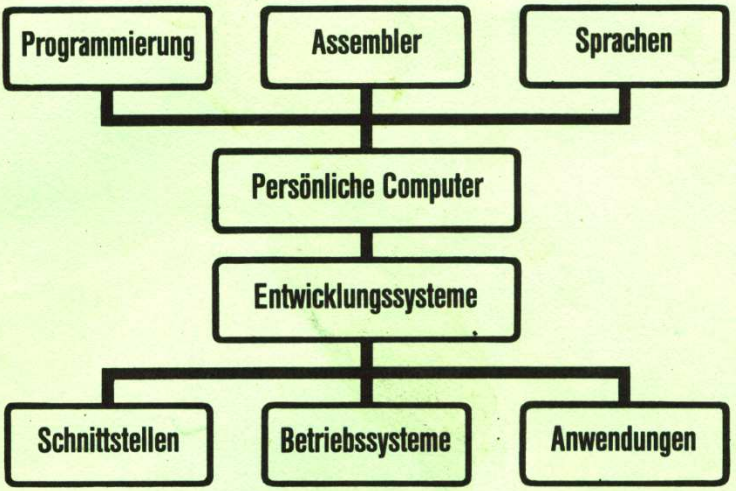
MICRO MAG

DM 9,50 Nr.44 August 1985



Inhaltsverzeichnis

Singl´n Sie mit!

Das Entwicklungsboard "FRITZ 05" und der Cross Assembler "AS 05" von MCT ermöglichen Ihnen einen preiswerten Einstieg in die vielfältige 6805 Single Chip Familie:

- komplettes Entwicklungsboard in aktueller CMOS-Bestückung
- komfortables Arbeiten am Host Computer
- hochleistungsfähige, ausgereifte Software
- kein lästiges Brennen von EPROM's
- Test der Software direkt im RAM
- Emulation von Betriebszuständen
- Cross Assembler für verschiedene Rechner
- leistungsfähige Debugging-Hilfsmittel durch "MONI 05"
- Anschluß problembezogener Hardware über 50-polige Steckverbindung
- bis zu 4K ROM durch 4K RAM on board emulierbar
- Stromaufnahme 6 mA
- Bedienung über serielle Schnittstelle
- 4 parallele Ports und Timereingang
- IRQ-Eingang
- einzig auf dem Markt

Preise:

FRITZ 05 (inkl. Handbuch, MONI 05, CMOS-Bestückung)	624,-- DM
AS 05 für CP/M 68 k	345,-- DM
AS 05 für PC/DOS bzw. MS/DOS	435,-- DM
EMU 05 Software Emulator/Debugger für CP/M 68 k	285,-- DM

68000 Single Board Computer

hervorragende Eignung als Software Entwicklungsrechner

- Betriebssystem CP/M 68 k
- C-Compiler
- Pascal MT+ Compiler lieferbar
- Basic Compiler lieferbar
- 68000 Assembler
- 6800, 6804, 6805 Assembler lieferbar
- 832K RAM on board
- zwei serielle Schnittstellen
- Druckerschnittstelle
- 20 bit parallel I/O
- Akku-gepufferte Real Time Clock
- Floppy Controller
- komplettes Rechnersystem
  wahlweise 3,5 oder 5,25" Laufwerke
  Single Board Computer
  CP/M 68 k
- Laufwerkskapazität 800K formatiert
- leistungsfähige 704K RAM-Disk kostenlos

Preise:

Betriebssystem CP/M 68 k	1.100,-- DM
SBC 68 k (832K Byte RAM)	2.250,-- DM
3,5" EPSON SMD 180 Laufwerk inkl. Kabel	545,-- DM

(Alle Preise zzgl. MWSt)

Mikrocomputertechnik

Auftragsentwicklung
32/16 bit-Systeme
Single-Chips
Hard- und Software

Paul & Scherer
Rostockerstr. 31
1000 Berlin 21

Tel. 030 - 392 30 11

Roland Löhr

Aufbau eines Assemblers

1. Einleitung

Im Verlauf der Jahre hat der Autor eine Reihe von Assemblern und Cross-Assemblern für verschiedene CPU-Typen entwickelt, formuliert in der Assemblersprache seines Entwicklungssystems oder auch in der Sprache FORTH. Damit ist einerseits das Prinzip der Touring-Maschine bestätigt, daß man grundsätzlich jeden Prozessor in einem anderen abbilden kann und daß andererseits auch jede Computersprache mit den Hilfsmitteln einer anderen nachgebildet werden kann. Natürlich kam es nicht auf diese Bestätigung an, sondern auf die Schaffung von Hilfsmitteln. Nachdem hier nun in den letzten Monaten ein Assembler von Grund auf neu bedacht und hochgezogen wurde, kann weiter unten eine Beschreibung des grundsätzlichen Aufbaues solcher Hilfsmittel und der Funktionsblöcke für ähnliche Vorhaben gegeben werden.

Die Implementierung irgendeiner Sprache auf einen Computer ist lehrreich hinsichtlich der Überlegungen, wie man als ASCII-Texte vorliegende Formulierungen in den Code einer Maschine und damit in ihre späteren Tätigkeiten umsetzt. Arbeiten dieser Art dürfen vor allem Informatikern und Programmierern empfohlen werden, die sich hinsichtlich einer freundlichen 'Benutzeroberfläche' um die Umsetzung sprachlich vom Anwender geäußerter Wünsche in Aktivitäten bemühen.

2. Zweck eines Assemblers

Ein Assembler soll einen in ASCII vorliegenden Quelltext in den einzig von der Maschine verstehbaren Binärcode umsetzen und ihn zur späteren Ausführung im Speicher selbst erzeugen oder ihn zum späteren Laden zunächst extern ablegen.

Die Erzeugung von solchem Objectcode aus dem Quelltext oder 'Source' soll nach den Wünschen des Anwenders dirigiert und mit verschiedenen Optionen versehen werden können, zu deren wichtigsten die Erzeugung einer formatierten Assembler-Liste gehört. Sie wird oft mit der Anweisung .OPT LIS befohlen. Sie dient nicht nur der notwendigen Dokumentation, sondern vor allem auch der Kontrolle. Neben dem erzeugten Code sind in ihr daher die Fehlerhinweise wichtig. Ein weiteres bedeutendes Hilfsmittel ist auch die Symboltafel.

3. Wünsche an einen Assembler

Viele vorhandene Assembler bauen auf Vorbilder auf, die aus einer Zeit beschränkter Hilfsmittel und engen Speicherplatzes stammen. So mag bei den Bezeichnern (Symbole und Label) eine Beschränkung auf wenige Zeichen gegeben sein. Im Sinne klaren sprachlichen Ausdruckes, sollte in ihrem Textstring auch ein Unterstreichungszeichen (underline character) zugelassen werden, der Haupt- und Nebenbegriffe für den Leser rein optisch trennt. Bezeichner sollten mehr als 5-6 Zeichen haben dürfen. Allerdings hat das Folgen für die Anordnung der Spalten in der zu erzeugenden Assemblerliste, die in den Zeilen ja auch noch Platz für Kommentare lassen sollte, damit das Listbild nicht zu sehr zerrissen wird.

Aus Platzgründen geben viele Assembler bei Fehlermeldungen nur eine Nummer aus. Wünschenswert ist eine Antwort in Klartext. Es ist gar nicht so schwer, eine solche Nummer in einen Pointer umzurechnen, der auf einen entsprechenden Textbeginn hinweist.

Mit dem Stichwort 'Text' ist darauf zu verweisen, daß ein Assembler in der Führung des Benutzers möglichst genaue Anleitungen zur weiteren Bedienung gibt, und das hierzulande möglichst in deutscher Sprache. - Weitere Wünsche mögen die bedingte Assemblierung und Makros betreffen. Zu letzteren ist zu sagen, daß sie eine Dokumentation erfordern und daß ihre Möglichkeiten nicht so häufig ausgenutzt werden. Anders ist die Implementierung hochsprachlicher Strukturierungsanweisungen zu sehen, wie BEGIN / UNTIL oder IF / THEN / ELSE usw.. Sie bilden Blöcke und bringen eine Entlastung von unnötigen Labeln.

4. Sprache und Syntax

Wer einen Assembler schreibt, darf sich im Prinzip frei fühlen, eine Programmiersprache nebst

Syntax nach dem eigenen Geschmack zu bilden, denn er schafft ja gerade die Mittel, um zugelassene sprachliche Elemente in Maschinencode zu übersetzen. Gleichwohl gibt es eine Warnung vor zuviel Streben nach Individualität. Ein Assembler, dessen Sprache auch von anderen Betreibern verstanden werden soll, muß auf das Rücksicht nehmen, was geläufig ist. Dazu gehören die in den Datenblättern dokumentierten mnemonischen Abkürzungen der Befehle und die Schreibweisen von Adressierungsarten. An diese sollte man sich halten. Das gilt auch für die Assembler-Direktiven.

Die Anlehnung an solche branchenüblichen Schreibweisen hat auch für den Schreiber eines Assemblers selbst den Vorteil, daß anderswo veröffentlichte Programmvorschläge zum Nachvollzug nicht erst mühsam gedanklich transponiert werden müssen.

5. Assembler sind zunächst Cross-Assembler. - Die Hilfsmittel

Es ist noch gar nicht lange her, daß eine Maschine mit Kippschaltern programmiert werden mußte oder, wie der KIM-1, mit der Eingabe handprogrammierter Halbbytes. Diese Zeiten sind überwunden. Gleichwohl bleibt die Aufgabe, Maschinencode für ein Zielsystem zu erzeugen, der auf dem Entwicklungssystem nicht zur Ausführung gebracht werden kann. Man denke dabei an folgende Umgebung: Es steht ein Assembler auf einer 6502-Maschine zur Verfügung. Es soll Code erzeugt werden, der auf dem 65816 mit dessen Möglichkeiten zur Ausführung gelangen soll. Vergleichsweise: Ein Assembler auf 68000 kann nur dessen Befehle zur Ausführung bringen. Es soll aber Code für 68020 erzeugt werden. Assembler dieser Art codieren zunächst für die gemeinsame Menge von Befehlen. Sie sind insoweit Cross-Assembler für eine Nachfolgemaschine. Erst wenn der nachfolgende Prozessor zum Laufen gebracht worden ist, kann man ihm einen noch leistungsfähigeren Assembler mit den Möglichkeiten seiner Befehle schreiben. - Damit ist die Aussage gmeint, daß man mit einem vorhandenen Assembler eine verbesserte Version 'hochzieht'.

6. Sprachliche Elemente

Der Eingabestrom eines Assemblers besteht aus (in ASCII formulierten) Texten der symbolischen Assemblersprache. Deren Hauptelemente sind:

Symbole
Mnemonische Befehle
Operanden
Adressierungsarten für die Operanden
Assembler-Direktiven
Kommentare
Trennzeichen

Zunächst noch eine Klarstellung zu den sprachlichen Elementen:

6.1 Symbole

Symbol ist der Oberbegriff für explizite Erklärungen vom Typ z.B. Carriage Return=$0D (=Symbolerklärung) und für Stellen im Programmablauf, die mit einem Namen als symbolische Adresse festgehalten werden. Solche Stellen bezeichnet man englisch als 'Label'. Beide Arten von Symbolen werden in einer Symboltafel verwaltet, in der die Namen und die ihnen zugewiesenen Werte geführt werden, ggfs. auch die Zeilennummer, in der die Zuweisung erfolgte.

6.2 Mnemonische Befehle

Ein großer Teil des Eingabestromes soll zu Maschinenbefehlen umgesetzt werden. Sie werden in mnemonischer Schreibweise aufgegeben, z.B. als LDA, STA, ORA usw. Es wird zunächst der Grund-Opcode der Befehle aufgeschlagen. Er wird im weiteren Verlauf oft durch die Art des Operanden und seiner Adressierung modifiziert.

6.3 Operanden

Operanden sind Objekte an denen ein (mnemonisch eingegebener) Befehl ausgeführt wird. Normalerweise sind es Inhalte im Datenspeicher, die mit einer als Operand angegebener (auch symbolischer) Adresse nebst Adressierungsart bezeichnet werden. Es können jedoch auch Operanden

sein, deren Wert im Programmspeicher steht, und zwar unmittelbar nach dem Opcode des Befehles. Solche Operanden bezeichnet man als 'immediate' oder 'unmittelbar'.

6.4 Adressierungsarten

Viele Befehle sind 'implied', d.h. deutsch 'mit inbegriffenem Operanden'. Es sind dies vor allem Austauschbefehle innerhalb der Zentraleinheit (CPU) oder zum Stack hin. Sie brauchen keine weiteren Bezeichnungen zu den Operanden, es wird elektrisch entsprechend durchgeschaltet.

Die meisten Befehle sprechen Operanden im Speicher an. Diese mögen per Symbol oder irgendwie interpretierbarer Adreßangabe endgültig bezeichnet sein. In 6502-Sprache heißt das 'absolute' oder 'zero page', bei Motorola 'extended' oder 'direct page'. Es gibt daneben je nach Befehlssatz weitere Adressierungsarten. Bei Verzweigungen haben wir z.B. Sprünge im Programmablauf relativ zum Programmzählerstand, die über gewisse Weiten nicht hinausgehen dürfen.

Die vom Schreiber eines Assemblers bestimmten ziemlich zwingenden Schreibweisen der Sprache definieren andere Adressierungsarten, z.B. zur Nach-Indizierung, zur Beschaffung einer Basisadresse mit möglicher Nachindizierung usw. Alle diese Besonderheiten erfordern umfangreiche Nachprüfungen.

6.5 Assembler-Direktiven

Bei der Auswertung von Zeilen mit Befehlen legt der Assembler normalerweise Code ab. Auch Pseudo-Befehle legen Code ab. lÜbliche Bezeichnungen dafür sind z.B. .BYT, .WOR oder bei Motorola FCB, FCW, FCC usw. Mit Ihnen werden Tabellen, Texte und sonstige Operanden erzeugt und abgelegt, als wenn sie Befehle wären.

6.6 Steuerung des Assemblers

Zu den Assembleranweisungen gehören auch solche, mit denen der Ablauf einer Assemblierung gesteuert wird. Das gilt nicht nur für das Erzeugen einer Liste, sondern vor allem auch hinsichtlich der Adressen, für die Code erzeugt werden soll. Hier ist besonders die Sternanweisung (*=... bzw. ORG) zu erwähnen, die den Zuordnungszähler des Assemblers auf einen definierten Wert setzt. Dieser virtuelle Programmzähler wird entsprechend der Befehlslänge hochgezählt. Sein augenblicklicher Stand wird bei der Definition eines Labels als dessen Wert eingesetzt. - Es gibt zahlreiche weitere Steueranweisungen. Wichtig ist meistens .FIL. Mit dieser Anweisung können Quelltext-Dateien miteinander verbunden werden.

6.7 Kommentare

Zu den Elementen einer Computersprache gehören Sonderzeichen, die dem Assembler den Beginn von Kommentaren kenntlich machen. Kommentare sind selbst keine Bestandteile einer Sprache, sie dienen vor allem der Dokumentation.

6.8 Trennzeichen

Der Assembler muß bei der Verarbeitung des Eingabestromes die sprachlichen Elemente unterscheiden und zuordnen können. Dazu gehören nicht nur Vorschriften für die zulässige Abfolge sprachlicher Elemente wie z.B. die Abfolge: Label, mnemonischer Befehl, Operand, Adressierungsart und Kommentar. Die einzelnen Elemente dürfen auch nicht einfach zusammenfließen. Trennzeichen zwischen ihnen sind normalerweise der Zwischenraum (Space) und das Zeichen für Zeilenende. Es sind aber auch andere Trenner denkbar, wie Semikolon, Klammern oder Akkoladen in Hochsprachen. Damit wird auch klar, daß es eine Hauptroutine zur Trennung der sprachlichen Elemente und für ihre Zuordnung geben muß. Man nennt sie Parser. Dieses englische Wort steht für das grammatikalische Zerlegen. Wir kommen darauf noch zurück.

7. Die Funktionsblöcke eines Assemblers

Der vorstehende Abschnitt 6 hat gezeigt, daß ein Assemblerprogramm auf die benutzten Sprachelemente und ihre Syntax zugeschnitten werden muß. Anders gesagt: Einen Assembler kann man erst dann schreiben, wenn man sich hinsichtlich des zugelassenen sprachlichen Ausdruckes schon ziemlich festgelegt hat. Natürlich sind noch Anpassungen im Verlaufe der Arbeit möglich. Unabhängig davon kann man folgende wichtigen Hauptblöcke eines Assemblers unterscheiden:

Initialisierung
Einlesen von Zeilen
Parsen/Erkennen
Bewerten
Code erzeugen
Verwaltungsinformation fortführen
Ausgabe von Listen
Tabellen

Zu diesen Blöcken nun im einzelnen:

7.1 Initialisierung

Es treten typisch folgende Arbeiten auf:

Lage der Symboltafel festlegen
Bestimmung des Quelltextes (Name des Files, Medium, von dem er kommt)
Initialisierung des Files
Verfügung über die Code-Ausgabe (File, Medium)
Verfügungen über die Einheit, die listen soll, ggfs. nur Fehler

7.2 Einlesen von Zeilen

Wie für die Initialisierung von E/A-Einheiten so gilt auch hier, daß Treibersoftware entweder auf dem System vorhanden ist oder aber geschrieben wird. Für die Zwecke des Parsens ist es am bequemsten, wenn die Zeilen des Quelltextes nacheinander in einen Eingabepuffer übertragen werden. Quelltext wird am schnellsten verarbeitet, wenn er im Speicher des Computers resident ist. Wenn man jedoch bedenkt, daß auch eine Symboltafel geführt werden muß und daß erzeugter Code möglicherweise ebenfalls in den Speicher abgelegt werden soll, dann wird klar, daß der Speicher bei größeren Programmen schnell zu eng wird. Man wird daher vorsehen, daß Quelltext zeilenweise von externen Medien eingelesen werden kann. Man wird sich natürlich vorwiegend für Floppy Disk entscheiden. Wesentlich ist, daß man das Ende des Quelltextes erkennen kann, um einen Durchgang (Pass) zu beenden. Für den zweiten Pass ist wieder auf die oberste Zeile zu setzen. Bei externen Medien bedeutet das Re-Initialisierung mit dem File-Namen, der also gespeichert sein muß.

Es tritt folgende Besonderheit hinzu: Wenn verschiedene Quelltext-Files mit der Assembleranweisung .FIL verbunden werden, so ist deren jeweiliger Name für die Initialisierung auszugeben. Der Name des allerersten Files darf jedoch nicht verloren gehen, weil er am Anfang des 2. Durchganges wieder gebraucht wird.

Bei der möglichen Einrichtung eines Kommandofiles, das nacheinander Quelltext von verschiedenen Medien aufruft, ist ferner zu bedenken, daß am Ende jeden Textabschnittes auf das Medium des Kommandofiles zurückgeschaltet wird. Am Ende einer Assemblierung ist auf die Standardeinheiten des Computers für die Ein- und Ausgabe zurückzuschalten.

7.3 Das Parsen

Auf die entscheidende Bedeutung der grammatikalischen Zerlegung einer Eingabezeile wurde schon hingewiesen. Die Zeichen am Beginn einer Zeile sollen dabei möglichst schon das Einschlagen eines logischen Fahrweges ermöglichen. Wenn eine Zeile z.B. mit einem Semikolon beginnt, so wird sie ohne weitere Nachprüfung als Kommentarzeile betrachtet. Zeilen mit einer Ziffer am Beginn werden verworfen, es sei denn, daß Zeilennummern zugelassen sind. Steht jedoch ein Buchstabe am Anfang, so wird angenommen, daß er entweder zu einem Symbol oder zu einer mnemonischen Befehlsbezeichnung gehört. (Assembler die fordern, daß ein Bezeichner immer in der ersten Schreibspalte stehen muß und daß alle anderen Sprachelemente erst in Spalten dahinter beginnen dürfen, sind unnötig beengend). Wenn die mit Alpha beginnende Stringlänge am Zeilenbeginn z.B. 3 ist, dann vermutet man zunächst einmal, daß es sich um eine mnemonische Befehlsbezeichnung handelt. Erstens beginnen die meisten Zeilen mit Mnemonics, zweitens sind diese im Tabellenteil des Assemblers schneller abgesucht, als eine umfangreiche Symboltafel. - Ein recht universell einsetzbarer Parser wurde in Heft 29 dieser Zeitschrift abgedruckt. Er tut auch hier gute

Dienste. - Ist ein reserviertes Befehlswort erkannt, so geht es mit der Bewertung des Operanden weiter (sofern der Befehl einen Operanden braucht).

Entspricht ein mit Buchstaben (Alpha) beginnender String nicht einem Befehl, so ist die Symboltafel abzusuchen. Wenn das Symbol noch nicht bekannt ist, so wird es mit dem Wert in die Tafel eingetragen, der auf ein mögliches nachgestelltes '=' folgt. Folgt das '=' nicht, so handelt es sich um ein Label im Programmverlauf. Es erhält den augenblicklichen Wert des Zuordnungszählers. Auf das Label können im Fahrweg der Zeile noch ein Befehl, Pseudo-Ops und anderes folgen.

Zeilen können aber nicht nur mit Kommentarzeichen, einem Label oder einem Befehl beginnen. Außer mit einer Sternanweisung kann eine Zeile auch mit einem Pseudo-Befehl beginnen, der typisch mit einem Punkt beginnt. Um den nachfolgenden Befehl zu entschlüsseln, wird man sich wieder eines Parsers und einer zugeordneten Tabelle bedienen.

7.4 Das Bewerten

Wie schon erwähnt, werden Label mit dem augenblicklich errreichten Stand des Zuordnungszählers bewertet. Explizit mit '=' erklärte Symbole erhalten den Wert des Ausdruckes, der auf das Zeichen '=' folgt. Das gilt auch für das Setzen des Zuordnungszählers mit '*='.

Ausdrücke finden wir auch zur Bezeichnung von Operanden nach Befehlen und Pseudo-Befehlen. Ausdrücke sind dabei nicht nur Zahlen des gewählten Zahlensystems (dezimal, hex, binär, oktal), sie enthalten typisch auch Bezeichner/Symbole und eine arithmetische oder logische Verknüpfung zwischen Zahlen und Symbolen. Symbole werden mit dem Wert eingesetzt, den der Assembler in der Symboltafel findet. Ist ein Symbol im ersten Pass noch nicht bekannt, so wird das bei der Bemessung der Befehlslänge zu berücksichtigen sein, wobei die vorläufig eingesetzte Länge in Pass 1 mit der endgültigen in Pass 2 identisch sein muß. Anderenfalls hat man Adreßverschiebungen in den Labels. Fehler dieser Art gehören zu den heimtückischen, weil man normalerweise für den Pass 1 keine Liste anfertigt, die man neben diejenige von Pass 2 legt. Die Ursache von Adreßverschiebungen (Phasenfehler) kann aber eigentlich nur auf diese Art eingekreist werden.

Typisch für die Bewertung von Ausdrücken ist eine Routine EVAL, die nicht nur Zahlen der verschiedenen Zahlensysteme auf binär bzw. hex umrechnet und miteinander verknüpft, sondern auch die Benutzung von (mit Alpha beginnenden) Symbolen erkennt und die der Symboltafel deren Wert entnimmt. An anderer Stelle in diesem Heft finden wir zum Thema eine Reihe nützlicher Routinen, wobei für die Zahlenwandlung insbesondere auf HORNER hinzuweisen ist. - Die Bewertung und Ablage von ASCII-Strings stellt einen Sonderfall dar.

7.5 Code erzeugen

Letztes Ziel des Assemblierens ist die Erzeugung maschinensprachlichen Codes. Bei Pseudo-Befehlen liegt mit der Bewertung in EVAL der entsprechende Code schon fest. Anders bei den mnemonischen Befehlen. Wird ein solcher Befehl erkannt, so entnimmt man nach dem Parsen den Grund-Opcode aus einer Tabelle. Dieser ist je nach erlaubter Adressierungsart noch abzuändern. Im Anschluß an die Bewertung des dem Befehl folgenden Operanden sind daher die möglichen Adressierungsarten zu überprüfen. Auch hier kann ein Parser der erwähnten Art nebst Tabelle nützlich sein, wenn es sich um zahlreiche Adressierungsmöglichkeiten handelt.

Die beabsichtigte Adressierungsart ist auf ihre Zulässigkeit zu prüfen und bei der Abänderung des Grund-Opcodes zu berücksichtigen. Ein optimierender Assembler wird dabei versuchen, die kürzere Befehlsform (Zero Page, Direct Page) wo immer möglich zu wählen. Auch das hat Einfluß auf den endgültigen Opcode.

Erzeugten Code wird man zunächst in einen Code-Buffer ablegen, von wo er später in Pass 2 auf die Zieleinheit abgesetzt wird, die im Initialisierungsteil festgelegt wurde.

7.6 Verwaltungsinformation fortführen

Wenn auch die Hauptttätigkeit eines Assemblers im Parsen, Bewerten und Suchen von Symbolen besteht, so dürfen Nebentätigkeiten nicht unerwähnt bleiben. So müssen z.B. die Zeiger der Symboltafel stets aktuell geführt werden. Der Zuordnungszähler (virtueller PC) ist entsprechend der Länge von Befehlen oder Pseudo-Befehlen zu erhöhen. Und auch die Zahl der eingetretenen Fehler

ist zu aktualisieren. Vor Beginn jeden Durchganges (Pass) und jeder Zeile des Quelltextes sind zahlreiche Variablen und Flags auf einen definierten Stand zu bringen, und zwar möglichst in einem übersichtlichen Zusammenhang.

7.7 Ausgabe von Listen

Der Programmierer hat erhebliche Arbeit auch mit dem Erzeugen eines formatierten Assembler-Listings in Pass 2. Es sollen der erzeugte Code, Symbole und der übrige Quelltext sowie Fehlermeldungen in einer möglichst klaren spaltenmäßigen Anordnung ausgegeben werden. Spielarten betreffen die Ausgabe nur von Fehlern oder die Option GEN für die Listung aller Bytes einer Textkette in der Anweisung .BYT.

Als zweite mögliche Liste ist diejenige mit der Symboltafel zu nennen. Sie sollte Namen, den Wert des Symbols und auch die Nummer der Zeile enthalten, in der dem Symbol sein Wert zugewiesen wurde. Die Wünsche der Anwender sind hier verschieden. Einige bevorzugen eine alphabetisch geordnete Symboltafel, andere haben lieber eine Reihenfolge, die der Folge des Quelltextes entspricht. - Eine dritte nützliche Liste ist die Kreuzreferenz (Cross Reference). Sie zeigt alle Stellen an, an denen ein Symbol benutzt wurde.

7.8 Tabellen

Auf die Bedeutung von Tabellen wurde bereits in den vorhergehenden Abschnitten hingewiesen. Für den Parser sind Tabellen mit den Sprachelementen vorzusehen, nämlich für mnemonische Befehle, für Pseudo-Befehle und Adressierungsarten. Weitere Tabellen sind vorzuhalten für den Grund-Opcode der Befehle, ggfs. für ihre Zugehörigkeit zu Befehlsgruppen und für ihre erlaubten Adressierungsarten. Zu den Tabellen sind auch Sprungleisten zu zählen, die das Assemblerprogramm in Fallentscheidungen verzweigen.

Im Sinne einer interaktiven Benutzerführung sind daneben Textttabellen für die Initialisierungsphase und für die Fehleranzeige in Klartext vorzusehen.

Nicht zum Assemblerprogramm selbst, aber zum Parser und zu EVAL gehört ferner die Symboltafel im RAM.

8. Zusammenfassung

Wenn wir hier auch nur die großen Funktionsblöcke des Assemblers besprechen konnten, so dürfte doch klar geworden sein, daß es beim Durcharbeiten des Quelltextes sehr viele Fallentscheidungen und damit sehr viele logische Fahrwege gibt. Sie führen schließlich beim Erzeugen einer Assemblerliste wieder zusammen. Sie setzt voraus, daß eine Zeile ausgewertet und der zu erzeugende Code bereits bekannt ist, damit er ausgedruckt werden kann. Bei der Verwirklichung von Assemblern hat es sich als zweckmäßig erwiesen, nicht etwa Kettenabfragen vom Stil CMP ... BEQ ... CMP ... BEQ zu programmieren, sondern Tabellen und Sprungleisten zu bilden, die eine Fallentscheidung im Stile CASE OF bzw. ON GOTO ermöglichen.

□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O□O

Bücher

Rappaport, A.: Erfahrungen mit Gate Arrays. Verlag te-wi, München 1985, 150 S. DM 66,- ISBN 3-921803-47-0. Dieses Buch aus dem Englischen ist im wesentlichen ein Erfahrungsbericht, wie es einem ergeht, wenn man die Herstellung eines kundenspezifischen IC's in Betracht zieht (Semi Custom IC). Die einzelnen Abschnitte sind dementprechend mit 'Bericht 1-9' überschrieben, die die verschiedenen zu durchlaufenden Phasen betreffen: Entwurf, Simulationen, Test, Übergabe zum Design, Produktion, Bilanz der Bemühungen. Wichtige im Buch herausgestellte Punkte sind Alternativen und Fragen der Wirtschaftlichkeit. - Kundenspezifische Schaltungen haben einen zunehmenden Markt, für den Auftraggeber bedeuten sie jedoch auch ein großes Risiko. Wo Themen dieser Art im Raume stehen, da sollte man diesen Bericht eines Vorreiters zur Hand nehmen.

