 tay
       2 076f 57bf 8a                 txa
20068  1 0770 57c0 20 e2 07           jsr findln
20069  1 0773 57c3 90 04              bcc aut110
20070  1 0775 57c5 a0 ff              ldy ##$ff
20075  1 0777 57c7 84 5a              sty strlen;flag f. overflow-error
20110  1 0779 57c9 60        aut110   rts
20120  1 077a 57ca                    ;-----------------------------------------
21000  1 077a 57ca                    ;neue zeile einfuegen
21005  1 077a 57ca                    ;
21010  1 077a 57ca a5 42     inlin    lda curlin
21020  1 077c 57cc a4 43              ldy curlin+1
21050  1 077e 57ce 20 e2 07           jsr findln;adr. neue zeile suchen
21060  1 0781 57d1 a5 30     inln10   lda txtend+1;textend neu berechn.
21070  1 0783 57d3 85 26              sta index2+1
21080  1 0785 57d5 a5 0e              lda count
21081  1 0787 57d7 aa                 tax
       2 0788 57d8 e8                 inx
       3 0789 57d9 8a                 txa
21090  1 078a 57da 18                 clc
21100  1 078b 57db 65 2f              adc txtend
21105  1 078d 57dd 85 25              sta index2
21110  1 078f 57df 90 02              bcc ren70
21120  1 0791 57e1 e6 26              inc index2+1
21130  1 0793 57e3 a5 2f     ren70    lda txtend
21140  1 0795 57e5 a4 30              ldy txtend+1
21150  1 0797 57e7 85 22              sta index1
21160  1 0799 57e9 84 23              sty index1+1
21170  1 079b 57eb a0 00              ldy ##$00
21180  1 079d 57ed b1 22     ren75    lda (index1),y;platz schaffen
21190  1 079f 57ef 91 25              sta (index2),y
21200  1 07a1 57f1 a5 6e              lda lowtr+1
21210  1 07a3 57f3 c5 23              cmp index1+1
21220  1 07a5 57f5 90 06              bcc ren80
21230  1 07a7 57f7 a5 6d              lda lowtr
```

```
21240  1 07a9 57f9 c5 22              cmp index1
21250  1 07ab 57fb f0 13              beq ren100
21251  1 07ad 57fd a5 22    ren80     lda index1
21252  1 07af 57ff d0 02              bne ren85
21253  1 07b1 5801 c6 23              dec index1+1
21260  1 07b3 5803 c6 22    ren85     dec index1
21270  1 07b5 5805 a5 25              lda index2
21280  1 07b7 5807 d0 02              bne ren90
21285  1 07b9 5809 c6 26              dec index2+1
21290  1 07bb 580b c6 25    ren90     dec index2
21330  1 07bd 580d 18                 clc
21340  1 07be 580e 90 dd              bcc ren75
21350  1 07c0 5810 a4 0e    ren100    ldy count
21360  1 07c2 5812 b1 88    ren110    lda (buffpt),y;zeile einfuegen
21370  1 07c4 5814 91 6d              sta (lowtr),y
21380  1 07c6 5816 88                 dey
21390  1 07c7 5817 10 f9              bpl ren110
21400  1 07c9 5819 4c a4 86           jmp lnkprg;link neu
21500  1 07cc 581c                    ;---------------------------------------
22000  1 07cc 581c                    ;alte zeile loeschen
22010  1 07cc 581c                    ;
22020  1 07cc 581c a0 00    outlin    ldy #$00
22030  1 07ce 581e b1 22    ren60     lda (index1),y
22040  1 07d0 5820 91 6d              sta (lowtr),y
22050  1 07d2 5822 c8                 iny
22060  1 07d3 5823 d0 f9              bne ren60
22070  1 07d5 5825 e6 6e              inc lowtr+1
22080  1 07d7 5827 e6 23              inc index1+1
22090  1 07d9 5829 a5 30              lda txtend+1
22100  1 07db 582b c5 6e              cmp lowtr+1
22110  1 07dd 582d b0 ef              bcs ren60
22115  1 07df 582f 4c a4 86           jmp lnkprg;link neu
23000  1 07e2 5832 85 1b    findln    sta linnum
23010  1 07e4 5834 84 1c              sty linnum+1
23020  1 07e6 5836 4c 1f 87           jmp fndlin;adr. neue zeile suchen
23050  1 07e9 5839 85 49    nxtgot    sta nxtlin
23060  1 07eb 583b 86 4a              stx nxtlin+1
23070  1 07ed 583d 4c 29 ba           jmp chrgot;letztes zeichen nochm.
23080  1 07f0 5840                    ;---------------------------------------
24000  1 07f0 5840 a2 88    off       ldx #>ncnnch;ind. sprung-adr.
24010  1 07f2 5842 a9 c2              lda #<ncnnch;zuruecksetzen
24020  1 07f4 5844 8d 84 02           sta icrnch
24030  1 07f7 5847 8e 85 02           stx icrnch+1
24040  1 07fa 584a 4c ca 85           jmp main
```

Bücher

Tatzl, G.: Praktische Anwendungen mit dem HP-71 B. Vieweg-Verlag, Braunschweig 1985, 152 S., ISBN 3 528 04348-2, DM 42,--. Für den Taschenrechner werden praktische Anwendungen zu folgenden Themenbereichen gegeben: Basis- und Lernprogramme, Mathematik und Statistik, Technik und Produktion, kaufm. Anwendungen, Spiel und Hobby und schließlich Zeitstudie vor Ort. Die Programm-Listen sind dabei kein Augenschmaus, weil mit einem Streifendrucker nicht sehr deutlich ausgedruckt. Sie sind jedoch kommentiert und mit einer Einführung zur Aufgabenstellung versehen.

Reparaturanleitungen für Computer: a) APPLE II, II PLUS, b) Commodore 64. Co-Produktion der Verlage Howard Sams & Co, USA und te-wi München 1985, jeweils in einer A4-Einsteckmappe, je DM 29,80, ISBN 3-921 803-54-3 bzw. -55-1.Es handelt sich um Anleitungen mit Schaltplänen. Enthalten ist eine Bauteile und Vergleichstypenliste, Prüfpunkte sind mit Oszilogrammen versehen, enthalten sind Logikangaben, Spannungsangaben, schnelle Tests und Anleitungen zur systematischen Fehlersuche. - Die Unterlagen machen einen sehr guten Eindruck. Sie sind in Deutsch geschrieben. Die Großfotos der Platinen z.T. mit Rasternetz und Bezeichnungen erleichtern das Zurechtfinden. Die Unterlagen dürfen den gewerblichen und Selbstreparateuren empfohlen werden.

P. Mueri, CH-5222 Umiken

Floppy Controller

für den AIM 65

Einleitung

Es wird der Aufbau eines Floppy-Controllers beschrieben, der zum AIM-DOS von Rockwell kompatibel ist. Als Vorzug gegenüber dem Rockwell FDC RM65-5101 NE erlaubt er das gleichzeitige Betreiben von 8" und 5" Drives. Die Karte hat folgende Eigenschaften:

- 8" und 5" gleichzeitig anschließbar
- Geringe Kosten (unter DM 100)
- Einfacher Abgleich
- Software-Kompatibilität zum AIM-DOS

Das Rockwell AIM-DOS A65-090 wurde gewählt, da es mit allen Sprachen zusammenarbeitet und gut dokumentiert ist (Rockwell Doc. 29801 N02). Es unterstützt prinzipiell auch den gemischten Betrieb von 8" und 5" Drives, das FDC-Module RM65-5101NE jedoch kann dies nicht (eine Jumper-Kombination muß gedreht werden).

Wahl des FDC

Der FDC soll möglichst kompatibel zum auf dem Rockwell-FDC verwendeten FD1793 sein. Durch die integrierte PLL und den nun günstigen Preis (ca. DM 60) wird ein FDC der Serie WD 279x vorgezogen, wobei beide Typen 2793 und 2797 in Frage kommen. Dabei gibt es jedoch kleine, aber wesentliche Unterschiede. Der WD 2793 ist voll kompatibel zum 1793 und weist einen internen /2-Clockteiler auf, während der 2797 einen Side Select Output hat (wird hier nicht benötigt). Trotzdem fällt die Wahl auf den 2797 wegen der im IBM-3740 Format größeren Flexibilität. Die 2797/Type-2 Kommandos R/W Sector sind nicht 100% kompatibel mit dem 1793 - und damit auch nicht mit dem DOS. Das DOS ist jedoch Dank der Dokumentation leicht an den 2797 anzupassen, und es mußte sowieso in ein 2532 EPROM übertragen werden, um die Korekturen nach /1/ zu übernehmen.

Aufbau des Controllers

Die Karte wird am J-3 Expansion Stecker des AIM betrieben. Sie decodiert ihre Adressen vollständig ($8Fxx) und liefert zugleich den CS für das DOS-EPROM, das auf einer EPROM-Karte oder auf der Controller-Karte selbst Platz finden kann. Der ROM-CS ist aktiv im Bereich hex 8F00 .. 8EFF. Bei 8F00 .. 8F03 liegen die FDC-Register und bei 8F04 die I/O Control und Statusregister (für Read und Write verschieden). Zusätzlich wird hex 8F15 vom DOS verwendet, um via die READY-Leitung die CPU anzuhalten zur Synchronisation mit dem FDC.

Als Neuigkeit steht je Drive ein Switch zur Verfügung, womit der Typ 8" oder 5" gewählt wird. Mit diesem Signal wird auf der Karte der FDC-Takt halbiert (2 MHz oder 1 MHz) und das bei 5" Drives normalerweise nicht vorhandene 'READY'-Signal simuliert. Das DOS selbst berücksichtigt das Drive-Typ-Signal bei jeder Aktion. Zusätzlich benötigt man einen 4 MHz Takt, der direkt vom AIM 65 vom IC Z14 P6 bezogen wird. - Neben dem FDC benötigt die Karte nur 10 TTL-IC's, so daß der Aufbau einer Europakarte bequem möglich ist.

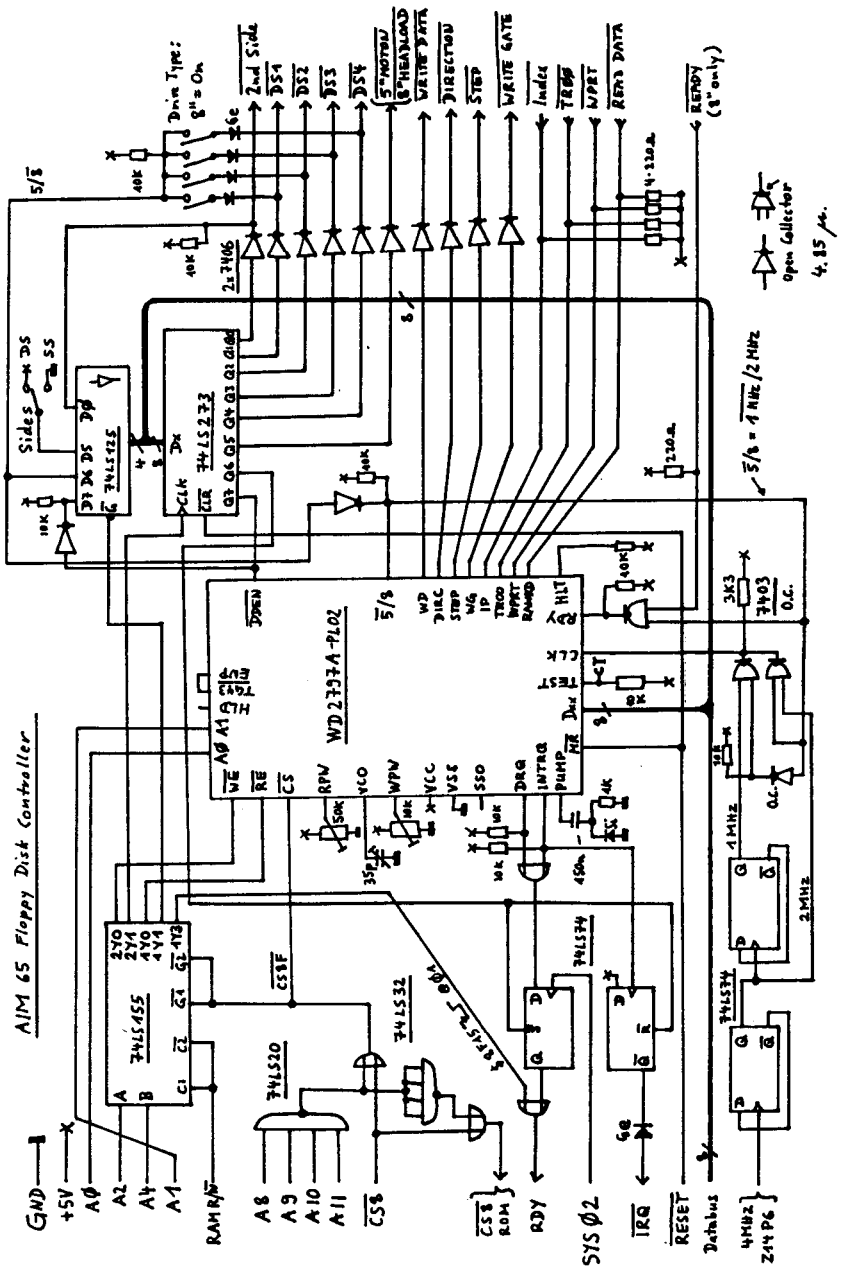
Abgleich

Der Abgleich erfolgt gemäß WDC-Spezifikation und ist einfach. Dazu braucht es nur den 4 MHZ-Takt und Vcc.

1. Alle Drive-Typ-Schalter auf 5"
2. Reset kurz aktivieren
3. Nun den Test-Pin (T) mit Masse verbinden
4. Mit Trimmer WPW eine Pulsbreite von 200 ns an Pin 31 (WD) einstellen
5. Mit Trimmer RPW eine Pulsbreite von 500 ns an Pin 29 (TG43) einstellen
6. Den VCO-Trimmer auf 250 kHz an Pin 16 (DIRC) abgleichen
7. Test-Pin (T) von Masse trennen

Betrieb

Es sind maximal 4 Drives ansprechbar, ein- oder zweiseitig, 8" oder 5". Der Schalter 'Sides' ist auf 'SS' zu stellen, wenn nur single Drives verwendet werden. Vom DOS aus zugänglich sind dann


AIM 65 Floppy Disk Controller
WD2793A-PL02
74LS455
74LS138
74LS32
74LS20
74LS74
74LS125
74LS273
2x7406
Drive Type: 8" = On
2nd Side
DS4
DS3
DS2
DS1
S/8
Sides
SS
DS
WRITE DATA
DIRECTION
STEP
WRITE GATE
Index
TRK0
WPRT
READ DATA
READY (8" only)
Open Collector
4.15 µs.
S/8 = 1MHz/2MHz
1MHz
2MHz
3K3
7403
O.C.
GND
+5V
A0
A2
A4
A1
RAM R/W
A8
A9
A10
A11
CS8
CS8 ROM
RDY
SYS Ø2
IRQ
RESET
Databus
4MHz
Z-44 P6

Disk-1 bis Disk-4. Falls ein oder mehrere doppelseitige Drives verwendet werden, ist 'DS' zu wählen. Dann stehen Disk-1 bis Disk-8 zur Verfügung, wobei jede Seite wie eine separate Disk behandelt wird. Für eine einseitige Disk gilt dann die ungerade Disk-Nummer. Für jeden Drive wird darauf der Type 5" oder 8" gesetzt.

Die Karte unterstützt bei 5" Single und Double Density (128 oder 256 Bytes/Sector) und bei 8" das IBM-3740 Format Single Density mit 128 Bytes/Sector. Dies ergibt beim jeweiligen Default-Format 5" 35-TR, 16-SC, SD 68K Bytes, bei DD 137 KB. Und bei 8" sind es 77-TR, 26-SC, SD 251K Bytes pro Seite.

Es ist nun problemlos möglich, z.B. von 5" DD auf 8" SD Files zu kopieren. Nach der Standard-DOS-Initialisierung *=8000 G/ sind alle Drives sofort gemäß ihrem Typ ansprechbar. Beim Verfasser sind mit dieser Karte seit vier Monaten zwei 8" Double Side Drives MFE 750 (Disk-1 bis Disk-4) und ein 5" Single Side SA 400 (Disk-5) problemlos in Betrieb.

DOS-Änderungen

Beim Kopieren des ROMs in ein EPROM sollten die folgenden Änderungen übernommen werden. Es sind alle Patches in der nachstehenden Liste enthalten.

Literatur
/1/ Zweygart, W.: Floppy Controller für den AIM 65, 65xx MICRO MAG Nr. 30, S. 58
/2/ Langermann, H.: AIM 65 DOS (Rockwell), 65xx MICRO MAG Nr. 33, S. 45
/3/ Western Digital Corp.: Datenblatt WD 279x-02 Floppy Disk Formatter/Controller Family