Martin Albrecht, 4350 Recklinghausen

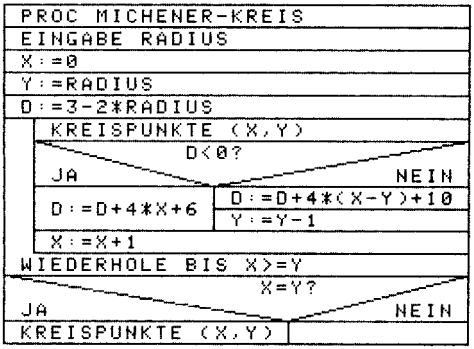
Kreisgenerator

mit Integer Arithmetik

Fast jeder Heim- oder Personal Computer verfügt heute über eine Rastergrafik. Das mitgelieferte BASIC ermöglicht meist auch die Kreiserzeugung mit einem einfachen Befehl. Die Programmierung in Maschinensprache wird jedoch kaum unterstützt.

Michener Kreisalgorithmus

Der im nachfolgenden Struktogramm dargestellte Algorithmus zur Kreiserzeugung eignet sich besonders gut für ein Maschinenprogramm.




```
PROC KREISPUNKTE (X,Y)
SETZE_PUNKT (X,Y)
SETZE_PUNKT (Y,X)
SETZE_PUNKT (X,-Y)
SETZE_PUNKT (Y,-X)
SETZE_PUNKT (-X,-Y)
SETZE_PUNKT (-Y,-X)
SETZE_PUNKT (-Y,X)
SETZE_PUNKT (-X,Y)
```

Die Prozedur MICHENER-KREIS erzeugt einen 360 Grad Vollkreis um den Koordinatenursprung (0,0). Der eigentliche Algorithmus liefert nur Kreispunkte im Winkelbereich von 90 bis 45 Grad (1/8 Kreisbogen). Die restlichen Kreispunkte erzeugt die Prozedur KREISPUNKTE durch Spiegelung an den Symmetrieachsen. Der Mittelpunkt kann vom Ursprung an jeden beliebigen Punkt verschoben werden, wenn vor dem Setzen der Kreispunkte entsprechende Konstanten zu X und Y addiert werden.
Der Algortihmus benötigt nur Additionen, Subtraktionen und Multiplikationen mit Potenzen von zwei. Die Multiplikation mit zwei bzw. vier wird in einem Maschinenprogramm durch einmaliges bzw. zweimaliges Linksschieben des Operanden erreicht.

Alle Kreispunkte werden inkremental erzeugt. Vom Punkt (0,Radius) beginnend wird der jeweils nächste Punkt aus den Koordinaten seines Vorgängers berechnet. Basis der Rechnung ist die Kreisgleichung: $X^2+Y^2=R^2$. Für jeden Punkt P(X,Y) im Winkelabschnitt von 90 bis 45 Grad gibt es nur zwei mögliche Nachfolger, (X+1,Y) und (X+1,Y-1) (siehe Bild).

Davon wird jeweils der gewählt, der den geringsten Fehler

(1) $D(P_K) = (X_K^2+Y_K^2) - R^2$

erzeugt und der Kreislinie am nächsten kommt. Für die Punkte $S_K(X_{K-1}+1, Y_{K-1})$ und $T_K(X_{K-1}+1, Y_{K-1}-1)$ als mögliche Nachfolger zum Punkt $P_{K-1}(X_{K-1}, Y_{K-1})$ ergeben sich die Fehler:

(2) $D(S_K) = [(X_{K-1}+1)^2+(Y_{K-1})^2] - R^2$
(3) $D(T_K) = [(X_{K-1}+1)^2+(Y_{K-1}-1)^2] - R^2$

und daraus die Entscheidung, welcher der beiden Punkte näher zur wirklichen Kreislinie liegt:

für $|D(S_K)| >= |D(T_K)|$ wähle T_K
für $|D(S_K)| < |D(T_K)|$ wähle S_K

oder zusammengefaßt zur Entscheidungsvariablen d_K:

(4) $d_K = |D(S_K)| - |D(T_K)|$
wähle T_K für $d_K >= 0$
wähle S_K für $d_K < 0$

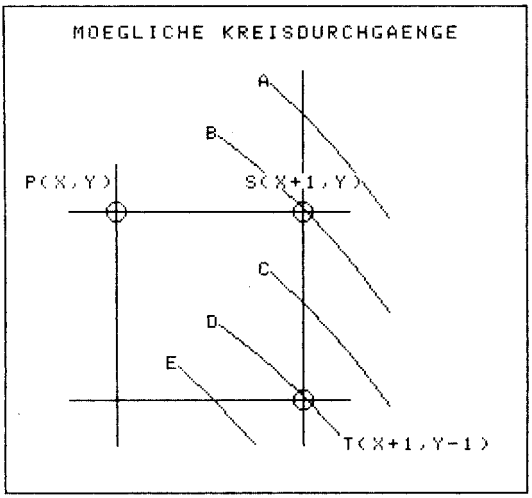
Betrachtet man die möglichen Durchgänge der wirklichen Kreislinie durch das Punktgitter im Verhältnis zu S_K und T_K (Fälle A,B,C,D und E in der Darstellung), dann ergeben sich folgende Resultate für $D(S_K)$ und $D(T_K)$:

Fall:	A	B	C	D	E
$D(S_K)$:	<0	=0	>0	>0	>0
$D(T_K)$:	<0	<0	<0	=0	>0
wähle:	S_K	S_K	S_K oder T_K	T_K	T_K

Mit diesen Ergebnissen können die Betragsstriche für $D(S_K)$ und $D(T_K)$ entfallen und man erhält für die Entscheidungsvariable d_K:

(5) $d_K = D(S_K) + D(T_K)$

Für die Fälle A und B ist d_K immer kleiner Null, so daß S_K gewählt wird, während für die Fälle D und E d_K immer größer Null ist und T_K gewählt wird. Im Fall C hängt die Auswahl vom größeren Fehler ab.



Ausgeschrieben erhält man:

$d_K = (X_{K-1}+1)^2 + (Y_{K-1})^2 - R^2 + (X_{K-1}+1)^2 + (Y_{K-1}-1)^2 - R^2$
(6) $d_K = 2X_{K-1}^2 + 4X_{K-1} + 2Y_{K-1}^2 - 2Y_{K-1} + 3 - 2R^2$

für K=1, Punkt (X_{K-1} = 0, Y_{K-1} = Radius R) erhält man durch Einsetzen:

(7) $d_1 = 3 - 2R$

für den neuen Punkt $S_K(X_{K-1}+1, Y_{K-1})$ ergibt sich für d_{K+1}:

$$\begin{aligned} d_{K+1} &= 2(X_{K-1}+1)^2 + 4(X_{K-1}+1) + 2Y_{K-1}^2 - 2Y_{K-1} + 3 - 2R^2 \\ &= 2X_{K-1}^2 + 4X_{K-1} + 2 + 4X_{K-1} + 4 + 2Y_{K-1}^2 - 2Y_{K-1} + 3 - 2R^2 \\ &= d_K + 4X_{K-1} + 6 \end{aligned} \quad (8)$$

und für den neuen Punkt $T_K(X_{K-1}+1, Y_{K-1}-1)$ ergibt sich für d_{K+1}:

$$\begin{aligned} d_{K+1} &= 2(X_{K-1}+1)^2 + 4(X_{K-1}+1) + 2(Y_{K-1}-1)^2 - 2(Y_{K-1}-1) + 3 - 2R^2 \\ &= 2X_{K-1}^2 + 4X_{K-1} + 2 + 4X_{K-1} + 4 + 2Y_{K-1}^2 - 4Y_{K-1} + 2 - 2Y_{K-1} \\ &\quad + 2 + 3 - 2R^2 \\ &= d_K + 4(X_{K-1} - Y_{K-1}) + 10 \end{aligned} \quad (9)$$

d_K bildet beginnend mit d_1 das Entscheidungskriterium. Gleichungen (8) und (9) geben an, wie die Entscheidungsvariable für den jeweils gewählten Punkt S_K oder T_K weiter berechnet wird. Daraus erhält man folgerichtig den im Struktogramm gezeigten Algorithmus.

Das vorgestellte Verfahren ist eine Vereinfachung des ursprünglichen Algorithmus von J. Bresenham [2], der ohne die Begrenzung auf den 45 Grad Sektor auskommt. Die Ausführungszeit steigt linear mit dem Radius R und das Verfahren arbeitet für beliebige Radien R >= 0.

Alternative Algorithmen

Tabellenverfahren
Benötigt wird eine Tabelle der Quadratzahlen von Null bis zum höchsten möglichen Radius R. Sie enthält für jeden Index K die entsprechende Quadratzahl K^2. Ausgehend von der Kreisgleichung $Y^2 = R^2 - X^2$ wird R^2 einmal aus der Tabelle entnommen. X wird von Null beginnend bis zum Abbruch bei X >= Y inkrementiert und für jedes X die Differenz $R^2 - X^2$ mit Hilfe der Tabellenwerte errechnet. Dann wird die Tabelle der Quadratzahlen K^2 durchsucht, bis ein Wert $K^2 <= (R^2 - X^2)$ gefunden ist. Der dazugehörende Index ergibt (näherungsweise) die Quadratwurzel aus $(R^2 - X^2)$ entsprechend Y. Die Prozedur KREISPUNKTE erzeugt die Kreispunkte in den weiteren Winkelabschnitten.

Die Größe der möglichen Radien ist hier durch die vorbestimmte Tabelle der Quadratzahlen stark eingeschränkt.

Rechenverfahren
X wird ebenso von Null beginnend bis zum Abbruch bei X >= Y inkrementiert. Die Quadratzahlen R^2 und X^2 werden mit einem Multiplikationsalgorithmus (z.B. nach [3]) bestimmt und die Quadratwurzel aus $(R^2 - X^2)$ mit einem Wurzelalgorithmus (z.B. nach [4]) berechnet. Die Kreispunkte in den weiteren Winkelabschnitten erzeugt wieder die Prozedur KREISPUNKTE.

Im Vergleich zum Michener Algorithmus entsteht ein wesentlich höherer Aufwand. Für große Y^2 steigt die Rechenzeit allein durch den Wurzelalgorithmus stark an. Eine Begrenzung für den Radius R besteht jedoch nicht.

Literatur
(1) James D. Foley, Andries van Dam: Fundamentals of Interactive Computer Graphics, Addison Wesley, 1982, p. 441-446
(2) Jack Bresenham: A Linear Algorithm for Incremental Digital Display of Circular Arcs, Communications of the ACM, February 1977, Vol. 20, No. 2, p. 100-106
(3) Peter Rix, Heiko Wolgast: Multiplikation und Division im FIG-FORTH, 65xx MICRO MAG Nr. 28, Dezember 1982, S. 45-46
(4) Peter Rix: Gerundete Integer Quadratwurzel ISQR, 65xx MICRO MAG Nr. 30, April 1983, S. 60

Peter W. Arps, 2000 Hamburg 73

Hardcopy mit Epson FX-80

Wenn man einen Drucker vom genannten Typ am C-64 angeschlossen hat, bereitet einem der Ausdruck einer Hardcopy vom Bildschirm nicht allzuviel Freude, da der FX-80 die Commodore-Grafik-Zeichen nicht direkt ausdrucken kann. Statt dessen werden auch diese Zeichen als ASCII-Zeichen ausgegeben. Es ist aber möglich, alle Commodore-Zeichen - also auch die grafischen - im Grafik-Modus auf dem FX-80 darzustellen. Von dieser Möglichkeit machen einige Drucker-Interfaces Gebrauch. Wenn man aber kein Geld für ein weiteres Interface ausgeben will, so kann man die Umsetzung, die sonst im Interface erfolgt, auch in einer kleinen Routine erledigen.

Das Hardcopy-Programm druckt alle Zeichen als Grafik, so als würde ein Grafik-Bildschirm ausgedruckt werden. Im Gegensatz zur Grafik-Hardcopy werden die benötigten Bitmuster aber nicht dem Grafik-RAM, sondern den Character-ROM entnommen. Für jedes zu druckende Zeichen werden dem Drucker acht Bytes übersandt, die das zu druckende Bitmuster des Zeichens enthalten. Damit der Drucker diese Bytes als Grafik und nicht als Zeichen druckt, muß er vorher mit der Steueranweisung 'ESC "K" Anzahl' in den Bitmuster-Modus gebracht werden.

Das für den Ausdruck erforderliche Bitmuster wird dem Character-ROM des C-64 entnommen. In diesem ROM sind je Zeichen die Bitmuster (0 = Punkt nicht gesetzt, 1 = Punkt gesetzt) in 8 Bytes abgelegt. Der erste Zeichensatz (Buchstaben/Grafik) liegt auf Adresse $D000, der zweite für Klein-/Großschreibung auf Adresse $D800. Um auf das Character-ROM zugreifen zu können, muß vorher das Steuerregister R6510 der CPU auf Adresse $01 umgeschaltet werden. In Abhängigkeit vom auszudruckenden Zeichen wird die entsprechende 8-Byte-Folge aus dem Character-ROM geholt.

Das so ermittelte Bitmuster kann aber noch nicht so auf dem Drucker ausgegeben werden. Jedes Zeichen setzt sich aus acht Zeilen zu je acht Spalten zusammen. Jedes Byte aus dem Character-ROM enthält aber das Bitmuster einer Zeile. Vom Drucker werden die Zeichen aber spaltenweise aufgebaut. Die Bitmuster sind dabei umzudrehen. Zum Beispiel für den Buchstaben 'C':

```
Ablage im ROM ==> 3C  0  0  1  1  1  1  0  0      ****
                  66  0  1  1  0  0  1  1  0     **  **
                  60  0  1  1  0  0  0  0  0     **
                  60  0  1  1  0  0  0  0  0     **
                  60  0  1  1  0  0  0  0  0     **
                  66  0  1  1  0  0  1  1  0     **  **
                  3C  0  0  1  1  1  1  0  0      ****
                  00  0  0  0  0  0  0  0  0
an FX-80 senden ==>  00 7C FE 82 82 86 44 00
```

Werden die so umgesetzten Bitmuster ausgedruckt, dann erhält man eine Hardcopy mit den originalen Commodore-Zeichen. - Die Arbeitsweise des Programms ist dem Struktorgramm zu entnehmen. Wird in der Routine HOL.CHRROM nicht auf das Character-ROM, sondern auf den Grafik-Speicher zugegriffen, dann kann das Programm nach Änderung auch zum Ausdruck von Grafik-Bildschirmen verwendet werden.

```
0030              .LS
0040              .BA $C000
0050              ;
0060              ;======================
0070              ;==                  ==
0080              ;==  H A R D C O P Y  ==
0090              ;==  AUF EPSON FX-80  ==
0100              ;==  IM GRAPHIK-MODUS ==
0110              ;==                  ==
0120              ;======================
0130              ;
0140 R6510        .DE 1
0150 CHRROM       .DE $31
```

```
H A R D C O P Y
Groß-/Klein-Schreibung?
ja                                  nein
CHR-ROM-Adr $D800 merken | CHR-ROM-Adr $D000 merken
Drucker eröffnen
Ausgabe auf Drucker legen
Drucker-Vorschub auf 8 Line/Inch einstellen
bis 25 Zeilen übertragen sind
    10 Leerstellen übertragen
    8-Bit-Modus am Drucker für 320 Bytes mit
    CHR$(27);"K";CHR$(64);CHR$(i) einschalten
    bis 40 Zeichen übertragen sind
        Byte aus Bildschirm-RAM holen
        Byte mit 8 multiplizieren = Offset
        Offset mit gemerkter CHR-ROM-Adresse
        addieren = relevante Adresse im CHR-ROM
        Inhalt R6510 (Adresse $01) sichern
        $31 nach R6510 = CHR-ROM einschalten
        ab relevanter Adresse 8 Bytes in den
        Buffer 1 übertragen
        alten Inhalt R6510 zurückschreiben
        horizontales Bitmuster aus Buffer 1 in
        vertikales ändern und in den Buffer 2
        schreiben
        Buffer 2 auf Drucker ausgeben
    CR auf Drucker ausgeben
    STOP gedrückt?
    ja                              nein
                                    weiter mit nächster Zeile
Drucker-Vorschub auf 6 Line/Inch einstellen
Ausgabe wieder auf Bildschirm legen
Drucker schließen
E N D E
```

```
                0160 ZEILEN      .DE $4B
                0170 SCREEN.ADR  .DE $4C
                0180 CHR.ADR     .DE $4E
                0190 PNT3        .DE $50
                0200 TMP         .DE $52
                0210 HI.BASE     .DE $288
                0220 BUFFER1     .DE $100
                0230 BUFFER2     .DE $108
                0240 CHR.ROM     .DE $D000
                0250 CHR.REG     .DE $D018
                0260 PRINT       .DE $FFD2
                0270 BRK.TEST    .DE $F6ED
                0280 SET.NAME    .DE $FFBD
                0290 SET.LFS     .DE $FFBA
                0300 OPEN        .DE $FFC0
                0310 CLOSE       .DE $FFC3
                0320 CHK.OUT     .DE $FFC9
                0330 CLR.CHN     .DE $FFCC
                0340             ;
                0350             ;--------------------------
                0360             ;- DRUCKER EROEFFNEN UND -
                0370             ;-  OUTPUT AUF DRUCKER   -
                0380             ;- ZEICHENSATZ ERMITTELN -
                0390             ;--------------------------
                0400             ;
C000- A9 00     0410 ENTRY       LDA #0
C002- AE 88 02  0420             LDX HI.BASE          ;ADR SCREEN MEM
C005- 85 4C     0430             STA *SCREEN.ADR
```

```
C007- 86 4D      0440          STX *SCREEN.ADR+1
C009- 85 4E      0450          STA *CHR.ADR
C00B- A2 D0      0460          LDX #H,CHR.ROM
C00D- A9 02      0470          LDA #2              ;SCHREIB-MODUS
C00F- 2C 18 D0   0480          BIT CHR.REG         ;TESTEN
C012- F0 02      0490          BEQ GX              ;BUCHSTABEN/GRAPHIK
C014- A2 D8      0500          LDX #H,CHR.ROM+$800
C016- 86 4F      0510 GX       STX *CHR.ADR+1
C018- A9 00      0520          LDA #0              ;OPEN 44,4,6
C01A- AA         0530          TAX
C01B- A8         0540          TAY
C01C- 20 BD FF   0550          JSR SET.NAME
C01F- A9 2C      0560          LDA #44             ;FILE-NR
C021- A2 04      0570          LDX #4              ;DEVICE
C023- A0 06      0580          LDY #6              ;SEC-ADR
C025- 20 BA FF   0590          JSR SET.LFS
C028- 20 C0 FF   0600          JSR OPEN
C02B- A2 2C      0610          LDX #44             ;CMD 44
C02D- 20 C9 FF   0620          JSR CHK.OUT
C030- A9 1B      0630          LDA #27             ;ESC
C032- 20 D2 FF   0640          JSR PRINT
C035- A9 30      0650          LDA #'0             ;8 LINE/INCH
C037- 20 D2 FF   0660          JSR PRINT
                 0670          ;
                 0680          ;--------------------
                 0690          ;- ZEILEN AUSGEBEN -
                 0700          ;--------------------
                 0710          ;
C03A- A9 19      0720          LDA #25             ;FUER 25 ZEILEN
C03C- 85 4B      0730          STA *ZEILEN
C03E- A2 0A      0740 LOOP0    LDX #10
C040- A9 20      0750          LDA #' '            ;BLANKS AUSGEBEN
C042- 20 D2 FF   0760 LOOP1    JSR PRINT
C045- CA         0770          DEX
C046- D0 FA      0780          BNE LOOP1
C048- A9 1B      0790          LDA #27             ;ESC
C04A- 20 D2 FF   0800          JSR PRINT           ;8-PUNKT
C04D- A9 4B      0810          LDA #'K             ;BITMUSTER-MODUS
C04F- 20 D2 FF   0820          JSR PRINT           ;FUER 320 ZEICHEN
C052- A9 40      0830          LDA #64             ;EINSCHALTEN
C054- 20 D2 FF   0840          JSR PRINT
C057- A9 01      0850          LDA #1
C059- 20 D2 FF   0860          JSR PRINT
C05C- A0 00      0870          LDY #0
C05E- B1 4C      0880 LOOP2    LDA (SCREEN.ADR),Y
C060- 20 A5 C0   0890          JSR HOL.CHRROM
C063- 20 D7 C0   0900          JSR DREH.BITS
C066- A2 00      0910          LDX #0
C068- BD 08 01   0920 LOOP3    LDA BUFFER2,X       ;ZEICHEN AUSGEBEN
C06B- 20 D2 FF   0930          JSR PRINT
C06E- E8         0940          INX
C06F- E0 08      0950          CPX #8              ;8 BYTES = 1 ZEICHEN
C071- D0 F5      0960          BNE LOOP3           ;WEITER
C073- C8         0970          INY
C074- C0 28      0980          CPY #40             ;ZEILEN-ENDE?
C076- D0 E6      0990          BNE LOOP2           ;NEIN - WEITER
C078- A9 0D      1000          LDA #13             ;CR AUSGEBEN
C07A- 20 D2 FF   1010          JSR PRINT
C07D- 18         1020          CLC
C07E- A9 28      1030          LDA #40             ;ADR NAECHSTE ZEILE
C080- 65 4C      1040          ADC *SCREEN.ADR
C082- 85 4C      1050          STA *SCREEN.ADR
C084- A9 00      1060          LDA #0
C086- 65 4D      1070          ADC *SCREEN.ADR+1
C088- 85 4D      1080          STA *SCREEN.ADR+1
C08A- 20 ED F6   1090          JSR BRK.TEST        ;STOP GEDRUECKT?
C08D- F0 04      1100          BEQ ENDE            ;JA - AUFHOEREN
C08F- C6 4B      1110          DEC *ZEILEN
```

```
C091- D0 AB      1120             BNE LOOP0              ;WEITER
                 1130             ;
                 1140             ;-----------------
                 1150             ;- ENDE-ROUTINE -
                 1160             ;-----------------
                 1170             ;
C093- A9 1B      1180 ENDE        LDA #27                ;ESC
C095- 20 D2 FF   1190             JSR PRINT
C098- A9 32      1200             LDA #'2                ;WIEDER 6 LINE/INCH
C09A- 20 D2 FF   1210             JSR PRINT
C09D- 20 CC FF   1220             JSR CLR.CHN            ;KANAL SCHLIESSEN
C0A0- A9 2C      1230             LDA #44                ;CLOSE44
C0A2- 4C C3 FF   1240             JMP CLOSE              ;CLOSE DRUCKER
                 1260             ;------------------------
                 1270             ;- BITMUSTER AUS DEM   -
                 1280             ;- CHARACTER-ROM HOLEN -
                 1290             ;------------------------
                 1300             ;
C0A5- A2 00      1310 HOL.CHRROM  LDX #0
C0A7- 86 52      1320             STX *TMP
C0A9- 0A         1330             ASL A                  ;BYTE MAL 8
C0AA- 26 52      1340             ROL *TMP
C0AC- 0A         1350             ASL A
C0AD- 26 52      1360             ROL *TMP
C0AF- 0A         1370             ASL A
C0B0- 26 52      1380             ROL *TMP
C0B2- 18         1390             CLC
C0B3- 65 4E      1400             ADC *CHR.ADR           ;RELEVANTE ADR
C0B5- 8D CA C0   1410             STA H0+1               ;DES CHARACTER-ROMS
C0B8- A5 52      1420             LDA *TMP
C0BA- 65 4F      1430             ADC *CHR.ADR+1
C0BC- 8D CB C0   1440             STA H0+2
C0BF- A5 01      1450             LDA *R6510
C0C1- 48         1460             PHA                    ;SAVE
C0C2- A9 31      1470             LDA #CHRROM
C0C4- 78         1480             SEI                    ;INTERRUPT AUS
C0C5- 85 01      1490             STA *R6510
C0C7- A2 07      1500             LDX #7                 ;8 BYTES UEBERTRAGEN
C0C9- BD FF FF   1510 H0          LDA $FFFF,X            ;WIRD MODIFIZIERT
C0CC- 9D 00 01   1520             STA BUFFER1,X
C0CF- CA         1530             DEX
C0D0- 10 F7      1540             BPL H0
C0D2- 68         1550             PLA
C0D3- 85 01      1560             STA *R6510             ;RESTORE
C0D5- 58         1570             CLI                    ;INTERRUPT EIN
C0D6- 60         1580             RTS
                 1590             ;
                 1600             ;--------------------------
                 1610             ;-   BITMUSTER UMDREHEN   -
                 1620             ;- HORIZONTAL => VERTIKAL -
                 1630             ;--------------------------
                 1640             ;
C0D7- 84 52      1650 DREH.BITS   STY *TMP               ;SAVE
C0D9- A0 00      1660             LDY #0
C0DB- A2 00      1670 DLOOP1      LDX #0
C0DD- A9 00      1680             LDA #0
C0DF- 1E 00 01   1690 DLOOP2      ASL BUFFER1,X
C0E2- 2A         1700             ROL A
C0E3- E8         1710             INX
C0E4- E0 08      1720             CPX #8
C0E6- D0 F7      1730             BNE DLOOP2
C0E8- 99 08 01   1740             STA BUFFER2,Y          ;BIT N AUS BYTES 0-7
C0EB- C8         1750             INY
C0EC- C0 08      1760             CPY #8
C0EE- D0 EB      1770             BNE DLOOP1
C0F0- A4 52      1780             LDY *TMP               ;RESTORE
C0F2- 60         1790             RTS
                 1800 ;           .EN
```

Klaus-Dieter Kossow, 2074 Mollhagen

AUTO für CBM 710/720

Für die genannten Commodore-Rechner sind bisher wenig Programmierhilfen veröffentlicht worden. Hier nun ein Programm AUTO, das für die BASIC-Programmierung automatisch die Zeilennummern in den gewählten Abständen erzeugt. Es wertet die Tatsache aus, daß viele Aufrufe von Systemroutinen soft-vektoriert sind, d.h. sie werden indirekt über einen im RAM abgelegten Vektor angesprungen. Diesen kann man nach Bedarf verändern. - Wenn auch diese Rechnertypen keine so große Verbreitung gefunden haben, so mag doch das Schema dieser Anwendung auch für andere Programmierungen interessant sein. Schließlich benutzt ja auch der neue PC 128 eine gleichartige Vektorierung.

AUTO 710 belegt in Bank 15 den Bereich $0400-04E1. Das Programm wird mit SYS(1024) initialisiert. Es wird zunächst, beginnend beim Label Auto, die Haupt-Interpreterschleife und die Token-Umwandlungsroutine des OS aufgetrennt..

In der Interpreter-Schleife wird jetzt zusätzlich das Zeilen-Inkrement ermittelt und die neue Zeilennummer vorgegeben, d.h. im Tastaturpuffer abgelegt. Sollte diese Zeilennummer bereits belegt sein, so wird die Vorgabe unterbunden. Nach dem Rücksprung ins OS (Zeile 11365) wird die aktuelle Zeilennummer aus dem Tastaturpuffer in den Eingabepuffer übernommen und auf dem Bildschirm ausgegeben.