```
*************************************************************
DOS PATCHES

0000          ;
0000          ; H. LANGERMANN, MICRO MAG # 33, S. 45
0000          ;
0000          ; P. MUERI : ANPASSUNG AN WD 2797
0000          ;              <F3> VEKTOR NICHT UEBERSCHREIBEN
0000          ;
0000 OUTFLG           =$A413
0000 CLSOUT           =$82AD
0000          ;
0000                  *=$8337
8337 LISEND 4CBE84    JMP LISEN2
833A          ;
833A                  *=$8343
8343          F0F2    BEQ LISEND        ; EOF FOUND
8345          ;
8345                  *=$84BE
84BE LISEN2 AD13A4    LDA OUTFLG
84C1          C955    CMP #'U'          ; OUT=U=DISK ?
84C3          D003    BNE LISRET        ; NO, RETURN
84C5          4CAD82  JMP CLSOUT        ; YES, CLOSE WRITE FILE
84C8 LISRET   60      RTS
84C9          ;
84C9                  *=$8433
8433          A90D    LDA #$0D
8435          8D13A4  STA OUTFLG        ; SET OUTPUT TO D/P
8438          EA      NOP
8439          ;
8439                  *=$8891
8891          0C      .BYT $0C          ; SET 2797 SECTOR LENGTH FLAG ALWAYS
8892          ;
8892                  *=$BBCE
BBCE          0902    ORA #2            ; 2797 SIDE OUT BIT 2 STATT 4
BBD0          ;
BBD0                  *=$801D
801D          A209    LDX #09           ; DO NOT OVERWRITE THE <F3>
801F          BD3980  LDA $8039,X       ; VECTOR, ONLY UIN,UOUT AND
8022          9D0801  STA $108,X        ; <F1> AND <F2>.
8025          ;
8025                  .END
 ERRORS= 0000
```

Roland Löhr

Intelligentes Terminal

In dieser Zeitschrift wurden vor einem Jahre bereits zwei Beiträge abgedruckt, die einen Rechner CBM 610/620 bzw. 710/720 als Terminal zu betreiben gestatten /1/, /2/. Beim praktischen Einsatz der Programme zeigte sich, daß oft Wünsche offen blieben, die mit der Geschwindigkeit, mit dem Bedienungskomfort und mit der Anpassung an den Datenaustausch zusammenhängen. Es wird daher nachstehend ein maschinensprachliches Programm abgedruckt, das diesen Wünschen weiter entgegenkommt. Sein Lösungsweg wird sich auf viele andere Computer übertragen lassen, die mit einer seriellen Schnittstelle betrieben werden sollen.

Einen Computer als Terminal für einen anderen Computer zu betreiben, mag zunächst einmal etwas paradox klingen, nimmt man meistens dafür doch ein Terminal. Nun wird man ein Terminal jedoch als einen spezialisierten Computer ansehen können, der die Funktionseinheiten Tastatur, Bildschirmanzeige und Regelung des Datenverkehrs zur Verfügung stellt, meistens in Mehrbenutzersystemen eingesetzt, in denen ein Zentralrechner Informationen und Dienstleistungen vorhält.

Es kann aber auch jeder mit einer seriellen Schnittstelle ausgerüsteter Heim- und Personal Computer als Terminal benutzt werden. Er hält dann ebenfalls Tastatur, Bildschirm und ein Programm für den Datenverkehr vor, was oft eine vorübergehende oder billigere Lösung darstellen kann. Die Intelligenz des Terminalrechners wird daneben aber häufig auch lokal am Arbeitsplatz benutzt werden können. Wir kommen weiter unten darauf zurück.

Das in Heft 37 abgedruckte BASIC-Programm 'Terminalbetrieb mit CBM 710' hat sich für die schnelle Erprobung einer Schnittstelle als recht nützlich erwiesen, weil die Betriebsparameter interaktiv eingegeben werden können. Es hat den Nachteil, daß bei hohen Übertragungsgeschwindigkeiten, insbesondere bei den üblichen 9600 Baud, Zeitengpässe auftreten. Es kommt zum Verlust übertragener Zeichen und damit auch zu Verwerfungen auf dem Bildschirm, wenn der Hostrechner z.B. beim Memory Dump ein Trommelfeuer auf das empfangende Terminal losläßt. Einerseits ist BASIC hier recht langsam, andererseits geht viel Zeit mit dem Hochschieben des Bildschirminhaltes verloren, wenn eine neue Zeile beginnt. Man kann eine bessere Synchronisation erhalten, wenn man das Bildschirmfenster klein hält, um die Menge der Transporte von Zeile zu Zeile geringer zu halten.

Die gleiche Beobachtung betrifft das ebenfalls in Heft 37 abgedruckte maschinensprachliche Programm 'CBM 6x0 und 7x0 als Terminal'. Auch hier können bei 9600 Baud noch Zeichen verloren gehen. Es war daher das Bestreben des Autors, diese Vorlage durch Linearisierung (Verzicht auf Unterprogramm-Aufrufe) zu beschleunigen. Sicher sind noch immer Beschleunigungen möglich, denn die benutzten E/A-Routinen des Kernels BASIN und BASOUT gehen weite verschlungene Wege. Die Geschwindigkeit hat sich gleichwohl als praktikabel gezeigt. - Mehr noch kam es darauf an, dem Benutzer ein Bedienungsinterface zur Verfügung zu stellen, wie er es etwa von der üblichen Arbeit an Bildschirm und Tastatur gewohnt ist. So soll er einen blinkenden Cursor für seine Eingaben vorfinden, seine Eingaben werden revers dargestellt, während die im Host ausgelösten Ausgaben in Normalschrift erfolgen. Cursorbewegungen nach rechts und links und DELETE sollen wie üblich wirken, andere Kontrolltasten sollen als Fehlbedienung totgelegt werden. Weiter waren die Besonderheiten der Zeichensatzdarstellung bei Commodore zu berücksichtigen.

Zum Programmaufbau:

Das Programm wird mit BLOAD "name" ON B15 an Adresse hex 2003 in Bank 15 geladen und mit SYS 8195 in dieser Adresse gestartet. Es erfolgt die Initialisierung der RS232-Schnittstelle als logisches File 14 mit Primäradresse (das ist ein Muß) Nr. 2 und Sekundäradresse 3 für bidirektionalen Betrieb. Die weiteren Betriebsparameter werden im OPEN-String, siehe Tabelle am Schluß, verankert. Zu den möglichen Parametern siehe /3/. Das Haupt-Verarbeitungsprogramm liegt zwischen den Labeln GETCH und TOBS. Es wird versucht, ein Zeichen von der Schnittstelle zu holen. War sie leer, so wird bei GETCH2 die Tastatur abgefragt. War auch sie leer, so geht es mit dem Warten zurück nach GETCH. Wird ein Zeichen im Tastaturbuffer vorgefunden, so wird es zu-

nächst bei CODBAR untersucht, ob es Kontrollzeichen ist. Wenn ja, so führt es über FUND und eine Sprungtafel JMPTBL auf ausführende Routinen für das Kontrollzeichen. Viele nicht benutzte Tasten CTRL/... führen zu GETCH direkt zurück. Solche Kontrollzeichen geraten also nicht an den Host-Rechner, können daher auch nicht zu Störungen führen, wenn sie von dort als Echo sonst zurückkommen würden. Die JMPTBL kann vom Benutzer bequem an das Protokoll seiner Host-Schnittstelle angepaßt werden.

Wegen der Besonderheiten der CBM-Zeichendarstellung, insbesondere auch bei DIN-ASCII, werden deutsche Sonderzeichen zunächst bei CODWAN abgefiltert und umgesetzt (beim Empfang bei TOBSCD), dann über die Umsetzungsroutine des CBM TOASCI (bzw. TOCBM) auf die normale Groß-Kleinschreibung gewandelt und bei RSOUT auf die Schnittstelle gesendet.

Der Host echot empfangene Zeichen zurück. Sie werden bei GETCH empfangen, gefiltert und normal bei BSAUSG auf den Schirm gegeben. Ist jedoch das Speicherflag eingeschaltet, so wird das empfangene Zeichen bei NORM1 in die Speicherbank 3 zunächst auch abgespeichert. Hier kann empfangener Text später mit dem VIEW-Befehl (CTRL/v) wieder betrachtet werden. Das Flag für die Mitspeicherung wird mit der C=-Taste (Commodore-Taste) als Flip/Flop ein- und ausgeschaltet. Mit CTRL/y wird der Textspeicher in Bank 3 geleert, zurückgesetzt.

Viele CTRL-Tasten sind totgelegt, sie senden nichts an den Host. Hier kann man anpassen. Die TASTE ESC führt zu BASIC zurück. ESC-Sequenzen sind damit nicht eingebaut, können aber nachimplementiert werden, indem man ESC an den Host durchgehen läßt.

Die Tasten CTRL/n und CTRL/a schalten im CBM den Zeichensatz auf normal bzw. alternativ. Mit entspr. Zeichensatzgenerator lassen sich dann Sonderzeichen alternativ als deutsche Umlaute oder Akkoladen und eckige Klammern auf den Bildschirm bringen.

Man beachte, daß für die genannten CBM verschiedene Betriebssysteme in Umlauf sind, so daß sich Unterschiede in den Adressen der Routinen ergeben mögen.

Anregungen

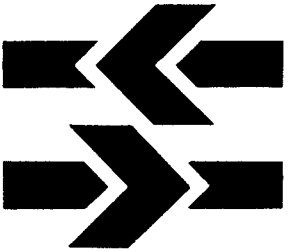
Für häufig benötigte Bedienungsabfolgen hat es sich als Bequemlichkeit erwiesen, wenn man die Funktionstasten des CBM mit solchen Zeichensequenzen belegt. Für die entsprechende Initialisierung des Terminalsystems kann ein auf Diskette gespeichertes BASIC-Vorprogramm ganz praktisch sein.

Einplatinencomputer und auch Entwicklungssysteme benutzen zunehmend serielle Schnittstellen. Für ihre Bedienung kann ein vorhandener Computer als Terminal benutzt werden. Andererseits treten oft Schwierigkeiten bei der Programm- und Datenübertagung zwischen verschiedenen Computern auf. Hier kann die serielle Schnittstelle mit der im Programm vorgesehenen Abspeicherung empfangener Daten eine einfache Hilfe bieten. Und schließlich sind Textübermittlungen zwischen Computern möglich, sie setzen allerdings höher organisierte Textprogramme voraus. – Man beachte z.B. bei der Übermittlung von Quelltexten z.B. an einen compilierenden FORTH-Computer, daß dieser nach der Auslösung einer Zeile mit 'CR' eine gewisse Zeit braucht, bis er sich mit einem 'OK' zurückmeldet. Einen solchen Computer kann man nicht mit einem Trommelfeuer übermittelter Daten zudecken, man wird per Programm von 'CR' zu 'CR' auf das 'OK' als Handshake warten müssen.

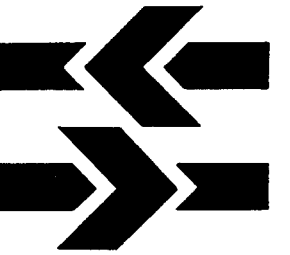
Literatur

/1/ Lange, H.-G.: CBM 6x0 und 7x0 als Terminal, MICRO MAG Nr. 37, Juni 1984
/2/ Löhr, R.: Terminalbetrieb mit CBM 710, ebenda
/3/ Löhr, R.: BASIC 4: Die neuen Befehle, 65xx MICRO MAG Nr. 32, August 1983

```
000000                  0004 ; PROGRAMM DIENT ZUM BETRIEB DES CBM 610/20 710/20
000000                  0005 ; ALS TERMINAL AN EINEM HOST-RECHNER UEBER RS232.
000000                  0006 ; DATEN WERDEN VOM KEYBOARD GELESEN
000000                  0007 ; UND UEBERTRAGEN. ANKOMMENDE BYTES WERDEN AUF
000000                  0008 ; DEN BILDSCHIRM GESCHRIEBEN. DIE
000000                  0009 ; BETRIEBSART IST 'FULL DUPLEX'
000000                  0010 ; 'DEL' ZEICHEN (ASCII 127) UND 'LF' (ASCII 10)
000000                  0011 ; WERDEN AUSGEBLENDET
```



ees



informiert:

OPTOCOS : DER SEHENDE PROZESSRECHNER

EUROCOS : DER FLEXIBLE MESS- UND STEUERRECHNER

PC-BOX : DIE VERBINDUNG IHRES PC'S ZUR AUSSENWELT

FORTH : DER INDUSTRIERECHNER MIT FORTH

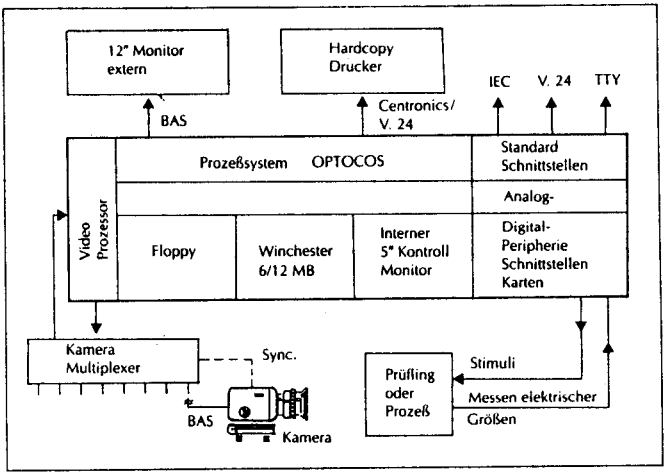
16 BIT : DIE ERSTEN RECHNER MIT 16 BIT 65000 PROZESSOI

ees

elektronik
entwicklung
service

fertigungs- und
vertriebs-gmbh
pelkovenstraße 51
8000 münchen 50
tel. 089/141 10 77 ∅
telex 5 28 424 igees

OPTOCOS – der sehende Prozeßrechner



Hardware

OPTOCOS ist ein aus der universell ausbaufähigen Industrierechnerfamilie EUROCOS entwickeltes Bildauswertungssystem, bei dem neben einer vielseitig anwendbaren Bildauswertung nicht auf die direkte Einbindung in Meß-, Steuer- und Prozeßaufgaben verzichtet werden muß.

Grundgerät

3 HE oder 6 HE 19" Einschübe mit 560 bzw. 330 mm Tiefe, ausgerüstet mit Netzteilkassette, Bus-Platine für rückseitige Einführung von Teileinschüben in EUROPA-Format (100 × 160 mm, VG 96 pol.), frontseitiger Bus zur Adaptierung von 33 TE breiten Kassetten (Floppy, Winchester, Monitor etc.).

Einschübe

Alle Kassetten 33 TE, 210 mm Tiefe, 3 HE. Kontaktierung über 37polige Steckerleiste.

- Monitor 5" grün, 80/40 Zeichen × 25 Zeilen
- Floppy 5¼", wahlweise 80/160 KB, maximal 4 Kassetten ansteuerbar
- Winchester 3,5", je nach Ausbau 2,5 bis 12 MB.

Karten

- CPU-Karte mit 24 KB Betriebssystem in EPROM, Taktfrequenz 2 MHz
- Memory I: 32 KB C-MOS mit Batterie-Pufferung, Speicherbausteine EPROM-kompatibel für Festprogramme
- Memory II: 128 KB C-MOS mit Banking bis 1 MB nutzbar
- Video: wahlweise 40/80 Zeichen und 25 Zeilen, Ausgang: TTL und BAS
- Graphik-Karte 200 × 320 Punkte, mit Firmware
- RTC-Realtime Clock, batteriegepuffert, Datum und Uhrzeit
- Diverse weitere Systemkarten

Bildauswertung

Bildeingang BAS-Signal, Bildwechselfrequenz 60 Hz mit synchronisierbarer Kamera. Einzelbildschaltung per Soft- oder Hardware möglich.
Bildauflösung:

Breite	320 (640)	Punkte
Höhe	200	Punkte
	400	Punkte optional

Zur Grauwertanpassung in 16 Stufen programmierbarer Komparator mit nachfolgender Schwarz-/Weiß-Speicherung des Bildes in 64 (128) KB Speicher

Kamera

Jede synchronisierbare Kamera ist anpaßbar.

Software

Über das bereits vom EUROCOS bekannte und bewährte peripherieunterstützende BASIC hinaus ist die Software vielfältig für die Bildauswertung des OPTOCOS erweitert worden.

Beispiele zu den derzeitigen Bild-Befehlen:

V. HL, x, y, dx
Horizontale Linie; Startpunkt x, y; Länge dx
Linie »an« oder »aus«, invertieren und analysieren
Analyse: wo und wie oft sind Hell/Dunkel-Übergänge auf der angegebenen Linie

V. VL, x, y, dx
Vertikale Linie; wie horizontal

V. SEGC, »Parameter« (12)
Komplexer Befehl zur Analyse einer 7-Segment-Anzeige

V. SEGA
Analyse der von SEGC gewonnenen Daten und Umsetzung an Hand einer Kodetabelle. Resultiert in der Ausgabe eines entsprechenden Zeichens.

V. AND, n, V
n-maliges Lesen eines Bildes und Integration der Dunkelstellen. (Korrektur unklarer Bilder)

V. EXOR, n, V
n-maliges Lesen eines Bildes mit Helligkeitsstufe V und EXOR-Verknüpfung der einzelnen Bilder.

V. READ, v
Einmaliges Lesen des Bildes mit programmierter Lichtempfindlichkeit v (v = 0–15)

V. SCAN, V
Bild auf Dauerabtastung z. B. bei Justage

V. FREEZE
Einfrieren des letzten Bildes

V. CLEAR x_0, y_0, x_1, y_1,
Löschen des Bildrechteckes

V. COUNT, x_0,y_0, x_1, y_1,
Zählen der »Hellen« Punkte in diesem Rahmen

V. CLEAN, x_0, y_0, x_1, y_1,
Versäubern in diesem Rahmen

V. ZPAR, Parameter (8)
setzt die Parameter zur Zeigerauswertung

ZRD, A%, B%
sucht Zeiger und übergibt Koordinaten des Zeigerende-Punktes in A%, B%

Alle G.-Befehle der Graphik-Beschreibung bleiben erhalten.

Alle G.-Befehle arbeiten relativ zu window und viewport und alle V.-Befehle arbeiten mit absoluten Koordinaten.
x: 0–319; y: 0–199

Zusätzliche BASIC-Hilfsprogramme wie z. B. Schnelleingabe via Fadenkreuz und Rechteck.

EUROCOS

Der flexible Industrierechner für Mess- und Steueraufgaben. Die Rechner der Eurocos-Familie bestehen immer aus einem 19" Grundrahmen in verschiedener Bauart mit Netzversorgung und einem oder mehreren Bussystemen. Der Eurocos kann mit einer großen Auswahl Euro-Karten bestückt werden, wodurch die verschiedenen Einsatzmöglichkeiten und optimale Lösung für den Anwender gewährleistet wird. Je nach Bauform gibt es verschiedene Kassetteneinschübe, wie Floppykassette, Monitorkassette, Winchesterkassette und Netzteilkassette.

Grundkarten:

- EC-SBC Prozessorkarte (6502/6520/6522) 27KB Betr.Sys. Exos.
- EC-MEM 32 Memorykarte mit 32KB statischen RAM für Programmerstellung frei verfügbar/max. 4 Karten pro Rechner.
- EC-FLC Floppy-Controllerkarte für max. 4 Laufwerke.
- EC-VIDEO Video-Karte 40/80 Zeichen pro Zeile.
- EC SEL/DRU Dekodier-Karte für Peripherie.
- EC-GPH Video-Graphik-Karte mit 320*200 Punkten und integriertem Befehlssatz auf E-PROM.
- EC-SB/IEC Aufsteck-Karte für SBC. IEC-Bus Interface gemäß IEEE-488. Anschluß 20 pol. Flachstecker.

Eine große Auswahl weiterer Karten steht zur Verfügung.

PC-BOX

- intelligentes Peripheriegerät für PC
- durch totale Trennung höchst störsicher
- 1/2 19"- Gehäuse, auch als Tischgehäuse geeignet
- Einschubkarten flexibel erweiterbar.
- microprozessorgesteuert 6502/65802
- Anschluß V24 / IEC /Current Loop 20 mA
- Assembler aus allen gängigen Programmiersprachen
- keine Treiber additiv notwendig, kein Link nötig, digitale Ein- und Ausgabe
- Es existieren Befehle zum Setzen und Löschen von Einzelbits, sowie Bitmustern. Es können alle Eingänge bitweise abgefragt werden.
- Setzen ganzer Gruppen von Digital I/O mit einem Befehl möglich.
- Ein/Ausgabe im BCD-Format möglich, bis zu 50 Stellen analoge Ein- und Ausgabe.
- Setzen von analogen Ausgängen und Abfragen von analogen Eingängen, Relais-Ausgänge.
- An/abschalten einzelner Relais unter Angabe mehrerer Relaisnummern.

Preis:	Gundgerät	DM 1.950,-
	Karten	DM 400,- bis DM 600,-

FORTH-COMPUTER

Der EUROCOS kann mit einem FORTH-Betriebssystem und einer fig-FORTH-Implementierung bestellt werden (EC-FORTH).

EC-FORTH unterstützt alle wichtigen Interfaces des EUROCOS sowie Vollgrafik und Floppys bzw. Festplatte.

Zur Einführung ist ein preiswerter Platinensatz mit EC-FORTH und ein Peripheriepaket lieferbar.
(Platinensatz: BUS, CPU, MEMory, Video, Centro, Floppyctrl)
(Peripheriesatz: Tastatur, 12'' Monitor, Floppylaufwerk, Netzteil)
EC-FORTH ist lieferbar für 6502 und 65802 CPUs.
Eine FORTH-83 Version unter 65816 ist in Vorbereitung.

Die neuen CPUs 65802 und 65816

Für die neuen CPUs der 65er-Reihe ist für EUROCOS folgende Software erhältlich oder in Vorbereitung:
EXOS-65000 (Betriebsystem), BASIC-65000 (BASIC-Interpreter), MOUSY-65000 (Assembler und symbolischer Debugger/Tracer), PASCAL-65000 (in Vorbereitung), FORTH-65000 (FORTH-83).

Die EUROCOS-Hardware kann weitgehend verwendet werden.
Bei Einsatz der CPU-65816 sind folgende Karten einzusetzen:
SBC-65000 (CPU-Karte), SEL-65000 (16-MB-Selektionskarte), MEM-65000 (Speicherkarten für RAM, CMOS-RAM oder EPROM).