Der Programmteil ab Label Auto50 wird in die Token-Umwandlungsroutine eingeschoben. Hier wird in der Hauptsache geprüft, ob die Zeilennummer in der ersten Stelle begann, d.h. ob die nächste Zeilennummer vorgegeben werden soll. - Durch SYS(1035), d.h. Ansprung von Autoff werden die geänderten indirekten Sprungadressen wieder zurückgesetzt und AUTO damit ausgeschaltet.

```
  1  1 0000 0000                    ;auto 710/6
 10  1 0000 0000                    ;************************************
 20  1 0000 0000                    ;*  auto 710/6 src                  *
 30  1 0000 0000                    ;*  bload"auto710"onb15,p1024       *
 40  1 0000 0000                    ;*  sys(1024)                       *
 50  1 0000 0000                    ;*  zeilen-nr. in der ersten stelle *
 60  1 0000 0000                    ;*  schaltet auto ein               *
 70  1 0000 0000                    ;*  zn in der zweiten stelle oder   *
 80  1 0000 0000                    ;*  sonst. eingabe schaltet auto aus*
 90  1 0000 0000                    ;*  sys(1027) setzt die indir.      *
100  1 0000 0000                    ;*  sprung-adr. zurueck             *
110  1 0000 0000                    ;************************************
500  1 0000 0000                    ;system-adressen
505  1 0000 0000          imain     =$0282;sprung zur interpreter-schleife
510  1 0000 0000          linnum    =$1b;zeilen-nr.
515  1 0000 0000          icrnch    =$0284;sprung zur token-routine
520  1 0000 0000          fndlin    =$871f;basic-zeile suchen
530  1 0000 0000          nmain     =$85cd;interpreter-schleife
540  1 0000 0000          buffpt    =$88;eingabe-puffer
545  1 0000 0000          qnum      =$ba50;test ob zeichen numerisch
550  1 0000 0000          ncnnch    =$88c2;token-routine
555  1 0000 0000          facho     =$72;fac #1
560  1 0000 0000          facmoh    =$73;fac #1
565  1 0000 0000          floatc    =$a220;
570  1 0000 0000          fout      =$a3e2;umw. gl.-komma/string
575  1 0000 0000          buf       =$0200;eingabe-puffer
595  1 0000 0000          keyd      =$03ab;tastatur-puffer
596  1 0000 0000          ndx       =$d1;anz. bytes im tast.-p.
597  1 0000 0000          chrgot    =$ba29;ein zeichen einlesen
600  1 0000 0000                    *=$0400
601  1 0400 0000                    &=$5450
700  1 0400 5450 4c 0d 04           jmp auto
710  1 0403 5453 4c d7 04           jmp autoff
800  1 0406 5456  *                 ;variable
805  1 0406 5456 00       autflg    .$00
```

```
  810  1 0407 5457 00 00     actlin    .$00,$00
  815  1 0409 5459 00 00     lstlin    .$00,$00
  820  1 040b 545b 00 00     lincnt    .$00,$00
10000  1 040d 545d                     ;-----------------------------------
10005  1 040d 545d a2 04     auto      ldx #>auto10
10010  1 040f 545f a9 24               lda #<auto10
10020  1 0411 5461 8e 83 02            stx imain+1
10030  1 0414 5464 8d 82 02            sta imain
10040  1 0417 5467 a9 ad               lda #<auto50
10050  1 0419 5469 a2 04               ldx #>auto50
10060  1 041b 546b 8d 84 02            sta icrnch
10070  1 041e 546e 8e 85 02            stx icrnch+1
10080  1 0421 5471 4c cd 85 auto6      jmp nmain
11000  1 0424 5474 2c 06 04 auto10     bit autflg
11010  1 0427 5477 10 f8               bpl auto6;keine zeilen-nr. vorgeben
11011  1 0429 5479 ad 09 04            lda lstlin;test ob 1. zn
11012  1 042c 547c 0d 0a 04            ora lstlin+1
11013  1 042f 547f d0 0c               bne auto11
11014  1 0431 5481 a9 0a               lda #$0a;zeilen-inkrement = 10
11015  1 0433 5483 8d 0b 04            sta lincnt
11016  1 0436 5486 a9 00               lda #$00
11017  1 0438 5488 8d 0c 04            sta lincnt+1
11018  1 043b 548b f0 19               beq auto13
11020  1 043d 548d 38          auto11  sec;zeilen-inkrement ermitteln
11030  1 043e 548e ad 07 04            lda actlin
11040  1 0441 5491 ed 09 04            sbc lstlin
11050  1 0444 5494 8d 0b 04            sta lincnt
11060  1 0447 5497 ad 08 04            lda actlin+1
11070  1 044a 549a ed 0a 04            sbc lstlin+1
11080  1 044d 549d 8d 0c 04            sta lincnt+1
11081  1 0450 54a0 10 04               bpl auto13
11085  1 0452 54a2 a2 00               ldx #0
11086  1 0454 54a4 f0 2f               beq auto16
11090  1 0456 54a6 18          auto13  clc
11100  1 0457 54a7 ad 07 04            lda actlin
11105  1 045a 54aa 8d 09 04            sta lstlin
11110  1 045d 54ad 6d 0b 04            adc lincnt
11120  1 0460 54b0 8d 07 04            sta actlin
11125  1 0463 54b3 ad 08 04            lda actlin+1
11126  1 0466 54b6 8d 0a 04            sta lstlin+1
11130  1 0469 54b9 90 03               bcc auto12
11140  1 046b 54bb ee 08 04            inc actlin+1
11150  1 046e 54be ad 07 04 auto12     lda actlin
11160  1 0471 54c1 ae 08 04            ldx actlin+1
11170  1 0474 54c4 85 1b               sta linnum
11180  1 0476 54c6 86 1c               stx linnum+1
11190  1 0478 54c8 20 1f 87            jsr fndlin;zeilen-adresse suchen
11200  1 047b 54cb 90 0d               bcc auto20
11205  1 047d 54cd a2 00               ldx #$00
11206  1 047f 54cf 8e 09 04            stx lstlin
11207  1 0482 54d2 8e 0a 04            stx lstlin+1
11208  1 0485 54d5 8e 06 04 auto16     stx autflg
11209  1 0488 54d8 f0 97               beq auto6
11229  1 048a 54da a6 1b     auto20    ldx linnum  ;zeilen-nr. umw.
11230  1 048c 54dc a5 1c               lda linnum+1;in string
11240  1 048e 54de 85 72               sta facho
11250  1 0490 54e0 86 73               stx facmoh
11260  1 0492 54e2 a2 90               ldx #$90
11270  1 0494 54e4 38                  sec
11280  1 0495 54e5 20 20 a2            jsr floatc
11290  1 0498 54e8 20 e2 a3            jsr fout
11300  1 049b 54eb a2 01               ldx #$01;1. stelle (bl) nicht ausg.
11310  1 049d 54ed bd 00 02 auto14     lda buf,x;zeilen-nr. in tast.-puffer
11320  1 04a0 54f0 f0 06               beq auto15
11325  1 04a2 54f2 9d aa 03            sta keyd-1,x
11340  1 04a5 54f5 e8                  inx
11350  1 04a6 54f6 d0 f5               bne auto14
11360  1 04a8 54f8 86 d1     auto15    stx ndx
11365  1 04aa 54fa 4c cd 85            jmp nmain;sprung ins os
```

```
12000  1 04ad 54fd a5 1b     auto50      lda linnum
12001  1 04af 54ff a6 1c                 ldx linnum+1
12002  1 04b1 5501 8d 07 04              sta actlin
12003  1 04b4 5504 8e 08 04              stx actlin+1
12004  1 04b7 5507 20 29 ba              jsr chrgot
12005  1 04ba 550a c9 00                 cmp #$00
12006  1 04bc 550c f0 07                 beq auto51
12018  1 04be 550e b1 88                 lda (buffpt),y
12019  1 04c0 5510 20 50 ba              jsr qnum;test numerisch
12020  1 04c3 5513 90 0a                 bcc auto55;ja
12030  1 04c5 5515 a2 00     auto51      ldx #$00;flag loeschen
12060  1 04c7 5517 8e 09 04              stx lstlin
12065  1 04ca 551a 8e 0a 04              stx lstlin+1
12066  1 04cd 551d f0 02                 beq auto60
12070  1 04cf 551f a2 ff     auto55      ldx #$ff;flag setzen
12080  1 04d1 5521 8e 06 04  auto60      stx autflg
12090  1 04d4 5524 4c c2 88              jmp ncnnch;sprung ins os
13000  1 04d7 5527 a9 cd     autoff      lda #<nmain;auto abschalten
13010  1 04d9 5529 a2 85                 ldx #>nmain
13020  1 04db 552b 8d 82 02              sta imain
13030  1 04de 552e 8e 83 02              stx imain+1
13040  1 04e1 5531 a9 c2                 lda #<ncnnch
13050  1 04e3 5533 a2 88                 ldx #>ncnnch
13060  1 04e5 5535 8d 84 02              sta icrnch
13070  1 04e8 5538 8e 85 02              stx icrnch+1
13080  1 04eb 553b 4c cd 85              jmp nmain
```

Peter W. Arps, 2000 Hamburg 73

MC68000-Befehle

Aufbau und Bitverschlüsselung

Als Programmierer des 65xx (und auch der IBM/370) ist man mit etwas Übung und einer Liste in der Lage, den Operations-Kode eines Befehls aus einem bestimmten Hexwert zu erkennen. Die evtl. benötigte Liste braucht lediglich 256 Eintragungen zu enthalten, da der Befehl und seine Adressierungsart in einem Byte verschlüsselt sind. Beim MC 68000 aber versagt dieses Verfahren, denn bei diesem Prozessor müßte die Liste 65536 Positionen umfassen.

Die Befehle des MC 68000 sind zwei Bytes (1 Wort) lang. Lediglich aus dem höchstwertigen Halbbyte läßt sich unmittelbar eine grobe Zuordnung zu einer Befehlsgruppe vornehmen. Die restlichen den Befehl betreffenden Informationen sind von einzelnen Bits abhängig. Dies liegt offenbar an der Architektur des MC 68000. Der Prozessor wurde nicht mit Standard-Logikfunktionen realisiert, sondern er enthält für jeden Befehl ein Mikroprogramm, das seinerseits Nanoprogramme zur Ausführung bringt. Die meisten 68000er-Befehle haben dabei das folgende Schema:

```
Bit 15 - 12:  Befehlsgruppe
Bit 11 -  9:  1. Operand (Quelle oder Ziel)
Bit  8 -  6:  Modus und Operationslänge
Bit  5 -  0:  2. Operand (Ziel oder Quelle)
```

Das genaue Bitmuster eines jeden Befehls läßt sich aus der beigefügten Aufstellung entnehmen.

Das nachfolgende BASIC-Programm zur Dekodierung eines MC-68000-Befehls sowie der jeweils benötigten Erweiterungsworte enthält zwei EXBASIC-Befehle. Diese konvertieren einen Hexa-String in eine Zahl (DEC(x)) bzw. eine Dezimalzahl in einen Hexa-String (HEX$(x)).

Literaturhinweise

Koch, Der 16bit-Mikroprozessor SC 68000, Bd. 1 Eigenschaften/ Bd. 2 Befehlsvorrat, VALVO (Verlag Boysen + Maasch)
Scanlon, Die 68'000er, AT-Verlag (Sonderdruck aus 'Elektroniker')

Bitmuster der Befehle des MC68000 (sortiert nach Bitmustern)

```
00000000 11eeeeee  ORI      #d,ea
00000010 11eeeeee  ANDI     #d,ea
00000100 11eeeeee  SUBI     #d,ea
00000110 11eeeeee  ADDI     #d,ea
00001000 00eeeeee  BTST     #d,ea
00001000 01eeeeee  BCHG     #d,ea
00001000 10eeeeee  BCLR     #d,ea
00001000 11eeeeee  BSET     #d,ea
00001010 11eeeeee  EORI     #d,ea
00001100 11eeeeee  CMPI     #d,ea
0000rrr1 00eeeeee  BTST     Dn,ea
0000rrr1 01eeeeee  BCHG     Dn,ea
0000rrr1 0x001rrr  MOVEP    d(An),Dn
0000rrr1 10eeeeee  BCLR     Dn,ea
0000rrr1 11eeeeee  BSET     Dn,ea
0000rrr1 1x001rrr  MOVEP    Dn,d(An)
0001zzzz zzqqqqqq  MOVE.B   ea,ea
0010zzz0 01qqqqqq  MOVEA.L  ea,An
0010zzzz zzqqqqqq  MOVE.L   ea,ea
0011zzz0 01qqqqqq  MOVEA.W  ea,An
0011zzzz zzqqqqqq  MOVE.W   ea,ea
01000000 11eeeeee  MOVE     SR,ea
01000000 11eeeeee  NEGX     ea
01000010 11eeeeee  CLR      ea
01000100 11eeeeee  MOVE     ea,CCR
01000100 11eeeeee  NEG      ea
01000110 11eeeeee  MOVE     ea,SR
01000110 11eeeeee  NOT      ea
01001000 00eeeeee  NBCD     ea
01001000 01000rrr  SWAP     Dn
01001000 01eeeeee  PEA      ea
01001000 1xeeeeee  MOVEM    list,ea
01001000 xx000rrr  EXT      Dn
01001010 11111100  ILLEGAL
01001010 11eeeeee  TAS      ea
01001010 11eeeeee  TST      ea
01001100 1xeeeeee  MOVEM    ea,list
01001110 0100vvvv  TRAP     #v
01001110 01010rrr  LINK     An,#d
01001110 01011rrr  UNLK     An
01001110 01100rrr  MOVE     An,USP
01001110 01101rrr  MOVE     USP,An
01001110 01110000  RESET
01001110 01110001  NOP
01001110 01110010  STOP     #d
01001110 01110011  RTE
01001110 01110101  RTS
01001110 01110110  TRAPV
01001110 01110111  RTR
01001110 10eeeeee  JSR      ea
01001110 11eeeeee  JMP      ea
0100rrr1 10eeeeee  CHK      ea,Dn
0100rrr1 11eeeeee  LEA      ea,An
0101cccc 11001rrr  DBcc     Dn,label
0101cccc 11eeeeee  Scc      ea
0101ddd0 11eeeeee  ADDQ     #d,ea
0101ddd1 11eeeeee  SUBQ     #d,ea
01100000 xxxxxxxx  BRA      label
01100001 xxxxxxxx  BSR      label

0110cccc xxxxxxxx  Bcc      label
0111rrr0 dddddddd  MOVEQ    #d,Dn
1000rrr0 11eeeeee  DIVU     ea,Dn
1000rrr0 11eeeeee  OR       ea,Dn
1000rrr1 00000rrr  SBCD     Dn,Dn
1000rrr1 00001rrr  SBCD     -(An),-(An)
1000rrr1 11eeeeee  DIVS     ea,Dn
1000rrr1 11eeeeee  OR       Dn,ea
1001rrr0 11eeeeee  SUBA.W   ea,An
1001rrr0 11eeeeee  SUB      ea,Dn
1001rrr1 11eeeeee  SUBA.L   ea,An
1001rrr1 11000rrr  SUBX     Dn,Dn
1001rrr1 11001rrr  SUBX     -(An),-(An)
1001rrr1 11eeeeee  SUB      Dn,ea
1010xxxx xxxxxxxx  A-Emu
1011rrr0 11eeeeee  CMPA.W   ea,An
1011rrr0 11eeeeee  CMP      ea,Dn
1011rrr1 11eeeeee  CMPA.L   ea,An
1011rrr1 11001rrr  CMPM     (An)+,(An)+
1011rrr1 11eeeeee  EOR      Dn,ea
1100rrr0 11eeeeee  MULU     ea,Dn
1100rrr0 11eeeeee  AND      ea,Dn
1100rrr1 00000rrr  ABCD     Dn,Dn
1100rrr1 00001rrr  ABCD     -(An),-(An)
1100rrr1 11eeeeee  MULS     ea,Dn
1100rrr1 11eeeeee  AND      Dn,ea
1100rrr1 xx00xrrr  EXG      Rn,Rn
1101rrr0 11eeeeee  ADDA.W   ea,An
1101rrr0 11eeeeee  ADD      ea,Dn
1101rrr1 11eeeeee  ADDA.L   ea,An
1101rrr1 11000rrr  ADDX     Dn,Dn
1101rrr1 11001rrr  ADDX     -(An),-(An)
1101rrr1 11eeeeee  ADD      Dn,ea
11100000 11eeeeee  ASR      ea
11100001 11eeeeee  ASL      ea
11100010 11eeeeee  LSR      ea
11100011 11eeeeee  LSL      ea
11100100 11eeeeee  ROXR     ea
11100101 11eeeeee  ROXL     ea
11100110 11eeeeee  ROR      ea
11100111 11eeeeee  ROL      ea
1110ddd0 11000rrr  ASR      #d,Dn
1110ddd0 11001rrr  LSR      #d,Dn
1110ddd0 11010rrr  ROXR     #d,Dn
1110ddd0 11011rrr  ROR      #d,Dn
1110ddd1 11000rrr  ASL      #d,Dn
1110ddd1 11001rrr  LSL      #d,Dn
1110ddd1 11010rrr  ROXL     #d,Dn
1110ddd1 11011rrr  ROL      #d,Dn
1110rrr0 11100rrr  ASR      Dn,Dn
1110rrr0 11101rrr  LSR      Dn,Dn
1110rrr0 11110rrr  ROXR     Dn,Dn
1110rrr0 11111rrr  ROR      Dn,Dn
1110rrr1 11100rrr  ASL      Dn,Dn
1110rrr1 11101rrr  LSL      Dn,Dn
1110rrr1 11110rrr  ROXL     Dn,Dn
1110rrr1 11111rrr  ROL      Dn,Dn
1111xxxx xxxxxxxx  F-Emu
```

Bitmuster der Befehle des MC68000 (sortiert nach Befehlen)

1100rrr1 00000rrr	ABCD	Dn,Dn
1100rrr1 00001rrr	ABCD	-(An),-(An)
1101rrr0 11eeeeee	ADD	ea,Dn
1101rrr1 11eeeeee	ADD	Dn,ea
1101rrr1 11eeeeee	ADDA.L	ea,An
1101rrr0 11eeeeee	ADDA.W	ea,An
00000110 11eeeeee	ADDI	#d,ea
0101ddd0 11eeeeee	ADDQ	#d,ea
1101rrr1 11000rrr	ADDX	Dn,Dn
1101rrr1 11001rrr	ADDX	-(An),-(An)
1100rrr0 11eeeeee	AND	ea,Dn
1100rrr1 11eeeeee	AND	Dn,ea
00000010 11eeeeee	ANDI	#d,ea
1110ddd1 11000rrr	ASL	#d,Dn
1110rrr1 11100rrr	ASL	Dn,Dn
11100001 11eeeeee	ASL	ea
1110ddd0 11000rrr	ASR	#d,Dn
1110rrr0 11100rrr	ASR	Dn,Dn
11100000 11eeeeee	ASR	ea
1010xxxx xxxxxxxx	A-Emu	
0110cccc xxxxxxxx	Bcc	label
0000rrr1 01eeeeee	BCHG	Dn,ea
00001000 01eeeeee	BCHG	#d,ea
0000rrr1 10eeeeee	BCLR	Dn,ea
00001000 10eeeeee	BCLR	#d,ea
01100000 xxxxxxxx	BRA	label
01100001 xxxxxxxx	BSR	label
0000rrr1 11eeeeee	BSET	Dn,ea
00001000 11eeeeee	BSET	#d,ea
0000rrr1 00eeeeee	BTST	Dn,ea
00001000 00eeeeee	BTST	#d,ea
0100rrr1 10eeeeee	CHK	ea,Dn
01000010 11eeeeee	CLR	ea
1011rrr0 11eeeeee	CMP	ea,Dn
1011rrr0 11eeeeee	CMPA.W	ea,An
1011rrr1 11eeeeee	CMPA.L	ea,An
00001100 11eeeeee	CMPI	#d,ea
1011rrr1 11001rrr	CMPM	(An)+,(An)+
0101cccc 11001rrr	DBcc	Dn,label
1000rrr1 11eeeeee	DIVS	ea,Dn
1000rrr0 11eeeeee	DIVU	ea,Dn
1011rrr1 11eeeeee	EOR	Dn,ea
00001010 11eeeeee	EORI	#d,ea
1100rrr1 xx00xrrr	EXG	Rn,Rn
01001000 xx000rrr	EXT	Dn
1111xxxx xxxxxxxx	F-Emu	
01001010 11111100	ILLEGAL	
01001110 11eeeeee	JMP	ea
01001110 10eeeeee	JSR	ea
0100rrr1 11eeeeee	LEA	ea,An
01001110 01010rrr	LINK	An,#d
1110ddd1 11001rrr	LSL	#d,Dn
1110rrr1 11101rrr	LSL	Dn,Dn
11100011 11eeeeee	LSL	ea
1110ddd0 11001rrr	LSR	#d,Dn
1110rrr0 11101rrr	LSR	Dn,Dn
11100010 11eeeeee	LSR	ea
0001zzzz zzqqqqqq	MOVE.B	ea,ea
0011zzzz zzqqqqqq	MOVE.W	ea,ea
0010zzzz zzqqqqqq	MOVE.L	ea,ea
01000100 11eeeeee	MOVE	ea,CCR
01000110 11eeeeee	MOVE	ea,SR
01000000 11eeeeee	MOVE	SR,ea
01001110 01100rrr	MOVE	An,USP
01001110 01101rrr	MOVE	USP,An
0011zzz0 01qqqqqq	MOVEA.W	ea,An
0010zzz0 01qqqqqq	MOVEA.L	ea,An
01001000 1xeeeeee	MOVEM	list,ea
01001100 1xeeeeee	MOVEM	ea,list
0000rrr1 0x001rrr	MOVEP	d(An),Dn
0000rrr1 1x001rrr	MOVEP	Dn,d(An)
0111rrr0 dddddddd	MOVEQ	#d,Dn
1100rrr1 11eeeeee	MULS	ea,Dn
1100rrr0 11eeeeee	MULU	ea,Dn
01001000 00eeeeee	NBCD	ea
01000100 11eeeeee	NEG	ea
01000000 11eeeeee	NEGX	ea
01001110 01110001	NOP	
01000110 11eeeeee	NOT	ea
1000rrr0 11eeeeee	OR	ea,Dn
1000rrr1 11eeeeee	OR	Dn,ea
00000000 11eeeeee	ORI	#d,ea
01001000 01eeeeee	PEA	ea
01001110 01110000	RESET	
1110ddd1 11011rrr	ROL	#d,Dn
1110rrr1 11111rrr	ROL	Dn,Dn
11100111 11eeeeee	ROL	ea
1110ddd0 11011rrr	ROR	#d,Dn
1110rrr0 11111rrr	ROR	Dn,Dn
11100110 11eeeeee	ROR	ea
1110ddd1 11010rrr	ROXL	#d,Dn
1110rrr1 11110rrr	ROXL	Dn,Dn
11100101 11eeeeee	ROXL	ea
1110ddd0 11010rrr	ROXR	#d,Dn
1110rrr0 11110rrr	ROXR	Dn,Dn
11100100 11eeeeee	ROXR	ea
01001110 01110011	RTE	
01001110 01110111	RTR	
01001110 01110101	RTS	
1000rrr1 00000rrr	SBCD	Dn,Dn
1000rrr1 00001rrr	SBCD	-(An),-(An)
0101cccc 11eeeeee	Scc	ea
01001110 01110010	STOP	#d
1001rrr0 11eeeeee	SUB	ea,Dn
1001rrr1 11eeeeee	SUB	Dn,ea
1001rrr0 11eeeeee	SUBA.W	ea,An
1001rrr1 11eeeeee	SUBA.L	ea,An
00000100 11eeeeee	SUBI	#d,ea
0101ddd1 11eeeeee	SUBQ	#d,ea
1001rrr1 11000rrr	SUBX	Dn,Dn
1001rrr1 11001rrr	SUBX	-(An),-(An)
01001000 01000rrr	SWAP	Dn
01001010 11eeeeee	TAS	ea
01001110 0100vvvv	TRAP	#v
01001110 01110110	TRAPV	
01001010 11eeeeee	TST	ea
01001110 01011rrr	UNLK	An

```
Erklärungen
===========

rrr      Daten- oder Adreßregister

ll       Länge
00       Byte
01       Wort
10       Langwort

mmm      Modus
   rrr   Register                                Assembler-   Aufbau
eeeeee   Effektive Adresse                       Syntax     Erweiterungswort(e)
000nnn   Datenregister (DR) direkt               Dn
001nnn   Adreßregister (AR)direkt               An
010nnn   AR indirekt                            (An)
011nnn   AR indirekt mit Postinkrement          (An)+
100nnn   AR indirekt mit Predekrement           -(An)
101nnn   AR indirekt mit Adreßdistanz           d(An)        vddddddd dddddddd
110nnn   AR ind. m. Index u. Adreßdistanz       d(An,Xi.l)   xiiil000 vddddddd
111000   Absolute kurze Adresse                 dddd         dddddddd dddddddd
111001   Absolute lange Adresse                 dddddddd     ddddd... ...ddddd (32 Bits)
111010   Programmzähler mit Adreßdistanz        d(PC)        vddddddd dddddddd
111011   PC mit Index und Adreßdistanz          d(PC,Xi.l)   xiiil000 vddddddd
111100   Daten unmittelbar                      #dB          00000000 dddddddd bei .B
                                                #d16         dddddddd dddddddd bei .W
                                                #d32         ddddd... ...ddddd bei .L
                                                v            v= Vorzeichen
                                                    l=0 .W
                                                    l=1 .L
                                                x=0 Di
                                                x=1 Ai

         bei MOVE
zzzzzz   Effektive Adresse Ziel   (rrrmmm)
qqqqqq   Effektive Adresse Quelle (mmmrrr)

cccc     Bedingungskode
0000     T  wahr     (nicht bei Bcc)
0001     F  falsch   (nicht bei Bcc)
0010     HI höher
0011     LS niedriger/identisch
0100     CC Cv=0
0101     CS Cy=1
0110     NE ungleich
0111     EQ gleich
1000     VC Ov=0
1001     VS Ov=1
1010     PL positiv
1011     MI negativ
1100     GE größer/gleich
1101     LT kleiner
1110     GT größer
1111     LE kleiner/gleich

dd...d   Daten

x...x    unterschiedliche Bedeutung
```

BASIC-Programm zur Dekodierung von MC 68000-Befehlen

```
100 PRINT "<CLR><CD>"
110 BL$="                                   "
120 PRINT TAB(7)"<RVS>"BL$
130 PRINT TAB(7)"<RVS>  MC68000 OP-CODE DECODER  "
140 PRINT TAB(7)"<RVS>"BL$
150 PRINT : PRINT
160 GOSUB 3110
170 :
```

```
180 INPUT "<CD>OP-CODE  0000<CL><CL><CL><CL><CL><CL>";HX$
190 IF LEFT$(HX$,1)="*" THEN END
200 OP=DEC(HX$)
210 GOSUB 1930: GOSUB 290
220 PRINT "<CU>";LEFT$(HX$+BL$,12)+" "+MN$
230 IF LEN(HX$)>12 THEN PRINT MID$(HX$,13)
240 GOTO 180
250 :
260 :
270 REM  OPERANDEN
280 :
290 RX=(OP/512) AND 7: RX$=MID$(STR$(RX),2): RY=YY AND 7:
    RY$=MID$(STR$(RY),2)
300 SI=(OP/64) AND 3: MM=(OP/8) AND 7: MO=9
310 ON CD+1GOTO 380,410,520,580,610,640,700,730,760,790,820
320 ON CD-10GOTO 850,880,920,960,1000,1030,1090,1130,1160,1200
330 ON CD-20GOTO 1230,1260,1300,1350,1380,1410,1440,1470,1500,1530
340 ON CD-30GOTO 1570,1620,1650,1730,1800,1050
350 STOP : REM  PROGRAMM-FEHLER
360 :
370 REM  0 = IMPLIED
380 LL=2: RETURN
390 :
400 REM  1 = ORI,ANDI,SUBI,ADDI,EORI,CMPI
410 GOSUB 1860: MN$=MN$+"#$": GOSUB 2760
420 IF SI=0 THEN MN$=MN$+HEX$(Y)+",": LL=4
430 IF SI=1 THEN MN$=MN$+HEX$(X)+HEX$(Y)+",": LL=4
440 IF SI<>2 THEN 460
450 MN$=MN$+HEX$(X)+HEX$(Y): GOSUB 2780: MN$=MN$+HEX$(X)+HEX$(Y)+",":
    LL=6
460 IF (YY AND 63)<>60 THEN GOSUB 2090: GOTO 490
470 IF SI=0 THEN MN$=MN$+"CCR": GOTO 490
480 MN$=MN$+"SR"
490 RETURN
500 :
510 REM  2 = MOVEP
520 SI=1: IF (YY AND 64) THEN SI=2
530 GOSUB 1870: GOSUB 2760: Y$="$"+HEX$(X)+HEX$(Y)+"(A"+RY$+")":
    LL=LL+2
540 IF (YY AND 128) THEN MN$=MN$+"D"+RX$+","+Y$: RETURN
550 MN$=MN$+Y$+",D"+RX$: RETURN
560 :
570 REM  3 = BTST/BCHG/BCLR/BSET DN,EA
580 GOSUB 1870: MN$=MN$+"D"+RX$+",": SI=0: GOTO 2090
590 :
600 REM  4 = BTST/BCHG/BCLR/BSET #DN,EA
610 GOSUB 2760: GOSUB 1870: MN$=MN$+"#"+MID$(STR$(Y),2)+",": SI=0:
    GOTO 2090
620 :
630 REM  5 = MOVE.X/MOVEA.X
640 GOSUB 1870: SI=(OP/4096) AND 3: SI=SI%(SI)
650 MO=0: GOSUB 2090: MN$=MN$+",": REM  QUELLE
660 MO=9: MM=(OP/64) AND 7: IF MM=1 THEN MO=1
670 RY=(OP/512) AND 7: GOTO 2090: REM  ZIEL
680 :
690 REM  6 = STOP
700 LL=2: GOSUB 2760: LL=4: MN$=MN$+HEX$(X)+HEX$(Y): RETURN
710 :
720 REM  7 = MOVE EA,CCR
730 MO=8: GOSUB 1870: SI=1: GOSUB 2090: MN$=MN$+",CCR": RETURN
740 :
750 REM  8 = MOVE EA,SR
760 MO=8: GOSUB 1870: SI=1: GOSUB 2090: MN$=MN$+",SR": RETURN
770 :
780 REM  9 = MOVE SR,EA
790 LL=2: SI=1: GOTO 2090
800 :
```

```
 810 REM  10= MOVE USP,AN
 820 LL=2: MN$=MN$+RY$: RETURN
 830 :
 840 REM  11= MOVE AN,USP
 850 LL=2: MN$=MN$+RY$+",USP": RETURN
 860 :
 870 REM  12= MOVEM EA,REGLISTE
 880 MO=2: IF MM=3 THEN MO=0
 890 GOSUB 2860: GOSUB 2090: MN$=MN$+","+Z$: RETURN
 900 :
 910 REM  13= MOVEM REGLISTE,EA
 920 MO=3: IF MM=4 THEN MO=0
 930 GOSUB 2860: MN$=MN$+Z$+",": GOTO 2090
 940 :
 950 REM  14= MOVEQ
 960 LL=2: X=YY AND 255: X$=HEX$(X)
 970 MN$=MN$+X$+",D"+RX$: RETURN
 980 :
 990 REM  15= BBB.X EA
1000 GOSUB 1860: GOTO 2090
1010 :
1020 REM  16= BBB   EA
1030 MO=2: GOSUB 1870: GOTO 2090
1040 :
1050 REM  36= BBB   EA
1060 GOSUB 1870: GOTO 2090
1070 :
1080 REM  17= EXT
1090 X=1: IF SI=3 THEN X=2
1100 SI=X: GOSUB 1860: MN$=MN$+RY$: RETURN
1110 :
1120 REM  18= SWAP/UNLK
1130 MN$=MN$+RY$: LL=2: RETURN
1140 :
1150 REM  19= LINK
1160 MN$=MN$+RY$: LL=2: GOSUB 2760: LL=4
1170 MN$=MN$+"$"+HEX$(X)+HEX$(Y): RETURN
1180 :
1190 REM  20= TRAP
1200 MN$=MN$+MID$(STR$(YY AND 15),2): LL=2: RETURN
1210 :
1220 REM  21= BCC
1230 X=(OP/256) AND 15: MN$=MN$+CC$(X)
1240 :
1250 REM  22= BSR/BRA
1260 GOSUB 1870: X=YY AND 255: IF X=0 THEN GOSUB 2760: X=X*256+Y:
     LL=LL+2
1270 MN$=MN$+"$"+HEX$(X): RETURN
1280 :
1290 REM  23= DBCC
1300 X=(OP/256) AND 15: MN$=MN$+CC$(X): GOSUB 1870
1310 MN$=MN$+"D"+RY$+",": GOSUB 2760: LL=LL+2
1320 MN$=MN$+"$"+HEX$(X)+HEX$(Y): RETURN
1330 :
1340 REM  24= SCC
1350 X=(OP/256) AND 15: MN$=MN$+CC$(X): GOSUB 1870: GOTO 2090
1360 :
1370 REM  25= ADDQ/SUBQ
1380 MO=1: GOSUB 1860: MN$=MN$+"#"+RX$+",": GOTO 2090
1390 :
1400 REM  26= ADBC/SBCD DY,DX
1410 MN$=MN$+"D"+RY$+",D"+RX$: LL=2: RETURN
1420 :
1430 REM  27= ADBC/SBCD -(AY),-(AX)
1440 MN$=MN$+"-(A"+RY$+"),-(A"+RX$+")": LL=2: RETURN
1450 :
1460 REM  28= CHK/DIV/MUL
1470 MO=8: GOSUB 1870: SI=1: GOSUB 2090: MN$=MN$+",D"+RX$: RETURN
1480 :
```

```
1490 REM  29= LEA
1500 GOSUB 1870: SI=2: GOSUB 2090: MN$=MN$+",A"+RX$: RETURN
1510 :
1520 REM  30= ADDA/SUBA/CMPA
1530 MO=0: SI=(OP/128) AND 3: IF SI=3 THEN SI=2
1540 GOSUB 1870: GOSUB 2090: MN$=MN$+",A"+RX$: RETURN
1550 :
1560 REM  31= ADDX,SUBX
1570 IF MM=0 THEN X$="D"+RY$+",D"+RX$
1580 IF MM=1 THEN X$="-(A"+RY$+"),-(A"+RX$+")"
1590 GOSUB 1860: MN$=MN$+X$: RETURN
1600 :
1610 REM  32= CMPM
1620 GOSUB 1860: MN$=MN$+"(A"+RY$+")+,(A"+RX$+")+": RETURN
1630 :
1640 REM  33= ADD/AND/CMP/EOR/OR/SUB
1650 X=(OP/256) AND 1: Y=(OP/4096) AND 3: Z=YY AND 256
1660 GOSUB 1860: Z$="D"+RX$
1670 IF X=0 THEN MO=0: IF Y=0 THEN MO=8
1680 IF X=1 THEN MO=5: IF Y=3 THEN MO=9
1690 IF Z=0 THEN GOSUB 2090: MN$=MN$+","+Z$: RETURN
1700 MN$=MN$+Z$+",": GOTO 2090
1710 :
1720 REM  34= EXG
1730 GOSUB 1870: X=(OP/8) AND 31
1740 IF X=8 THEN MN$=MN$+"D"+RX$+",D"+RY$
1750 IF X=9 THEN MN$=MN$+"A"+RX$+",A"+RY$
1760 IF X=17 THEN MN$=MN$+"D"+RX$+",A"+RY$
1770 RETURN
1780 :
1790 REM  35= ASX/LSX/ROX/ROXX
1800 MO=5: GOSUB 1860: MM=(MM/4) AND 1
1810 Y$=",D"+RY$: X=RX-8*(RX=0)
1820 IF MM=1 THEN MN$=MN$+"D"+RX$+Y$: RETURN
1830 MN$=MN$+"#"+MID$(STR$(X),2)+Y$: RETURN
1840 :
1850 :
1860 MN$=MN$+LL$(SI)
1870 MN$=LEFT$(MN$+BL$,8): LL=2
1880 RETURN
1890 :
1900 :
1910 REM  BEFEHL UND GRUPPE ERMITTELN
1920 :
1930 P=INT(OP/8192): II=35
1940 FOR N=V(P) TO B(P) STEP -1
1950 YY=OP: IF OP>32767 THEN YY=OP-65536
1960 X=YY AND MX%(N)
1970 IF MA%(N)=X THEN II=N: N=-1
1980 NEXT N
1990 CD=CD%(II): MN$=MN$(II)
2000 RETURN
2010 :
2020 :
2030 REM  EFFEKTIVE ADRESSE
2040 REM      MO=8 DATEN
2050 REM      MO=4 SPEICHER
2060 REM      MO=2 STEUERUNG
2070 REM      MO=1 AENDERBAR
2080 :
2090 RY$=MID$(STR$(RY),2): '''' PRINT: PRINT"MO/MM/RY";MO;MM;RY: PRINT
2100 ON MM+1GOTO 2140,2180,2220,2250,2290,2330,2360,2430
2110 STOP : REM  PROGRAMM-FEHLER
2120 :
2130 REM  EA=DN
2140 IF (MO AND 6)<>0 THEN 2700
2150 MN$=MN$+"D"+RY$: RETURN
2160 :
```

```
2170 REM  EA=AN
2180 IF (MO AND 14)<>0 THEN 2700
2190 MN$=MN$+"A"+RY$: RETURN
2200 :
2210 REM  EA=(AN)
2220 MN$=MN$+"(A"+RY$+")": RETURN
2230 :
2240 REM  EA=(AN)+
2250 IF (MO AND 2)<>0 THEN 2700
2260 MN$=MN$+"(A"+RY$+")+": RETURN
2270 :
2280 REM  EA=-(AN)
2290 IF (MO AND 2)<>0 THEN 2700
2300 MN$=MN$+"-(A"+RY$+")": RETURN
2310 :
2320 REM  EA=XXXX(AN)
2330 GOSUB 2760: MN$=MN$+"$"+HEX$(X)+HEX$(Y)+"(A"+RY$+")": LL=LL+2:
     RETURN
2340 :
2350 REM  EA=XX(AN,XI)
2360 GOSUB 2760: X$=MID$(STR$(Y),2): IF Y>127 THEN X$=STR$(Y-256)
2370 LL=LL+2: MN$=MN$+X$+"(A"+RY$+",": X$="D": IF (X AND 128) THEN
     X$="A"
2380 Y=(X/8) AND 1
2390 Z=(X/16) AND 7: X$=X$+MID$(STR$(Z),2)
2400 IF Y=0 THEN MN$=MN$+X$+".W)": RETURN
2410 MN$=MN$+X$+".L)": RETURN
2420 :
2430 ON RY+1GOTO 2470,2490,2540,2590,2670
2440 MN$=MN$+"???": RETURN
2450 :
2460 REM  EA = NEXT WORD
2470 GOSUB 2760: MN$=MN$+"$"+HEX$(X)+HEX$(Y): LL=LL+2: RETURN
2480 :
2490 REM  EA = NEXT 2 WORDS
2500 GOSUB 2760: X$="$"+HEX$(X)+HEX$(Y): GOSUB 2780:
     X$=X$+HEX$(X)+HEX$(Y)
2510 MN$=MN$+X$: LL=LL+4: RETURN
2520 :
2530 REM  EA = (PC)+D
2540 IF (MO AND 1)<>0 THEN 2700
2550 GOSUB 2760: X$="$"+HEX$(X)+HEX$(Y)
2560 MN$=MN$+X$+"(PC)": LL=LL+2: RETURN
2570 :
2580 REM  EA = (PC)+(RN)+D
2590 IF (MO AND 1)<>0 THEN 2700
2600 GOSUB 2760: LL=LL+2: MN$=MN$+"$"+HEX$(Y)+"(PC,"
2610 X$="D": IF (X AND 128) THEN X$="A"
2620 Y=(X/16) AND 7: X$=X$+MID$(STR$(Y),2)
2630 Y$=".W)": IF (X AND 8) THEN Y$=".L)"
2640 MN$=MN$+X$+Y$: RETURN
2650 :
2660 REM  EA = (PC)+2
2670 IF (MO AND 3)<>0 THEN 2700
2680 GOSUB 2760: X$=HEX$(X): Y$=HEX$(Y): MN$=MN$+"#$"
2690 ON SI+1GOTO 2710,2720,2730
2700 MN$=MN$+"???": RETURN
2710 MN$=MN$+Y$: LL=LL+2: RETURN
2720 MN$=MN$+X$+Y$: LL=LL+2: RETURN
2730 GOSUB 2780: MN$=MN$+X$+Y$+HEX$(X)+HEX$(Y): LL=LL+4: RETURN
2740 :
2750 :
2760 X=LL/2: GOTO 2790
2770 :
2780 X=(LL/2)+1
2790 PRINT "<CU>";STR$(X);: INPUT ".ERWEITERUNGSWORT  0000<CL><CL><CL>
     <CL><CL><CL>";XX$
2800 Y=DEC(XX$): X=INT(Y/256): Y=Y-X*256
2810 HX$=HX$+XX$: RETURN
```

```
2820 :
2830 :
2840 REM   REGLISTE FUER MOVEM
2850 :
2860 SI=(SI AND 1)+1: GOSUB 1860: Z$=""
2870 GOSUB 2760: LL=4: IF MM=4 THEN 2980
2880 X$="01234567"
2890 FOR Z=7 TO 0 STEP -1
2900 U=X AND (2^Z): IF U<>0 THEN Z$=Z$+"/A"+MID$(X$,Z+1,1)
2910 NEXT
2920 FOR Z=7 TO 0 STEP -1
2930 U=Y AND (2^Z): IF U<>0 THEN Z$=Z$+"/D"+MID$(X$,Z+1,1)
2940 NEXT
2950 Z$=MID$(Z$,2)
2960 RETURN
2970 :
2980 X$="76543210"
2990 FOR Z=7 TO 0 STEP -1
3000 U=X AND (2^Z): IF U<>0 THEN Z$=Z$+"/D"+MID$(X$,Z+1,1)
3010 NEXT
3020 FOR Z=7 TO 0 STEP -1
3030 U=Y AND (2^Z): IF U<>0 THEN Z$=Z$+"/A"+MID$(X$,Z+1,1)
3040 NEXT
3050 Z$=MID$(Z$,2)
3060 RETURN
3070 :
3080 :
3090 REM  INITIALISIERUNG
3100 :
3110 DIM MA%(101),MX%(101),CD%(101),MN$(101),V(7),B(7),CC$(15),LL$(3),
     SI%(3)
3120 Z=32768
3130 FOR I=0 TO 101
3140 READ X$,Y$,CD%(I),MN$(I)
3150 X=DEC(X$): IF X=>Z THEN X=X-Z
3160 Y=DEC(Y$): IF Y=>Z THEN Y=Y-Z
3170 MA%(I)=X: MX%(I)=Y
3180 NEXT
3190 FOR I=0 TO 7: READ B(I),V(I): NEXT
3200 FOR I=0 TO 15: READ CC$(I): NEXT
3210 SI%(1)=0: SI%(2)=2: SI%(3)=1
3220 LL$(0)=".B": LL$(1)=".W": LL$(2)=".L": LL$(3)=".?"
3230 RETURN
3240 :
3250 :
3260 REM   DATAS
3270 :
3280 DATA  0000,FF00, 1,"ORI"
3290 DATA  0100,F1C0, 3,"BTST"
3300 DATA  0108,F138, 2,"MOVEP"
3310 DATA  0140,F1C0, 3,"BCHG"
3320 DATA  0180,F1C0, 3,"BCLR"
3330 DATA  01C0,F1C0, 3,"BSET"
3340 DATA  0200,FF00, 1,"ANDI"
3350 DATA  0400,FF00, 1,"SUBI"
3360 DATA  0600,FF00, 1,"ADDI"
3370 DATA  0800,FFC0, 4,"BTST"
3380 DATA  0840,FFC0, 4,"BCHG"
3390 DATA  0880,FFC0, 4,"BCLR"
3400 DATA  08C0,FFC0, 4,"BSET"
3410 DATA  0A00,FF00, 1,"EORI"
3420 DATA  0C00,FF00, 1,"CMPI"
3430 DATA  1000,F000, 5,"MOVE.B"
3440 DATA  2000,F000, 5,"MOVE.L"
3450 DATA  2040,F1C0, 5,"MOVEA.L"
3460 DATA  3000,F000, 5,"MOVE.W"
3470 DATA  3040,F1C0, 5,"MOVEA.W"
3480 DATA  4000,FF00,15,"NEGX"
```

```
3490 DATA  40C0,FFC0, 9,"MOVE    SR,"
3500 DATA  4180,F1C0,28,"CHK"
3510 DATA  41C0,F1C0,29,"LEA"
3520 DATA  4200,FF00,15,"CLR"
3530 DATA  4400,FF00,15,"NEG"
3540 DATA  44C0,FFC0, 7,"MOVE"
3550 DATA  4600,FF00,15,"NOT"
3560 DATA  46C0,FFC0, 8,"MOVE"
3570 DATA  4800,FF38,17,"EXT"
3580 DATA  4800,FFC0,36,"NBCD"
3590 DATA  4840,FFC0,16,"PEA"
3600 DATA  4840,FFF8,18,"SWAP    D"
3610 DATA  4880,FF80,13,"MOVEM"
3620 DATA  4A00,FF00,15,"TST"
3630 DATA  4AC0,FFC0,36,"TAS"
3640 DATA  4AFC,FFFF, 0,"ILLEGAL"
3650 DATA  4C80,FF80,12,"MOVEM"
3660 DATA  4E40,FFF0,20,"TRAP    #"
3670 DATA  4E50,FFF8,19,"LINK    A"
3680 DATA  4E58,FFF8,18,"UNLK    A"
3690 DATA  4E60,FFF8,11,"MOVE    A"
3700 DATA  4E68,FFF8,10,"MOVE    USP,A"
3710 DATA  4E70,FFFF, 0,"RESET"
3720 DATA  4E71,FFFF, 0,"NOP"
3730 DATA  4E72,FFFF, 6,"STOP    #"

3740 DATA  4E73,FFFF, 0,"RTE"
3750 DATA  4E75,FFFF, 0,"RTS"
3760 DATA  4E76,FFFF, 0,"TRAPV"
3770 DATA  4E77,FFFF, 0,"RTR"
3780 DATA  4E80,FFC0,16,"JSR"
3790 DATA  4EC0,FFC0,16,"JMP"
3800 DATA  5000,F100,25,"ADDQ"
3810 DATA  5100,F100,25,"SUBQ"
3820 DATA  50C0,F0C0,24,"S"
3830 DATA  50C8,F0F8,23,"DB"
3840 DATA  6000,F000,21,"B"
3850 DATA  6000,FF00,22,"BRA"
3860 DATA  6100,FF00,22,"BSR"
3870 DATA  7000,F100,14,"MOVEQ   #"
3880 DATA  8000,F000,33,"OR"
3890 DATA  80C0,F1C0,28,"DIVU"
3900 DATA  8100,F1F8,26,"SBCD"
3910 DATA  8108,F1F8,27,"SBCD"
3920 DATA  81C0,F1C0,28,"DIVS"
3930 DATA  9000,F000,33,"SUB"
3940 DATA  90C0,F1C0,30,"SUBA.W"

3950 DATA  9100,F130,31,"SUBX"
3960 DATA  91C0,F1C0,30,"SUBA.L"
3970 DATA  A000,F000, 0,"A-EMU"
3980 DATA  B000,F100,33,"CMP"
3990 DATA  B0C0,F1C0,30,"CMPA.W"
4000 DATA  B100,F100,33,"EOR"
4010 DATA  B108,F138,32,"CMPM"
4020 DATA  B1C0,F1C0,29,"CMPA.L"
4030 DATA  C000,F000,33,"AND"
4040 DATA  C0C0,F1C0,28,"MULU"
4050 DATA  C100,F130,34,"EXG"
4060 DATA  C100,F1F8,26,"ABCD"
4070 DATA  C108,F1F8,27,"ABCD"
4080 DATA  C1C0,F1C0,28,"MULS"
4090 DATA  D000,F000,33,"ADD"
4100 DATA  D0C0,F1C0,30,"ADDA.W"
4110 DATA  D100,F130,31,"ADDX"
4120 DATA  D1C0,F1C0,30,"ADDA.L"
4130 DATA  E000,F118,35,"ASR"
4140 DATA  E008,F118,35,"LSR"
4150 DATA  E010,F118,35,"ROXR"
4160 DATA  E018,F118,35,"ROR"

4170 DATA  E0C0,FFC0,16,"ASR"
4180 DATA  E100,F118,35,"ASL"
4190 DATA  E108,F118,35,"LSL"
4200 DATA  E110,F118,35,"ROXL"
4210 DATA  E118,F118,35,"ROL"
4220 DATA  E1C0,FFC0,16,"ASL"
4230 DATA  E2C0,FFC0,16,"LSR"
4240 DATA  E3C0,FFC0,16,"LSL"
4250 DATA  E4C0,FFC0,16,"ROXR"
4260 DATA  E5C0,FFC0,16,"ROXL"
4270 DATA  E6C0,FFC0,16,"ROR"
4280 DATA  E7C0,FFC0,16,"ROL"
4290 DATA  F000,F000, 0,"F-EMU"
4300 :
4310 DATA  0,15,16,19,20,55,56,59,60,68,69,74,75,84,85,101
4320 :
4330 DATA  "T ","F ",HI,LS,CC,CS,NE,EQ
4340 DATA  VC,VS,PL,MI,GE,LT,GT,LE
```

Roland Löhr

Viele nützliche Routinen

Der Autor hat sich seit einiger Zeit mit dem grundsätzlich neuen Aufbau eines mehr strukturierten Assemblers befaßt, zu dem viele Routinen für die Ein-und Ausgabe und solche für die Bewertung gehören. Im Ergebnis sind etliche mitzuteilende Programm-Module entstanden, die hier dargelegt werden. Man kann sie in Anwenderprogramme einbinden oder auch ihre Grundgedanken in Systemen verwenden, die auf sich allein gestellt (stand alone) solche Dienste benötigen.

Ein- und Ausgabe

In jedem Computersystem finden wir im Betriebssystem Programmabschnitte für die E/A. Oft sind sie mit anderen Abschnitten verwoben, so daß sie nicht isoliert verwendet werden können. Es soll das Ziel sein, möglichst einfache Module zu besitzen, auf denen höhere Dienstleistungen aufgebaut werden können. Beginnen wir mit der Textausgabe: Eine größere Zahl von ASCII-Strings für die Textausgabe befinde sich irgendwo im Speicher. Das ist eine typische Situation in Systemen, die sich um eine interaktive Benutzerführung mit größeren Textausgaben bemühen. Die Menge des Textes läuft schnell über 256 Bytes hinaus. Es bleibt damit die Frage, wie man gleichwohl Textabschnitte gezielt zur Ausgabe aufrufen kann, möglichst in einer Pointeradressierung in Stile LDA (Pointer),y und anschließend JSR OUTPUT. Im Laufe der Zeit sind hierzu einige Vorschläge in dieser Zeitschrift veröffentlicht worden. Bei weiterem Nachdenken stellt sich die Überlegung ein, daß es sich bei solchen Ausgaben um keine zeitkritische Angelegenheit handelt, so daß man in der Programmierung etwas großzügiger verfahren kann. Die Routine TEXTOUT in Bild 1 macht von von einer Tafel Gebrauch, die die mit einem Label versehenen Textanfänge bezeichnet. Sie wird als MESTAF bezeichnet. Im Akku wird eine hexadezimale Nummer übergeben, welcher Text aufgerufen wird. Die Routine TEXTOUT multipliziert diese Nummer zunächst mit 2 und lädt den Zeiger auf den Textbeginn nach MESPTR. Das ist ein Pointer in der Zero Page, der mit LDA (MESPTR),Y benutzt werden kann.

Mit dieser Grundidee kann man bis zu 128 Texte gezielt aufrufen, was in den meisten Fällen ausreichen dürfte.

```
65XXX-ASSEMBLER COPYRIGHT 1985 BY ROLAND LOEHR
DURCHGANG MIT ALTEN PARAMETERN? J/CR=N
J
1. DURCHGANG


2. DURCHGANG
0001 000000            START     =$5000

0171 00503A            START1
0173 00503A                      .OPT LIS
0174                             ;$KO"PROG-BEISPIELE"
0175                             ;AUSGABE EINES TEXTES, DIE TEXT-NUMMER MUSS IM AK
KU SEIN
0176                             ;TEXTBEGRENZUNG DURCH BYTE 00 ODER DURCH BIT7=1
0177                             ;IM LETZTEN ZEICHEN
0178                             ;FERNER DURCH EIN CR
0179
0180 00503A 0A         TEXTOUT   ASL A                  A ;MULTIPLIKATION NUMMER IM A
                                 KKU MIT 2
0181 00503B A8                   TAY
0182 00503C B97050               LDA MESTAF,Y            ;POINTER ZUR MESSAGE ZURECHTL
                                 EGEN
0183 00503F 8566                 STA MESPTR
0184 005041 B97150               LDA MESTAF+1,Y
0185 005044 8567                 STA MESPTR+1            ;MESSAGE-POINTER
0186 005046 A000                 LDY #0
```

```
0187 00504B B166      TXT1      LDA (MESPTR),Y      ;ZEICHEN HOLEN
0188 00504A F00E                BEQ CRLF            ;WENN TEXT MIT 00 BEGRENZT IS
                                T
0189 00504C C90D                CMP #CR             ;BEGRENZUNG MIT CARRIAGE RETU
                                RN
0190 00504E F00A                BEQ CRLF            ;JA
0191 005050 48                  PHA
0192 005051 297F                AND #$7F            ;REINES ASCII HERSTELLEN
0193 005053 201F50              JSR OUTALL          ;ZEICHENAUSGABE
0194 005056 C8                  INY                 ;POSITIONIERUNG AUF FOLGENDES
                                 BYTE
0195 005057 68                  PLA                 ;PRUEFUNG BIT7 GESETZT ALS TE
                                XTBEGRENZER?
0196 005058 10EE                BPL TXT1            ;NEIN, WEITER AUSGEBEN
0197 00505A A90D      CRLF      LDA #CR
0198 00505C 201F50              JSR OUTALL
0199 00505F 60                  RTS
0200
0201                            ;TEXTAUSGABEN, BEISPIELE MIT VERKUERZTEM LISTENUM
FANG
0202 005060 36355858 MES0       .BYT '65XXX-ASSEMBLER COPYRIGHT 1985 BY ROLAND L
                                OEHR'$
0203 005068 53594D42 MES1       .BYT 'SYMBOLTAFEL ZU KLEIN'$
0204                            ;USW.
0205
0206                            ;WORTADRESSEN DER TEXTBEGINNE
0207 005070           MESTAF                        ;MESSAGE-TAFEL
0208 005070 60506850            .WOR MES0,MES1      ;USW
0209
```

Die Textausgabe ist somit auf den Anfang eines Ausgabestrings positioniert. Das Indexregister Y wird bei der vorgeschlagenen Art der Adressierung auf den String laufend erhöht. Es bleibt die Frage, wann die Textausgabe abzubrechen ist. Hier haben sich nun im wesentlichen drei verschiedene Trennzeichen zum Abbruch eingebürgert. Manche Programme verwenden das Byte 00 als Endmarke, was einen gewissen Mehraufwand der Abspeicherung bedingt. In anderen Fällen ist das CARRIAGE RETURN CR das Abbruchkriterium. Diesen Trenner muß man für viele Fälle als 'natürlich' bezeichnen, weil viele Textausgaben mit einem Vorschub in eine neue Zeile enden sollen. Und drittens wird speichersparend im letzten Byte das Bit 7 zu '1' gesetzt, um das Ende eines Strings zum erkennen. Das hat zur Erkennung des 'negativen' Vorzeichens in der Programmierung einen kleinen Mehraufwand. - Die in Bild 1 enthaltenene Routine TEXTOUT erkennt alle drei Abbruchkriterien und darf damit für den allgemeinen Gebrauch empfohlen werden.

Im Zuge der Zeilenausgabe von Text ist es für den Programmierer bequem, wenn er einfach schreiben kann: JSR CRLF, Ausgabe eines CR und eines Linefeed. Diese Routine wird normalerweise im Betriebsssystem vorgehalten, so daß man sie mit einer einfachen symbolischen Erklärung benutzen kann. Bei ihr verhält es sich so, wie bei anderen Standardausgaben, die man für ein Zielsystem selber programmieren muß: Man sollte sie einmalig anlegen, um sie dann mit symbolischen Namen immer wieder benutzen zu können. Übliche Ausgaben dieser Art sind solche für Zwischenräume und Tabulierungen, auch solche für das Erzeugen eines Prompts. Man halte sich aber vor Augen, daß je nach der Umgebung der Peripherie manchmal nur ein CR ausgegeben werden muß, manchmal aber auch ein zusätzliches Linefeed. Man kann das leicht mit einem Zusatzparameter vor dem Aufruf von TEXTOUT steuern, der im Speicher oder in einem Register übergeben wird. In unserem Beispiel ist CRLF einfach der letzte Teil von TEXTOUT.

In der beifolgenden Dokumentation der Programmabschnitte bezeichnet OUTALL die Ausgabe auf die jeweils aktive Ausgabe per Treiberprogramm.

Zur Ausgabe von Text gehören meistens auch Routinen, die gespeicherte binäre Zahlen in einen ASCII-String verwandeln. Sie gehören nicht zur eigentlichen Textausgabe und werden daher im Zusammenhang der Umwandlungen betrachtet.

Die drei Haupt-Ablaufsblöcke der Datenverarbeitung sind Eingabe, Verarbeitung und Ausgabe. In diesem Abschnitt sind wir speziell erst auf die Textausgabe eingegangen. Jetzt also weiter zur Eingabe: Eingaben erfolgen heutzutage als ASCII-Zeichen, die entweder von einem Tastaturdekoder bereits parallel mit Strobe-Impuls angeliefert oder die in einem seriellen Interfacebaustein z.B. vom Typ ACIA/UART empfangen werden. Bei vielen Computern legt die Zentraleinheit über ein Interface auch selbst ein Netz, eine Matrix aus, um festzustellen, an welchem Kreuzungspunkt eine Taste gedrückt wurde. Aus dem Kreuzungspunkt wird dann per Tabelle ein ASCII-Zeichen gebildet. Die Dinge sind also hardware-abhängig. Gleichwohl kann man sagen, daß es überall eine Routine GETKEY geben muß, die ein Zeichen von der Tastatur holt, das dann im Akku als bereits geformtes ASCII-Zeichen übergeben wird. Ebenso ist auch die Abfrage 'any key', ob irgendeine Taste betätigt wurde, von der Umgebung abhängig.

In den Programmbeipielen finden wir die Routine GETSTRING. Wenn wir GETKEY als ein primitives Unterprogramm ansehen, dann ist GETSTRING eine Ebene höher. Es werden hier Zeichen in einem Eingabepuffer, der beliebig eingerichtet werden kann, zu einem String zusammengesetzt. Diese Routine wird sofort verlassen, wenn das Zeichen ESCAPE empfangen wurde. Im Beispiel erfolgt bei diesem Zeichen die Rückkehr in das übergeordnete Betriebsprogramm. In den übrigen Fällen wird die Zeichenkette im Eingabepuffer aufgebaut. Das empfangene Zeichen RUBOUT, auch DELETE genannt, führt zu einem Abbau des Zeigers in den Eingabepuffer. Dieser ist mit INBUFP bezeichnet. Der Zeilenabschluß erfolgt mit dem empfangenen Zeichen CR. Es führt dazu, daß im Eingabepuffer eine '00' an den bisher empfangenen String als Begrenzer angefügt wird.

Dienstleistungen wie TEXTOUT oder GETSTRING gehören findet man eigentlich auf allen aktuellen Computersystemen, nach dem Studium des Autors jedoch seltener in einer glatten linearen Form. Meistens sind da noch andere Dinge eingewoben. Insofern bleibt anzuregen, die hier gegebenen Vorschläge vielleicht noch weiter zu vereinfachen oder auf eine allgemeinere Ebene zu bringen.

Nachdem ein String empfangen worden ist, stellt sich die Frage, was weiter mit ihm geschehen soll. Das hängt natürlich von der Aufgabenstellung ab. Wenn es sich um ein Textverarbeitungsprogramm handelt, dann wird man den eingegebenen String an die beabsichtigte Stelle transportieren und auf neue Eingaben warten. Abbruchkriterium für die laufende Texteingabe könnte dann das Zeichen ESCAPE sein oder eine empfangene Leerzeile (nur CR). Man kehrt dann in die Befehlsebene zurück.