```
000000                  0013 ; SONDERFUNKTIONEN:
000000                  0014 ;'C='   (COMMODORE_TASTE)
000000                  0015 ;       ALLE ANKOMMENDEN ZEICHEN WERDEN AB $0002
000000                  0016 ;       IN BANK 2 ABGESPEICHERT BIS ZUM ABERMALIGEN
000000                  0017 ;       BETAETIGEN DER C=-TASTE
000000                  0018 ;CTRL-V AUSGABE DES SPEICHERINHALTS, AUCH WIEDERHOLT,
000000                  0019 ;       AN DEN BILDSCHIRM
000000                  0020 ;CTRL-F UEBERGANG IN DEN TEXTEDITOR/SEPARATES PROGRAMM
000000                  0021 ;CTRL-Y LOESCHT DEN TEXTSPEICHER
000000                  0022 ;CTRL-A STELLT ASCII-ZEICHENSATZ EIN
000000                  0023 ;CTRL-N STELLT DEN DIN-ZEICHENSATZ EIN
000000                  0024 ; ESCAPE-TASTE FUEHRT ZU BASIC ZURUECK
000000                  0025 ;-----------------------------------------
000000                  0026 ;FOLGENDE CTRL-TASTEN LEITEN DIREKT AUF DEN COMPUTER:
000000                  0027 ;C,D,E,H,I,J,M,R,S,P,T,U,X,Z
000000                  0028 ;FERNER CURSOR LEFT, RIGHT, DELETE
000000                  0029
000000                  0030 ;-----------------------------------------
000000                  0031 ;VERKEHRSZEICHEN:
000000                  0032 CR      =$0D            ;CARRIAGE RETURN
000000                  0033 LF      =$0A            ;LINEFEED
000000                  0034 ;
000000                  0035 ; CBM ROM-ROUTINEN:
000000                  0036 ;
000000                  0037 REVSCR =$EA27           ;BILSCHIRM REVERS SCHALTEN
000000                  0038 NRMSCR =$EA3B           ;BILDSCHIRM NORMAL SCHALTEN
000000                  0039 BAKCHR =$E58E           ;CURSOR LINKS RUECKEN, KEIN SP
ACE
000000                  0040 ALTSET =$EA1C           ;UMSCHALTEN AUF BENUTZER-ZEICH
ENSATZ
000000                  0041 TOCBM  =$F3DC           ;CODEWANDLANG ASCII TO BS-CODE
000000                  0042 TOASCI =$F3C7           ;UMWANDLUNG TASTATUR TO ASCII-
CODE
000000                  0043 BSOUT=$FFD2            ;AUSGABE EINES ZEICHENS
000000                  0044 BASIN=$FFE4            ;BASIN=GETIN: EINGABE EINES ZEI
CHENS
000000                  0045 CHKIN  =$FFC6           ;TALK
000000                  0046 CHKOUT=$FFC9           ;LISTEN
000000                  0047 CLRCH=$FFCC            ;RESTORE I/O
000000                  0048 OPEN    =$FFC0          ;OEFFNEN LOG.FILE
000000                  0049 ;
000000                  0050 ;*** HARDWARE:
000000                  0051 I6509   =1              ;REGISTER FUER BANKUMSCHALTUNG
000000                  0052 CRTC    =$D800          ;ADRESSE DES BS-CONTROLLERS
000000                  0053 ;
000000                  0054 ;VARIABLEN UND IHRE ADRESSEN
000000                  0055 RS232   =14             ;LOG FILE RS232
000000                  0056 MEMFLG =$72             ;SPEICHERFLAG IM FAC
000000                  0057 JUMP    =$73            ;SPRUNGVERTEILER
000000                  0058 LA      =$9E            ;LOG. GERAETEADRESSE
000000                  0059 FA      =$9F            ;PRIMAERADRESSE
000000                  0060 SA      =$A0            ;SEKUNDAERADRESSE
000000                  0061 FNADR   =$90            ;STARTADRESSE DES FILENAMENS:L
OW,HIGH,SEGMENTNR
000000                  0062 FNLEN   =$9D            ;LAENGE DES FILENAMENS
000000                  0063 QTSW    =$D2            ;GAENSEFUESSCHEN-FLAG
000000                  0064 RVS     =$0397          ;REVERSE FLAG BILDSCHIRM
000000                  0065 ;
000000                  0066 BANK2   = 2             ;USER SEGMENT
000000                  0067 ANFANG =0002            ;ZEIGER AUF TEXTBEGINN
000000                  0068 ;
000000                  0069 ;ZERO PAGE-VARIABLEN
000000                  0070 ;
000000                  0071         * =$F0          ;FREIER BEREICH
0000F0                  0072 NOWLN   *=*+2           ;POINTER LAUFENDE ZEILE
0000F2                  0073 BOTLN   *=*+2           ;POINTER FUER DAS ABSPEICHERN
0000F4                  0074 ;
0000F4                  0075         *=$2003         ;FREIER RAM-BEREICH IM SEGMENT
 15
002003                  0076 ;
********************************************************************************
```

```
002003                  0078 ;
002003                  0079 START
002003  A0 0A           0080        LDY #$0A        ;BLINKENDEN CURSOR EINSCHALTEN
002005  8C 00 DB        0081        STY CRTC        ;KONTROLLREGISTER
00200B  A9 60           0082        LDA #$60
00200A  8D 01 DB        0083        STA CRTC+1      ;DATENREGISTER
00200D                  0084 INIT                   ;OEFFNEN DER RS232 SCHNITTSTEL
LE
00200D  A9 0E           0085        LDA #RS232
00200F  85 9E           0086        STA LA          ;LOG. FILE 14
002011  A9 02           0087        LDA #2          ;PRIMAERADR. 2=RS232
002013  85 9F           0088        STA FA
002015  A9 03           0089        LDA #3          ;BIDIREKTIONAL OHNE CODEWANDLU
NG
002017  85 A0           0090        STA SA
002019  A9 B9           0091        LDA #<OPNSTR    ;LOW-BYTE ADR. FILENAME
00201B  85 90           0092        STA FNADR
00201D  A9 21           0093        LDA #>OPNSTR
00201F  85 91           0094        STA FNADR+1
002021  A9 0F           0095        LDA #15         ;LAEUFT IN UND HAT FILENAME IN
 SEG. 15
002023  85 92           0096        STA FNADR+2
002025  A9 04           0097        LDA #04         ;NAMENSLAENGE
002027  85 9D           0098        STA FNLEN
002029  18             0099        CLC
00202A  20 C0 FF        0100        JSR OPEN        ;OEFFNEN RS232
00202D  20 42 21        0101        JSR INITPT      ;INITIALISIERE POINTER
002030  A9 00           0102        LDA #0
002032  85 72           0103        STA MEMFLG      ;=0, NICHTS ABSPEICHERN
002034  20 1C EA        0104        JSR ALTSET      ;ZEICHENSATZ DES BENUTZERS
**********************************************************************************
HAUPTSCHLEIFE

002037                  0106 GETCH                  ;HOLE ZEICHEN VON SCHNITTSTELL
E
002037  A2 0E           0107        LDX #RS232
002039  20 C6 FF        0108        JSR CHKIN       ;HOLE ZEICHEN VON RS232C TALK
00203C  20 E4 FF        0109        JSR BASIN       ;EINGABE EINES ZEICHENS
00203F  48              0110        PHA
002040  20 CC FF        0111        JSR CLRCH       ;RESTORE I/O
002043  68              0112        PLA
002044  29 7F           0113 RESTRT AND #$7F        ;STRIP OFF MSB
002046  F0 3C           0114        BEQ GETCH2      ;SKIP WENN KEIN ZEICHEN DA
00204B  C9 7F           0115        CMP #$7F        ;DELETE?
00204A  F0 38           0116        BEQ GETCH2
00204C  C9 0A           0117        CMP #LF
00204E  F0 34           0118        BEQ GETCH2      ;LINEFEED?
002050                  0119
002050  A2 06           0120        LDX #06         ;7 ITEMS GGFS. ZU WANDELN
002052  DD C5 21        0121 TOBSCD CMP CODESS,X    ;BS-CODE MACHEN
002055  F0 6B           0122        BEQ TOBS        ;JA, WANDELN
002057  CA              0123        DEX
002058  10 F8           0124        BPL TOBSCD
00205A  20 DC F3        0125 NORM   JSR TOCBM
00205D  C9 08           0126        CMP #$08        ;KOMMT DELETE?
00205F  D0 03           0127        BNE NORM1
002061  20 BE E5        0128        JSR BAKCHR      ;CURSOR LINKS RUECKEN
002064  A6 72           0129 NORM1  LDX MEMFLG      ;SPEICHERFLAG GESETZT?
002066  F0 19           0130        BEQ BSAUSG      ;NEIN, NUR AUSGEBEN
002068  48              0131        PHA             ;ZEICHEN SICHERN
002069  48              0132        PHA             ;FUER ABSPEICHERN UND AUSGABE
00206A  A0 00           0133        LDY #0          ;ZUM POINTERN
00206C  A6 01           0134        LDX $01         ;ALTE BANK SICHERN
00206E  A9 02           0135        LDA #BANK2      ;ABLAGE IN BANK 2
002070  85 01           0136        STA I6509       ;BANKUMSCHALTUNG
002072  68              0137        PLA             ;ZEICHEN
002073  91 F2           0138        STA (BOTLN),Y   ;ABLAGE
002075  E6 F2           0139        INC BOTLN       ;POINTER ERHOEHEN
002077  D0 02           0140        BNE INCP2
002079  E6 F3           0141        INC BOTLN+1
00207B  98              0142 INCP2  TYA             ;ACCU=0
00207C  91 F2           0143        STA (BOTLN),Y   ;NULLBEGRENZER
```

```
00207E  86 01        0144          STX 16509          ;ALTE BANK WIEDER HERSTELLEN
002080  68           0145          PLA                ;EMPFANGENES ZEICHEN
002081  20 D2 FF     0146 BSAUSG JSR BSOUT            ;ZEICHEN AUF BILDSCHIRM
002084                0147
002084  A9 00        0148 GETCH2 LDA #00
002086  85 D2        0149          STA QTSW           ;IMMER OHNE QUOTE-MODUS
002088  20 E4 FF     0150          JSR BASIN          ;TASTATUR PRUEFEN
00208B  F0 AA        0151          BEQ GETCH          ;KEINE TASTE
00208D               0152 ;HIER SPRUNGVERTEILER FUER CTRL-CODES
00208D  A2 1E        0153          LDX #30            ;31 ITEMS
00208F  DD 9A 21     0154 CODPAR CMP CTRLCD,X
002092  F0 34        0155          BEQ FUND           ;CTRL-CODE GEFUNDEN
002094  CA           0156          DEX
002095  10 F8        0157          BPL CODPAR
002097               0158
002097  A2 06        0159          LDX #6             ;7 CODES WANDELN
002099  DD BE 21     0160 CODWAN CMP CODEST,X       ;KEYBOARD ZU ASCII
00209C  F0 3C        0161          BEQ WANDEL
00209E  CA           0162          DEX
00209F  10 F8        0163          BPL CODWAN
0020A1               0164
0020A1  20 C7 F3     0165          JSR TOASCI         ;UMWANDLUNG TASTATUR ZU ASCII-
CODE
0020A4               0166
0020A4  48           0167 RSOUT  PHA
0020A5  A9 12        0168          LDA #$12           ;SCHALTE REVERSE EIN
0020A7  8D 97 03     0169          STA RVS            ;INS FLAG
0020AA  68           0170          PLA
0020AB  48           0171 RSOUT1 PHA                  ;AUSGABE AUF SCHNITTSTELLE RS2
32C
0020AC  A2 0E        0172          LDX #RS232
0020AE  20 C9 FF     0173          JSR CHKOUT         ;LISTEN TO RS232
0020B1  68           0174          PLA
0020B2  20 D2 FF     0175          JSR BSOUT          ;ZEICHEN AUSGEBEN
0020B5  48           0176          PHA
0020B6  20 CC FF     0177          JSR CLRCH          ;RESTORE I/O
0020B9  A9 92        0178          LDA #$92           ;NIMM REVERSE ZURUECK
0020BB  8D 97 03     0179          STA RVS
0020BE  68           0180          PLA
0020BF  4C 37 20     0181          JMP GETCH          ;ANTWORT HOLEN
0020C2               0182
0020C2               0183 ;**** CODE ASCII/BILDSCHIRM ***
0020C2  BD BE 21     0184 TOBS   LDA CODEST,X       ;UMSETZEN
0020C5  4C 64 20     0185          JMP NORM1
0020C8               0186
0020C8               0187 ;***** KONTROLLZEICHEN GEFUNDEN ***
0020C8  48           0188 FUND   PHA                ;RETTE TASTATURZEICHEN
0020C9  8A           0189          TXA                ;DEKODE MIT X*2
0020CA  0A           0190          ASL A
0020CB  AA           0191          TAX
0020CC  BD 5C 21     0192          LDA JMPTBL,X       ;ZIELADRESSE LOW
0020CF  85 73        0193          STA JUMP
0020D1  BD 5D 21     0194          LDA JMPTBL+1,X
0020D4  85 74        0195          STA JUMP+1
0020D6  68           0196          PLA                ;TASTATURZEICHEN
0020D7  6C 73 00     0197          JMP (JUMP)         ;SPRUNGVERTEILER
0020DA               0198
0020DA               0199 ;**** ZEICHENWANDLUNG TASTATUR/ASCII ***
0020DA  BD C5 21     0200 WANDEL LDA CODESS,X
0020DD  4C A4 20     0201          JMP RSOUT          ;AUSGABE AN COMPUTER
0020E0  A5 72        0202 MEMIN  LDA MEMFLG         ;UMSCHALTEN SPEICHERFLAG
0020E2  49 FF        0203          EOR #$FF
0020E4  85 72        0204          STA MEMFLG         ;UMGESCHALTET
0020E6  A9 0D        0205          LDA #CR            ;AUSGABEVORBEREITUNG
0020E8  4C A4 20     0206          JMP RSOUT          ;VORSCHUB IN NEUE ZEILE
0020EB               0207
0020EB  A6 01        0208 VIEW   LDX 16509          ;TEXTAUSGABE AUS BANK 2
0020ED  A9 02        0209          LDA #BANK2         ;NEUE DATENBANK
0020EF  85 01        0210          STA 16509
0020F1  A0 00        0211          LDY #0
0020F3  A9 00        0212          LDA #<ANFANG
0020F5  85 F1        0213          STA NOWLN+1        ;POINTER ZURUECKSETZEN
```

```
0020F7  A9 03        0214           LDA #<ANFANG+1
0020F9  85 F0        0215           STA NOWLN          ;EINGESTELLT AUF $0003
0020FB  20 27 EA     0216           JSR REVSCR         ;BILDSCHIRM INVERS
0020FE  B1 F0        0217 VIEW1     LDA (NOWLN),Y      ;GESPEICHERTE ZEICHEN
002100  F0 0C        0218           BEQ VIEW2          ;ENDE-MARKE
002102  20 D2 FF     0219           JSR BSOUT          ;AUSGABE
002105  E6 F0        0220           INC NOWLN
002107  D0 F5        0221           BNE VIEW1          ;WEITER
002109  E6 F1        0222           INC NOWLN+1
00210B  4C FE 20      0223 VIEW1A    JMP VIEW1          ;WEITERE ZEICHEN
00210E  86 01        0224 VIEW2     STX I6509          ;ALTE BANK
002110  20 3B EA     0225           JSR NRMSCR         ;BILDSCHIRM NORMAL
002113  4C 44 20      0226           JMP RESTRT
002116                0227
002116  A9 00        0228 ESCAPE    LDA #00            ;POINTER FUER SYS4, MONITOR
002118  85 04        0229           STA 04
00211A  60            0230           RTS                ;RUECKKEHR ZU BASIC
00211B                0231 ;
00211B  A9 0D        0232 LINFED    LDA #CR            ;WANDELN
00211D  4C A4 20      0233           JMP RSOUT
002120  A9 08        0234 DELETE    LDA #$08
002122  4C A4 20      0235           JMP RSOUT          ;COMPUTER ANTWORTET $08,$20,$0
8, MIT SPACE ALSO
002125                0236 ZEICHN
002125  A9 20        0237 CURRIG    LDA #' '           ;SPACE AUSGEBEN
002127  4C A4 20      0238           JMP RSOUT
00212A  20 1C EA      0239 ALTERN    JSR $EA1C          ;SCHALTE MIT CTRL-A ZEICHENSAT
Z ASCII EIN
00212D  4C 37 20      0240           JMP GETCH
002130                0241
002130  20 33 EA      0242 NORMAL    JSR $EA33          ;SCHALTE DEUTSCHE ZEICHEN EIN
002133  4C 37 20      0243           JMP GETCH
002136                0244
002136  20 42 21      0245 INIMEM    JSR INITPT         ;TEXTSPEICHER LEEREN
002139  4C 37 20      0246           JMP GETCH
00213C                0247
00213C  20 29 30      0248 EDITOR    JSR $3029          ;UEBERGANG IN SEPARATEN EDITOR
00213F  4C 37 20      0249           JMP GETCH
002142                0250
002142                0251 ;-------------------------
002142  A6 01        0252 INITPT    LDX I6509          ;RETTE BANK
002144  A9 02        0253           LDA #BANK2
002146  85 01        0254           STA I6509
002148  A0 00        0255           LDY #00            ;INITIALISIERE SPEICHERZEIGER
00214A  A9 02        0256           LDA #<ANFANG       ;ANFANGSWERT $0002
00214C  85 F2        0257           STA BOTLN
00214E  A9 00        0258           LDA #>ANFANG
002150  85 F3        0259           STA BOTLN+1
002152  98          0260           TYA                ;ACCU=0
002153  91 F2        0261           STA (BOTLN),Y      ;BEGRENZERZEICHEN VOR TEXT
002155  E6 F2        0262           INC BOTLN          ;AUSGANGSSTELLUNG JETZT $0003
002157  91 F2        0263           STA (BOTLN),Y      ;MARKIERE ZUNAECHST TEXTENDE
002159  86 01        0264 INT1      STX I6509          ;BANK ZURUECK
00215B  60            0265           RTS
00215C                0266 ;-------------------------
00215C                0267 ;SPRUNGTAFEL FUER VERSCH. FUNKTIONEN - 31 POSITIONEN
00215C                0268 ;CONTROL_CODES USW. GETCH= ZEICHEN NICHT AUSWERTEN
00215C                0269 ;RSOUT = WEITERGABE AN COMPUTER
00215C  37 20        0270 JMPTBL .WOR GETCH,ALTERN,MEMIN,RSOUT,RSOUT,RSOUT ;CTRL
-E
00215E  2A 21        0270
002160  E0 20        0270
002162  A4 20        0270
002164  A4 20        0270
002166  A4 20        0270
002168  3C 21        0271           .WOR EDITOR,GETCH,DELETE,RSOUT,LINFED,ZEICHN ;C
TRL-F-K
00216A  37 20        0271
00216C  20 21        0271
00216E  A4 20        0271
002170  1B 21        0271
002172  25 21        0271
```

```
002174  37 20          0272          .WOR GETCH,RSOUT,NORMAL,GETCH,RSOUT,LINFED ;CTR
L-L-Q
002176  A4 20          0272
002178  30 21          0272
00217A  37 20          0272
00217C  A4 20          0272
00217E  1B 21          0272
002180  A4 20          0273          .WOR RSOUT,RSOUT,DELETE,RSOUT,VIEW,RSOUT ;CTRL-
R-W
002182  A4 20          0273
002184  20 21          0273
002186  A4 20          0273
002188  EB 20          0273
00218A  A4 20          0273
00218C  A4 20          0274          .WOR RSOUT,INIMEM,RSOUT,ESCAPE ;CTRL-X-Z, ESCAP
E
00218E  36 21          0274
002190  A4 20          0274
002192  16 21          0274
002194  37 20          0275          .WOR GETCH,CURRIG,DELETE ;2 CURSORTASTEN
002196  25 21          0275
002198  20 21          0275
00219A                 0277 ;******** BYTES FUER TAFEL JMPTBL ******
00219A  00             0278 CTRLCD .BYT 0,1,2,3,4,5,6,7,8,9,$A,$B,$C,$D,$E,$F
00219B  01             0278
00219C  02             0278
00219D  03             0278
00219E  04             0278
00219F  05             0278
0021A0  06             0278
0021A1  07             0278
0021A2  08             0278
0021A3  09             0278
0021A4  0A             0278
0021A5  0B             0278
0021A6  0C             0278
0021A7  0D             0278
0021A8  0E             0278
0021A9  0F             0278
0021AA  10 11          0279          .DBY $1011,$1213,$1415,$1617,$1819,$1A1B ;BIS E
SCAPE
0021AC  12 13          0279
0021AE  14 15          0279
0021B0  16 17          0279
0021B2  18 19          0279
0021B4  1A 1B          0279
0021B6  1C 1D          0280          .DBY $1C1D          ;XX, CURSOR RIGHT
0021B8  9D             0281          .BYT $9D            ;CURSOR LEFT
0021B9                 0282
0021B9                 0283 OPNSTR                       ;'FILENAME' DER RS 232, OPEN-S
TRING
0021B9  BE             0284          .BYT $BE            ;9600 BAUD, 8 DATENBITS,KEINE
PARITAET
0021BA  10             0285          .BYT $10            ;ECHO MODE AN
0021BB  00             0286          .BYT 0              ;2 BYTE MUESSEN NOCH KOMMEN
0021BC  00             0287          .BYT 0              ;OHNE BEDEUTUNG
0021BD                 0288 ;KONTROLL-BYTES
0021BD  C0             0289 BITBYT .BYT $C0              ;KONTROLLE EMPFANGENES ZEICHEN
0021BE                 0290 ;CODES FUER UMWANDLUNG BEIM SENDEN/EMPFANGEN
0021BE  BB             0291 CODEST .BYT $BB,$BC,$BD,$DB,$DC,$DD,$BE ;TASTEN AKKOLA
DEN++
0021BF  BC             0291
0021C0  BD             0291
0021C1  DB             0291
0021C2  DC             0291
0021C3  DD             0291
0021C4  BE             0291
0021C5  7B             0292 CODESS .BYT $7B,$7C,$7D,$5B,$5C,$5D,$7E ;ASCII FUER CO
MPUTER
0021C6  7C             0292                    0021CA  5D          0292
0021C7  7D             0292                    0021CB  7E          0292
0021C8  5B             0292                    0021CC              0293          .END
0021C9  5C             0292                     ERRORS=0000
```

Kurt Peter, 8000 München 80

6502 als Frequenzzähler

Übersicht

Es wird ein Programm beschrieben, das es ermöglicht, mit einem Rechner Frequenzen bis 400 kHz zu messen. Für zusätzliche Hardware muß nur etwa DM 1,- investiert werden.

Wenn der Rechner komplexe, frei programmierbare Peripheriebausteine besitzt, so verfügt man auf sehr kleinem Raum über eine Menge digitaler Funktionen, die sonst nur durch viele externe Bausteine erreicht werden könnten. - Mit diesem Programm soll gezeigt werden, wie man die VIA 6522 und deren Fähigkeit zur Ereigniszählung ausnutzen kann, um Frequenzen zu messen.

Eigenschaften

Der hier realisierte Frequenzzähler zählt ohne Vorteiler Frequenzen bis etwa 400 kHz mit einer Auflösung von 6 Stellen bei einer Torzeit von 1 Sekunde bzw. 7 Stellen bei einer Torzeit von 10 Sekunden. Die Begrenzung ergibt sich aus der System-Clock des 6522, die bei 1 MHz liegt. Neben dem Rechner mit 6502 und VIA 6522 wird an Hardware eine Anzeige benötigt, die mindestens 16 ASCII-Zeichen darstellen kann. Wenn man auf etwas Komfort verzichtet, dann genügt auch eine einfache Hex-Anzeige für die Ziffern. Für Frequenzen über 50 kHz empfiehlt sich ein UND-Gatter als Gate.

Die Software von weniger als 0,5 KB nutzt intensiv die Timer und Interrupts der VIAs. Sie benötigt nur eine einzige Systemroutine, die man praktisch auf allen Systemen findet: Ausgabe eines ASCII-Zeichens auf das Display. Wichtig ist, daß keine anderen Interrupts bearbeitet werden müssen als die des Freqenzzählerprogrammes.

Bisherige Veröffentlichungen zum Thema weisen einige Nachteile auf: In /1/ wird z.B. ein Teil des Zählvorganges auf externe Hardware verlegt, der Rechner dient dabei im wesentlichen als Anzeigebaustein. /2/ benutzt zur Frequenzzählung nicht den Timer des 6522, sondern den IRQ, so daß die Grenzfrequenz sich etwas oberhalb des NF-Bereiches bewegt. In /3/ schließlich werden die Nulldurchgänge von der CPU selbst gezählt, was ebenfalls auf die Grenzfrequenz drückt.

Zur Software

Da das Programm ausführlich kommentiert ist, soll hier nur auf das Konzept eingegangen werden: Die Hauptarbeit wird vom 6522 geleistet. Timer 2 wird als Ereigniszähler programmiert. Da er 16 Bit breit ist, kann er in einer Sekunde ca. 65000 negative Flanken zählen. Um den Meßbereich zu erweitern, wird ein Interrupt durch den Zählerüberlauf erzeugt, so daß die CPU einen Zähler inkrementieren kann. - Timer 1 wird als Zeitbasis herangezogen. Er wird in den Free Run–Modus versetzt und erzeugt Interrupts im Abstand von 50 ms. Diese werden ebenfalls von der CPU gezählt. Nach 20 Interrupts von Timer 1 wird das Gate geschlossen, die Zählerwerte gerettet und die Timer neu geladen. Über das Byte NEWVAL wird dem in einer Warteschleife hängenden Hauptprogramm mitgeteilt, daß es einen neuen Wert in das Display bringen muß. Es erfolgt eine Umwandlung der Binärwerte in BCD-Ziffern, wobei die Wandelroutine nach /4/ gute Dienste leistet. Jetzt erfolgt noch die ASCII-Konvertierung und die Ausgabe auf das Display.

Es ergaben sich einige Probleme, die hier noch kurz angedeutet werden sollen, weil sie u.U. auch für andere Anwendungen relevant sind. Zum einen müssen die Timer immer byteweise bedient werden. Da z.B. das Auslesen des niederwerigen Bytes nicht mit einem Latchen des gesamten Timerwortes verbunden ist, sollte ein Gate vorgesehen werden, damit nicht zwischen dem Auslesen des Low und des High Bytes ein Übertrag aus dem Low Byte entsteht, den das High Byte dann noch registrieren würde. Bei niedrigen Frequenzen kommt eine solche Verfälschung des Ergebnisses allerdings selten vor, da zwischen dem Auslesen der beiden Bytes gar keine Flanke mehr auftritt.

Zum zweiten ergibt sich eine gewisse Reaktionszeit für jeden Interrupt, was sich vor allem nach dem 20. und 200. Interupt der Zeitbasis Timer 1 schmerzhaft bemerkbar macht. In dieser Zeit werden natürlich einige Impulse mehr gezählt. Ein Ausgleich wird dadurch erzielt, daß beim

Neusetzen der Timer zuerst die Zeitbasis und dann, mit einer gewissen Verzögerung, der Ereigniszähler geladen wird, so daß die Reaktionszeit wieder ausgeglichen wird..

Des weiteren läßt sich das Interrupt-Flag des Timers T2 nur durch ein Schreiben von T2H löschen. Dadurch wird aber auch T2L neu vom Latch geladen, obwohl er vielleicht schon, vor allem bei hohen Frequenzen, einige Flanken gezählt hat. Dies wird durch eine kleine Korrekturrechnung ausgeglichen, auf die in der Programm-Liste hingewiesen wird.

Trotz all' dieser Unzulänglichkeiten kann auch bei 400 kHz noch die letzte Stelle ausgewertet werden. Sicher wurden die Möglichkeiten für eine optimale Korrektur nicht voll ausgeschöpft, so daß sich Gelegenheit für eine weitere Verbesserung des Programms ergibt.

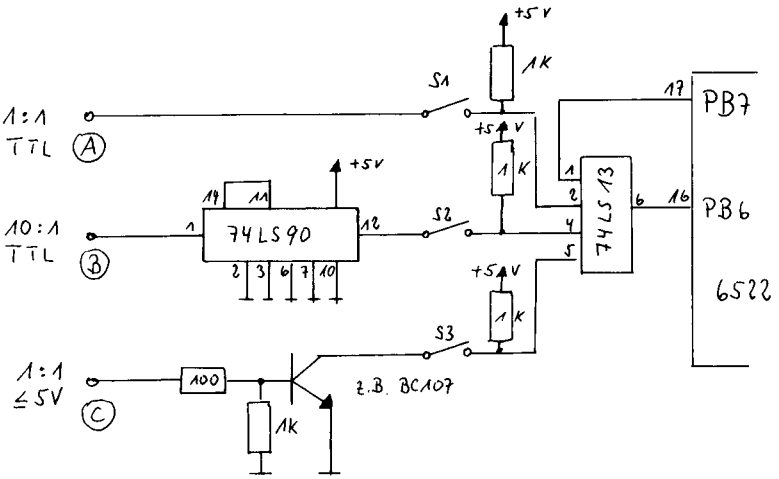
Auf ein Anwendermenue wurde bewußt verzichtet, um das Programm maschinen-unabhängig zu halten. Den wichtigsten Parameter, die Zeitbasis, kann man im dritten Programm-Byte 'per Hand' ändern, ohne neu assemblieren zu müssen (hex 14 = 1 Sek., hex C8 = 10 Sek.). Bei Bedarf kann das Programm natürlich anwenderfreundlicher ausgestaltet werden.

Hardware-Ergänzungen

Bedingt durch die oben beschriebenen kleinen Unzulänglichkeiten des 6522 sollte als Minimalaufwand ein Gate vorgesehen werden, um den vollen Frequenzbereich ausnutzen zu können. Die bei folgende Skizze zeigt ein Beispiel für eine etwas komfortablere Hardware. An Punkt A werden TTL-Signale 1:1 gezählt, an Punkt B ist ein 10:1-Teiler eingeschleift und an Punkt C können auch Pegel kleiner als 5 Volt gemessen werden. Die Umschaltung kann man durch einen kleinen DIL-Schlater bewirken. Natürlich gibt es auch Varianten dazu. Z.B ist in /5/ ein sehr guter 1000:1-Teiler für Frequenzen bis 1 GHz beschrieben, der mit geringem Aufwand selbst aufgebaut werden kann.

Literatur

/1/ Helbig, Frank: AIM 65 als interrupt-gesteuerter Frequenzzähler, Chip-Spezial 4/81, Vogel-Verlag, Würzburg
/2/ Sullivan, C.: uP als Minizähler, Elektor 5/82, Elektor-Verlag, Gangelt
/3/ Feichtinger, Herwig: KIM als Nf-Zähler, Funkschau 6/79, Franzis Verlag, München
/4/ Rix, Peter: Binär-BCD-Wandlung, 65xx MICRO MAG, Heft 22
/5/ Mazur, Hartmut: Einfacher Vorteiler für Frequenzen bis über 1 GHz mit Dezimalisierung des Teilerfaktors, cq-DL 2/83, Clubzeitschrift des DARC, Fachorgan für den Amateurfunkdienst



Die Beschaltung

```
0000          ;****************************************
0000          ;*                                      *
0000          ;* 6502-RECHNER ALS FREQUENZZAEHLER     *
0000          ;* BEISPIEL: AIM65/PC100                *
0000          ;* V030385                              *
0000          ;* (C)          KURT PETER              *
0000          ;*               BAUMKIRCHNER STR. 3    *
0000          ;*               8000 MUENCHEN 80       *
0000          ;*                                      *
0000          ;****************************************
0000
0000
0000 START            =$0200
0000
0000                  *=$010C
010C          4C0002 JMP INIT
010F
010F          ; #### EINZIGE SPEZIFISCHE SYSTEMROUTINE! ####
010F OUTALL           =$E9BC            ;OUTPUT ONE ASCII-CHAR TO DISPLAY
010F
010F IRQVEK           =$A404
010F
010F VIA              =$A000
010F
010F UDRB             =VIA+$0
010F UDDRB            =VIA+$2
010F UT1CL            =VIA+$4
010F UT1CH            =VIA+$5
010F UT1LL            =VIA+$6
010F UT2L             =VIA+$8
010F UT2H             =VIA+$9
010F UACR             =VIA+$B
010F UIFR             =VIA+$D
010F UIER             =VIA+$E
010F
010F
010F DELVAL           = 65              ;VERZOEGERUNG FUER GATE-UNLOCK
010F DBLEN            = 20              ;DISPLAY BUFFER LENGTH
010F CMASK            =%01100000        ;T2=CNT PB6-PULS, T1=FREE-RUN
010F ENFLAG           =%11100000        ;IRENABLE FUER IER-REG
010F DSFLAG           =%00011111        ;DISABLE ALL OTHER IR
010F T2VALU           =$FFFF            ;TIMER 2 VALUE
010F T1VALU           = 49998           ;TIMER 1 VALUE
010F                ;49998 BIS 0 = 49999, + 1 FUERS LADEN BEI 0
010F                ;=> 50000 US = 50 MS
010F PMASK            =%10000000        ;PB7=OUT, PB6=IN
010F T1IR             =%01000000        ;T1-IR-FLAG
010F T2IR             =%00100000        ;T2-IR-FLAG
010F LOCK             =%01000000        ;GATE VERRIEGELN
010F UNLOCK           =%11000000        ;GATE FREIGEBEN
010F TB1S             = 20              ;TIME BASE 1 S == 20 * 50MS
010F TB10S            = 200             ;TIME BASE 10 S == 200 * 50MS
010F N                = 3               ;ANZAHL DER BINAERSTELLEN
010F I                = 24              ;ANZAHL DER HEX-BITS
010F M                = 4               ;ANZAHL DER BCD-STELLEN
010F SPACE            =$20
010F CR               =$0D
010F                  *=0               ;ZEROPAGE VARIABLES
0000
0000 HEX
0000 VALMSB           *=*+1             ;MSB DES MESSWERTS <= T2IRCT
0001 VALHI            *=*+1             ;HIGH VALUE VON READ T2
0002 VALLOW           *=*+1             ;LOW VALUE VON READ T2
0003 TBASE            *=*+1             ;TIME BASE VALUE LOW
0004 TBCNT            *=*+1             ;TIME BASE COUNT, ZAEHLT T1 IRS
0005 TBFLAG           *=*+1             ;TIME BASE FLAG
0006 NEWVAL           *=*+1             ;NEW VALUE AVAILABLE
0007 T2IRCT           *=*+1             ;ZAEHLT DIE UEBERLAEUFE VON T2
0008 TEMP             *=*+1
0009 BCD              *=*+M             ;BUFFER FUER DIE BCD-ZAHL
000D DISBUF           *=*+DBLEN         ;DISPLAY BUFFER
```