Oft soll die als String vorliegende Eingabe unmittelbar geprüft oder umgesetzt werden. Wenn es sich um Ketten von mehr als einem Zeichen handelt, die in irgeneiner Weise als Befehle dienen sollen, so wird man sich eines Parsers bedienen, um sie zu entschlüsseln. In anderen Fällen wird man sie auf zulässige Zeilen filtern oder als Strings vorliegende Zahlen nach binär umwandeln wollen. Routinen für diesen Zweck finden sich in den nachstehenden Abschnitten.

```
                                ;EINGABE EINES STRINGS NACH INBUFF
0211                            ;LESEN MIT RUBOUT, ZEICHENAUSGABE IM ECHO
0212                            ;WEITERVEARBEITUNG FUER Z.B. FILENAME ODER ADRESS
EN
0213 005074 A200      GETSTRING LDX #0
0214 005076 8684                STX STRLEN          ;SCHUTZ VOR VERAENDERUNG DURC
                                H OUTALL
0215 005078 202E50    GETST1    JSR GETKEY         ;HOLE ZEICHEN VON TASTATUR
0216 00507B C91B                CMP #ESCAPE        ;ABBRUCH?
0217 00507D F028                BEQ GETST4         ;JA, ZURUECK ZUM MONITOR!
```

```
0218 00507F 48              PHA
0219 005080 201F50          JSR OUTALL          ;ECHO DES ZEICHENS
0220 005083 68              PLA
0221 005084 C90D            CMP #CR             ;ZEILENABSCHLUSS?
0222 005086 F016            BEQ GETST3          ;JA
0223 005088 C97F            CMP #RUBOUT
0224 00508A F009            BEQ GETST2
0225 00508C A684            LDX STRLEN
0226 00508E 9500            STA INBUFF,X        ;IN DEN EINGABEPUFFER
0227 005090 E684            INC STRLEN          ;ZAHL GUELTIGER ZEICHEN
0228                        ;GGFS. PRUEFEN, OB BUFFER VOLL
0229 005092 4C7850          JMP GETST1          ;NAECHSTES ZEICHEN
0230 005095 A684    GETST2  LDX STRLEN          ;RUBOUT BEARBEITEN
0231 005097 F0DF            BEQ GETST1          ;PUFFER NOCH LEER
0232 005099 C684            DEC STRLEN          ;ZAHL GUELTIGER ZEICHEN
0233 00509B 4C7850          JMP GETST1
0234 00509E A900    GETST3  LDA #0              ;BEGRENZER IN DEN PUFFER
0235 0050A0 A684            LDX STRLEN
0236 0050A2 9500            STA INBUFF,X
0237 0050A4 A500            LDA INBUFF          ;1. ZCH. ZUR KONTROLLE LADEN,
                             OB LEER
0238 0050A6 60              RTS                 ;ENDE DER EINGABE
0239
0240 0050A7 68      GETST4  PLA                 ;ABBBRUCH BEI ESCAPE
0241 0050A8 68              PLA
0242 0050A9 4C3450          JMP MONITR
```

Filterung von Zeichen

Für die folgenden Routinen gilt, daß das Carry-Flag zu '1' gesetzt wird, wenn das gesuchte Merkmal zutrifft, wahr ist. Die Routine ALPHA? prüft, ob ein Zeichen ein Buchstabe ist, der als Groß- oder Kleinbuchstabe vorliegen darf. Im vorliegenden Beispiel wird auch der Unterstreichungsstich zu Alpha gerechnet. In anderen Fällen wird man auch die deutschen Sonderzeichen je nach ihrer Codierung zu den Buchstaben zählen.

Die Routine ALFNUM? filtert Zeichen auf ihre Eigenschaft als Buchstabe oder Dezimalziffer. Diese Routine akzeptiert auch ein Fragezeichen als zugehörig, um z.B. Bezeichner folgender Schreibweise zuzulassen: IS_ASCII?

Mit DRUCKBAR? ist ein Filter gegeben, ob ein ASCII-Zeichen zur Gruppe der druckbaren Zeichen gehört. Analog filtern die Routinen BINZIF? auf die zugelassenen Zeichen 0 und 1, OCTZIF? auf 0-7, DEZZIF? auf 0-9 sowie HEXZIF auf 0-9 und A-F der jeweiligen Zahlensysteme. Je nach Bedarf kann man ähnliche Filter formulieren. Zu den Filtern ist noch zu bemerken, daß sie das im Akku übergebene Zeichen nicht verändern. Wenn ein Zahlenzeichen mit einem solchen Filter bearbeitet worden ist, kann es unmittelbar in eine Dezimalzahl oder Hexadezimalzahl konvertiert werden. Dazu der nächste Abschnitt.

```
0293
0294                        ;UNTERPROGRAMME ZUM FILTERN
0295                        ;VEREINBARUNG: CARRY SET WHEN TRUE, ELSE CLEAR
0296
0297 0050F6         ALFNUM?                     ;ERKENNE DEZIMALE ZIFFERN UND
                             BUCHSTABEN
0298 0050F6 C93F            CMP #'?'            ;ALS FOLGEZEICHEN AUCH ZUGELA
                            SSEN
0299 0050F8 F01E            BEQ TRUE
0300 0050FA C930            CMP #'0'            ;KLEINER 0?
0301 0050FC 900D            BCC FALSE1          ;STEUERZEICHEN
0302 0050FE C93A            CMP #':'            ;ZEICHEN FOLGT AUF '9'?
0303 005100 9016            BCC TRUE            ;NEIN, NOCH ZIFFER
0304
0305 005102         ALPHA?                      ;FILTER FUER BUCHSTABEN UND U
                            NDERLINE-CHAR
0306                        ;GROSS- UND KLEINBUCHSTABEN WERDEN ERKANNT
0307                        ;ZIFFERN UND '?' WERDEN AUSGESCHIEDEN
0308 005102 C95F            CMP #'_'            ;UNDERLINE? IST ZUGELASSEN
0309 005104 F012            BEQ TRUE            ;CARRY SET
0310 005106 C97B            CMP #$7B            ;GROESSER ALS KLEINES 'Z'?
```

```
0311 005108 9002                BCC ALPHA3              ;NEIN
0312 00510A 18          FALSE   CLC                     ;NOT A LETTER
0313 00510B 60          FALSE1  RTS
0314 00510C C941        ALPHA3  CMP #$41                ;KLEINER 'A'?
0315 00510E 90FB                BCC FALSE1
0316 005110 C95B                CMP #$5B                ;GROSSES Z +1?
0317 005112 9004                BCC TRUE                ;GROSSBUCHSTABE
0318                            ;NOCH SIND $5B BIS $7A IM TOPF
0319 005114 C961                CMP #$61                ;KLEINES 'A'
0320 005116 90F3                BCC FALSE1              ;SONDERZEICHEN $5B-$60
0321 005118 38          TRUE    SEC                     ;SET TRUE
0322 005119 60                  RTS
0323
0324 00511A             DRUCKBAR?                       ;PRUEFE HEXBYTE, OB ALS ASCII
                                 DRUCKBAR
0325 00511A C920                CMP #' '                ;KLEINER SPACE?
0326 00511C B001                BCS DRBS                ;EVTL. JA
0327 00511E 60          NDRBAR  RTS                     ;CARRY CLEAR, NEIN
0328 00511F C97F        DRBS    CMP #$7F                ;RUBOUT?
0329 005121 B002                BCS DRBC
0330 005123 38                  SEC                     ;DRUCKBAR
0331 005124 60                  RTS                     ;AKKU NICHT VERAENDERT DURCH
                                DRUCKB
0332 005125 18          DRBC    CLC                     ;CARRY UMKEHREN
0333 005126 60                  RTS                     ;CARRY CLEAR
0334
0335 005127             BINZIF?                         ;ERKENNE BIAERZIFFERN
0336 005127 C930                CMP #'0'
0337 005129 90E0                BCC FALSE1              ;DRUNTER
0338 00512B C932                CMP #'2'                ;DRUEBER?
0339 00512D B0DB                BCS FALSE
0340 00512F 38                  SEC                     ;TRUE
0341 005130 60                  RTS

0343 005131             HEXZIF?                         ;ERKENNE HEX-ZIFFERN GROSS UN
                                D KLEIN
0344 005131 C967                CMP #$67                ;KLEINES G
0345 005133 B0D5                BCS FALSE
0346 005135 C961                CMP #$61                ;KLEINES A
0347 005137 B0DF                BCS TRUE
0348 005139 C947                CMP #'G'                ;NUN GROSSBUCHST.
0349 00513B B0CD                BCS FALSE
0350 00513D C941                CMP #'A'
0351 00513F 9005                BCC DEZZIF?             ;NOCH DEZIMALZIFFERN FILTERN
0352 005141 60                  RTS                     ;TRUE, CARRY=1
0353
0354 005142             OCTZIF?                         ;ERKENNE OKTALZIFFERN 0-7
0355 005142 C938                CMP #'8'
0356 005144 B0C4                BCS FALSE               ;UEBER 7
0357
0358 005146             DEZZIF?                         ;FILTER FUER DEZ. ZIFFERN 0-9
0359 005146 C930                CMP #'0'
0360 005148 90C1                BCC FALSE1
0361 00514A C93A                CMP #':'                ;GROESSER ALS ZIFFER '9'
0362 00514C B0BC                BCS FALSE               ;KEINE ZIFFER
0363 00514E 38                  SEC                     ;SETZE 'TRUE'
0364 00514F 60                  RTS
```

Bewertung eingegebener Zahlenstrings

Es sollen eingegebene ASCII-Zeichen, die Ziffern eines gewählten Zahlensystems repräsentieren, nach dezimal oder binär umgewandelt werden. Die erste Stufe der Behandlung ist JSR MAKE_ZIF. Aus jeweils einer Eingabeziffer wird der Bestandteil des Wertes aus der ASCII-Verschüsselung herausgelöst. Eine Besonderheit tritt in dieser Routine dadurch ein, daß sie auch Hexziffern umwandeln soll, sogar dann, wenn diese als kleine Buchstaben geschrieben sind.

Wir fordern, daß der Eingabestrom der Ziffern von links nach rechts gelesen und umgewandelt werden und daß das Ergebnis der Bewertung im Ergebnisfeld zexpr abgelegt werden soll. Für diese Art der Verarbeitung bietet sich das

Horner-Schema an. Vorbild für die hier vorliegende Codierung war ein Vorschlag in /1/. Das Schema wurde jedoch ausgebaut. Mit dem Aufruf JSR HORNERDEZ erfolgt eine Umwandlung in BCD-Zahlen, mit JSR HORNERBIN eine solche in Binärzahlen. Je nach der für die Eingabe geltenden Zahlenbasis wird man eine der folgenden Routinen wählen: BINVAL, OCTVAL, DEZVAL oder HEXVAL.

```
0366                            ;MACHE VOR-GEFILTERTE ASCII-ZAHL ZU BINAERWERT
0367                            ;ANZUWENDEN NACH JSR HEXZIF/DEZZIF/OCTZIF/BINZIF
0368                            ;ZEICHEN IM ACCU ZU UEBERGEBEN
0369 005150 C93A      MAKE_ZIF  CMP #':'              ;ZEICHEN FOLGT AUF '9'
0370 005152 9004                BCC MAKZI1
0371 005154 295F                AND #%01011111        ;MACHE KLEINES A-F GROSS
0372 005156 E907                SBC #7                ;CARRY=SET
0373 005158 290F      MAKZI1    AND #$0F              ;STRIP OFF
0374 00515A 858F                STA ZIFFER            ;ZWISCHENSPEICHERUNG FUER HOR
                                NER
0375 00515C 60                  RTS
0376
0377 00515D A000      ZEXNUL    LDY #0                ;ZWISCHENERGEBNIS=0 FUER BEWE
                                RTUNG
0378 00515F 8490                STY ZEXPR
0379 005161 8491                STY ZEXPR+1
0380 005163 8492                STY ZEXPR+2           ;ANFANGSWERT=0
0381 005165 849A                STY TRUEFL            ;00 BEDEUTET: OHNE BEWERTUNG
                                GEBLIEBEN
0382 005167 60                  RTS
0383
0384                            ;ZAHLENWANDLUNG: EINGABEZIFFER NACH HEX
0385 005168 F8        HORNERDEZ SED                   ;DEZIMALER MODUS/WANDLUNG NAC
                                H DEZ
0386 005169 8689      HORNERBIN STX COUNT             ;RETTE ZEIGER NACH INBUFF
0387                            ;ACHTUNG: ES WIRD ZU AUFSTEIGENDEN ADRESSEN GEARB
EITET
0388 00516B A592                LDA ZEXPR+2           ;KURZWEG, WENN AUSDRUCK NOCH
                                NULL
0389 00516D 0591                ORA ZEXPR+1
0390 00516F 0590                ORA ZEXPR
0391 005171 F01D                BEQ HORNAD
0392 005173 A200                LDX #0                ;3 ITEMS, AUFSTEIGENDE ADRESS
                                EN
0393 005175 8699                STX CARRY             ;1. STELLE OHNE UEBERTRAG
0394 005177 A599      HORN1     LDA CARRY             ;UEBERTRAG AUS VOR-STELLE
0395 005179 A000                LDY #0
0396 00517B 8499                STY CARRY             ;CARRY=0
0397 00517D A48E                LDY BASIS             ;ZAHL DER MULTIPLIKATIONEN
0398 00517F 18        HORN2     CLC                   ;IMMER OHNE CARRY AUS EIGENER
                                 STELLE
0399 005180 7590                ADC ZEXPR,X
0400 005182 9002                BCC HORN3             ;NICHTS MERKEN
0401 005184 E699                INC CARRY             ;UEBERTRAG IN FOLGESTELLE
0402 005186 88        HORN3     DEY
0403 005187 D0F6                BNE HORN2             ;WIEDERHOLTE ADDITION
0404 005189 9590                STA ZEXPR,X           ;STELLENERGEBNIS
0405 00518B E8                  INX
0406 00518C E003                CPX #3                ;3 BYTES UEBERARBEITET?
0407 00518E D0E7                BNE HORN1             ;NAECHSTE STELLE
0408 005190 18        HORNAD    CLC                   ;NUN EINGABEZIFFER ADDIEREN
0409 005191 A58F                LDA ZIFFER
0410 005193 6590                ADC ZEXPR             ;LOW BYTE
0411 005195 8590                STA ZEXPR
0412 005197 900E                BCC HORN4
0413 005199 A591                LDA ZEXPR+1
0414 00519B 6900                ADC #0                ;UEBERTRAG
0415 00519D 8591                STA ZEXPR+1
0416 00519F 9006                BCC HORN4
0417 0051A1 A592                LDA ZEXPR+2
0418 0051A3 6900                ADC #0
0419 0051A5 8592                STA ZEXPR+2
0420 0051A7 A689      HORN4     LDX COUNT             ;RESTORE
0421 0051A9 E69A                INC TRUEFL            ;ZEIGE ERFOLGREICHE BEWERTUNG
                                 AN
```

```
0422 0051AB D8                CLD                   ;FALLS NACH DEZIMAL GEWANDELT
                               WURDE
0423 0051AC 60                RTS                   ;OVERFLOW ON CARRY SET!
0424
0425 0051AD A010      HEXVAL  LDY #16               ;ZAHLENBASIS
0426 0051AF B48E              STY BASIS
0427 0051B1 E8        HEXVA1  INX
0428 0051B2 B500              LDA INBUFF,X
0429 0051B4 203151            JSR HEXZIF?
0430 0051B7 904A              BCC ABBRUCH           ;ZEICHEN DANACH BEHANDELN
0431 0051B9 205051            JSR MAKE_ZIF          ;ASCII ZU HEX
0432 0051BC 206951            JSR HORNERBIN
0433 0051BF 4CB151            JMP HEXVA1
0434
0435 0051C2 A00A      DEZVAL  LDY #10               ;ZAHLENBASIS
0436 0051C4 B48E              STY BASIS
0437 0051C6 D008              BNE DEZVA2            ;ZEICHEN NOCH IM AKKU!
0438 0051C8 E8        DEZVA1  INX
0439 0051C9 B500              LDA INBUFF,X
0440 0051CB 204651            JSR DEZZIF?
0441 0051CE 9033              BCC ABBRUCH           ;ZEICHEN DANACH BEHANDELN
0442 0051D0 205051    DEZVA2  JSR MAKE_ZIF          ;ASCII ZU HEX
0443 0051D3 206951            JSR HORNERBIN
0444 0051D6 4CC851            JMP DEZVA1
0445
0446 0051D9 A002      BINVAL  LDY #02               ;ZAHLENBASIS
0447 0051DB B48E              STY BASIS
0448 0051DD E8        BINVA1  INX
0449 0051DE B500              LDA INBUFF,X
0450 0051E0 202751            JSR BINZIF?
0451 0051E3 901E              BCC ABBRUCH           ;ZEICHEN DANACH BEHANDELN
0452 0051E5 205051            JSR MAKE_ZIF          ;ASCII ZU HEX
0453 0051E8 206951            JSR HORNERBIN
0454 0051EB 4CDD51            JMP BINVA1
0455
0456 0051EE A008      OCTVAL  LDY #08               ;ZAHLENBASIS
0457 0051F0 B48E              STY BASIS
0458 0051F2 E8        OCTVA1  INX
0459 0051F3 B500              LDA INBUFF,X
0460 0051F5 204251            JSR OCTZIF?
0461 0051F8 9009              BCC ABBRUCH           ;ZEICHEN DANACH BEHANDELN
0462 0051FA 205051            JSR MAKE_ZIF          ;ASCII ZU HEX
0463 0051FD 206951            JSR HORNERBIN
0464 005200 4CF251            JMP OCTVA1
0465
0466 005203           ABBRUCH
```

Die typische Verarbeitungsabfolge für die Bewertung von ASCII-Zeichen wird damit z.B. für Hexadezimal-Ziffern:

```
lda inbuff,x  ;hole zeichen aus dem puffer
jsr hexzif    ;filter auf Hex-Ziffern
bcc abbruch   ;zeichen repräsentiert nicht hex
jsr make-zif  ;Wert der Ziffer isolieren
jsr hexval    ;Umwandlung über Horner-Schema
```

Diese Sequenz wird man in einer Schleife mehrfach durchlaufen, bis alle Ziffern umgewandelt sind. Im Einzelfall kann man sicher etwas kürzer programmieren. Die Beispiele sollen jedoch als ein 'Instrumentenkasten' dienen, der für alle Fälle geeignetes Werkzeug enthält.

Nachdem nun gezeigt ist, wie man Eingaben filtert und umwandelt, erhebt sich die Frage, wie man im Speicher enthaltene Zahlen für eine ASCII-Ausgabe aufbereitet. Eine typische Routine in den Betriebbsystemen ist NUMA: Ein im Akku übergebenes Hexbyte wird in zwei ASCII-Zeichen umgeformt, die sofort auf die aktive Ausgabe gesandt werden. Manchmal möchte man das Ergebnis der Umformung jedoch auch im Speicher weiterverwenden können. Hier ist daher die Routine NUMWA aufgeführt, die die eigentliche Umformung durchführt und im

Akku die höherwertige ASCII-Ziffer übergibt, in TEMP die niederwertige. NUMA ist daher hier als reine Ausgaberoutine implementiert.

```
0244                                ;KONVERTIERE EINEN ZAHLENSTRING IN INBUFF NACH HE
X
0245                                ;WEISE MIT CARRY=0 AUF UNZULAESSIGE ZEICHEN HIN
0246 0050AC 205D51    ADRCON      JSR ZEXNUL          ;ZWISCHENERGEBNIS ZU 0
0247 0050AF A910                  LDA #16             ;ZAHLENBASIS FUER HORNER
0248 0050B1 858E                  STA BASIS
0249 0050B3 A200                  LDX #0              ;ZEIGER IN DEN PUFFER
0250 0050B5 B500      ADRCO1      LDA INBUFF,X        ;HOLE ZEICHEN
0251 0050B7 F011                  BEQ ADRCOS          ;VERLASSE ROUTINE ERFOLGREICH
                                   MIT CARRY=1
0252 0050B9 858F                  STA ZIFFER          ;FUER HORNER
0253 0050BB 203151                JSR HEXZIF?         ;FILTER FUER HEXZAHLEN
0254 0050BE 900B                  BCC ADRCOC          ;VERLASSE MIT CARRY=0
0255 0050C0 205051                JSR MAKE_ZIF        ;STRIP OFF
0256 0050C3 206951                JSR HORNERBIN       ;ZIFFER KONVERTIEREN
0257 0050C6 E8                    INX                 ;ZEIGER NACH INBUFF
0258 0050C7 4CB550                JMP ADRCO1
0259 0050CA 38        ADRCOS      SEC                 ;WANDLUNG ERFOLGREICH
0260 0050CB 60        ADRCOC      RTS
0261                                ;*********************
0262                                ;WANDLE 1 HEXBYTE IN 2 ASCII-BYTES
0263                                ;BEIM VERLASSEN DER ROUTINE:
0264                                ;HOHES BYTE IM ACCU, NIEDRIGES IN TEMP
0265
0266 0050CC 40        BIT6        .BYT $40
0267
0268 0050CD 48        NUMWA       PHA
0269 0050CE 2CCC50                BIT BIT6            ;SETZE V-FLAG
0270 0050D1 290F                  AND #$0F
0271 0050D3 20DB50                JSR NUMWA2
0272 0050D6 68                    PLA
0273 0050D7 4A                    LSR A               A ;ISOLIERE HOEHERWERTIGEN TE
                                   IL
0274 0050D8 4A                    LSR A
0275 0050D9 4A                    LSR A
0276 0050DA 4A                    LSR A               A ;4XRECHTS
0277 0050DB 0930      NUMWA2      ORA #$30            ;MACH ZU ASCII
0278 0050DD C93A                  CMP #$3A            ;UEBERLAUF?
0279 0050DF 9004                  BCC NUMWA3
0280 0050E1 08                    PHP                 ;V-BIT NICHT D. ADC ZERSCHIES
                                   SEN LASSEN!
0281 0050E2 6906                  ADC #6              ;FUER A...F
0282 0050E4 28                    PLP
0283 0050E5 5003      NUMWA3      BVC NUMWA4          ;BEIM 2. DURCHGANG
0284 0050E7 857D                  STA TEMP            ;NIEDERWERTIGEN TEIL
0285 0050E9 B8                    CLV                 ;RESET V-FLAG
0286 0050EA 60        NUMWA4      RTS
0287
0288 0050EB           NUMA                            ;AUSGABE EINER ZAHL
0289 0050EB 20CD50                JSR NUMWA           ;UMWANDLUNG
0290 0050EE 201F50                JSR OUTALL
0291 0050F1 A57D                  LDA TEMP
0292 0050F3 4C1F50                JMP OUTALL          ;RTS DORT
```

Bücher

Rolle, G.: Mikkro Wissen A–Z. Vieweg-Verlag, Braunschweig 1985, 124 S., DM 24,80, ISBN 3 528 04329-6. Das Buch enthält eine alphabetisch geordnete Sammlung und Erklärung von Begriffen aus Hard- und Software, Datenkommunikation und Bildschirmtext. Es sind zahlreiche schematische Abbildungen enthalten und Tabellen. Für die etwa 950 Stichworte ist im Anhang ein Glossarium Deutsch-Englisch und Englisch-Deutsch enthalten. - Die Stichworte sind auf ein größeres interessiertes Publikum zugeschnitten, die Erklärungen sind daher knapp und verständlich abgefaßt. Wer sich schon länger mit dem Fachgebiet befaßt, darf daher keine Vertiefung seines Wissens erwarten.

Peter Engels, 5308 Rheinbach-Niederdrees

C-64 Screen Monitor

Dieses Programm entstand aus dem Wunsch heraus, Bildschirminhalte von abgebrochenen Programmen, z.B. Spielszenen, nach einem Reset sichtbar zu machen und gegebenenfalls auf einen Drucker auszugeben.

Das Programm funktioniert ähnlich dem Monitor der bekannten CBM-Rechner. D.h. nach dem Start wird auf eine Eingabe (Tastendruck) gewartet, und es folgt danach eine direkte Abarbeitung des Befehls. - Nach der Eingabe von RUN meldet sich das Programm mit einem Menue (Bild 1) und erwartet eine Eingabe. Alle Änderungen von Parametern werden direkt auf dem Schirm angezeigt. Es sind folgende Eingaben/Änderungen möglich:

Taste	Wert	Wirkung
B	0 - 3	Dem VIC-Chip wird einer der 16KB-Bereiche des C-64 Speichers zugewiesen
V	0 - 15	Das Video-RAM wird in 1KB-Schritten weitergeschaltet
Z	0 - 7	Der Zeichengenerator wird in 2KB-Schritten weitergeschaltet
M	HI/LO	Es wird zwischen Hires- und Textgrafik umgeschaltet
P	1 / 2	Bei Hires-Grafik wird zwischen dem unteren und oberen 8K-Block umgeschaltet
C	MUL/NOR	Der Farbmodus wird zwischen normal und Multicolor umgeschaltet
F1		Durch diese Taste werden alle eingestellten Werte in den VIC übertragen, d.h. auf dem Schirm wird der eingestellte RAM-Bereich im eingestellten Modus sichtbar. Nochmaliger Druck von F1 schaltet ins Menue
F3		In Verbindung mit dem EPSON-VC-Interface wird eine Hardcopy des dargestellten Schirms über den User-Port vorgenommen
F5		Wie oben, jedoch invertierte Hardcopy
F7		Der VIC wird auf die Standardwerte eingestellt, und es erfolgt ein Sprung zu BASIC

Mit diesen Funktionen können auch die RAM-Bereiche bequem dargestellt werden, die unter den ROM- bzw. I/O-Bausteinen liegen. Wenn im Display-Modus ein Bild im Speicher gefunden wurde, dann kann durch erneutes Drücken der F1-Taste anhand der im Menue dargestellten Daten die Lage des Bildes leicht ermittelt werden:

Hires-Modus		
Bildstart	Bank * 16384 + Page * 8192	Länge: 8192 Bytes
Farbramstart	Bank * 16384 + VRAM * 1024	Länge: 1000 Bytes
Text-Modus		
Bildstart	Bank * 16384 + VRAM * 1024	Länge 1000 Bytes
Zeichengen.	Bank * 16384 + ZGEN * 2048	Länge: 2048 Bytes

Mit der Kenntnis dieser Werte ist es möglich, das Bild - nach eventuellem vorherigen Abschalten von überlagerten ROMs - auf Cassette oder Disk abzuspeichern und später in einen beliebigen anderen Bereich einzuladen.