```
0021                    *=START
0200
0200 INIT
0200            78      SEI
0201            A914    LDA #TB1S       ;ZEITBASIS FESTLEGEN
0203            8503    STA TBASE
0205            A213    LDX #DBLEN-1    ;DISPLAYBUFFER LOESCHEN
0207            A920    LDA #SPACE
0209 CLDB       950D    STA DISBUF,X
020B            CA      DEX
020C            10FB    BPL CLDB
020E            A953    LDA #'S'        ;DIVERSE HINWEISE INS DISPLAY
0210            8510    STA DISBUF+3
0212            A948    LDA #'H'
0214            8518    STA DISBUF+14
0216            A95A    LDA #'Z'
0218            851C    STA DISBUF+15
021A            A503    LDA TBASE
021C            C914    CMP #TB1S
021E            D007    BNE INI1
0220            A931    LDA #'1'
0222            850E    STA DISBUF+1
0224            4C2F02  JMP INI2
0227 INI1       A930    LDA #'0'
0229            850E    STA DISBUF+1
022B            A931    LDA #'1'
022D            850D    STA DISBUF+0
022F INI2       20E102  JSR OUTDB
0232            A940    LDA #LOCK       ; GATE VERRIEGELN
0234            8D00A0  STA UDRB
0237            A900    LDA #0
0239            8506    STA NEWVAL
023B            8507    STA T2IRCT
023D            209F02  JSR LDTBCT      ;TIME BASE COUNTER LADEN
0240            209402  JSR DECTB
0243            A980    LDA #PMASK
0245            8D02A0  STA UDDRB       ;SET PORT DIRECTION
0248            A96C    LDA #<IRROUT    ;IRQ-VEKTOR UMBIEGEN
024A            8D04A4  STA IRQVEK
024D            A903    LDA #>IRROUT
024F            8D05A4  STA IRQVEK+1
0252            A960    LDA #CMASK      ; SET VIA MODE
0254            8D0BA0  STA UACR
0257            20A802  JSR LDT1        ; TIMER LADEN
025A            20B302  JSR LDT2
025D            A91F    LDA #DSFLAG     ; DISABLE INVALID IRS
025F            8D0EA0  STA UIER
0262            A9E0    LDA #ENFLAG     ; IR-ENABLE-REG FREIGEBEN
0264            8D0EA0  STA UIER
0267            A9C0    LDA #UNLOCK     ;GATE FREIGEGEN
0269            8D00A0  STA UDRB
026C            58      CLI
026D WAIT
026D            A506    LDA NEWVAL
026F            F0FC    BEQ WAIT
0271            203A03  JSR CORREC
0274            20BE02  JSR HEXBCD      ; ... WERT UMWANDELN ...
0277            A204    LDX #4          ;AUSGABEPOSITION FUER DISBUF
0279            A000    LDY #0
027B LAB3       B90900  LDA BCD,Y       ;BCD-ZIFFERN IN DISBUF
027E            20F302  JSR BCDASC
0281            E8      INX
0282            C8      INY
0283            C004    CPY #M
0285            D0F4    BNE LAB3
0287            201303  JSR DELBL       ;FUEHRENDE NULLEN RAUS
028A            20E102  JSR OUTDB       ; DISPLAY BUFFER AUSGEBEN
028D            A900    LDA #0          ;CLEAR NEW VALUE FLAG
028F            8506    STA NEWVAL
0291            4C6D02  JMP WAIT
0294
0294            ;***************** UNTERROUTINEN ***************
```

```
0294 DECTB
0294
0294            ; DIESE ROUTINE DEKREMENTIERT DEN
0294            ; TIME BASE COUNTER UND MELDET DESSEN
0294            ; ABLAUF IN TBFLAG
0294            A504    LDA TBCNT       ; TBASE DEKREMENTIEREN
0296            C604    DEC TBCNT       ; TBASE ABGELAUFEN....
0298            D004    BNE TBCNT2      ; ...DANN VORWARNEN
029A            A9FF    LDA #$FF
029C            8505    STA TBFLAG
029E TBCNT2     60      RTS
029F
029F LDTBCT
029F            ; LADE TIME BASE COUNTER UND .....
029F            ; LOESCHE TIME BASE FLAG
029F            A900    LDA #0
02A1            8505    STA TBFLAG      ;TIME BASE FLAG LOESCHEN
02A3            A503    LDA TBASE       ;ZEITBASIS LADEN
02A5            8504    STA TBCNT
02A7            60      RTS
02A8
02A8 LDT1
02A8            ; LADE TIMER 1
02A8            A94E    LDA #<T1VALU
02AA            8D06A0  STA UT1LL
02AD            A9C3    LDA #>T1VALU
02AF            8D05A0  STA UT1CH
02B2            60      RTS
02B3
02B3 LDT2
02B3            ; LADE TIMER2
02B3            A9FF    LDA #<T2VALU
02B5            8D08A0  STA UT2L
02B8            A9FF    LDA #>T2VALU
02BA            8D09A0  STA UT2H
02BD            60      RTS
02BE
02BE HEXBCD
02BE            ;WANDELROUTINE VON HEX NACH BCD
02BE            F8      SED
02BF            A900    LDA #0
02C1            A203    LDX #M-1
02C3 AAA        9509    STA BCD,X       ;FELD LOESCHEN
02C5            CA      DEX
02C6            10FB    BPL AAA
02C8            A018    LDY #I          ;BITZAEHLER HOLEN
02CA BBB        A202    LDX #N-1
02CC CCC        3600    ROL HEX,X
02CE            CA      DEX
02CF            10FB    BPL CCC
02D1            A203    LDX #M-1
02D3 DDD        B509    LDA BCD,X
02D5            7509    ADC BCD,X
02D7            9509    STA BCD,X
02D9            CA      DEX
02DA            10F7    BPL DDD
02DC            88      DEY
02DD            D0EB    BNE BBB         ;NAECHSTES BIT
02DF            D8      CLD
02E0            60      RTS
02E1
02E1 OUTDB
02E1            ; GIBT DEN DISPLAYBUFFER AUS
02E1            A90D    LDA #CR
02E3            20BCE9  JSR OUTALL
02E6            A200    LDX #0
02E8 OUTDB1
02E8            B50D    LDA DISBUF,X
02EA            20BCE9  JSR OUTALL
02ED            E8      INX
02EE            E014    CPX #DBLEN
02F0            D0F6    BNE OUTDB1
02F2            60      RTS
```

```
02F3 BCDASC
02F3          ; WANDELT EINE BCDZAHL IN ZWEI ....
02F3          ; ...ASCII-ZEICHEN UM
02F3          ; ERWARTET IN X DIE ERSTE DISPLAYPOSITION
02F3          48      PHA
02F4          4A      LSR A
02F5          4A      LSR A
02F6          4A      LSR A
02F7          4A      LSR A
02F8          200E03  JSR BCDAS1
02FB          E00A    CPX #10
02FD          D00B    BNE BCDAS0
02FF          A503    LDA TBASE       ; BEI TBASE = 10S MUSS ....
0301          C914    CMP #TB1S       ; .. JETZT DER DEZIMALPUNKT ...
0303          F005    BEQ BCDAS0      ; .. FOLGEN
0305          E8      INX
0306          A92E    LDA #'.'
0308          950D    STA DISBUF,X
030A BCDAS0   E8      INX
030B          68      PLA
030C          290F    AND #$0F
030E BCDAS1   0930    ORA #$30
0310          950D    STA DISBUF,X
0312          60      RTS
0313
0313 DELBL
0313          ; LOESCHT FUEHRENDE NULLEN IM
0313          ; DISPLAYBUFFER
0313          A020    LDY #SPACE
0315          A204    LDX #4
0317 DELBL1   B50D    LDA DISBUF,X
0319          C930    CMP #'0'
031B          F004    BEQ DELBL2
031D          C920    CMP #SPACE
031F          D007    BNE DELEND
0321 DELBL2   98      TYA
0322          950D    STA DISBUF,X
0324          E8      INX
0325          4C1703  JMP DELBL1
0328 DELEND   60      RTS
0329
0329 CLT2IR
0329          ; CLEAR IR-FLAG VON T2 UND ENABLE KUENFTIGE
0329          ; INTERRUPTS.
0329          ; HINWEIS: NUR DAS SCHREIBEN VON T2CH LOESCHT
0329          ; DAS IR-FLAG  **UND**  ENABLED KUENFTIGE IRS
0329          ; LEIDER BEWIRKT DAS SCHREIBEN VON T2CH ABER
0329          ; AUCH EIN NEULADEN VON T2CL. DORT SIND ABER
0329          ; - BESONDERS BEI HOHEN FREQUENZEN - SCHON WIEDER
0329          ; EINIGE IMPULSE GEZAEHLT WORDEN, DIE DURCH DAS
0329          ; NEULADEN VON T2CL MIT T2LL VERLOREN GEHEN.
0329          ; DESHALB FOLGENDE VORGEHENSWEISE:
0329          ; ERST WIRD DER WERT FUER T2CH GEHOLT.
0329          ; VOR DEM EINSCHREIBEN IN T2CH WIRD T2CL AUSGELESEN
0329          ; UND IN T2LL EINGESCHRIEBEN. JETZT WIRD SOFORT T2CH
0329          ; GELADEN, DER AUCH T2LL MITLAEDT.
0329          ; BEI FREQUENZEN > CA. 100 KHZ GEHEN ZWISCHEN
0329          ; DEN BEIDEN SCHREIBAKTIONEN TROTZDEM NOCH IMPULSE
0329          ; VERLOREN (ETWA PROPORTIONAL ZUR FREQUENZ). DIES
0329          ; MUSS DANN EINE KORREKTURRECHNUNG (CORREC)
0329          ; AUSGLEICHEN
0329          A9FF    LDA #>T2VALU    ;T2CH-VALUE HOLEN
032B          AE08A0  LDX UT2L        ;T2CL AUSLESEN
032E          8E08A0  STX UT2L        ;ZURUECKSCHREIBEN
0331          8D09A0  STA UT2H        ;EBENSO T2CH
0334          A9FF    LDA #<T2VALU    ;T2LL WIEDERHERSTELLEN
0336          8D08A0  STA UT2L
0339          60      RTS
033A
033A CORREC
033A          ; BEI ZWEI UND MEHR IR VON T2 WAEHREND
033A          ; DER MESSZEIT LIEGT ALSO AN PB2 EINE FREQUENZ
```

```
033A            ; > 100 KHZ AN, DARAUS FOLGT, DASS WAEHREND
033A            ; DES LOESCHENS VON T2-IR-FLAG IMPULSE VERLOREN
033A            ; GEHEN UND ZWAR UM SO MEHR JE HOEHER DIE
033A            ; FREQUENZ IST. DESHALB IST EINE KORREKTUR DES
033A            ; MESSWERTS IN ABHAENGIGKEIT VON DEN T2-IRS NOETIG
033A            A503    LDA TBASE
033C            C914    CMP #TB1S       ;TIME BASE 1 S?
033E            F009    BEQ CORR1
0340            A500    LDA VALMSB
0342            C914    CMP #20
0344            3025    BMI COREND
0346            4C4F03  JMP CORR
0349 CORR1      A500    LDA VALMSB      ;2 ODER MEHR IRS?
034B            C902    CMP #2
034D            301C    BMI COREND      ;NEIN
034F CORR       8508    STA TEMP        ;ZWISCHENSPEICHERN
0351            0608    ASL TEMP        ; * 2
0353            18      CLC
0354            6508    ADC TEMP        ; INSGESAMT ALSO * 3
0356            8508    STA TEMP
0358            18      CLC             ; UND EVTL. EINEN CARRY DURCHZIEHEN
0359            A502    LDA VALLOW
035B            6508    ADC TEMP
035D            8502    STA VALLOW
035F            A501    LDA VALHI
0361            6900    ADC #0
0363            8501    STA VALHI
0365            A500    LDA VALMSB
0367            6900    ADC #0
0369            8500    STA VALMSB
036B COREND     60      RTS
036C
036C
036C            ; ************* INTERRUPT HANDLING ***********
036C
036C
036C IRROUT
036C            48      PHA             ;3
036D            98      TYA             ;2
036E            48      PHA             ;3
036F            8A      TXA             ;2
0370            48      PHA             ;3
0371            A940    LDA #T1IR       ;IR VON TIME BASE? (2)
0373            2C0DA0  BIT UIFR        ;4
0376            504B    BVC T2HNDL      ;2
0378            2405    BIT TBFLAG      ;JA, TIME BASE ABGELAUFEN? (4)
037A            503E    BVC IRR1        ;2
037C            A940    LDA #LOCK       ;JA, ALSO .... (2)
037E            8D00A0  STA UDRB        ;GATE OFF (4)   SUMME = 33 +IR
0381            AD08A0  LDA UT2L        ;WERT AUSLESEN....
0384            49FF    EOR ##$FF       ;COUNTER ZAEHLT RUECKWAERTS
0386            8502    STA VALLOW
0388            AD09A0  LDA UT2H
038B            49FF    EOR ##$FF       ;COUNTER ZAEHLT RUECKWAERTS
038D            8501    STA VALHI
038F            A920    LDA #T2IR       ;WENN NOCH EIN IR VON T2 ...
0391            2C0DA0  BIT UIFR        ; ... ANSTEHT, MUSS ER AUCH ...
0394            F005    BEQ CONT1       ; .. BERUECKSICHTIGT WERDEN
0396            E607    INC T2IRCT
0398            202903  JSR CLT2IR      ;T2-IR LOESCHEN
039B CONT1      A507    LDA T2IRCT
039D            8500    STA VALMSB
039F            A900    LDA #0
03A1            8507    STA T2IRCT
03A3            A9FF    LDA ##$FF       ; .. UND MELDUNG AN HAUPTPROGRAMM
03A5            8506    STA NEWVAL
03A7            209F02  JSR LDTBCT      ;TIME BASE COUNTER LADEN
03AA            20A802  JSR LDT1        ;TIMER 1 NEU AUFSETZEN (RTS=6US)
03AD            20B302  JSR LDT2        ;TIMER 2 NEU + CLEAR IR (6+18)
03B0            EA      NOP             ;2
03B1            EA      NOP             ;2
```

```
03B2          EA     NOP          ;2
03B3          EA     NOP          ;2
03B4          EA     NOP          ;2
03B5          A9C0   LDA #UNLOCK  ;2
03B7          8D00A0 STA UDRB     ;4  SUMME =46 US
03BA IRR1     209402 JSR DECTB    ;TIME BASE COUNTER DECR.
03BD IRR2     AD04A0 LDA UT1CL    ; T1-IR-FLAG LOESCHEN
03C0          4CC803 JMP IRRDY
03C3
03C3 T2HNDL
03C3          E607   INC T2IRCT   ;IR ZAEHLEN
03C5          202903 JSR CLT2IR   ;IR FLAG LOESCHEN
03C8
03C8 IRRDY
03C8          68     PLA
03C9          AA     TAX
03CA          68     PLA
03CB          A8     TAY
03CC          68     PLA
03CD          40     RTI
03CE
03CE                 .FIL
03CE                 .END
```

○≡

Dipl.-Ing. H. Grunewald, 2150 Buxtehude

DISMOS V1.0 (2)

In Heft 41 dieser Zeitschrift (Februar 1985) wurde mit dem Abdruck dieses Disassemblers begonnen. Er ist für den AIM 65 geschrieben und disassembliert Maschinenprogramme, die für die CMOS-Version der CPU 65C02 geschrieben worden sind. Hier folgt der Rest des Programmes.

```
70F4
70F4 COL7   202171 JSR BITNUM     ; RMB, SMBO...7
70F7        200D71 JSR NEXT       ; Zero-Page ausgeben
70FA
------------------------------------------------------------
ENTRY FÜR

70FA ONE    A901   LDA #1         ; Zwei-Byte-Befehle
70FC        2C     .BYT $2C
70FD TWO    A902   LDA #2         ; Drei-Byte-Befehle
70FF        2C     .BYT $2C
7100 ZER    A900   LDA #0         ; Ein-Byte-Befehle
7102        85EA   STA LENGTH
7104        60     RTS
------------------------------------------------------------
HILFSROUTINEN

7105 NEXT2  A002   LDY #2         ; zwei Zeichen holen/ausgeben
7107        2056EB JSR PCLLD      ;LDA ($A425),Y
710A        2046EA JSR NUMA
710D NEXT   A001   LDY #1         ; nächstes Zeichen holen/ausgeben
710F        2056EB JSR PCLLD      ;LDA ($A425),Y
7112        2046EA JSR NUMA
7115        60     RTS
7116 KOMMAX A92C   LDA #','       ; ',X' ausgeben
7118        207AE9 JSR OUTPUT
711B        A958   LDA #'X'
711D        207AE9 JSR OUTPUT
7120        60     RTS
```

```
7121 BITNUM A000    LDY #0          ; Bit-Nummer ausgeben
7123        2056EB  JSR PCLLD       ;LDA ($A425),Y
7126        2970    AND #$70
7128        4A      LSR A
7129        4A      LSR A
712A        4A      LSR A
712B        4A      LSR A
712C        2051EA  JSR NOUT
712F        203EE8  JSR BLANK
7132        60      RTS
7133 REL    18      CLC             ; rel. Sprungadresse ermitteln
7134        AD25A4  LDA SAVPC       ; und ausgeben
7137        6902    ADC #2
7139        8D1701  STA REG+1
713C        AD26A4  LDA SAVPC+1
713F        6900    ADC #0
7141        8D1801  STA REG+2
7144        18      CLC
7145        A001    LDY #1
7147        2056EB  JSR PCLLD       ;LDA ($A425),Y
714A        1003    BPL AC4
714C        CE1801  DEC REG+2
714F AC4    6D1701  ADC REG+1
7152        8D1701  STA REG+1
7155        9003    BCC AC5
7157        EE1801  INC REG+2
715A AC5    AD1801  LDA REG+2
715D        2046EA  JSR NUMA
7160        AD1701  LDA REG+1
7163        2046EA  JSR NUMA
7166        60      RTS
```

TABELLEN

X-Index für Adresse in ASCII-Tabelle

```
7167 ASCPOS
7167        00      .BYT 0,$10,$2,$1A,$12,$1D,$20,6,$17,$27,$2A,$25,$3A
7168        10
7169        02
716A        1A
716B        12
716C        1D
716D        20
716E        06
716F        17
7170        27
7171        2A
7172        25
7173        3A
7174        3A      .BYT $3A,$25,$31,$04,8,$D,$34,$37,$27,$2A,$25,$B,$3A
7175        25
7176        31
7177        04
7178        08
7179        0D
717A        34
717B        37
```

```
717C        27
717D        2A
717E        25
717F        0B
7180        3A
7181        3A      .BYT $3A,$2E,$23,$2C,$15
7182        2E
7183        23
7184        2C
7185        15
7186
-------------------------------------------------------------
Code-Tabelle

7186 CODE
7186        80      .BYT $80        ; BRA
7187        12      .BYT $12,$32,$52,$72,$92,$B2,$D2,$F2,$04,$14,$34,$64
7188        32
7189        52
718A        72
718B        92
718C        B2
718D        D2
718E        F2
718F        04
7190        14
7191        34
7192        64
7193        74      .BYT $74,$89,$1A,$3A,$5A,$7A,$DA,$FA,$0C,$1C,$3C,$7C
7194        89
7195        1A
7196        3A
7197        5A
7198        7A
7199        DA
719A        FA
719B        0C
719C        1C
719D        3C
719E        7C
719F        9C      .BYT $9C,$9E
71A0        9E
71A1        07      .BYT $07        ; Spalte 07 bis F7
71A2        87      .BYT $87
71A3        0F      .BYT $0F        ; Spalte 0F bis FF
71A4        8F      .BYT $8F        ; BBS0 ZP
71A5
-------------------------------------------------------------
ASCII-Tabelle

71A5 ASCTAB
71A5        4252    .BYT 'BRANDECMPHYJMPLYORAD'
71B9        4342    .BYT 'CBBSBCEORSTALDASMBIT'
71CD        5342    .BYT 'SBTRBBRMBINCPHXPLXST'
71E1        5A      .BYT 'Z'
71E2
-------------------------------------------------------------
Sprungweiten-Tabelle
```

```
71E2 SBRTAB
71E2        9D      .BYT COL1-OPERAN+1,COL2-OPERAN+1,COL2-OPERAN+1,COL2-OPERAN+1
71E3        3F
71E4        3F
71E5        3F
71E6        3F      .BYT COL2-OPERAN+1,COL2-OPERAN+1,COL2-OPERAN+1,COL2-OPERAN+1
71E7        3F
71E8        3F
71E9        3F
71EA        3F      .BYT COL2-OPERAN+1,COL41-OPERAN+1,COL41-OPERAN+1
71EB        15
71EC        15
71ED        27      .BYT COL42-OPERAN+1,COL41-OPERAN+1,COL42-OPERAN+1
71EE        15
71EF        27
71F0        52      .BYT COL9-OPERAN+1,COLA1-OPERAN+1,COLA1-OPERAN+1
71F1        60
71F2        60
71F3        12      .BYT COLA2-OPERAN+1,COLA2-OPERAN+1,COLA2-OPERAN+1
71F4        12
71F5        12
71F6        12      .BYT COLA2-OPERAN+1,COLC1-OPERAN+1,COLC1-OPERAN+1
71F7        1E

71F8        1E
71F9        33      .BYT COLCE-OPERAN+1,COLC2-OPERAN+1,COLC1-OPERAN+1
71FA        70
71FB        1E
71FC        33      .BYT COLCE-OPERAN+1,COL7-OPERAN+1,COL7-OPERAN+1
71FD        A6
71FE        A6
71FF        86      .BYT COLF-OPERAN+1,COLF-OPERAN+1
7200        86
7201
7201                .END E DISMOS V1.0 $7
ERRORS=0000
```

Bücher

Anderson, J., J.: C-64 Akustik und Graphik. te-wi-Verlag, München 1985, 208 S., ISBN 3-921-803-31-4, DM 49,--. Das Buch beschreibt auf den ersten 72 Seiten für den Anfänger die Hantierung des beliebten Computers und die Möglichkeiten, mit dem normalen BASIC Graphik und Sound zu erzeugen, wofür bekanntlich ja keine besonderen Befehle vorhanden sind. Der Rest des Buches befaßt sich mit zwei BASIC-Erweiterungen für den C-64, nämlich das deutsche Structured BASIC und das Simons's BASIC, und hier besonders mit den für die genannten Effekte vorgesehenen Möglichkeiten und Befehlen. Diese beiden Erweiterungen werden Punkt für Punkt bei gleichen Aufgabestellungen mit ihren notwendigen Formulierungen und mit Programmbeispielen belegt.Der Betreiber und auch der Interessent erhalten damit einen klaren Überblick für den Einsatz des Computers für Effekte mit Graphik und Tonerzeugung. Zum Structured BASIC ist zu bemerken, daß es auch als Cartridge nebst 376 Seiten Dokumentation vom te-wi-Verlag angeboten wird, Preis DM 199,--, ISBN 3-921803-51-9. Selbige haben hier nicht vorgelegen.

Klaus Schünemann, 5000 Köln 60

Datenaustausch

zwischen Commodore und Apple II per IEC-Bus

Beruflich benutze ich CBM-Computer 3032, privat dagegen Apple II. Mein Wunsch war, Daten am Commodore eingeben zu lassen, um sie danach auf dem Aplle weiterverarbeiten zu können. Ein Programm, mit dessen Hilfe der Apple Tonbandcassetten vom CBM verstehen sollte (mc 3/83, S. 80), lief bei mir nicht sofort. Deshalb beschloß ich, einen Datenaustausch über den IEC-Bus zu realisieren, der auf Dauer weiter reichende Möglichkeiten zur Koppelung verschiedener Rechner bieten würde. Da CBM und Apple räumlich getrennt stehen, war die Verwendung zweier Knotenrechner erforderlich. Im vorletzten Absatz wird erläutert, wie die Kopplung auch ohne Knotenrechner funktioniert, falls die beiden Hauptrechner nebeneinander stehen. - Am Commodore entstand kein zusätzlicher Hardware-Bedarf.

Die Hardware der beiden Knotenrechner wurde aus mehreren vorhandenen Modul-Karten (kölner Hersteller Mertes und Eigenbau) auf einer Buskarte zusammengesteckt. Die Bauteile hätten aber bei Neuentwicklung durchaus auf jeweils einer Euro-Karte Platz gehabt. Je Knotenrechner wurden 16 KB statisches RAM als Zwischenspeicher verwendet (der RAM-Bedarf richtet sich nach dem Übertragungsvolumen). Jeweils zwei 6532 Port-Bausteine (RAM, I/O, Interval Timer Device) bedienten den IEC-Bus, eine Leuchtdiode sowie das Cassetten-Interface (Nachbau des KIM-Interfaces). Vervollständigt wurde der Aufbau durch je eine 6502-CPU sowie die Stromversorgungen. Die Übertragungsprogramme benötigten zusammen nicht mehr als 360 Bytes.

Die Tri-State-Ausgänge des Listeners arbeiteten ohne Pufferung gemeinsam mit zwei verschiedenen Druckern ziemlich problemlos am Bus des Commodore. Beim Lesen von der Floppy wurden allerdings einzelne Zeichen verschluckt. Zwischen dem Talker und dem Apple ergaben sich keine Bus-Probleme.

Am Apple genügte eine zusätzliche 6532-Karte der Fa. Neucom (mc 10/83, S. 110) plus IEC-Stecker. Das Programm paßte in die 128 Bytes RAM des 6532 RIOT. Die Pin-Belegung wurde aus mc 4/1982, S. 68 entnommen, mit Ausnahme der IFC-Leitung, die direkt auf eine allen Geräten gemeinsame Reset Leitung gelegt wurde. Das bewirkte einen heilsamen Zwang zu sauberer Initialisierung und stets gleichen Startbedingungen an allen Geräten.

Nachdem ein einfaches Übertragungsprogramm (alles in Maschinensprache) per Hand in einen der beiden Knotenrechner eingetippt worden war, wurde das eigentliche Programm auf dem Apple in Assembler entwickelt und in den jeweiligen Entwicklungsstufen über die IEC-Schnittstelle an den Knotenrechner übertagen. Erst nach dem Austesten aller Programme wurden die vorhandenen Eurokarten auf zwei verschiedene Rechner aufgeteilt. Die Kassetten-Routinen entstammen dem KIM-Monitor sowie dem 'First Book of KIM'. - Die Zwischenspeicherung kann je nach vorhandener Hardware auch anders als mit handelsüblicher Tonbandcassette und KIM-Monitor gelöst werden, z.B. mit zwei Diskettenlaufwerken.

Übertragen werden stets CBM-Programme, Daten also ggfs. als DATA-Statements von BASIC-Programmen. Dies ermöglichst einfache Dateneingabe, Korrekturlesen über Listen vom Drucker und einfache Datenänderung ohne Benutzung einer Datenbank. Selbstverständlich sind nur solche Programme auf dem Apple sofort lauffähig, deren Befehle in den beiden BASICs völlig identisch sind. Nach den Befehlen OPEN 10,10,1 und CR, dann CMD 10 und LIST (CBM-BASIC 3) gelangen die Programme bzw. Daten normal auf den IEC-Bus. Der erste Knotenrechner übernimmt die ASCII-Zeichen und speichert sie nach Erkennen des EOI-Signals auf einen handelsüblichen Cassettenrekorder. Damit der CBM das EOI-Signal abgibt, muß sofort nach der Rückkehr des Cursors nach CLOSE 10, CR und PRINT# nebst CR eingegeben werden. Die Übertragungsgeschwindigkeit war für meine bisherigen Anwendungen unerheblich. Deshalb habe ich sie nicht gemessen. Auf jeden Fall sind die eigentlichen Handshake-Routinen länger und damit langsamer als die Routinen im schon erwähnten Heft mc 4/82. Ich habe sie ohne Rücksicht auf Speicherbedarf genau nach IEC Norm geschrieben, d.h. niemals zwei Leitungen gemeinsam abgefragt oder gemeinsam

gesetzt. - Wenn alles gut läuft, funktionieren die kürzeren Routinen durchaus, ich verwende sie gelegentlich selbst. Insbesondere beim Austesten von Selbstbau-Platinen erscheinen mir aber die pedantischeren Routinen sicherer.

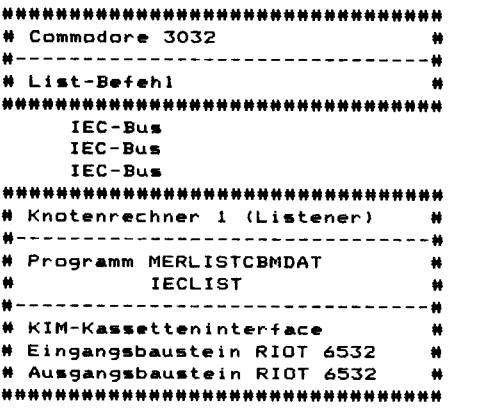
Beim Einlesen übernimmt der zweite Knoten-Rechner die Daten von einem Cassettenrekorder und bietet sie dem Apple an, der sie nach Verbiegen des Eingabevektors auf die IEC-Neucom-Karte (durch IN#4) problemlos als Input akzeptiert. Das Listener-Programm des ersten Knoten-Rechners wird bei hex 7D60 gestartet. Es geht nach einer abgeschlossenen Cassetten-Speicherung erneut in den Listener-Zustand. In der vorliegenden Version muß es mit Reset abgebrochen werden. Die Datenspeicherung erfolgt ab hex 0200. RAM muß aber ab Adresse 0000 vorhanden sein.

Das Talker-Programm des zweiten Knoten-Rechners startet bei 7C60. Nach Übergabe eines Programmes an den Apple geht es bei Adresse 7CB3 in die Tastaturabfrage zurück.Auch auf diesem Rechner muß RAM ab Adresse 0000 zur Verfügung stehen.

Wenn CBM und Apple nebeneinander stehen, dann können die beiden Knoten-Rechner auch zu einem einzigen Rechner zusammengefaßt werden. Praktisch ist das sinnlos, weil man unter diesen Umständen völlig auf Zwischenspeicherung verzichten würde. Nur der logischen Vollständigkeit halber wird es hier erwähnt. Ein direkter Sprung von 7DCA nach 7C68 unter Verzicht auf jede externe Speicherung ist aber nicht möglich. Wenn nämlich der Apple von Anfang an am IEC-Bus hört, bekommt er nicht nur das CBM-Listing mit, sondern auch noch das CLOSE10 und das PRINT#. Dieses trifft dann auf dem Bus mit dem aus dem Knoten-Rechner noch einmal kommenden Programm zusammen und zerstört die erste Programmzeile. - Wenn auf Knoten-Rechner völlig verzichtet wird, weil CBM und Apple räumlich nicht getrennt sind, wird der Busausgang des CBM direkt mit dem Apple verbunden und auch nur das Listener-Programm des Apple verwendet.. Diese Einlese-Programm des Apple ist für Slot 4 geschrieben, kann aber durch Änderung der Portadresse C4 für jeden anderen Slot außer Null und 7 umgeschrieben werden. Es wird mit IN#4 und CR gestartet und mit Reset abgebrochen. Es kann auch für Apple-Platinen angepaßt werden, die mit den Bausteinen 6520, 6522 oder 6821 an Stelle des 6532 bestückt sind. Der CBM hat invertierende IEC-Bus-Treiber. Diese Invertierung wird in C34F rückgängig gemacht. Wenn am Apple eine echte IEC-Karte verwendet wird, die ebenfalls inverviert, muß die Invertierung aus dem Apple-Programm herausgenommen werden.

Über die Rechner-Konfiguration und die benutzten Programmteile unterrichtet das nachfolgende Schema.

Literatur
IEC-Bus, Elektronik-Sonderheft Nr 47, Franzis-Verlag 1980
IEC-625, Philips, digital instrument course, Part 4, ohne Jahr
mc 4/82, Seite 68
mc 3/83, Seite 80
mc 10/83, Seite 110



manueller Kassetten-Transport

```
#####################################
# Knotenrechner 2 (Talker)          #
#-----------------------------------#
# Programm MERTALKAPDAT             #
#          IECTALK                  #
#-----------------------------------#
# KIM-Kassetteninterface            #
# Eingangsbaustein RIOT 6532        #
# Ausgangsbaustein RIOT 6532        #
#####################################
     IEC-Bus
     IEC-Bus
     IEC-Bus
#####################################
# APPLE II                          #
#-----------------------------------#
# Programm APPLEFROMMERTES          #
#-----------------------------------#
# Eingangsbaustein RIOT 6532        #
# NEUCOM-Karte in Slot 4            #
#####################################
```

```
                1    */MERLISTCBMDAT27
                2    *
                3              ORG  $7D60
                4    *
                5    * FUER MERTES-RECHNER
                6    * LISTEN CBM UND
                7    * SAVE AUF KIM-HYPERTAPE
                8    *
                9    IECLIST  EQU  $7F80
                10   DATBUFF  EQU  $7FEE
                11   STATBUFF EQU  $7FEF
                12   * RIOT 6532
                13   VEBIN    EQU  $E92C      ;VOLATILE EXECUTION BLOCK
                14   SAL      EQU  $E935      ;START-ADR LOW
                15   SAH      EQU  $E936
                16   EAL      EQU  $E937      ;END-ADR LOW
                17   EAH      EQU  $E938
                18   ID       EQU  $E939
                19   * RIOT 6532
                20   EBDRB    EQU  $EB82
                21   EBDDRB   EQU  $EB83
                22   *
                23   INCVEBIN EQU  $F9EA
                24   DELAY1   EQU  $FA6B
                25   DUMPT5   EQU  $FAFD
                26   *
7D60: A9 04     27   LEDON    LDA  #%00000100 ;LED AN
7D62: 8D 83 EB  28            STA  EBDDRB
7D65: AD 82 EB  29            LDA  EBDRB
7D68: 29 FB     30            AND  #%11111011
7D6A: 8D 82 EB  31            STA  EBDRB
7D6D: A9 8D     32            LDA  #$8D       ;OPCODE STA
7D6F: 8D 2C E9  33            STA  VEBIN+0
7D72: A9 00     34            LDA  #$0        ;SPEICHERUNG AB LOW
7D74: 8D 2D E9  35            STA  VEBIN+1
7D77: A9 02     36            LDA  #$02       ;SPEICHERUNG AB HIGH
7D79: 8D 2E E9  37            STA  VEBIN+2
7D7C: A9 60     38            LDA  #$60       ;OPCODE RETURN SUB
7D7E: 8D 2F E9  39            STA  VEBIN+3
7D81: A9 80     40            LDA  #%10000000  ;INITPORT
7D83: 8D 82 EB  41            STA  EBDRB
7D86: A9 E4     42            LDA  #%11100100
7D88: 8D 83 EB  43            STA  EBDDRB
```

```
7D8B: 20 80 7F 44   INBYTE   JSR  IECLIST
7D8E: AD EF 7F 45            LDA  STATBUFF
7D91: 4A       46            LSR
7D92: B0 F7    47            BCS  INBYTE
7D94: AD EE 7F 48            LDA  DATBUFF
7D97: C9 D5    49            CMP  #$D5        ;PRIM10=42 INVERTIERT
7D99: D0 F0    50            BNE  INBYTE
7D9B: 20 80 7F 51            JSR  IECLIST
7D9E: AD EE 7F 52            LDA  DATBUFF
7DA1: C9 9E    53            CMP  #$9E        ;SEC1=97 INVERTIERT
7DA3: F0 03    54            BEQ  LEDOFF
7DA5: 4C 8B 7D 55            JMP  INBYTE      ;REPEAT
7DA8: AD 83 EB 56   LEDOFF   LDA  EBDDRB      ;LED AUS
7DAB: 29 FB    57            AND  #%11111011
7DAD: 8D 83 EB 58            STA  EBDDRB
7DB0: 20 80 7F 59   GETBYTE  JSR  IECLIST
7DB3: AD EE 7F 60            LDA  DATBUFF
7DB6: 20 2C E9 61            JSR  VEBIN       ;STORE
7DB9: 20 EA F9 62            JSR  INCVEBIN
7DBC: AD 2E E9 63            LDA  VEBIN+2
7DBF: C9 7C    64            CMP  #$7C        ;HIMEM
7DC1: F0 07    65            BEQ  SAVE
7DC3: AD EF 7F 66            LDA  STATBUFF    ;EOI
7DC6: 29 08    67            AND  #%00001000
7DC8: D0 E6    68            BNE  GETBYTE
               69   *
               70   * SAVE (KIM-HYPERTAPE)
7DCA: A9 0F    71   SAVE      LDA  #$0F
7DCC: 8D 35 E9 72            STA  SAL
7DCF: A9 02    73            LDA  #$02
7DD1: 8D 36 E9 74            STA  SAH
7DD4: AD 2D E9 75            LDA  VEBIN+1
7DD7: 8D 37 E9 76            STA  EAL
7DDA: AD 2E E9 77            LDA  VEBIN+2
7DDD: 8D 38 E9 78            STA  EAH
7DE0: A9 01    79            LDA  #$01
7DE2: 8D 39 E9 80            STA  ID
7DE5: A9 0A    81            LDA  #$0A
7DE7: 20 6D FA 82            JSR  DELAY1+2    ;10 SEK
7DEA: 20 FD FA 83            JSR  DUMPT5
7DED: 20 6B FA 84            JSR  DELAY1      ;1 SEK
7DF0: 20 FD FA 85            JSR  DUMPT5
7DF3: 20 6B FA 86            JSR  DELAY1
7DF6: 20 FD FA 87            JSR  DUMPT5
7DF9: 20 6B FA 88            JSR  DELAY1
7DFC: 4C 60 7D 89            JMP  LEDON
7DFF: 00       90            HEX  00          ;TAPE