C-64 SCREEN-MONITOR V1.2

(Standard-Werte nach RUN)

F1=SWITCH F3=COPY F5=COPY RVS F7=EXIT

BANK	VRAM	ZGEN	MODE	PAGE	CMODE
0	1	2	LO		NOR

C-64 SCREEN-MONITOR V1.2

(geänderte Werte)

f1=SWITCH f3=COPY f5=COPY RVS f7=EXIT

BANK	VRAM	ZGEN	MODE	PAGE	CMODE
2	12	7	HI	2	MUL

```
                        0010 ;  SCREEN-MONITOR
                        0020 ; BASICVERSION 1.2
                        0030 ;
                        0040 ;
                        0050              .BA $0801
                        0060              .MC $0801
                        0070              .OS
                        0080 ;
                        0090 VIC          .DE $D000
                        0100 CIA2         .DE $DD00
                        0110 CLSFIL       .DE $F642
                        0120 GET          .DE $FFE4
                        0130 INPUT        .DE $FFCF
                        0140 TALK         .DE $FFB4
                        0150 SECTALK      .DE $FF96
                        0160 IECIN        .DE $FFA5
                        0170 UNTALK       .DE $FFAB
                        0180 LISTEN       .DE $FFB1
                        0190 SECLIST      .DE $FF93
                        0200 IECOUT       .DE $FFA8
                        0210 UNLIST       .DE $FFAE
                        0220 BASOUT       .DE $FFD2
                        0230 OPEN         .DE $FFC0
                        0240 CLOSE        .DE $FFC3
                        0250 SETPAR       .DE $FFBA
                        0260 SETNAM       .DE $FFBD
                        0270 SENDNAM      .DE $F3D5
                        0280 CHKIN        .DE $FFC6
                        0290 CKOUT        .DE $FFC9
                        0300 CLRCH        .DE $FFCC
                        0310 CLALL        .DE $FFE7
                        0320 READY        .DE $E37B
                        0330 LNPRT        .DE $BDCD
                        0340 STATUS       .DE $90
                        0350 LF           .DE $B8
                        0360 SA           .DE $B9
                        0370 FA           .DE $BA
                        0380 FNADR        .DE $BB
                        0390 FNLEN        .DE $B7
                        0400 QUOTFLG      .DE $D4
                        0410 CR           .DE $0D
                        0420 NCMDS        .DE $11
                        0430 BUFFER       .DE $033C
                        0440 BUFF         .DE BUFFER+$30
                        0450 BNKBUFF      .DE BUFF
                        0460 VRAMBUFF     .DE BUFF+1
                        0470 ZGENBUFF     .DE BUFF+2
                        0480 MODEBUFF     .DE BUFF+3
                        0490 PAGEBUFF     .DE BUFF+4
                        0500 CMODEBUFF    .DE BUFF+5
                        0510 SWITCHBUFF   .DE BUFF+6
                        0520 RVSFLAG      .DE BUFF+7
                        0530 VICBUFF      .DE BUFF+8
                        0531 ;
0801- 2A 08             0532              .SI NEXTLINE          ; POINTER NAECHSTE ZEILE
0803- C0 07             0533              .BY $C0 $07 ; ZEILENNR.
0805- 9E                0534              .BY $9E               ; SYS-TOKEN
0806- 28 32 31          0535              .BY '(2120) :
0809- 32 30 29
080C- 20 3A 20
080F- 0D 53 43          0536              .BY $0D  SCREENMONITOR BY P. ENGELS'
0812- 52 45 45
0815- 4E 4D 4F
```

```
0818- 4E 49 54
081B- 4F 52 20
081F- 42 59 20
0821- 50 2E 20
0824- 45 4E 47
0827- 45 4C 53
082A- 00 00       0537 NEXTLINE   .BY $00 $00
                  0538 ;
                  0539            .BA $0848
                  0540            .MC $0848
                  0541 ;
0848- A2 2F       0550 INIT       LDX #$2F
084A- AD 00 DD    0560            LDA CIA2
084D- 9D 3C 03    0570            STA BUFFER,X
0850- CA          0580            DEX
0851- BD 00 D0    0590 INIT1      LDA VIC,X
0854- 9D 3C 03    0600            STA BUFFER,X
0857- CA          0610            DEX
0858- 10 F7       0620            BPL INIT1
085A- A9 93       0630            LDA #$93
085C- 20 D2 FF    0640            JSR BASOUT
085F- 20 15 09    0650            JSR INITBUFF
                  0660 ;
0862- 20 34 09    0670 START      JSR SETSCR          ; SCHIRM UMSCHALTEN
0865- A2 00       0680            LDX #$00
0867- BD 7F 0A    0690 PRT        LDA HLIN,X
086A- F0 06       0700            BEQ DECODE
086C- 20 D2 FF    0710            JSR BASOUT
086F- E8          0720            INX
0870- D0 F5       0730            BNE PRT
                  0740 ;
0872- 20 DA 09    0750 DECODE     JSR MONITOR
0875- 20 E4 FF    0760            JSR GET
0878- A2 10       0770            LDX #NCMDS-1
087A- DD 07 0B    0780 S1         CMP CMDS,X
087D- D0 0E       0790            BNE S2
087F- 8A          0800            TXA                 ; X*2
0880- 0A          0810            ASL A
0881- AA          0820            TAX
0882- E8          0830            INX
0883- BD 11 0B    0840            LDA ADRESS,X
0886- 48          0850            PHA
0887- CA          0860            DEX
0888- BD 11 0B    0870            LDA ADRESS,X
088B- 48          0880            PHA
088C- 60          0890            RTS
088D- CA          0910 S2         DEX
088E- 10 EA       0920            BPL S1
0890- 4C 62 08    0930            JMP START
                  0940 ;
0893- EE 6C 03    0950 BANK       INC BNKBUFF
0896- 4C 62 08    0960            JMP START
                  0970 ;
0899- EE 6D 03    0980 VRAM       INC VRAMBUFF
089C- 4C 62 08    0990            JMP START
                  1000 ;
089F- EE 6E 03    1010 ZGEN       INC ZGENBUFF
08A2- 4C 62 08    1020            JMP START
                  1030 ;
08A5- AD 6F 03    1040 MODE       LDA MODEBUFF
08A8- 49 FF       1050            EOR #$FF
08AA- 8D 6F 03    1060            STA MODEBUFF
08AD- 4C 62 08    1070            JMP START
                  1080 ;
08B0- AD 6F 03    1090 PAGE       LDA MODEBUFF
08B3- F0 08       1100            BEQ NOPAGE
08B5- AD 70 03    1110            LDA PAGEBUFF
08B8- 49 FF       1120            EOR #$FF
08BA- 8D 70 03    1130            STA PAGEBUFF
08BD- 4C 62 08    1140 NOPAGE     JMP START
                  1150 ;
08C0- AD 71 03    1160 CMODE      LDA CMODEBUFF
08C3- 49 FF       1170            EOR #$FF
08C5- 8D 71 03    1180            STA CMODEBUFF
```

```
08C8- 4C 62 08  1190            JMP START
                1200 ;
08CB- AD 72 03  1210 SWITCH     LDA SWITCHBUFF
08CE- 49 FF     1220            EOR #$FF
08D0- 8D 72 03  1230            STA SWITCHBUFF
08D3- 4C 62 08  1240            JMP START
                1250 ;
08D6- A9 00     1260 COPY       LDA #0
08D8- 2C        1270            .BY $2C
08D9- A9 01     1280 RVSCOPY    LDA #1
08DB- 8D 73 03  1290            STA RVSFLAG
08DE- A9 07     1300            LDA #7                  ; DEVICE 7
08E0- 85 BA     1310            STA *FA
08E2- A9 7E     1320            LDA #126
08E4- 85 B8     1330            STA *LF
08E6- A9 00     1340            LDA #0
08E8- 85 B7     1350            STA *FNLEN
08EA- 85 B9     1360            STA *SA
08EC- 20 C0 FF  1370            JSR OPEN
08EF- A6 B8     1380            LDX *LF
08F1- 20 C9 FF  1390            JSR CKOUT
08F4- A9 0D     1400            LDA #$0D
08F6- 20 D2 FF  1410            JSR BASOUT
08F9- AD 73 03  1420            LDA RVSFLAG
08FC- 20 D2 FF  1430            JSR BASOUT
08FF- A9 0D     1440            LDA #$0D
0901- 20 D2 FF  1450            JSR BASOUT
0904- 20 CC FF  1460            JSR CLRCH
0907- A9 7E     1470            LDA #126
0909- 20 C3 FF  1480            JSR CLOSE
090C- 4C 62 08  1490            JMP START
                1500 ;
090F- 20 21 09  1510 EXIT       JSR SETMONSCR           ; NORMALE SCHIRMANZEIGE
0912- 4C 7B E3  1520            JMP READY
                1530            ;
0915- A2 07     1540 INITBUFF   LDX #7                  ; DEFAULTWERTE SETZEN
0917- BD FF 0A  1550 INITBUFF1  LDA DEFAULT,X
091A- 9D 6C 03  1560            STA BUFF,X
091D- CA        1570            DEX
091E- 10 F7     1580            BPL INITBUFF1
0920- 60        1590            RTS
                1600 ;
0921- A2 2F     1610 SETMONSCR  LDX #$2F                ; SCHIRM AUF STARTWERT
0923- BD 3C 03  1620            LDA BUFFER,X
0926- 8D 00 DD  1630            STA CIA2
0929- CA        1640            DEX
092A- BD 3C 03  1650 SETMON1    LDA BUFFER,X
092D- 9D 00 D0  1660            STA VIC,X
0930- CA        1670            DEX
0931- 10 F7     1680            BPL SETMON1
0933- 60        1690            RTS
                1700 ;
0934- AD 72 03  1710 SETSCR     LDA SWITCHBUFF
0937- F0 E8     1720            BEQ SETMONSCR
0939- AD 6C 03  1730            LDA BNKBUFF             ; BANK SETZEN
093C- 49 03     1740            EOR #%00000011
093E- 29 03     1750            AND #%00000011
0940- 8D 74 03  1760            STA VICBUFF
0943- AD 00 DD  1770            LDA CIA2
0946- 29 FC     1780            AND #%11111100
0948- 0D 74 03  1790            ORA VICBUFF
094B- 8D 00 DD  1800            STA CIA2
                1810 ;
094E- AD 11 D0  1820            LDA VIC+17
0951- AE 6F 03  1830            LDX MODEBUFF
0954- D0 03     1840            BNE SETHI
0956- 29 DF     1850            AND #%11011111          ; TEXTMODE SETZEN
0958- 2C        1860            .BY $2C
0959- 09 20     1870 SETHI      ORA #%00100000          ; HIRESMODE SETZEN
095B- 8D 11 D0  1880 SETMODE    STA VIC+17
                1890 ;
095E- A9 00     1900            LDA #0
```

```
0960- 8D 74 03  1910            STA VICBUFF
0963- AD 6D 03  1920            LDA VRAMBUFF        ; VRAM SETZEN
0966- 0A        1930            ASL A
0967- 0A        1940            ASL A
0968- 0A        1950            ASL A
0969- 0A        1960            ASL A
096A- 8D 74 03  1970            STA VICBUFF
                1980 ;
096D- AD 6F 03  1990            LDA MODEBUFF
0970- D0 0F     2000            BNE SETPAGE
0972- AD 6E 03  2010            LDA ZGENBUFF        ; ZGEN SETZEN
0975- 29 07     2020            AND #%00000111
0977- 0A        2030            ASL A
0978- 0D 74 03  2040            ORA VICBUFF
097B- 8D 74 03  2050            STA VICBUFF
097E- 4C 9B 09  2060            JMP SETCMODE
                2070 ;
0981- AD 70 03  2080 SETPAGE    LDA PAGEBUFF
0984- 29 01     2090            AND #1
0986- F0 0B     2100            BEQ PAG1
0988- AD 74 03  2110            LDA VICBUFF
098B- 09 08     2120            ORA #%00001000      ; PAGE2 SETZEN
098D- 8D 74 03  2130            STA VICBUFF
0990- 4C 9B 09  2140            JMP SETCMODE
                2150 ;
0993- AD 74 03  2160 PAG1       LDA VICBUFF
0996- 29 F7     2170            AND #%11110111      ; PAGE1 SETZEN
0998- 8D 74 03  2180            STA VICBUFF
                2190 ;
                2200 ;
099B- AD 74 03  2210 SETCMODE   LDA VICBUFF
099E- 8D 18 D0  2220            STA VIC+24
09A1- AD 6F 03  2230            LDA MODEBUFF
09A4- D0 17     2240            BNE HICMODE
09A6- AD 71 03  2250            LDA CMODEBUFF
09A9- F0 09     2260            BEQ CLRLOCMODE
09AB- AD 16 D0  2270            LDA VIC+22
09AE- 09 10     2280            ORA #%00010000      ; SET MULTICOLOR
09B0- 8D 16 D0  2290            STA VIC+22
09B3- 60        2300            RTS
                2310 ;
09B4- AD 16 D0  2320 CLRLOCMODE LDA VIC+22
09B7- 29 EF     2330            AND #%11101111      ; CLEAR MULTICOLOR
09B9- 8D 16 D0  2340            STA VIC+22
09BC- 60        2350            RTS
                2360 ;
09BD- AD 16 D0  2370 HICMODE    LDA VIC+22
09C0- AE 71 03  2380            LDX CMODEBUFF
09C3- F0 03     2390            BEQ CLRHICMODE
09C5- 09 10     2400            ORA #%00010000      ; SET HIRES MULTICOLR
09C7- 2C        2410            .BY $2C
09C8- 29 EF     2420 CLRHICMODE AND #%11101111      ; CLEAR HIRES MULTICOLR
09CA- 8D 16 D0  2430            STA VIC+22
09CD- 60        2440            RTS
                2450 ;
09CE- 20 D1 09  2460 SPAC4      JSR SPAC2           ; 4 * SPACE
09D1- 20 D4 09  2470 SPAC2      JSR SPACE           ; 2 * SPACE
09D4- A9 20     2480 SPACE      LDA #$20            ; SPACE
09D6- 20 D2 FF  2490            JSR BASOUT
09D9- 60        2500            RTS
                2510 ;
09DA- 20 CE 09  2520 MONITOR    JSR SPAC4
09DD- 20 D1 09  2530            JSR SPAC2
09E0- AD 6C 03  2540            LDA BNKBUFF
09E3- 29 03     2550            AND #3
09E5- AA        2560            TAX
09E6- A9 00     2570            LDA #0
09E8- 20 CD BD  2580            JSR LNPRT           ; BANK AUSGEBEN
09EB- 20 D1 09  2590            JSR SPAC2
09EE- 20 D4 09  2600            JSR SPACE
09F1- AD 6D 03  2610            LDA VRAMBUFF
09F4- 29 0F     2620            AND #$0F
```

```
09F6- C9 0A      2630              CMP #10
09F8- B0 08      2640              BCS PRTVRAM
09FA- 20 D4 09   2650              JSR SPACE
09FD- AD 6D 03   2660              LDA VRAMBUFF
0A00- 29 0F      2670              AND #$0F
0A02- AA         2680 PRTVRAM      TAX                    ; VRAM AUSGEBEN
0A03- A9 00      2690              LDA #0
0A05- 20 CD BD   2700              JSR LNPRT
0A08- 20 CE 09   2710              JSR SPAC4
0A0B- 20 D4 09   2720              JSR SPACE
0A0E- A2 00      2730              LDX #0
0A10- AD 6E 03   2740              LDA ZGENBUFF
0A13- 29 07      2750              AND #7
0A15- AA         2760              TAX
0A16- A9 00      2770              LDA #0
0A18- 20 CD BD   2780              JSR LNPRT              ; ZGEN AUSGEBEN
0A1B- 20 CE 09   2790              JSR SPAC4
0A1E- AD 6F 03   2800              LDA MODEBUFF
0A21- F0 0D      2810              BEQ PRTLO
0A23- A9 48      2820              LDA #'H'               ; HI = HIRES AUSGEBEN
0A25- 20 D2 FF   2830              JSR BASOUT
0A28- A9 49      2840              LDA #'I'
0A2A- 20 D2 FF   2850              JSR BASOUT
0A2D- 4C 3A 0A   2860              JMP PRTPAGE
0A30- A9 4C      2870 PRTLO        LDA #'L'               ; LO = LORES AUSGEBEN
0A32- 20 D2 FF   2880              JSR BASOUT
0A35- A9 4F      2890              LDA #'O'
0A37- 20 D2 FF   2900              JSR BASOUT
0A3A- 20 CE 09   2910 PRTPAGE     JSR SPAC4              ; HIRESPAGE AUSGEBEN
0A3D- 20 D4 09   2920              JSR SPACE
0A40- AD 6F 03   2930              LDA MODEBUFF
0A43- F0 0B      2940              BEQ NOPAG
0A45- AD 70 03   2950              LDA PAGEBUFF
0A48- F0 03      2960              BEQ PRTPAG1
0A4A- A9 32      2970              LDA #'2'
0A4C- 2C         2980              .BY $2C
0A4D- A9 31      2990 PRTPAG1      LDA #'1'
0A4F- 2C         3000              .BY $2C
0A50- A9 20      3010 NOPAG        LDA #' '
0A52- 20 D2 FF   3020              JSR BASOUT
0A55- 20 CE 09   3030              JSR SPAC4
0A58- AD 71 03   3040              LDA CMODEBUFF
0A5B- F0 12      3050              BEQ PRTNOR
0A5D- A9 4D      3060              LDA #'M'               ; MUL AUSGEBEN
0A5F- 20 D2 FF   3070              JSR BASOUT
0A62- A9 55      3080              LDA #'U'
0A64- 20 D2 FF   3090              JSR BASOUT
0A67- A9 4C      3100              LDA #'L'
0A69- 20 D2 FF   3110              JSR BASOUT
0A6C- 4C 7E 0A   3120              JMP NEWLN
0A6F- A9 4E      3130 PRTNOR       LDA #'N'               ; NOR AUSGEBEN
0A71- 20 D2 FF   3140              JSR BASOUT
0A74- A9 4F      3150              LDA #'O'
0A76- 20 D2 FF   3160              JSR BASOUT
0A79- A9 52      3170              LDA #'R'
0A7B- 20 D2 FF   3180              JSR BASOUT
0A7E- 60         3190 NEWLN        RTS
                 3200 ;
0A7F- 13 0D 0D   3210 HLIN         .BY $13 $0D $0D $20
0A82- 20
0A83- 20 20 20   3220              .BY $20 $20 $20 $20 'C-64  SCREEN-MONITOR  V1.2
0A86- 20 43 2D
0A89- 36 34 20
0A8C- 20 53 43
0A8F- 52 45 45
0A92- 4E 2D 4D
0A95- 4F 4E 49
0A98- 54 4F 52
0A9B- 20 20 56
0A9E- 31 2E 32
0AA1- 0D 0D 0D   3230              .BY $0D $0D $0D $0D $0D
0AA4- 0D 0D
```

```
0AA6- 20 12 46  3240            .BY $20 $12 'F1' $92 '=SWITCH ' $12 'F3' $92 '=
0AA9- 31 92 3D
0AAC- 53 57 49
0AAF- 54 43 48
0AB2- 20 12 46
0AB5- 33 92 3D
0AB8- 43 4F 50
0ABB- 59 20
0ABD- 12 46 35  3250            .BY $12 'F5' $92 '=COPY RVS ' $12 'F7' $92 '=EX
0AC0- 92 3D 43
0AC3- 4F 50 59
0AC6- 20 52 56
0AC9- 53 20 12
0ACC- 46 37 92
0ACF- 3D 45 58
0AD2- 49 54
0AD4- 0D 0D 20  3260            .BY $0D $0D $20 $20 $20 $20
0AD7- 20 20 20
0ADA- 42 41 4E  3270            .BY 'BANK VRAM  ZGEN  MODE  PAGE  CMODE'
0ADD- 4B 20 56
0AE0- 52 41 4D
0AE3- 20 20 5A
0AE6- 47 45 4E
0AE9- 20 20 4D
0AEC- 4F 44 45
0AEF- 20 20 50
0AF2- 41 47 45
0AF5- 20 20 43
0AF8- 4D 4F 44
0AFB- 45
0AFC- 0D 0D 00  3280            .BY $0D $0D $00
                3290 ;
0AFF- 00        3300 DEFAULT    .BY 0               ; BANK
0B00- 01        3310            .BY 1               ; VRAM
0B01- 01        3320            .BY 1               ; ZGEN
0B02- 00        3330            .BY 0               ; MODE = TEXT
0B03- 00        3340            .BY 0               ; HIRESPAGE=1
0B04- 00        3350            .BY 0               ; CMODE = NOR
0B05- 00        3360            .BY 0               ; SWITCHFLAG = AUS
0B06- 00        3370            .BY 0               ; RVSFLAG=AUS
                3380 ;
0B07- 42        3390 CMDS       .BY 'B'             ; BANK
0B08- 56        3400            .BY 'V'             ; VRAM
0B09- 5A        3410            .BY 'Z'             ; ZGEN
0B0A- 4D        3420            .BY 'M'             ; MODE
0B0B- 50        3430            .BY 'P'             ; PAGE
0B0C- 43        3440            .BY 'C'             ; CMODE
0B0D- 85        3450            .BY 133             ; SWITCH
0B0E- 86        3460            .BY 134             ; COPY
0B0F- 87        3470            .BY 135             ; RVSCOPY
0B10- 88        3480            .BY 136             ; EXIT
                3490 ;
0B11- 92 08     3500 ADRESS     .SI BANK-1
0B13- 98 08     3510            .SI VRAM-1
0B15- 9E 08     3520            .SI ZGEN-1
0B17- A4 08     3530            .SI MODE-1
0B19- AF 08     3540            .SI PAGE-1
0B1B- BF 08     3550            .SI CMODE-1
0B1D- CA 08     3560            .SI SWITCH-1
0B1F- D5 08     3570            .SI COPY-1
0B21- DB 08     3580            .SI RVSCOPY-1
0B23- 0E 09     3590            .SI EXIT-1
                3600 ;
                3610            .EN
END OF ASSEMBLY!
```

Die neuen Computer

PC 128 von Commodore. Die Auslieferung des neuen Rechners hat begonnen, und zwar zunächst an Kaufhäuser und Versandhäuser, während die Vertragshändler von Commodore erst später beliefert werden sollen. Commodore hat damit die Händlerschaft unnötig verärgert.

Der Herausgeber hatte bei Redaktionsschluß Gelegenheit, eines der ersten Geräte kurz zu studieren. Der Eindruck: Gut gemacht!. Wir finden von Anfang an eine Tastatur und eine Bildschirmausgabe, die entweder mit dem ASCII-Zeichensatz oder mit dem deutschen Alphabet betrieben werden kann. Dem Bildschirmspeicher ist ein Attribut-Speicher für die Zeichen beigeordnet, der nicht nur die Farbe der Zeichen (vor gewähltem Hintergrund) speichert, sondern auch, ob die Darstellung normal oder invers oder gar auch blinkend sein soll. In einem Buchhaltungsprogramm könnte man damit 'rote Zahlen' auch wirklich rot darstellen und andere auffällige Merkmale, wie z.B. Fristüberschreitungen blinkend. Dem Herausgeber würde allerdings ein einfarbiger Monitor reichen, auf dem man ein ruhig stehendes Bild hat. Bei farbiger Monitorausgabe hängt es ein wenig vom Geschmack ab, welche Zeichen- und Hintergrundfarbe man für gute Lesbarkeit wählt.
Der PC 128 erlaubt mehrere Betriebsweisen, auf die in weiteren Heften noch eingegangen werden soll. In der ihm eigenen Betriebsweise benutzt er BASIC 7, eine außerordentlich leistungsfähige Version der Sprache. Sie erlaubt jetzt einen klar strukturierten Programmierstil mit IF/THEN/ELSE (wie schon beim CBM 710/720), daneben aber auch die Blockbildung mit BEGIN/UNTIL oder DO/WHILE/UNTIL usw. Es gibt neue Befehle zur Benutzung/Umwandlung von hexadezimalen und dezimalen Argumenten, das PRINT USING, binäres Laden (BLOAD) und viele neue Befehle für die Benutzung der Grafik-Fähigkeiten und des Sound Synthesizers. Melodien können z.B. als Alpha-Strings programmiert werden. Mit anderen Befehlen erhält man Zeiger auf Variable, womit die innere Transparenz der Sprache verbessert wird. - Der vielleicht größte Fortschritt bei Commodore ist die von Anfang an mitgelieferte Dokumentation in deutscher Sprache. Es handelt sich um drei Teile mit einigen hundert Seiten Gesamtumfang. Alle Befehle sind dort erklärt und die Belegung des Speichers mit den Variablen des Betriebssystems und des BASIC 7. Ein kleinerer Band beschreibt besonders den Betrieb unter CP/M.

Von BASIC her erreicht man mit dem Befehl MONITOR ein großzügiges Monitorprogramm, das in seinen Leistungen z.B. etwa dem bekannten Newtim-S entspricht. Wir finden dort die Speicheranzeige (auch mit ASCII-Ausdeutung), die Speicheränderung, einen Assembler Line by Line, einen Disassembler, die Suche nach Zeichenketten und vieles mehr.

Der Preis der gut geformten Computer-Konsole liegt bei DM 998. In der Kalkulation muß man natürlich einen Bildschirm-Monitor hinzurechnen, Geräte für Massenspeicherung und für die Textausgabe. Vom Preis her gerät man damit in den Bereich der Personalcomputer, allerdings auch von der Leistungsfähigkeit her. Einerseits durch das gute BASIC 7, durch eine gut zu bedienende Tastatur mit besonderem Zehnerblock und eine Reihe mit Funktions- und Cursortasten. Man wird den PC 128 sicher häufig für ernsthafte Zwecke einsetzen, im Laufe der Zeit ggfs. auch als Entwicklungssystem. Mit den grafischen und akustischen Fähigkeiten wird er aber auch für künstlerische und spielerische Zwecke benutzt werden. Wenn er auch keinen Prozessor mit 16 Bit enthält, so doch drei der verbreiteten Typen mit 8 Bit, für die es bereits ein großes Softwareangebot gibt, so daß man von Anfang an gut betriebsfähig ist. - Beim Herausgeber als Lizenznehmer wird wahrscheinlich in Kürze schon das wohl leistungsfähigste deutsche Textsystem zum PC 128 lieferbar sein.

Atari 520 ST: Seit der zweiten Juli-Hälfte ist eine Vorserie des neuen Rechners mit der Zentraleinheit MC 68000 an die Händlerschaft ausgeliefert worden. Atari zufolge soll das Entwicklungspaket mit dem Computer, Mini-Floppy Laufwerk, Systemdisketten, Monitor und Dokumentation rund 200mal ausgeliefert worden sein. Die Mini-Floppies nehmen einseitig auf und haben ein Fassungsvermögen von 360 KB. Die Dokumentation hat etwa die Dicke von zwei Brockhausbänden. Gleichwohl muß sie als noch unvollständig gelten, denn es fehlen noch viele der beabsichtigten Abbildungen und die Beschreibung vieler Parameter, die beim Aufruf von Systemroutinen

übergeben werden müssen. Der größte Teil des für ROMs vorgesehenen Betriebssystems muß noch von Diskette in das RAM geladen werden.

Zum Entwicklungspaket (ca. DM 4000) gehört das GEM (Grafik-Bildschirmsystem, in 'C' geschrieben), die Sprache LOGO, beide von der Maus unterstützt, das Betriebssystem CP/M-68K (in einigen Punkten abgemagert), ein 'C'-Compiler zum CP/M von Digital Research nebst dem dazugehörigen 68000 Assembler und Linker. Ein BASIC, auch in 'C' geschrieben, soll in Kürze folgen. Nach einem Ondit sollte man jedoch seine Geschwindigkeit genau beobachten.

Außer zum Kennenlernen und zur Programmierung in LOGO ist der neue Computer in der Vorserie allerdings noch von wenig Nutzen, denn es gibt noch keinen Text-Editor z.B. für irgendeine andere Sprache oder zur Verwendung als reines Textsystem. Die Dinge sollen sich natürlich verbessern. Der erreichte Stand, der als 'planmäßig' bezeichnet wird, ist also noch nicht überall befriedigend. Positiv muß man sagen, daß das Bild auf dem Schirm hervorragend klar und ruhig ist. Auch die Bedienung der Maus bringt große Annehmlichkeiten bei der Auswahl in den Menues. Man kann also Atari nur wünschen, daß man bald in den Sattel kommt. Der Computer kann gut werden, wenn er akzeptiert und vielseitig unterstützt wird. - Zum Rechner sind bereits Bücher erschienen, ehe die Entwicklungspakete ausgeliefert wurden. Wenn man bedenkt, daß der Rechner noch gar nicht seine endgültige Version hat, dann ist sicher sehr zu beachten, wie kompetent ihr Inhalt denn schon sein kann.

Bei Atari wurde Dr. Hans Riedl zum Leiter der Abteilung Software-Support berufen. Peter Henselt übernahm den Bereich Hardware-Support und Service.

O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡O≡

Roland Löhr

Fritz 05

Entwicklungsboard für 6805-Prozessoren

In Heft 27 des MICRO MAG wurde nach dem Stand von Mitte 1982 die Familie der 6805- und 68705- Einchip-Prozessoren vorgestellt. Sie wird von Motorola und Second Sources angeboten. Es gibt in dieser Familie eine Vielfalt von Ausführungen. Der Leser sei hierzu besonders auf das Buch 'Single Chip Microcomputer Data 1984/85' von Motorola hingewiesen. Ihre Merkmale kann man grob wie folgt sortieren: Abhängig von der Pinzahl (28 oder 40) steht eine unterschiedliche Menge von Portpins zur Verfügung, Pins können als Eingang für A/D-Wandler dienen oder als serielles Interface. Es gibt Ausführungen mit erhöhter Treiberkraft, z.B. für LED's usw. Wir finden Ausführungen in HMOS/NMOS und CMOS. Die Typen mit maskenprogrammiertem ROM sind vor allem für industrielle Serien von Bedeutung. Für Kleinserien sind jedoch solche interessant, die ein UV-löschbares EPROM, natürlich nebst RAM und Timer, auf dem Chip haben. Es wird durch ein kleines Einbrennprogramm im ROM solcher Typen unterstützt.

Der Befehlssatz dieser CPU-Typen ist sehr regelmäßig aufgebaut. Er erlaubt eine sehr codesparende Programmierung mit Befehlen zum Setzen und Löschen von Bits und zur Verzweigung in Abhängigkeit von gesetzten oder gelöschten Bits. - Gleichwohl bleibt für den Entwickler die Frage, mit welchen Hilfsmitteln er kostensparend und produktiv programmieren kann. Und danach bleibt die Frage des Testens. Wie man den Code in die Einchipper mit EPROM hineinkriegt, das ist in den Datenblättern beschrieben. Man baut sich eine Einbrennschaltung auf und fertigt sich eine Vorlage des Betriebsprogrammes als EPROM an, von dem in den Einchipper hinübergebrannt wird. Als Hilfsmittel für die Programmierung steht vom Autor z.B. der 6805 Cross-Assembler für den AIM 65 oder kompatible Systeme zur Verfügung. Die Programmerprobung hat dabei noch immer die etwas umständlichen Zyklen des Einbrennens, Prüfens, Löschens, Korrigierens usw.