               1    */MERIECHANDSHAKE
               2    *
               3            ORG  $7F80
               4    *
               5    * FUER MERTES-RECHNER
               6    * IEC-HANDSHAKE
               7    * OHNE PORT-INITIALISIERUNG
               8    *
               9    *
               10   EBDRA    EQU  $EB80       ;6532
               11   EBDRB    EQU  $EB82
               12   *
7F80: AD 82 EB 13   IECLIST  LDA  EBDRB
7F83: 29 10    14            AND  #%00010000
7F85: D0 F9    15            BNE  IECLIST
7F87: AD 82 EB 16            LDA  EBDRB       NRFDLOW
7F8A: 29 7F    17            AND  #%01111111
7F8C: 09 20    18            ORA  #%00100000
7F8E: 8D 82 EB 19            STA  EBDRB
7F91: AD 80 EB 20            LDA  EBDRA       INBYTE
7F94: 8D EE 7F 21            STA  DATBUFF
7F97: AD 82 EB 22            LDA  EBDRB
7F9A: 8D EF 7F 23            STA  STATBUFF    STATUS
```

```
7F9D: AD 82 EB 24             LDA  EBDRB       NDACHIGH
7FA0: 09 40    25             ORA  #%01000000
7FA2: 8D 82 EB 26             STA  EBDRB
7FA5: AD 82 EB 27   DAVHI     LDA  EBDRB
7FA8: 29 10    28             AND  #%00010000
7FAA: F0 F9    29             BEQ  DAVHI
7FAC: AD 82 EB 30             LDA  EBDRB       NDACLOW
7FAF: 29 BF    31             AND  #%10111111
7FB1: 8D 82 EB 32             STA  EBDRB
7FB4: AD 82 EB 33             LDA  EBDRB       IECTALK
7FB7: 09 80    34             ORA  #%10000000
7FB9: 8D 82 EB 35             STA  EBDRB
7FBC: 60       36             RTS
               37   *
7FBD: AD 82 EB 38   IECTALK   LDA  EBDRB
7FC0: 0A       39             ASL
7FC1: 90 FA    40             BCC  IECTALK
7FC3: AD F0 7F 41             LDA  BUFFOUT
7FC6: 8D 80 EB 42             STA  EBDRA
7FC9: AD 82 EB 43             LDA  EBDRB       IECLISTW
7FCC: 29 EF    44             AND  #%11101111
7FCE: 8D 82 EB 45             STA  EBDRB
7FD1: AD 82 EB 46   NRFDLO    LDA  EBDRB
7FD4: 0A       47             ASL
7FD5: B0 FA    48             BCS  NRFDLO
7FD7: AD 82 EB 49   NDACHI    LDA  EBDRB
7FDA: 0A       50             ASL
7FDB: 0A       51             ASL
7FDC: 90 F9    52             BCC  NDACHI
7FDE: AD 82 EB 53             LDA  EBDRB       DAVHI
7FE1: 09 10    54             ORA  #%00010000
7FE3: 8D 82 EB 55             STA  EBDRB
7FE6: AD 82 EB 56   NDACLO    LDA  EBDRB
7FE9: 29 40    57             AND  #%01000000
7FEB: D0 F9    58             BNE  NDACLO
7FED: 60       59             RTS
               60   *
7FEE: 00       61   DATBUFF  HEX  00
7FEF: 00       62   STATBUFF HEX  00
7FF0: 00       63   BUFFOUT  HEX  00
               64   *

               1    */APPLEFROMMERTES21
               2    *
               3              ORG  $C400
               4              OBJ  $C400
               5    *
               6    * (C) 1984 COPYRIGHT KLAUS SCHUENEMANN
               7    * AN DER SCHANZ 2 WO.04/9
               8    * 5000 KOELN 60
               9    * TELEFON (0221)7601931
               10   *
               11   * PORT IST RIOT 6532
               12   DATBUFF  EQU  $C47E
               13   C4DRA    EQU  $C480
               14   C4DDRA   EQU  $C481
               15   C4DRB    EQU  $C482
               16   C4DDRB   EQU  $C483
               17   *
C400: AD 83 C4 18             LDA  C4DDRB
C403: C9 E4    19             CMP  #%11100100 ;SCHON INITIALISIERT?
C405: F0 0F    20             BEQ  DAVLO      ;JA
C407: A9 00    21             LDA  #$0        ;INITPORT
C409: 8D 81 C4 22             STA  C4DDRA
C40C: A9 80    23             LDA  #%10000000
C40E: 8D 82 C4 24             STA  C4DRB
C411: A9 E4    25             LDA  #%11100100
C413: 8D 83 C4 26             STA  C4DDRB
C416: AD 82 C4 27   DAVLO     LDA  C4DRB
C419: 29 10    28             AND  #%00010000
C41B: D0 F9    29             BNE  DAVLO
C41D: AD 82 C4 30             LDA  C4DRB       ;NRFDLOW
```

```
C420: 29 7F     31              AND  #%01111111
C422: 09 20     32              ORA  #%00100000
C424: 8D 82 C4  33              STA  C4DRB
C427: AD 80 C4  34              LDA  C4DRA       ;INBYTE
C4  : 8D 7E C4  35              STA  DATBUFF
C42D: AD 82 C4  36              LDA  C4DRB       ;NDACHIGH
C430: 09 40     37              ORA  #%01000000
C432: 8D 82 C4  38              STA  C4DRB
C435: AD 82 C4  39  DAVHI       LDA  C4DRB
C438: 29 10     40              AND  #%00010000
C43A: F0 F9     41              BEQ  DAVHI
C43C: AD 82 C4  42              LDA  C4DRB       NDACLOW
C43F: 29 BF     43              AND  #%10111111
C441: 8D 82 C4  44              STA  C4DRB
C444: AD 82 C4  45              LDA  C4DRB       ;NRFDHI
C447: 09 80     46              ORA  #%10000000
C449: 8D 82 C4  47              STA  C4DRB
C44C: AD 7E C4  48              LDA  DATBUFF
C44F: 49 FF     49              EOR  #%11111111  ;CBM HAT INVERT-TREIBER
C451: 09 80     50              ORA  #%10000000  ;MASK APPLE-FORMAT
C453: C9 8A     51              CMP  #$8A        ;LF
C455: F0 BF     52              BEQ  DAVLO       ;APPLE WILL NUR CR
C457: 60        53              RTS


                 1  */MERTALKAPDAT28
                 2  *
                 3  * LADE KIM-TAPE
                 4  * UND TALK ZUM APPLE
                 5  *
                 6  * (C) 1984 COPYRIGHT KLAUS SCHUENEMANN
                 7  * AN DER SCHANZ 2 WO.04/9
                 8  * 5000 KOELN 60
                 9  * TELEFON (0221)7601931
                10  *
                11          ORG  $7C60
                12  *
                13  IECTALK  EQU  $7FBD
                14  BUFFOUT  EQU  $7FF0
                15  * TAPE-ADRESSEN RIOT 6532
                16  VEBIN    EQU  $E92C      ;VOLATILE EXECUTION BLOCK
                17  ID       EQU  $E939      ;IDENTIFIKATION
                18  * IEC-ADRESSEN RIOT 6532
                19  VEBOUT   EQU  $EB2C      ;VOLATILE EXECUTION BLOCK
                20  EBDRA    EQU  $EB80
                21  EBDDRA   EQU  $EB81
                22  EBDRB    EQU  $EB82
                23  EBDDRB   EQU  $EB83
                24  *
                25  LOADT    EQU  $F873
                26  DISPL1   EQU  $FC50
7C60: A9 01     27              LDA  #$01        ;STANDARD-IDENTIFIKATION
7C62: 8D 39 E9  28              STA  ID
7C65: 20 73 F8  29              JSR  LOADT       TAPEIN
7C68: A9 AD     30              LDA  #$AD        ;OPCODE LDA
7C6A: 8D 2C EB  31              STA  VEBOUT+0
7C6D: A9 0F     32              LDA  #$0F        ;DATEN NACH HEADER
7C  : 8D 2D EB  33              STA  VEBOUT+1
7C72: A9 02     34              LDA  #$02
7C74: 8D 2E EB  35              STA  VEBOUT+2
7C77: A9 60     36              LDA  #$60        ;OPCODE RETURN SUB
7C79: 8D 2F EB  37              STA  VEBOUT+3
7C7C: A9 FF     38              LDA  #%11111111  ;INITPORT
7C7E: 8D 80 EB  39              STA  EBDRA
7C81: 8D 81 EB  40              STA  EBDDRA
7C84: 8D 82 EB  41              STA  EBDRB
7C87: A9 1F     42              LDA  #%00011111
7C89: 8D 83 EB  43              STA  EBDDRB
7C8C: 20 2C EB  44  LDAVEBO     JSR  VEBOUT
7C8F: 8D F0 7F  45              STA  BUFFOUT
7C92: 20 BD 7F  46              JSR  IECTALK     ;UEBERTRAGUNG AN APPLE
7C95: EE 2D EB  47              INC  VEBOUT+1    ;INCVEBOUT
```