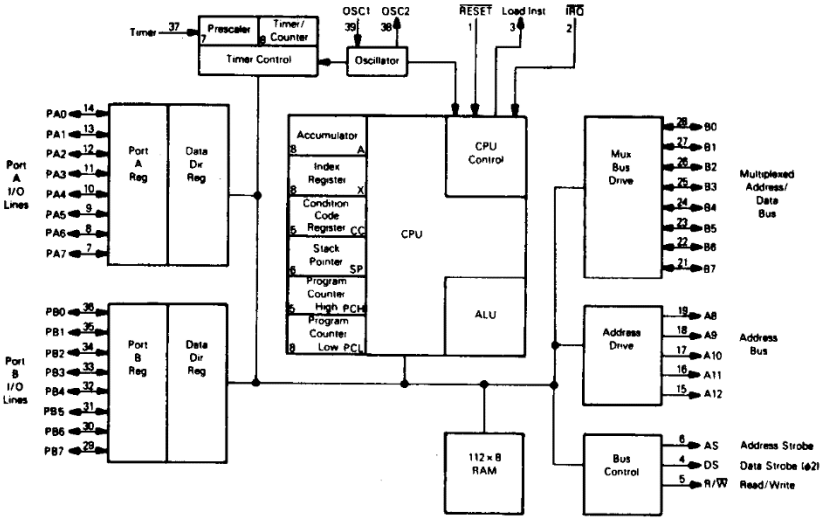
Das hier getestete Entwicklungsboard 'fritz05' der berliner Firma Mikrocomputertechnik Paul & Scherer, auf das wir schon in Heft 43 hinwiesen, erlaubt die Programmentwicklung und Erprobung ohne die Zwischenstufen des Brennens und Löschens auf eine sehr bequeme Weise:

Das Entwicklungboard 'fritz05' hat die Größe einer einfachen Europakarte. Zentraleinheit ist die CMOS-Version MC146805E2 der Prozessorfamilie. Sie bietet sich insofern für Entwicklungen an, weil sie einen herausgeführten und gemultiplexten Daten- und Adreßbus hat, mit dem sie extern 8 KB Speicher ansprechen kann. Die Verwendung von 16 Pins für diese Busse und ihre Kontrolle zieht jedoch nach sich, daß für diesen CPU-Typ nur zwei Ports mit je 8 Pins übrig bleiben (siehe Schema der CPU).Um nun mit dem Entwicklungsboard auch CPU-Typen mit mehr Ports emulieren zu können, enthält es eine CMOS PIA 6521. Auch das RAM, der Festwertspeicher und die übrigen Bausteine des Boards sind in CMOS ausgeführt, wodurch sich ein Stromverbrauch von typisch 13 mA ergibt, im WAIT-Status von 6 mA. Diese Eigenschaften läßt es auch als Computer dort in Betracht kommen, wo eine stromsparende Steuerung benötigt wird.

Die Prozessor- und Interfacesignale sind doppelt aus der Platine herausgeführt, einmal auf eine 64polige VG-Leiste und dann auf eine 50polige Pfostenleiste. Seitlich an der Platine findet man eine 25polige Buchse für das RS232c-Interface, das mit einer ACIA 65C51 und Treiber- und Empfangsbausteinen ausgelegt ist. Die Baudrate kann mit Jumpern auf 300, 1200, 2400 und 9600 festgelegt werden. Ohne Jumper gelten 9600 Baud. Die Betriebsspannungen von +5, +12 und -12 Volt werden entweder auf Buchsen auf dem Board gelegt oder auf die erwähnten Interface-Leisten.

An Speicher stehen dem Anwender außerhalb der CPU 4 KB RAM und 4 KB EPROM zur Verfügung. Im EPROM ist ein interaktiver Mini-Monitor enthalten, der diese Karte als Entwicklungssystem bequem nutzbar macht. Wir gehen gleich auf seine Leistungen ein. Was man jetzt noch zur Inbetriebnahme der Karte braucht, das ist ein Terminal. Besser ist jedoch ein Computer mit der V24-Schnittstelle, mit einem Cross-Assembler und mit der Möglichkeit, getestete Programme in ein EPROM zu brennen. - Für externe Beschaltungen findet sich auf der Platine ein Lochrasterfeld von ca. 57x40 mm.

FIGURE 1 — MICROPROCESSOR BLOCK DIAGRAM



Das Monitorprogramm: Es richtet nach einem Reset den seriellen Datenkanal ein, meldet sich mit einem Text und ist dann zur Entgegennahme von Befehlen bereit. Die Inbetriebnahme ist also nach dem Anlegen der Spannungen denkbar einfach. Normalerweise beginnt man jetzt mit der Erprobung seiner Befehle: Der M-Befehl gestattet es, Speicherzellen-Inhalte anzuzeigen und zu verändern, man kann vorwärts und rückwärts blättern. D (Dump) zeigt Blöcke mit Zeilen zu 16

Bytes hexadezimal an, auch in ASCII-Ausdeutung dahinter. A, X, P und C setzen den Akku auf definierte Werte, das Register X, den Programmzähler und das Condition Code Register. R zeigt neben den Registern auch den Stackpointer an, B setzt einen Breakpoint (vorübergehende Absetzung des Befehles SWI an der bezeichneten Stelle), F füllt einen Speicherbereich mit Zeichen und W stellt den WAIT-Modus der CPU ein. Mit *G (Go) oder G Adresse wird ein Programm zur Ausführung gebracht, S führt einen einzelnen Befehl aus, T führt wiederhote Einzelschritte aus. Dabei werden die jeweiligen Folgebefehle in disassemblerter Form angezeigt. Mit dem Setzen von Speicherstellen und Registern sowie mit dem Einzelschritt sind daher eigentlich alle notwendigen Hilfen gegeben, um ein Programm zu erproben.*

Ein wichtiger weiterer Punkt ist auch geregelt: Ein auf einem Host-Rechner entwickeltes Proggramm kann auf den 'fritz05' heruntergeladen werden, und zwar in der Form von S-Records, die über die serielle Schnittstelle gesendet und am Entwicklungsboard empfangen werden. Für die Entschlüsselung und Abspeicherung enthält das Monitor-Programm den Befehl 'L'. Nach einem erfolgreichen Laden meldet es sich mit einem 'o.k' zurück, andernfalls mit einer Fehlermeldung. Wenn wir nun vom Ei gewissermaßen auf die Henne backtracen, so wird damit die für den 'fritz05' angebotene Entwicklungssoftware angesprochen. Auch hierfür ist gesorgt. Dem Autor wurde für sein MCT-Entwicklungssystem ein Cross-Assembler AS05 zur Erprobung zur Verfügung gestellt. Er ist in der Sprache 'C' formuliert worden und steht damit auch für Systeme mit MS-DOS bzw. PC-DOS zur Verfügung. Er verarbeitet im Editor geschriebene und in Motorola-Syntax formulierte Quelltexte, was hier mit den Hilfsmitteln nachvollzogen werden konnte.

Die das Entwicklungsboard begleitende 42seitige Dokumentation darf als gut bezeichnet werden: Sie beschreibt die Inbetriebnahme und die Signale auf allen Pins. Dokumentiert ist auch das Schaltbild. Für das Monitorprogramm ist nicht nur die Hantierung der Befehle erklärt, es wird sogar auch das Assembler-Quellprogramm des Monitors abgedruckt. Insofern ist auch ein kleines Lehrstück gegeben, wie man 6805-Prozessoren programmiert.

Bei eigener früherer Softwareentwicklung für 6805 hat der Autor keine Hilfsmittel wie den 'fritz05' und die ihn begleitenden Programme gehabt. Bei den Einchippern, in die man ja nicht hineinsehen kann, vermißte er dabei besonders die Möglichkeit der Ablaufkontrolle im Einzelschritt-Modus. Diese Dinge sind nun für eine effektive Programmierung gut gelöst, und man kann nur wünschen, daß die vielen Möglichkeiten der Einchipper für Steuerungen zunehmend benutzt werden.

Roland Löhr

S - Records

für den Datenaustausch

Übersicht

Die große Zahl der am Markt befindlichen Computertypen zieht nach sich, daß selten einmal Kompatibilität für den Datenaustausch zwischen ihnen besteht. Bei der Hardware sind nicht nur unterschiedliche Bussysteme in Benutzung, sondern auch die verschiedensten Formate für Floppy Disk und Aufzeichnungen auf Magnetbandgeräten. Wenn es sich nicht um sehr große Datenmengen handelt, kann man sich manchmal mit dem Brennen eines EPROMs behelfen, das man dann auf den empfangenden Computer aufsetzt und dort auswertet. In anderen Fällen bedient man sich des seriellen Datenverkehrs z.B. über RS232C-Schnittstellen. Wenn hier die Dinge auch elektrisch geregelt sind, so bleibt doch die Frage einer zuverlässigen Datenübertragung, für die verschiedene Kontrollen notwendig sind.

Die nachfolgend beschriebenen S-Records stellen ein Übermittlungsprotokoll für Motorola-Rechner dar. Sie enthalten zusätzlich zu den Daten Informationen, die der Kontrolle dienen. Mit Ihnen ist daher eine sehr sichere Übertragung von Programmen und Daten möglich. — Mit der zunehmenden Verbreitung von Rechnern mit einer Motorola-CPU wird daher jetzt oder später für den Leser die Frage bedeutungsvoll, wie man mit solchen S-Records umgeht. - Bei Rechnern mit dem Betriebssystem CP/M-68K hat man für das Senden solcher Records das Programm SENDC68, so z.B. auch beim neuen Atari 520ST.

Aufbau der S-Records

Es gibt verschiedene Typen der Records, sie haben jedoch den gleichen Aufbau. Zunächst zum benutzten Zeichensatz: Es werden nur die ASCII-Zeichen für 'S' (hex 53) sowie für die Zahlen 0-9 (hex 30-39) sowie für die Buchstaben A-F (hex 41 bis 46) benutzt. Dateninformation wird in 2 Halbbytes zerlegt, die nacheinander als 2 ASCII-Bytes gesendet wird, wobei eben nur ASCII-Repräsentationen für 0-9 und A-F benutzt werden. Jeder Record beginnt mit einem großen 'S':

S	Typ	Länge	Adresse	Daten	Kontrollsumme
Länge 1 Zch	1 Zch	2 Zch.	4/6/8 Zch.	variabel	2 Zch
ASCII	ASCII	1 Paar	2/3/4 Paare	Zeichenzahl= Paare	1 Paar

In vorstehendem Diagramm bedeutet der Begriff 'Paar', daß ein Hexbyte als 2 ASCII-Bytes gesendet wird. Wir haben damit den grundsätzlich einfachen Aufbau, daß die beiden ersten Zeichen eines Records unveränderte ASCII-Zeichen sind, während bei den Begriffen Länge, Adresse, Daten und Kontrollsumme aus Gründen der Datensicherheit eine Ausgabe in Paaren erfolgt.

Das Attribut Länge bezieht sich auf die Menge der Paare in Adresse, Daten und Kontrollsumme.

Die Kontrollsumme bezieht sich auf die Felder Länge, Adresse und Daten. Jedes zu diesen Feldern gehörende Bytepaar wird mit seinem Wert auf das Feld Kontrollsumme addiert. Der 'Wert' steht dabei im rechten Nibble des Ascii-Bytes, unter Abstreifung der ASCII-Vercodung also. Der Wert des höheren ASCII-Bytes eines Paares wird in das linke Nibble der Kontrollsumme addiert, der eines niedrigen in die rechte. Praktisch heißt das: Man kann die ASCII-Repräsentationen der Adressen und Daten direkt addieren und muß nur bei der 'Länge' aufpassen. - Nachdem die Kontrollsumme für die drei Felder Länge, Adresse und Daten gebildet ist, wird sie vor ihrer Ausgabe als 1er-Komplement umgewandelt (EOR #$FF).

Typen der S-Records

Es gibt die mit S0 bis S9 bezeichneten Record-Typen. S0 leitet dabei im allgemeinen eine Datenübertragung ein. Mit diesem Typ kann optional ein Name übertragen werden (Filename). Das Adressenfeld ist ausgenullt. Je nach Menge der zu übertragenden Daten schließen sich nun Daten-Records vom Typ S1, S2 oder S3 an. Hier wird jeweils im Feld 'Adresse' mitgeteilt, von welcher Adresse das erste übermittelte Datenbyte stammt. - Die Typen S1, S2 und S3 unterscheiden sich lediglich in der Breite des Feldes 'Adresse', die aus 2, 3 oder 4 Bytes besteht (gleiche Zahl von 'Paaren'). Man kann auch so sagen: S1 für 16 Bit Adreßbus, S2 für 24 und S3 für 32 Bit breiten Bus.

Auf die notwendige Zahl von Daten-Records folgt ein Schluß-Record. Es ist einer der Typen S9 (Abschluß für Typ S1), S8 (für S2) oder S7 (für S3). Wir haben damit folgendes Ablaufschema:

S0-Record (Name) .... Sx-Record (Daten) .... S(10-x)-Record als Schluß

Senden von S-Records

Obwohl der erwähnte Aufbau im Grunde genommen einfach ist, so zieht er doch einigen notwendigen Code beim Senden nach sich. Beim S0-Record zur Eröffnung muß man bereits die Namenslänge abgescannt haben, ehe man für diesen Record das Längenattribut hat (JSR NAMLEN). Der Name wird in einem Eingabepuffer erwartet. Hierzu und auch hinsichtlich der Beschickung der Pointeradressen BEGINN und ENDE sei auf 'Nützliche Routinen' in diesem Heft verwiesen.

Das Paket der Dienste ist modularisiert: KOPF erzeugt immer die ersten 2 Zeichen eines Records, abhängig vom übergebenen Parameter TYP. BYTES sendet immer 1 Paar, ADDCKS addiert zur Kontrollsumme, CKSOUT sendet diese. Im Bereich des Labels RECORD wird zunächst geprüft, ob die zu sendenden Daten schon erschöpft sind. Wenn ja, dann wird bei LASTRC der Abschluß-Record gesendet, wobei der notwendige Typ (S9, 8 oder 7) automatisch berechnet wird. - Die Einsprungspunkte S1REC, S2REC und S3REC dienen der Erzeugung von Datenrecords vom Typ S1, S2 oder S3.

Empfang von S-Records

Die Logik beim Empfang von S-Records ist ähnlich wie beim Senden: Es wird zunächst auf einen S0-Record gewartet. Wenn dieser einen Namen enthält, so wird dieser im Eingabepuffer festgehalten (beim EMPF2). Ab Label EMPREC werden Records vom Typ S1, S2 und S3 empfangen. Je nach Typ ist die unterschiedliche Länge der Adressenfelder zu berücksichtigen. Deren Inhalt wird in den Laufpointer übertragen und dient hernach für die indirekt adressierte Ablage der empfangenen Bytes. Bei EMPFEND schließlich werden Schluß-Records abgearbeitet. - Man beachte, daß der empfangende Prozessor bei hohen Baudraten in zeitliche Schwierigkeiten kommen kann, wenn er, wie der AIM 65, für den Empfang serieller Signale eine zeitaufwendige Routine wie GETTTY benutzt.

```
000000                  0001 ;PROGRAMM ZUM SENDEN UND EMPFANGEN VON MOTOROLA S-RECO
RDS
000000                  0002 ;AUG. 85
000000                  0003
000000                  0004 ;VEREINBARUNG VON KONSTANTEN
000000                  0005 NULLCH =$30              ;ASCII-NULL
000000                  0006
000000                  0007 PAKET  =16               ;MENGE DER DATENBYTES JE RECOR
D
000000                  0008
000000                  0009 ;VARIABLEN UND POINTER IN DER ZERO PAGE
000000                  0010         *=0
000000                  0011 INBUFF *=*+80            ;EINGABEPUFFER, NAMEN ENTHALTE
ND
000050                  0012 INBUFP *=*+1             ;ZEIGER IN DEN EINGABEPUFFER
000051                  0013 BEGINN *=*+4             ;STARTADRESSE BEIM SENDEN/EMPF
ANGEN
000055                  0014 LAUFPT =BEGINN           ;LAUFPOINTER, SPEICHERZEIGER
000055                  0015 ENDE   *=*+2             ;ENDADRESSE FUER DAS ABSENDEN
000057                  0016 TYP    *=*+1             ;TYP DES S-RECORDS
000058                  0017 MENGE  *=*+1             ;BYTES IM S-RECORD, ALS 2 ASCI
I-ZEICHEN SENDEN
000059                  0018 CKSUM  *=*+1             ;KONTROLLSUMME, DITO
00005A                  0019 TEMP   *=*+1             ;ZWISCHENSPEICHER
00005B                  0020 COUNT  *=*+1             ;MENGE DATENBYTES ZU SENDEN
00005C                  0021 TRUEFL *=*+1             ;FLAG F. FEHLERHAFTEN EMPFANG
00005D                  0022
00005D                  0023 ;BENUTZTE SYSTEMROUTINEN
00005D                  0024 OUTALL =$E9BC            ;AUSGABE 1 ZCH. AUF AKTIVEN KA
NAL
00005D                  0025 GETTTY =$EBDB            ;EINGABE 1 ZCH. VOM SERIELLEN
KANAL
00005D                  0026 MONITR =$E182            ;SPRUNG ZUM BETRIEBSSYSTEM
00005D                  0027
00005D                  0028         *=$6000          ;PROGRAMMSTART FUER DAS SENDEN
006000                  0029 START
006000  A9 30           0030 SOREC   LDA #NULLCH      ;INIT UND SENDE 'S0'
006002  85 57           0031         STA TYP          ;TYP S0
006004  20 CF 60        0032         JSR KOPF         ;S0 GESENDET
006007  20 DD 60        0033         JSR NAMLEN       ;BESTIMMUNG GESAMTLAENGE
00600A  20 EE 60        0034         JSR LENOUT       ;LAENGE AUSGEBEN, 2 BYTES
00600D  A0 04           0035         LDY #4           ;4 NULLEN AUSGEBEN
00600F  20 36 61        0036         JSR NULOUT
006012  A9 00           0037         LDA #0
006014  85 50           0038         STA INBUFP       ;ZEIGER EINGABEPUFFER
006016  A6 50           0039 SOREC1 LDX INBUFP
00601B  B5 00           0040         LDA INBUFF,X     ;HOLE NAMEN
00601A  F0 09           0041         BEQ SOREC2       ;STRING FERTIG
00601C  E8              0042         INX
00601D  86 50           0043         STX INBUFP       ;ZEIGE AUF FOLGENDES ZEICHEN
00601F  20 F0 60        0044         JSR BYTES        ;ADDIERE Z. CKSUM UND GIB AUS
006022  4C 16 60        0045         JMP SOREC1
006025                  0046
006025  20 06 61        0047 SOREC2 JSR CKSOUT        ;CKSUM AUSGEBEN
006028  A9 0D           0048         LDA #$0D         ;CARRIAGE RETURN NUR FUER BILD
SCHIRM!
00602A  20 BC E9        0049         JSR OUTALL
00602D                  0050
00602D  A9 31           0051 S1REC   LDA #'1'         ;SENDE S1-RECORD
```

```
00602F  85 57        0052 ALLTYP STA TYP
006031  A5 57        0053 RECORD LDA TYP          ;TYP BESTIMMT LAENGE
006033  29 0F        0054        AND #$0F
006035  18           0055        CLC
006036  69 12        0056        ADC #PAKET+2     ;# BYTES ADRESSE + CKSUM
006038  85 58        0057        STA MENGE        ;DEFAULT LAENGE RECORD
00603A  A2 10        0058        LDX #PAKET       ;DEFAULT MENGE DATENBYTES
00603C  18           0060        CLC              ;BESTIMME RESTMENGE ZU SENDEND
ER DATEN
00603D  A5 55        0061        LDA ENDE         ;LOW VALUE
00603F  69 01        0062        ADC #1           ;BYTE BEI ENDE EINZUSCHLIESSEN
006041  38           0063        SEC
006042  E5 51        0064        SBC LAUFPT
006044  85 5A        0065        STA TEMP
006046  A5 56        0066        LDA ENDE+1       ;HIGH
006048  E5 52        0067        SBC LAUFPT+1
00604A  D0 16        0068        BNE WEITER       ;NOCH MEHR ALS 1 PAGE
00604C  A5 5A        0069        LDA TEMP
00604E  F0 54        0070        BEQ LASTRC       ;DATEN FERTIG, NUN LAST RECORD
006050  C9 10        0071        CMP #PAKET
006052  B0 0E        0072        BCS WEITER       ;NORMALMENGE SENDEN
006054  AA           0073        TAX              ;MENGE MERKEN
006055  85 5A        0074        STA TEMP
006057  A5 57        0075        LDA TYP          ;MENGE VOM TYP ABHAENGIG
006059  29 0F        0076        AND #$0F
00605B  18           0077        CLC
00605C  69 02        0078        ADC #2           ;F. ADRESSBYTES UND CKSUM
00605E  65 5A        0079        ADC TEMP
006060  85 58        0080        STA MENGE        ;PARAMETER DER AUSGABE
006062  86 5B        0081 WEITER STX COUNT        ;DATENMENGE
006064  20 CF 60     0082        JSR KOPF         ;BEGINN DER AUSGABE
006067  20 EE 60     0083        JSR LENOUT       ;LAENGE AUSGEBEN
00606A  A5 57        0084        LDA TYP          ;BESTIMMT ADRESS-LAENGE
00606C  29 0F        0085        AND #$0F
00606E  A8           0086        TAY              ;Y= ADRESSER + ZAEHLER
00606F  B9 51 00     0087 WEITRO LDA LAUFPT,Y     ;ADRESSE AUSGEBEN
006072  20 F0 60     0088        JSR BYTES
006075  88           0089        DEY
006076  10 F7        0090        BPL WEITRO
006078  A0 00        0091        LDY #0           ;NUN DATENBYTES AUSGEBEN
00607A  B1 51        0092 WEITR1 LDA (LAUFPT),Y   ;HOLEN
00607C  20 F0 60     0093        JSR BYTES
00607F  C8           0094        INY
006080  C4 5B        0095        CPY COUNT
006082  90 F6        0096        BCC WEITR1       ;NOCH NICHT FERTIG
006084  20 06 61     0097        JSR CKSOUT       ;CKSUM AUSGEBEN
006087  A9 0D        0098        LDA #$0D         ;CR NUR F. BILDSCHIRM!
006089  20 BC E9     0099        JSR OUTALL       ;RECORD NUN FERTIG
00608C  18           0100        CLC              ;NUN LAUFPOINTER ERHOEHEN
00608D  A5 5B        0101        LDA COUNT        ;MENGE GESENDETER DATENBYTES
00608F  65 51        0102        ADC LAUFPT
006091  85 51        0103        STA LAUFPT
006093  90 0A        0104        BCC WEITR2       ;KEIN UEBERTRAG
006095  E6 52        0105        INC LAUFPT+1
006097  D0 06        0106        BNE WEITR2
006099  E6 53        0107        INC LAUFPT+2
00609B  D0 02        0108        BNE WEITR2
00609D  E6 54        0109        INC LAUFPT+3
00609F  A5 57        0110 WEITR2 LDA TYP
0060A1  4C 31 60     0111        JMP RECORD       ;WEITEREN RECORD ODER ENDE
0060A4               0112
0060A4               0113 LASTRC              ;LAST RECORD ABHAENGIG VON TYP
0060A4  38           0114        SEC
0060A5  A9 3A        0115        LDA #$3A         ;ENTSP. ASCII '9' +1
0060A7  E5 57        0116        SBC TYP
0060A9  09 30        0117        ORA #$30         ;WIEDER ZU ASCII
0060AB  85 57        0118 LASTR2 STA TYP          ;TYP NUN BESTIMMT
0060AD  A2 03        0119        LDX #3           ;ZU SENDENDE LAENGE
0060AF  86 5B        0120        STX MENGE
0060B1  20 CF 60     0121        JSR KOPF
0060B4  20 EE 60     0122        JSR LENOUT       ;AUSGABE LAENGE
0060B7  A0 04        0123        LDY #4           ;4 NULLEN
0060B9  20 36 61     0124        JSR NULOUT
```

```
0060BC  20 06 61    0125            JSR CKSOUT          ;CHKSUM
0060BF  A9 0D       0126            LDA #$0D            ;CR NUR F. BILDSCHIRM!
0060C1  20 BC E9    0127            JSR OUTALL
0060C4  4C 82 E1    0128            JMP MONITR          ;ZUM BETRIEBSSYSTEM
0060C7              0130 ;EINSPRUNGSPUNKTE    R RECORD-TYPEN S2 UND S3
0060C7  A9 32       0131 S2REC      LDA #'2'
0060C9  2C          0132            .BYT $2C            ;SKIP
0060CA  A9 33       0133 S3REC      LDA #'3'
0060CC  4C 2F 60    0134            JMP ALLTYP          ;WEITER WIE UEBLICH
0060CF              0135
0060CF              0136 KOPF
0060CF  A9 00       0137            LDA #0              ;S-RECORD TYP AUSGEBEN
0060D1  85 59       0138            STA CKSUM           ;NUN S0, S1 ETC. SENDEN
0060D3  A9 53       0139            LDA #'S'
0060D5  20 BC E9    0140            JSR OUTALL          ;SENDE DAS S
0060D8  A5 57       0141            LDA TYP
0060DA  4C BC E9    0142            JMP OUTALL          ;RTS FINDET SICH DORT
0060DD              0143
0060DD  A2 00       0144 NAMLEN     LDX #0              ;NAME IM EINGABEPUFFER, BEGREN
ZT DURCH 0
0060DF  B5 00       0145 NAMLE1     LDA INBUFF,X
0060E1  F0 04       0146            BEQ NAMEND
0060E3  E8          0147            INX
0060E4  4C DF 60    0148            JMP NAMLE1 ,LET'S DO IST AGAIN
0060E7  8A          0149 NAMEND     TXA                 ;X ENTHIELT DIE NAMENS-LAENGE
0060E8  18          0150            CLC                 ;NUN + 3 ZUR LAENGE FUER 2 NUL
LEN +
0060E9  69 03       0151            ADC #3              ;UND ZU SENDENDE CKSUM
0060EB  85 58       0152            STA MENGE
0060ED  60          0153            RTS
0060EE              0154
0060EE  A5 58       0155 LENOUT     LDA MENGE           ;MENGE ALS 2 ASCIBYTES AUSGEBE
N
0060F0  20 FE 60    0156 BYTES      JSR ADDCKS          ;UPDATE CKSUM
0060F3  20 16 61    0157            JSR NUMWA           ;HEX ZU ASCII MACHEN
0060F6  20 FB 60    0158            JSR BYTOUT          ;AUSGEBEN + ZU CKSUM
0060F9  A5 5A       0159            LDA TEMP            ;DITO, NIEDERWERTIGE ANGABE
0060FB  4C BC E9    0160 BYTOUT     JMP OUTALL          ;RTS FINDET SICH DORT
0060FE              0161
0060FE  48          0162 ADDCKS     PHA                 ;ADDIERE BYTE ZU CKSUM
0060FF  18          0163            CLC                 ;ADDIERE WERT DES ZEICHENS ZUR
 KONTROLLSUMME
006100  65 59       0164            ADC CKSUM
006102  85 59       0165            STA CKSUM
006104  68          0166            PLA                 ;ZEICHEN ZURUECK
006105  60          0167            RTS
006106              0168
006106  A5 59       0169 CKSOUT     LDA CKSUM           ;WERT DER KONTROLLSUMME NACHSE
NDEN
006108  49 FF       0170            EOR #$FF            ;BILDE 1ER KOMPLEMENT
00610A  20 16 61    0171            JSR NUMWA           ;ZU ASCII MACHEN
00610D  20 BC E9    0172            JSR OUTALL
006110  A5 5A       0173            LDA TEMP
006112  4C BC E9    0174            JMP OUTALL          ;RTS DORT
006115              0175
006115  40          0176 BIT6       .BYT $40
006116              0177
006116  48          0178 NUMWA      PHA                 ;UMWANDLUNG 1 HEXBYTE=2 ASCII-
BYTES
006117  2C 15 61    0179            BIT BIT6            ;SETZE V-FLAG
00611A  29 0F       0180            AND #$0F
00611C  20 24 61    0181            JSR NUMWA2
00611F  68          0182            PLA
006120  4A          0183            LSR A               ;ISOLIERE HOEHERWERTIGEN TEIL
006121  4A          0184            LSR A
006122  4A          0185            LSR A
006123  4A          0186            LSR A               ;4XRECHTS
006124  09 30       0187 NUMWA2     ORA #$30            ;MACH ZU ASCII
006126  C9 3A       0188            CMP #$3A            ;UEBERLAUF?
006128  90 04       0189            BCC NUMWA3
00612A  08          0190            PHP                 ;V-BIT NICHT D. ADC ZERSCHIESS
EN LASSEN!
00612B  69 06       0191            ADC #6              ;FUER A...F
00612D  28          0192            PLP
```

```
00612E  50 03        0193 NUMWA3 BVC NUMWA4       ;BEIM 2. DURCHGANG
006130  85 5A        0194        STA TEMP         ;NIEDERWERTIGEN TEIL
006132  B8           0195        CLV              ;RESET V-FLAG
006133  60           0196 NUMWA4 RTS
006134               0197
006134  A0 20        0198 DELAY  LDY #32          ;GEGENSTATION ZEIT GEBEN
006136  84 5A        0199 NULOUT STY TEMP
006138  A9 30        0200 DELAY1 LDA #NULLCH      ;NULLEN SENDEN
00613A  20 BC E9     0201        JSR OUTALL
00613D  C6 5A        0202        DEC TEMP
00613F  D0 F7        0203        BNE DELAY1
006141  60           0204        RTS
006142               0205
006142               0206 ;*****************
006142               0207 ;EMPFANGEN VON S-RECORDS
006142               0208 ;WARTE ZUNAECHST AUF S0-RECORD
006142               0209        *=START+$200
006200  A9 00        0210 EMPF0  LDA #0           ;INITIALISIERUNG
006202  85 59        0211        STA CKSUM        ;KONTROLLSUMME
006204  85 5C        0212        STA TRUEFL       ;=KEIN FEHLER BISHER
006206  AA           0213        TAX              ;X=0 F. EVTL. NAMENSABLAGE
006207  20 DB EB     0214 EMPF1  JSR GETTTY       ;WARTE AUF DAS 1. 'S'
00620A  C9 53        0215        CMP #'S'
00620C  D0 F9        0216        BNE EMPF1
00620E  20 DB EB     0217        JSR GETTTY       ;0 MUSS FOLGEN
006211  C9 30        0218        CMP #'0'
006213  D0 F2        0219        BNE EMPF1        ;FANGE VON VORNE AN!
006215  20 C8 62     0220        JSR GET2         ;LAENGE MUSS FOLGEN
006218  85 58        0221        STA MENGE
00621A  20 C8 62     0222        JSR GET2         ;PACKE 2 BYTES UND ADDIERE
00621D  20 C8 62     0223        JSR GET2         ;ADRESSE MIT 00 ERWARTET/ERLED
IGT
006220  A5 58        0224 EMPF2  LDA MENGE        ;WENN 1, DANN NAME ERLEDIGT
006222  C9 01        0225        CMP #1
006224  F0 0B        0226        BEQ EMPCKS       ;KONTROLLSUMME ERWARTEN/VERGLE
ICHEN
006226  20 C8 62     0227        JSR GET2         ;ZEICHEN DES NAMENS KOMMEN
006229  A5 5A        0228        LDA TEMP
00622B  95 00        0229        STA INBUFF,X
00622D  E8           0230        INX
00622E  4C 20 62     0231        JMP EMPF2
006231               0232
006231  A9 00        0233 EMPCKS LDA #0
006233  95 00        0234        STA INBUFF,X     ;NAMEN IM PUFFER BEGRENZEN
006235  20 D0 62     0235        JSR GETHEX       ;KONTROLLSUMME?
006238  49 FF        0236        EOR #$FF         ;1ER KOMPLEMENT DER EMPFANGENE
N SUMME
00623A  C5 59        0237        CMP CKSUM        ;GLEICH?
00623C  F0 05        0238        BEQ EMPREC       ;JA, HOLE JETZT DATEN-RECORDS
00623E  E6 5C        0239 EMPFER INC TRUEFL       ;FEHLER
006240  4C 82 E1     0240 EMPFZZ JMP MONITR       ;ZWANGSABBRUCH
006243               0241
006243  20 DB EB     0242 EMPREC JSR GETTTY       ;JETZT DATEN- ODER SCHLUSS-SAT
Z
006246  C9 53        0243        CMP #'S'         ;MUSS MIT 'S' BEGINNEN
006248  D0 F9        0244        BNE EMPREC       ;FEHLER ODER SONDERZEICHEN
00624A  20 DB EB     0245        JSR GETTTY       ;TYP
00624D  C9 37        0246        CMP #'7'         ;EIN SCHLUSS-TYP?
00624F  B0 3C        0247        BCS EMPEND       ;JA
006251  85 57        0248        STA TYP          ;FESTHALTEN
006253  20 C8 62     0249        JSR GET2         ;HOLE LAENGE
006256  85 58        0250        STA MENGE
006258               0251
006258  A5 57        0252        LDA TYP          ;WIEVIELE ADRESS-BYTES? (-1)
00625A  29 0F        0253        AND #$0F
00625C  85 5B        0254        STA COUNT        ;SICHERN
00625E  20 C8 62     0255 EMPRC1 JSR GET2
006261  A4 5B        0256        LDY COUNT
006263  99 51 00     0257        STA LAUFPT,Y     ;ADRESSE IN DEN POINTER
006266  C6 5B        0258        DEC COUNT
006268  10 F4        0259        BPL EMPRC1
00626A               0260
00626A  A0 00        0261        LDY #0           ;Z. POINTERADRESSIERUN
00626C  84 5B        0262        STY COUNT        ;ABLAGE ZUR VORSICHT
```

```
00626E  A5 58        0263 EMPRC2 LDA MENGE     ;WENN 1, DANN DATEN ERLEDIGT
006270  C9 01        0264        CMP #1
006272  F0 0E        0265        BEQ EMPRCX    ;KONTROLLSUMME ERWARTEN/VERGLE
ICHEN
006274  20 C8 62     0266        JSR GET2      ;DATEN KOMMEN
006277  A5 5A        0267        LDA TEMP
006279  A4 5B        0268        LDY COUNT
00627B  91 51        0269        STA (LAUFPT),Y
00627D  E6 5B        0270        INC COUNT
00627F  4C 6E 62     0271        JMP EMPRC2
006282               0272
006282  20 D0 62     0273 EMPRCX JSR GETHEX    ;KONTROLLSUMME VERGLEICHEN
006285  49 FF        0274        EOR #$FF      ;1ER KOMPLEMENT DER EMPFANGENE
N SUMME
006287  C5 59        0275        CMP CKSUM     ;GLEICH?
006289  D0 B3        0276        BNE EMPFER    ;FEHLER, ABBRUCH
00628B  F0 B6        0277        BEQ EMPREC    ;HOLE WEITERE DATEN-RECORDS
00628D               0278
00628D  20 DB EB     0279 EMPEND JSR GETTTY    ;JETZT DATEN- ODER SCHLUSS-SAT
Z
006290  C9 53        0280        CMP #'S'      ;MUSS MIT 'S' BEGINNEN
006292  20 BC E9     0281        JSR OUTALL
006295  D0 F6        0282        BNE EMPEND    ;FEHLER ODER SONDERZEICHEN
006297  20 DB EB     0283        JSR GETTTY    ;TYP
00629A  85 5A        0284        STA TEMP
00629C  20 BC E9     0285        JSR OUTALL
00629F  A5 5A        0286        LDA TEMP
0062A1  18           0287        CLC           ;SUMME DER TYPEN MUSS $6A ERGE
BEN
0062A2  65 57        0288        ADC TYP
0062A4  C9 6A        0289        CMP #$6A
0062A6  D0 1D        0290        BNE EMPEN2    ;FEHLER
0062A8  20 C8 62     0291        JSR GET2      ;LAENGE MUSS FOLGEN
0062AB  85 58        0292        STA MENGE
0062AD  20 C8 62     0293        JSR GET2      ;PACKE 2 BYTES UND ADDIERE
0062B0  20 C8 62     0294        JSR GET2      ;ADRESSE MIT 00 ERWARTET/ERLED
IGT
0062B3  A5 58        0295 EMPEN1 LDA MENGE     ;WENN 1, DANN ERLEDIGT
0062B5  C9 01        0296        CMP #1
0062B7  D0 85        0297        BNE EMPFER    ;FEHLER
0062B9  20 D0 62     0298        JSR GETHEX    ;KONTROLLSUMME?
0062BC  49 FF        0299        EOR #$FF      ;1ER KOMPLEMENT DER EMPFANGENE
N SUMME
0062BE  C5 59        0300        CMP CKSUM     ;GLEICH?
0062C0  D0 03        0301        BNE EMPEN2
0062C2  4C 82 E1     0302        JMP MONITR
0062C5  4C 3E 62     0303 EMPEN2 JMP EMPFER    ;FEHLER
0062C8               0304
0062C8  20 D0 62     0305 GET2   JSR GETHEX    ;HOLE 2 BYTE, ADDIERE WERT ZU
CKSUM
0062CB  C6 58        0306        DEC MENGE
0062CD  4C FE 60     0307        JMP ADDCKS    ;ADDIERE ZU CKSUM, RTS DORT
0062D0               0308
0062D0  20 DB EB     0309 GETHEX JSR GETTTY    ;ZEICHEN VON EINGABE HOLEN
0062D3  C9 3A        0310        CMP #$3A      ;VERWANDLE ASCII ZU HEX
0062D5  90 03        0311        BCC GETHX1
0062D7  38           0312        SEC
0062D8  E9 07        0313        SBC #7
0062DA  0A           0314 GETHX1 ASL A         ;STRIP OFF ASCII
0062DB  0A           0315        ASL A         ;BILDE MSD
0062DC  0A           0316        ASL A
0062DD  0A           0317        ASL A
0062DE  85 5A        0318        STA TEMP
0062E0  20 DB EB     0319        JSR GETTTY    ;LOW BYTE
0062E3  C9 3A        0320        CMP #$3A      ;VERWANDLE ASCII ZU HEX
0062E5  90 03        0321        BCC GETHX2
0062E7  38           0322        SEC
0062E8  E9 07        0323        SBC #7
0062EA  29 0F        0324 GETHX2 AND #$0F      ;STRIP OFF ASCII
0062EC  05 5A        0325        ORA TEMP
0062EE  85 5A        0326        STA TEMP
0062F0  60           0327        RTS
```