```
7C98: D0 03     48            BNE  LASTHI
7C9A: EE 2E EB 49             INC  VEBOUT+2
7C9D: AD 2E EB 50   LASTHI    LDA  VEBOUT+2
7CA0: CD 2E E9 51             CMP  VEBIN+2
7CA3: F0 03     52            BEQ  LASTLO
7CA5: 4C 8C 7C 53             JMP  LDAVEBO
7CA8: AD 2D EB 54   LASTLO    LDA  VEBOUT+1
7CAB: CD 2D E9 55             CMP  VEBIN+1
7CAE: F0 03     56            BEQ  STOP
7CB0: 4C 8C 7C 57             JMP  LDAVEBO
7CB3: 4C 50 FC 58   STOP      JMP  DISPL1
7CB6: 00        59            HEX  00          ;TAPE
```

RLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRLRL

Editorial

Diese Zeitschrift erscheint seit Juni 1978, seit mehr als sieben Jahren also. Der Leser wird erinnern, daß wir seither in der Hard- und Software einen weiten Weg gegangen sind. Anfangs hatten wir den KIM-1 zu etwa DM 900,-. Er hatte nur gut 1KB RAM, eine hexadezimale Eingabe und eine nur sechsstellige Leuchtziffernanzeige. Heute erhalten wir in der gleichen oder doppeltem preislichen Größenordnung leistungsfähige Personal Computer der 8 Bit-Generation. Und ab DM 2800 werden in diesen Tagen die ersten Atari 520ST mit CPU 68000 und u.a. 512 KB RAM erwartet.

Wenn man aus der Vergangenheit heraus einmal blind solche Merkmale der Hardware auf die nächsten Jahre, z.B. auf etwa 1990 hochrechnet, dann wären in dieser Preisklasse dann Rechner mit einer sehr schnellen 32 Bit CPU, Megabyte-Speichern und großen externen Massenspeichern zu erwarten, von weiteren möglichen Attributen z.B. in der Grafik-Fähigkeit einmal abgesehen.

Ein Blick in die Zukunft ist immer Spekulation, aber so ganz wird man mit solchen Erwartungen nicht daneben liegen, denn es wird weltweit viel in die Halbleitertechnologie und in die Massenspeicherung investiert. In der Tendenz wird man sagen können, daß die nächste Computergeneration jeweils noch leistungsfähiger und relativ zur Leistung noch preiswerter sein wird.

Und hier liegt eigentlich auch schon ein Haken, weil die Softwareentwicklung da nicht Schritt hält. Ein Computer mit noch so schönen Merkmalen in der Hardware ist wenig wert, wenn für ihn nicht gute Systemsoftware, Sprachen, ggfs. Anwenderprogramme und vor allem auch Dokumentationen zur Verfügung stehen. Bei der Kurzlebigkeit von Computermodellen hat es schon Fälle gegeben, wo gute Software erst dann zur Verfügung stand, als das Modell schon nicht mehr gebaut wurde. - Die Industrie denkt nach wie vor zu sehr in Hardware und daran, daß diese möglichst schnell mit einem Betriebssystem zum Laufen und damit zum Verkauf gebracht wird, vielleicht gehören für den Anfang noch einige Demoprogramme dazu. Und Schnelligkeit heißt oft, daß man auf Bekanntes und Bewährtes zurückgreift, ohne die zeitgemäßen Möglichkeiten maschinen-spezifisch und fortschreitend zu benutzen. Betriebssysteme müssen also durchaus nicht 'up to date' sein. So stören den Herausgeber z.B. am CP/M-68K von DRI Beschränkungen in der Länge von Dateinamen (es fällt oft schwer, in acht Buchstaben eine Aussage unterzubringen) oder solche im Editor. Hier hatten wir schon mit den Full Screen Editor-Erweiterungen des AIM 65 bequemere Arbeitsmöglichkeiten. Auch bemerkt man, daß Dienstleistungsprogramme dieser Art viel Code beanspruchen, was bei den heutigen Speicherpreisen und -größen an sich kein Beinbruch ist. Er läßt aber vermuten, daß die Leistung zwar da ist, daß aber nicht geschickt programmiert wurde. - So wird es auch bei den jetzt aktuellen Maschinen vermutlich noch viel Arbeit in der Software geben, ehe man halbwegs zufrieden sein kann.

Oder man stelle einmal die Frage, warum das Microsoft-BASIC für 65xx noch immer nicht dezimal rechnen kann, obwohl die CPU es kann. Es war wohl bequemer, immer wieder in dieselbe Schublade zu greifen.

Rechner werden im harten Wettbewerb oft sehr frühzeitig angekündigt und von manchen Verlagen in Zeitschriften und Büchern schon hochgejubelt, ehe sie am Markt und in der Hand von kritischen Testern sind. Wenn man auf die vorstehenden Absätze schaut, dann ist sicher Zurückhaltung gegenüber den lauten Ankündigungen am Platz, denn nicht die Hardware allein macht einen

guten Rechner aus. - Als Käufer und Anwender sind wir in der Lage, durch unser Verhalten mehr Druck auf die Lieferanten hinsichtlich der Bereitstellung von Betriebssystemen, Sprachen und Dokumentation auszuüben. Im Sinne einer 'Entbündelung' von Leistungen muß das ja nicht gratis sein, Hauptsache ist ja doch, daß man auf eine vernünftige Art arbeitsfähig wird.

Sieben Jahre des Erscheinens von zunächst 65xx MICRO MAG und jetzt einfach MICRO MAG haben gezeigt, daß das Zusammenfließen von Information und Ausarbeitungen der Leser alle Betreiber gefördert hat. Berichten Sie bitte also auch weiterhin aus ihren Arbeitsgebieten. Und Sie sollten sich nicht scheuen, auch einmal brieflich oder telefonisch Wünsche zu äußern, was dargestellt werden sollte. Oft kann solchen Wünschen entsprochen oder ein nützlicher Hinweis gegeben werden. Oder Sie schauen einmal (nach Anmeldung) kurz herein, wenn sie im hamburger Raum sind.

Zum Schluß noch kurze Hinweise: Die Auslieferung des PC 128 von Commodore verzögert sich wohl noch bis September 1985. - Die neuen CPU-Typen G6SC02 und G65SC816 werden wohl erst am Ende des dritten Quartals regelmäßig lieferbar sein. Es ist sehr damit zu rechnen, daß zu diesem Zeitpunkt etwa leistungsfähige Computer auf Euro-Karten zur Verfügung stehen werden, die auch zu größeren Systemen ausgebaut werden können.

Bücher

Coffron, J. W.: Der Mikroprozessor 68000 - Anwendung und Fehlersuche. Vieweg-Verlag, Braunschweig 1985, 195 S., ISBN 3 528 04330-X, DM 49,50. Das Buch wurde aus dem Amerikanischen übersetzt.Es richtet sich 'an alle, die sich zum ersten Mal mit einem 16 Bit Prozessor beschäftigen'. Nach der Erläuterung der Hardware-Funktionen und eines vollständigen 68000-Systems wird eine Fehlersuchtechnik beschrieben, die ohne aufwendige Einrichtungen auskommt. - Schaut man nun in das Buch hinein, so findet man lange Passagen aus den bekannten Datenblättern, z.T. mit Text in eine lesbare Form gebracht. Aber dann wird man schon aufmerksam, daß zum Systemaufbau so antiquitierte Bausteine verwendet werden wie EPROM 2716 und RAM 2114. Die Tests beziehen sich daher im wesentlichen auf solche Bausteine. Zum Dekodieren werden 7442 eingesetzt. Der Betrieb dynamischer Bausteine ist nicht abgehandelt. Bei Interfaces findet man nur solche mit 6821. – Der Interessent sollte sich beim Durchblättern des Buches die Frage stellen, weiweit es ihn noch befördern kann.

Cohen, Ph.: APPLE II - Bewegte 3D-Grafik. te-wi-Verlag, München 1985, 200 S., ISBN 3-921-803-32-2, DM 49,--. Auch dieses Buch ist eine Übersetzung aus dem Englischen. Es beschreibt mit Text und Programmen die vielseiten grafischen Fähigkeiten des Gerätes in Kunst, Technik und Computerspiel. Enthalten sind ein Zeichenprogramm für Bildschirmentwürfe, Techniken der ebenen und räumlichen Bewegung von Computergrafiken, Ablegen und Ausdruck von Grafiken. Das Buch dürfte sich damit sehr gut als Grundlage für die Einarbeitung und für Versuche auf diesem Spezialgebiet eignen.

Aus der Branche

Atari 520ST: Einem Händlerrundschreiben von Anfang Juni ist zu entnehmen, daß der neue Computer nun für Händler als Entwicklungspaket lieferbar sein soll. Es umfaßt den Rechner, eine 3,5" Floppy, Monitor, Maus, fünf Disketten und Dokumentation. Die Software auf Disketten umfaßt das Betriebssystem TOS, 'C'-Compiler und Linker von DRI, als Tools die download/upload untility Kermit, Debugger, Assembler und schließlich einen Mince-Editor. Bei Redaktionsschluß war noch bei keinem Bekannten des Herausgebers ein Gerät eingetroffen. Wir kommen möglichst bald darauf zurück.

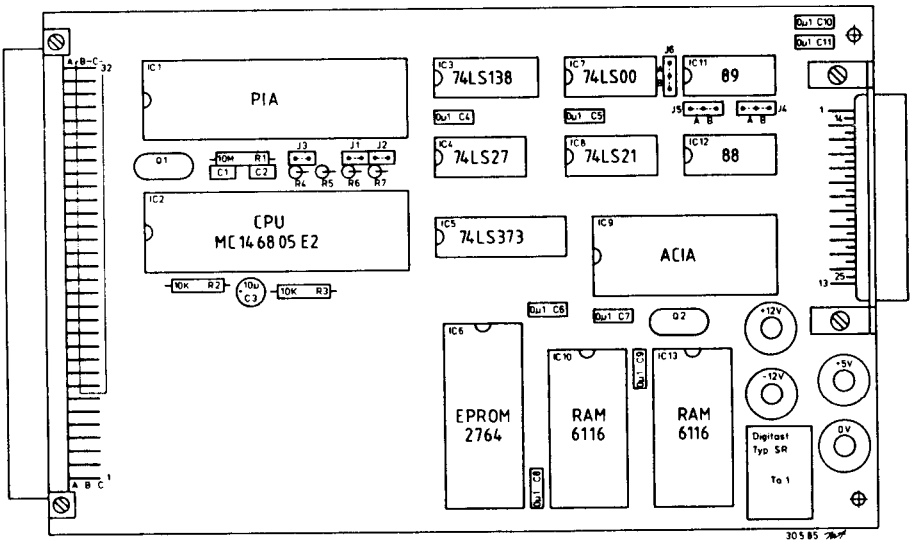
Intel-Schulungen. Das Schulungszentrum, Nymphenburger Str. 1/III in 8 München 2 legte jetzt einen Katalog für die in 2. Halbjahr angebotenen Schulungen vor, die nicht nur Microkomputer betreffen, sondern auch Betriebssysteme (XENIX/UNIX) und höhere Sprachen, wie z.B. 'C'.

Interfaces für Computer mit Applebus heißt ein 190-seitiger Katalog, der zugleich auch eine Dokumentation ist. Herausgeber ist die Fa. IBS Computertechnik, Olper Straße 10, 4800 Bielefeld 14.

Fachbücher, Fachzeitschriften 85/86 - Elektrotechnik, Elektronik ist ein Nachschlagewerk der Fachgemeinschaft Elektro, ige. Der neue Katalog erscheint im Oktober 1985. Gleichzeitig erscheint als neuer Titel der ige-Computerkatalog 'Informatik und Mikrocomputer-Fachbücher'. Im Katalog des Vorjahres waren über 1300 Buch- und Zeitschriftentitel enthalten, der Benutzer hat also einen sehr großen Überblick für seine Auswahl.

40 MHz Dotfrequenz mit dem ICB2675T von Valvo für monochrom- und Farbterminals. Mit dem ab Lager lieferbaren Attribut-Controller lassen sich Datensichtgeräte für moderne anspruchsvolle Forderungen auslegen, lt. Presse-Info 47/85.

Entwicklungsboard für Einchipper der Familie 6805 der Fa. MCT, Berlin. Das ggfs. auch als Single Board Computer einsetzbare System mit der Bezeichung 'fritz 05' dient vor allem der Programmentwicklung für die vielseitige Familie der Einchipper von Motorola und den Second Sources. Bekanntlich ist die Programmentwicklung für die Einchipper mit EPROM 'on chip' etwas umständlich, wenn nur einfache Hilfsmittel zu r Verfügung stehen. Auf einen Programmierzyklus folgt einer für das Brennen, einer für das Erproben und dann schließen sich gleiche Korrekturzyklen an. Mit der CMOS-Version der CPU und dem dazugehörigen Monitorprogramm MONI05 ist jedoch ein zügigeres Arbeiten und eine Emulation der Betriebszustände möglich. Das nachstehende Layout der Platine zeigt besser als Worte die Möglichkeiten: Der Verkehr mit dem programmierenden Host-Computer erfolgt über die ACIA und Schnittstelle V24. Von CPU und PIA stehen 32 Portleitungen zur Verfügung, daneben Timer, 4 KB RAM und EPROM, das in Blöcken von 4 KB dekodierbar ist. Alle Leitungen sind über eine 50-polige Pfostensteckverbindung oder einen optionalen VG64-Anschluß nach außen geführt. Eine reine CMOS-Bestückung ist möglich. Der Monitor MONI05 hat folgende Befehle: Display/Change Memory, Single Step, Load S-Records, Set/Display Breakpoints, Calculate Branch Offset, Trace, Execute Program, Dump Memory, Set/Display Accu or Index Register or Condition Code Register. Wir werden auf dieses System zurückkommen. Lieferant: MCT Mikrocomputertechnik Paul & Scherer, Rostocker Str. 31, 1000 Berlin 21. T. 030 - 392 30 11.



Fritz 05 Testboard
MCT Paul &Scherer

Assembler

Neu: Assembler für 65802/816, für R65C02 und für 6502

2 Pass Assembler mit einstellbaren Befehlssätzen für jede CPU zur Vermeidung von Irrtümern. Geschrieben in 6502-Code, damit auf vorhandenen Systemen einsetzbar. Generierbar für beliebige Adreßbereiche. Standardmäßig lieferbar für AIM 65 und kompatible (für SYSTEM 65 a. Anfr.). Für andere Fälle dokumentierte Sprungleiste auf externe Routinen des E/A-Systems. Die Syntax folgt dem Rockwell-Assembler. Symboltafel alphabet. geordnet mit Zeilenangabe, wo definiert. Formatierte Assembler-Liste mit 80 Zch/Z, wie in Heft 41 dokumentiert. Kommandofiles möglich, die Quelltexte von verschiedenen Eingabemedien verbinden. Damit zügige Durchläufe mit Library-Funktion. Assemblerung mit Offset in beliebige Bereiche als Vorlage für EPROMs. Für AIM 65, inkl. Dokumentation, Preis inkl. MWSt DM 300,--

Cross-Assembler für 6800/6802/6803/6303

für die CPU-Typen von Motorola und Hitachi, für den AIM 65 und kompatible Systeme. Erzeugt aus den Mnemonics von Motorola Maschinencode. Zwei-Pass-Assembler mit gewohnten Komfort, Syntax jedoch nach 6502, 2 EPROMs mit Dokumentation DM 300,--

6805/68705 Cross-Assembler

für AIM 65 und kompatible Systeme. 2-Pass-Assembler mit allem gewohnten Komfort und 6502-Syntax. Erzeugt aus den Mnemonics von Motorola 6805-Code. 2 EPROMs wie vor DM 280,--

Mathe-ROM für 6502

Implementierung nach Peter Rix (s. Hefte 28/29). Fließkommaarithmetik und höhere mathematische Funktionen wie in Microsoft-BASIC für AIM-65-FORTH und für jedes 6502-Assemblerprogramm (20 S. Dokumentation mit Einsprungspunkten und Argumenten), für Sockel $D000 DM 124,30

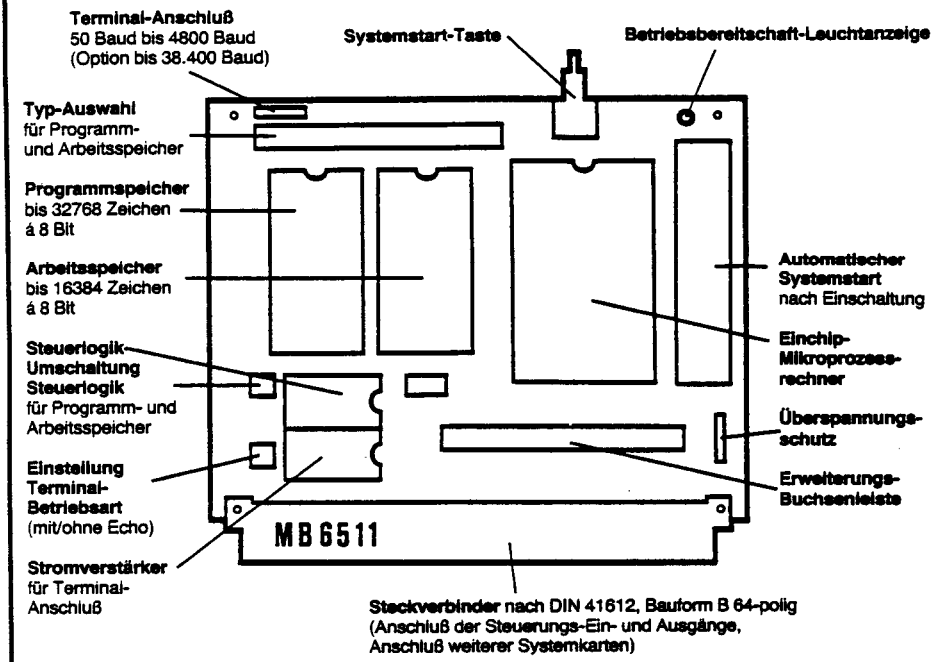
Video-Terminal mit Cherry-Tastatur (im Gehäuse), MC-Video-Platine (Regge) und ggfs. Sanyo-Monitor DM 5912CX (grün, 18 MHz Bandbreite) funktionierend abzugeben. Betrieb mit Schnittstellen RS232C, TTY, einstellbare Baudraten und Übertragungsarten. R. Löhr, Tel. 04102-55 816.

VMEbus-Karte FORCE SYS68k, CPU1 mit MC68000 (8 MHz), 128 KB RAM (on board erweiterbar bis 512 KB), 4 EPROM-Steckplätze (bis 64 KB), 3 RS232C-Kanäle mit einstellbaren Parametern, Interfacebaustein PI/T MC68230 (Parallel Interface/Timer), mit Betriebsystem/Monitor/Debugger/line by line Assembler wowie FORCE IDEAL Screen Editor und 2 Pass-Assembler, mit Dokumentation, Neuwert ca. DM 4200 günstig abzugeben: R. Löhr, Tel. 04102 - 55 816.

Roland Löhr, Hansdorfer Str. 4, D-2070 Ahrensburg
Tel.: 04 102 - 55 816

8408

MINIBEE-MB6511

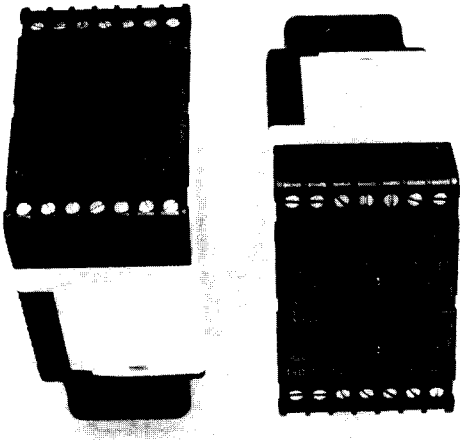


TECHNISCHE DATEN

Leiterplattengröße	100mm x 80mm
Gesamtmaße	100mm x 88mm x 14mm
Kartenmaterial	Glasfaser-Epoxy 1,5mm, beidseitig Lötstoplack
Gewicht	80g
Leistungsbedarf	2W
Spannungsversorgung	5V ± 5%
Systemtakt	2MHz (Option 1MHz)
Arbeitstemperaturbereich	0°C bis +70°C
Programmspeicher	2kByte bis 32kByte BYTEWIDE
Arbeitsspeicher	2kByte bis 16kByte BYTEWIDE
extern erweiterbar bis	4MByte (256 x 16kByte)
Terminal-Anschluß	TTL-kompatibel

RECHNER DATEN

Einchip-Mikroprozessor	R6511Q (Rockwell/NCR)
Rechnerstruktur	6500
Programmierung	softwarekompatibel zu 6500 60 Befehle, 14 Adressierungsarten
Ein-/Ausgabe-Anschlüsse	24 O.C. mit Pullup, 8 Tristate, alle TTL kompatibel
Zähler/Zeitmesser/Takt	1 voll programmierbar, retriggerbar 1 für serielle Schnittstelle
Serielle Schnittstelle	voll duplex, asynchron, synchron
Programm-Unterbrechung (IRQ)	10 (je 2 positiv, negativ, Zähler, Serielle Schnittstelle; 1 unbedingte Unterbrechung (NMI)



Neu: V.24/V.28 & RS 232 C <——> 20 mA Current Loop 2 Kanalkonverter

in einem Gehäuse mit 11-poligem Stecksockel für Schaltschrankmontage

Der 2 Kanalkonverter, DBGM 8432 684.0, ist eine Weiterentwicklung des bewährten Steckerkonverters, der im Gehäuse des Subminiatur "D" Steckers der V.24-schnittstelle eingebaut wird und dort als 20mA Current Loop Interface fungiert.

Technische Daten:

- 19200 bit/sec bis 2000 Meter
- 2400 bit/sec bei 5700 Meter
- 100% galvanische Trennung
- ein Netzteil für jeden Kanal
- einfache Montage auf Hutschiene
- LED-Kontrolle der Übertragungsstrecke
- 100% kompatibel mit anderen 20mA Systemen
- Betriebsart aktiv oder passiv für jede Stromschleife

Die Datenübertragung mit 20mA Stromschleifen gehört zur digitalen Basisübertragung. Das Stromschleifenprinzip, ein dem Wellenwiderstand angepaßter Leitungsabschluß und ein überduchschnittlich hoher Triggerpegel, logisch 1 => 13 mA und logisch 0 <= 11 mA, garantieren große Reichweiten mit hoher Übertragungsrate und größter Störsicherheit.

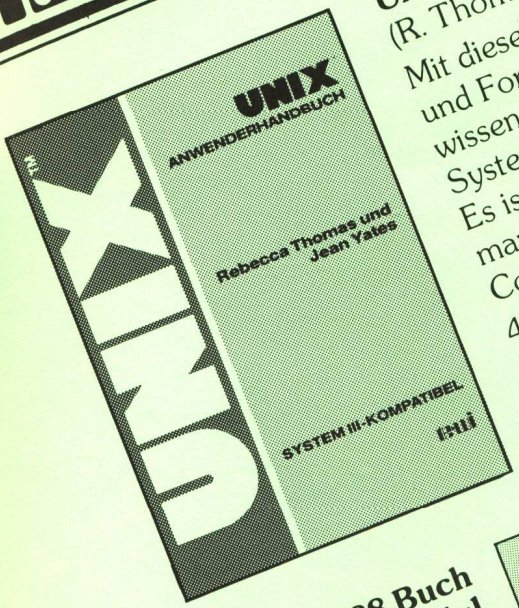
Der Einsatz von einem Netzteil je Konverter ermöglicht sowohl die Potentialtrennung zwischen der Übertragungsstrecke und den angeschlossenen Datengeräten, als auch zwischen den Stromschleifen im selben Kabel. Letztere vermindert die Betriebskapazität der Datenleitung und sorgt somit für eine höhere Datenübertragungsrate, die sich mit 19200 bit/sec bei einer Leitungslänge von 2000 Metern dokumentiert.

Getestet werden kann die Datenübertragungsstrecke dank der in den Stromschleifen liegenden Leuchtdioden auch ohne die angeschlossenen Datengeräte, da es sich um ein Ruhestromsystem handelt, d.h. im Ruhezustand wird die logische 1 übertragen. - Bei der Errichtung von Punkt zu Punkt Datennetzen mittels 20 mA Current Loop Systemen hat der Anwender einen beachtlichen Kostenvorteil. Ggfs. können freie Adern eines Telefonkabels benutzt werden. Bereits vorhandene EDV-Geräte werden netzwerkfähig, womit der Ankauf neuer Geräte mit teueren Netzwerkinterfaces entfällt.

INGENIEURBÜRO STECKER

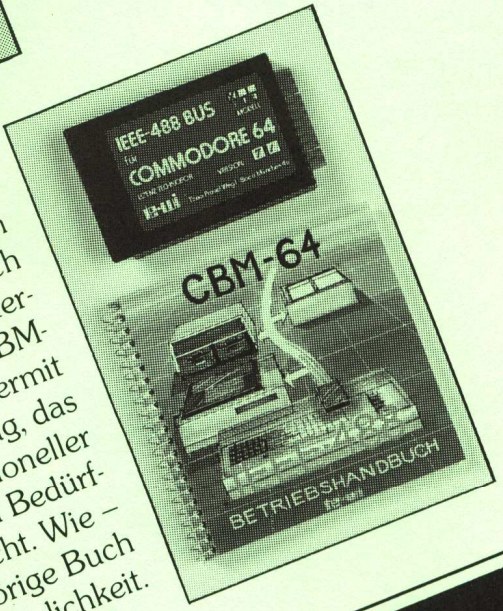
5000 Köln 60 (Niehl)
Postfach 60 07 66
Delmenhorster Str. 20
Tel. (02 21) 7 12 40 18

te-wi aktuell...



UNIX™ – Anwenderhandbuch

(R. Thomas, J. Yates)
Mit diesem Buch können sich Interessierte und Fortgeschrittene ein fundiertes Basiswissen über das UNIX-Betriebssystem, einem System, dem die Zukunft gehört, aneignen. Es ist so praxisnah geschrieben, daß man schon in kurzer Zeit mit UNIX am Computer arbeiten kann.
478 Seiten, Softcover, DM 79,–



C-64 /IEEE-488 Buch und Steckmodul

Hardware im te-wi Verlag! Mit diesem Steckmodul schaffen Sie sich Mehrfachnutzung durch nur ein Interface, das speziell den C-64 an die CBM-Großperipherie führt. Hiermit haben Sie zugleich ein Werkzeug, das z.B. sämtliche Elemente professioneller Meß- und Regelsysteme Ihren Bedürfnissen zugänglicher macht. Wie – das sagt Ihnen das dazugehörige Buch in aller Ausführlichkeit.
40 Seiten plus Modul, DM 239,–

te-wi

te-wi Verlag GmbH
technisch wissenschaftliche Elektronik-Literatur
Theo-Prosel-Weg 1 8000 München 40

MICRO MAG

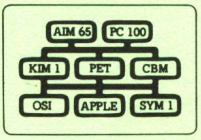
HERAUSGEBER/EDITOR:
DIPL.-VOLKSWIRT ROLAND LÖHR
HANSDORFER STRASSE 4
D-2070 AHRENSBURG
☎ (04102) 5 58 16

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- im Inland, bzw. DM 59,-- im Ausland (surface mail). Luftpostzustellung auf Anfrage. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu 4 Wochen vor deren Ablauf. - Nachliefermöglichkeiten siehe unten.

Private Besteller werden um Überweisung oder Scheck (auch Auslandsschecks) zusammen mit Bestellungen gebeten. Konto Roland Löhr, Nr. 654 70-202 Postgiroamt Hamburg, BLZ 200 100 20.

Leser-Service des Herausgebers

Das Buch 7-13 des 65.. MICRO MAG



340 Seiten, DM 42,--

Nachliefermöglichkeiten:

Vergriffen sind 'Das Buch 1-6 des 65xx MICRO MAG' sowie die Hefte 1-13 und 16 der Zeitschrift. Es erfolgt keine Neuauflage.

Lieferbar: 'Das Buch 7-13 des 65xx MICRO MAG' zu DM 42,-- sowie die Hefte 14, 15 und 17-42 zu DM 7,80/St. (ab 10 St. in einer Sendung: DM 6,--/St.):

Mathe-ROM nach P. Rix für FORTH und Einbindung in Assembler-Programme, mit Dokumentation DM 124,30.

Assembler für 6502/65C02/65802/65816 (drei Befehlssätze!) sowie für MC6805 und für 6800/6802/6303 (drei Befehlssätze): Siehe im Heftinneren.

Vorstehende Preise inkl. MWSt, zuzüglich DM 2,50/Sendung + ggfs. NN DM 2,--