Editorial

Fast täglich ist jetzt den Medien zu entnehmen, daß die Computerindustrie Absatz- und Beschäftigungssorgen hat. Auch große Hersteller legen ganze Produktionslinien still. Der mangelnde Absatz wirkt auf die Hersteller von Halbleitern zurück. Selbst für die modernen hochintegrierten Schaltungen, wie RAM 41256 oder EPROM 27256 haben haben sich äußerst niedrige Preise am Markt eingestellt, so daß es sich lohnt, sie für neue Entwicklungen in Betracht zu ziehen: Die Bauteile für einen Megabyte-RAM-Speicher kosten nur noch etwa DM 300. Man halte sich das vor Augen, wenn man die Beschaffung von Erweiterungsmodulen erwägt. In vielen Fällen wird man deutliche Abschläge gegenüber den früher herausgelegten Preisen aushandeln können, zumal ja auch der Wert des Dollars zurückgegangen ist.

Der preisliche Wettkampf betrifft natürlich auch die Computer selbst. Beim Verkauf vieler Gerätetypen kann der Handel kaum noch etwas verdienen. Jede zusätzliche Bemühung in der Beratung oder bei der Bearbeitung einer Reklamation kostet bares Geld, soweit man dafür überhaupt Fachpersonal hat. Man lasse sich also nicht allein vom günstigen Preis leiten, denn erfahrungsgemäß braucht man im Laufe der Zeit doch einmal Hilfe, weil ein Teil defekt geworden ist. Bei den Millionen Schaltkreisen, die ein Rechner heute hat, ist das, statistisch gesehen, eigentlich auch kein Wunder. Preis und Service sind sicher wichtige Merkmale. Wie hier aber immer wieder betont wird, ist sehr auch auf die Software und die Dokumentation zu achten, die zum Rechner zur Verfügung steht. Wenn man nicht immer wieder alles selber machen will oder kann, dann sind dies vielleicht die entscheidenden Punkte.

Computer werden nach wie vor sehr früh angekündigt. Trotz des Überflusses an Bauteilen werden dabei die ursprünglich genannten Termine der Lieferbarkeit überschritten. Das gibt dem Wettbewerb Gelegenheit, auf einer Pressekonferenz oder bei einer Ausstellung das 'noch leistungsfähigere' Modell in einer Geräteklasse zu präsentieren. Der Interessent ist damit zunächst einmal verunsichert und hält sich wohlmöglich zurück. - Jetzt im Spätsommer 1985 läßt sich schwer voraussagen, wohin die Reise im Herbst und Winter bei Personal Computern gehen wird. Man sieht hier einserseits zwar eine Hinwendung zum IBM-PC und seinen Nachbauten (der PC 10 von Commodore wird gut verkauft), bei den übrigen hinzutretenden PC's muß zunächst einmal der Markt entscheiden, wie sie aufgenommen werden. In der Anfangsphase einer Markteinführung gibt es immer einen gewissen aufgestauten Bedarf von Neugierigen. Erst Monate später weiß man, ob sich ein Modell wirklich durchsetzen wird, weil es nach Leistung, Bedienungskomfort und Programmen einen großen Kreis von Betreibern anspricht.

Im vorliegenden Heft sind wieder viele Lösungsvorschläge für häufig vorkommende Aufgaben abgedruckt worden. Sie mögen dem Leser bei seinen eigenen Entwicklungen hilfreich sein. Der Herausgeber hat in vielen Gesprächen den Eindruck gewonnen, daß Vorschäge in mehr grundlegenden Darstellungen notwendig sind, um die Programmentwicklung zu erleichtern. Es kommt auf die geeigneten Instrumente an und natürlich auch auf das bessere Verstehen. – In diesem Sinne darf ich die Leser bitten, auch künftig Beiträge zu dieser Zeitschrift einzusenden, die für einen größeren Leserkreis von Interesse sein können.

Bücher

Field, T.: MacWrite und MacPaint. Verlag McGraw-Hill, Hamburg 1985, 152 S., DM 39,80, ISBN 3-89-28-029-3. Dieses Buch zu den Programmen des Macintosh wurde aus dem Amerikanischen übersetzt. Es erklärt die Benutzung der Text- und Grafikmöglichkeiten anhand zahlreicher Abbildungen, die der Darstellung auf dem Bildschirm des Rechners entsprechen. Die Beispiele beziehen sich auf die Gestaltung von Geschäftsberichten, Brief- und Formularköpfen, Formularsatz, Illustrationen usw. Dabei wird die Erfindungsgabe des Lesers angeregt. Das sauber aufgemachte Buch darf den Betreibern daher empfohlen werden, wenn sie sich in Text und Grafik des Rechners einarbeiten wollen.

Thies, Klaus-Dieter: Die 8087/80287 Numerischen Prozessor-Erweiterungen. te-wi-Verlag, München 1985,355 S., DM 69,--, ISBN 3-921803-53-5. Für Systeme mit einer CPU 8086 oder 80286 gibt es die Möglichkeit, einen numerischen Coprozessor vom Typ 8087/80287 einzusetzen, der

Rechenabläufe wesentlich beschleunigt.Diesem Thema ist das vorliegende Buch gewidmet. Es hat folgende Kapitel: Das 8086/8087-System, Architektur des 8087, Environment, Interruptverarbeitung bei 8086/8087, Initialisieren des 8087, Datentypen und -formate, Adressierung der NDP-Operanden, Befehlssatz des 8087, Befehlsliste, Unterschiede 8087/80287, Programmierbeispiele in Assemblersprache. - Mit dem Einsatz eines Gleitpunktprozessors gelangt man als Programmierer in eine vollkommen neue Umgebung. Nicht nur, daß man etwa 120 zusätzliche numerische Befehle zur Verfügung hat, man kann auch mit sieben unterschiedlichen Datentypen arbeiten. Zusätzlich muß man auch den vom Hauptprozessor erwarteten Rapport beachten. Ein solches Umfeld braucht Dokumentation und Erklärungen. - Das Buch ist aus einer Schulungsunterlage auf Industriestandard entstanden, es ist in seinen Befehls- und Funktionsbeschreibungen datenblatt-genau. Auf fast allen Seiten findet man Systembilder, mit denen die Ausführungsphasen verfolgt werden können, und die Merkpunkte sind grafisch hervorgehoben. Der Rezensent bekennt, daß er kein Fachmann für die Prozessorfamilie und für das besondere Thema ist. Er sieht jedoch, daß dieses Buch übersichtlich aufgemacht und daß sein Text klar formuliert worden ist, so daß man bei der Einarbeitung und beim späteren Nachschlagen zur Kontrolle sehr gut zurechtfindet.

West, Raeto: C-64/SX-64 Computer Handbuch. te-wi-Verlag, München 1985, 688 Seiten, Softcover, ISBN 3-921 803-24-1, DM 66,--. Der C-64 von Commodore ist in den letzten drei Jahren weltweit einige millionenmal verkauft worden und wurde damit zum bisher größten Computererfolg. Er zog eine ganze 'Zulieferindustrie' nach sich für Anschlußplatinen, Spiele und Veröffentlichungen in z.T. Spezialzeitschriften und Büchern. Man wird wohl auch sagen dürfen, daß er Computer insgesamt verbilligte, nicht nur durch seinen eigenen Preis (z.Zt. etwas unter DM 500,--), sondern auch dadurch, daß er bei Peripheriegeräten, z.B. Druckern und Schreibwerken eine kostengünstigere Massenproduktion ermöglichte. Wenn man es so sieht, dann hat er als Heimcomputer wohl weniger Arbeitsplätze vernichtet, als weltweit ermöglicht.

Zum vorliegenden Buch wird vermerkt, daß Raeto West etwa ein Jahr Arbeit in seine Abfassung investiert habe. Nicht nur nach seinem Umfang, sondern auch nach seinem Inhalt darf es damit als das umfassendste und wohl auch seriöseste Buch zum Computer bezeichnet werden. Mancher Leser mag zwar bedauern, daß es erst jetzt erscheint. Gleichwohl erscheint es wohl nicht zu spät, denn der C-64 läuft noch immer, und das Buch wird wohl auch solchen Betreibern helfen, die jetzt den größeren Nachfolger, den PC 128 anschaffen. Dieser hat auch eine Betriebsweise 'C-64' und dürfte, nach näherem Studium, wohl auch sonst viele Parallelen aufweisen.

Das Buch enthält auf eng bedruckten Seiten soviel Information, daß wir nur im Überblick darauf eingehen können. Es sind im Ablauf der Darstellungen über 300 Programmbeispiele in BASIC und in Assemblersprache enthalten. Viele dieser Beispiele decken ernsthafte Aufgaben und Lösungen ab. Man muß auch sagen, daß der Autor ohne überflüssige Worte immer das Wesentliche herausarbeitet. So ist der Kern des BASIC nebst Beispielen und Hinweisen auf Besonderheiten in bereits 57 Seiten abgehandelt. Aus den nachfolgenden Kapiteln erwähnen wir folgende abgehandelten Themen: Effektives Programmieren in BASIC (Laufzeiten, Sortieren, Suchen, Mischen), Architektur des C-64, BASIC für Professionelle, maschinensprachliche Programmierung und ihr Anschluß an BASIC, ROM-Führer, Speicherbelegung, Grafik, Ton und Musik, Betrieb mit Cassette, Diskette, Spielports, Drucker, Plotter und Modems. Der Anhang enthält etwa 16 Dokumentationen und zwei Monitorprogramme. - Als interessierter Betreiber des C-64 und seines Nachfolgers sollte man nach Auffassung des Rezensenten dieses Buch als allererstes zur Arbeitsgrundlage wählen.

Bassel, H.: Umweltdynamik, 30 Programme für kybernetische Umwelterfahrungen auf jedem BASIC-Rechner. te-wi-Verlag, München 1985, 466 S., DM 59,--, ISBN 3-921 803-36-5. Der Autor ist Professor für Umweltschutz an der GHS Kassel und schreibt seit 1960 Simulationsprogramme für dynamische Systeme. Er war Mitarbeiter am 2. Weltmodell des Club of Rome, dessen Veröffentlichungen in der zweiten Hälfte der 70er Jahre zu einiger Unruhe und auch zu Widerspruch führten, weil das Hochrechnen von Trends in die Zukunft mit Vorsicht zu beurteilen ist. Gleichwohl ist dieses Buch interessant. Es macht den Versuch, Bewegungsgesetze unserer komplexen Umwelt in ihrer Verwobenheit zu erkennen. Es zeigt, wie man Modelle aufstellt. Für den Nachvollzug auf Home-Computern (C-64, IBM-PC, Sinclair-Spectrum, Apple II und Epson HX 20) werden in der Sprache BASIC 29 Modelle zu aktuellen Themen berechenbar gemacht.Diese Modelle beziehen

sich auf folgende Bereiche: Wachstum, Zerfall und Fließgleichgewicht, Fressen und Gefressenwerden, Aufbau und Zersetzung, Das lautlose Wirken der Sonne, Wettstreit um Nährstoffe und Licht, Wasser, Säen, Wachsen, Ernten, Essen, Energie, Rohstoffe und Dynamik über Jahrhunderte. - In den einzelnen Kapiteln sind Grafiken für die Modelle und die Ausgaben des Computers enthalten, so daß man hier schon die Auswirkung der wesentlichen Einflußgrößen erkennen kann. Etwas bequemer wird man sicherlich an den Stoff herangeführt, wenn man die zu den einzelnen Computern lieferbare Diskette (DM 29,--) benutzt, um selber ein Gespür für die Zusammenhänge zu bekommen. Abgesehen von den Einwänden, die man zu mehr mathematisch hergeleiteten Prognosen nicht vergessen sollte, scheint die hier gegebene Einarbeitung in komplexe Modelle gleichwohl sehr interessant.

Aus der Branche

Die neuen Prozessoren 65802 und 65816 mit 8/16 Bit-Verarbeitung und erweitertem Befehlssatz: Der Zweithersteller GTE Microcircuits, München, teilte Ende Juli mit: Preise sowie Muster für die 2 und 4 MHz-Versionen sollen ab Aug./Sept. verfügbar sein. Im Nov./Dez. sollen diese Versionen die Produktionsfreigabe erhalten und damit regelmäßig lieferbar sein. - Der Ersthersteller, Western Design Center ist in Düsseldorf durch die Firma Unitronic vertreten. Man ist dort in der Lage, Musterstücke abzugeben. Der Preis ist noch relativ hoch. Man erwartet jedoch, daß er nach der Aufnahme der regelmäßigen Produktion am Jahresende deutlich unter DM 100 sinken sollte. - Es wird berichtet, daß das Western Design Center Pläne zur Entwicklung noch leistungsfähigerer Zentraleinheiten in der 65xx-Familie hat.

Motorola: Einem Rundschreiben der Firma ist zu entnehmen, daß der 32 Bit Prozessor 68020 jetzt ab Lager lieferbar ist. Von Benchmarks wird berichtet, daß er ohne weiteres mit einer VAX 11/780 mithalten kann. - Weitere Peripheriebausteine für die Familie sind der Floating Point Coprozessor (68881), der Multifunktionschip 68901, die DMA-Controller 68440/442/450, der X25-Controller 68605 und die Paged Memory Management Unit 68461. - Für die schnellen 68020-Systeme wurden sehr schnelle statische RAMS mit 25-55 ns. Zugriffszeit entwickelt: MCM6178H, 6164H und 6167H. - Im Verlaufe der SYSTEMS in München ist am 30.10. und 31.10. eine Veranstaltung aller VME-Hersteller (z.Zt. ca. 250) im Hotel Hilton vorgesehen.

FORTH: euroFORML Conference vom 25.-27. Okt. 1985 auf Schloss Stettenfels bei Heilbronn. Die FORTH-Gesellschaft eV. und die amerikanische FORTH Interest Group, Inc. führen ein internationales Treffen mit Konferenz zum Thema Software Metrics durch. Anmeldungen sollten bis Anfang September bei der Forth-Gesellschaft erfolgen: Schanzenstr. 27, 2000 Hamburg 6, Tel. 43 50 70.

ATARI: Über den neuen Rechner 520ST wurde an anderer Stelle in diesem Heft vorläufig berichtet. Einer Pressemitteilung der Firma ist zu entnehmen, daß man bis zum Herbst die Herausgabe umfassender Buchreihen plant. Weiterhin würde an einer 'revolutionären' Programmiersprache PASCULA entwickelt, die nicht nur ein Standard PASCAL, sondern auch die Sprache MODULA umfassen soll.

Rockwell: Der Zweitlieferant für 68000 kündigt an, daß man die Zentraleinheit als erste Firma gem. miltitärischem Standard MIL-Std-883 (Rev. C) für den Temperaturbereich von -55—+125 Grad Celsius anbieten kann. - Hingewiesen wird auch auf auf den für ein breites Spektrum von Sprach- und Datenübertragungen entworfenen Übertrager R8070, der als 1 Chip an die Stelle der bisher benötigten drei- bis vierteiligen Chipsätze tritt. Er kann in eine Vielzahl von Mikroprozessor-Busstrukturen integriert wrden. Er hat autonome Sender- und Empfängerteile und TTL/CMOS-kompatible Ein- und Ausgänge.

Birgit Roßmöller Computertuning, Maxstr. 52, 5300 Bonn 1, Tel. 02225/15575. Die Roßmöller GmbH hat zum 1.8. ein Spezialgeschäft in Bonn eröffnet, in dem nicht nur die bisher erfolgreichen Computerprodukte der GmbH vertrieben werden, sondern auch über 1000 verschiedene IC's für den Selbstbauer, darunter sämtliche TTL-Bausteine.

Internationale Funkausstellung vom 30.8. bis 8.9.85 in Berlin. Das sog. Computerzentrum wird sich in Halle 19 befinden. Vor allem für Fachbesucher werden Veranstaltungen und Demonstrationen zu folgenden Themen abgehalten: Neue Medien, Video-Programmsignal, Telekommunikation, Bildschirmtext und Computer. Kontakt: Deutsches Video Institut, Budapester Str. 44, Berlin 30, Tel. 030/261 16 70.

Directory of Microcomputer Software for Mechanical Engeneering Design. Das Handbuch führt etwa 235 Software-Pakete auf, und zwar mit erforderlicher Umgebung in Hard- und Software, Preis, Dokumentation und möglichem Training. Die Themen sind: Automatic Control, Building Design, Drafting, Finite Element Mehod, Fluid Mechanics, Geometric Modelling, Heating/Cooling Air Conditioning, Machine Design, Numerical Methods, Pressure Vessels, Stress Analysis, Thermal Analysis usw. ISBN 0-8247-7413-2, Preis SFr 139,--, beziehbar bei Marcel Dekker, Inc. Postfach 133, CH-4010 Basel.

VALVO: Im zweiten Halbjahr werden Seminare zu folgenden Themen abgehalten: 68000-Workshop, VMEbus, C-Workshop, IFL, CMOS Design, Sprachsynthese. Kontakt: Valvo, Abt. DWE, Burchardstr. 19, 2000 Hamburg 1, Tel. 040-3296-278. - Kurse werden auch geplant für die neue SMD-Technik (Surface Mounted Devices) im Oktober. Den Rundschreiben der Firma ist zu entnehmen, daß eine Fülle von aktiven und passiven Bauteilen bereits für diese Montagetechnik lieferbar ist.

Controller für DRAMs 74LS764: Dieser Baustein führt sämtliche Funktionen aus, die nötig sind, um zwei Mikroprozessoren den Zugriff auf ein gemeinsames DRAM zu gewähren, das eine Blockgröße bis 256K hat. Die Refresh-Prozedur, die eine höhere Priorität als der Speicherzugriff hat, wird selbstätig durchgeführt. Der Refresh-Takt kann dabei wahlweise vom System abgeleitet oder extern vorgegeben werden. Die maximale Taktfrequenz beträgt 30 MHz (Rundschr. 55/85). - Lieferbar ist jetzt auch die **Fehlererkennungs- und Korrekturschaltung N2960.** Diese bipolare LSI-Schaltung ersetzt etwa 30 MSI-Schaltungen und erhöht die Speicherzuverlässigkeit in schnellen Systemen. Alle Einzelbitfehler werden korrigiert, Doppelbitfehler werden erkannt. Die worst case-Zeiten für Fehlererkennung bzw. 16 Bit Datenkorrektur liegen bei 32-65 ns. Es kann auf 32- und 64 Bit-Architekturen kaskadiert werden. - (Rundschr. 48/85). - Neu ist auch der **Video-Attribut-Controller SCB2675T für Monochrom- und Farbterminals.** Er erlaubt eine Dotfrequenz von 40 MHz. Datenblatt unter Kennziffer 64 von Valvo. - Der Vertrieb der VMEbus-Produkte ist auf den Unternehmensbereich Elektronik für Wissenschaft und Industrie der Philips in Kassel übertragen worden, z.Hd. Herrn Krostewitz, Tel. 0561-501-348.

SYSTEMS '85 vom 28. Okt.-1. Nov. 1985 in München: Einige Leser gaben die Anregung, auch in diesem Jahre wieder einmal ein zwangloses Treffen ernsthafter Computerbetreiber im Anschluß an den Messebesuch abzuhalten. Der Herausgeber nimmt diesen Gedanken gerne auf und schlägt ein Treffen am 29. oder 30. Oktober ab 19 Uhr in München oder in einem gut erreichbaren Vorort vor. Interessenten sollten sich frühzeitig telefonisch oder brieflich anmelden, damit sie rechtzeitig benachrichtigt werden können. In diesem Zusammenhang werden die Leser in München um den Vorschlag eines geeigneten Lokals gebeten, in dem das Treffen stattfinden könnte.

Eine dringende Bitte an die Leser: **Geben Sie bitte eine Veränderung Ihrer Briefanschrift frühzeitig bekannt.** Die Post macht sich meistens nicht die Mühe, eine neue Adresse aufzuschlagen und Sendungen weiterzuleiten. Aus diesen Gründen reißt häufig die weitere Zustellung dieser Zeitschrift unbeabsichtigt ab.

Wenn ein Abonnement abgelaufen ist, dann erhalten Sie auf brieflichem Wege eine Anschlußrechnung mit beigefügter Zahlkarte (im Inland). Bitte schreiben Sie links auf den **Empfängerabschnitt immer Ihren Namen und die Rechnungs-Nummer.** Wenn diese Angaben fehlen, dann müssen die Nachforschungsstellen der Geldinstitute unnötig bemüht werden, was Mehrarbeit und Verzögerungen bedeutet.

Leser im Ausland: Wählen Sie den einfachsten und für Sie billigsten Weg, um eine auf DM lautende Rechnung zu bezahlen. Wenn Sie ein Postscheck-Konto haben, dann können Sie problemlos von Konto zu Konto überweisen. Andernfalls sparen Sie viel Gebühren, wenn Sie eine Einzahlung beim Postamt vornehmen oder einen Euro-Scheck senden, der immer gebührengünstig abgewickelt wird. Der teuerste Weg ist es, wenn Sie Ihre Bank um die Ausstellung eines Schecks bitten, der in DM auf eine deutsche Korrespondenzbank gezogen ist.

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,--

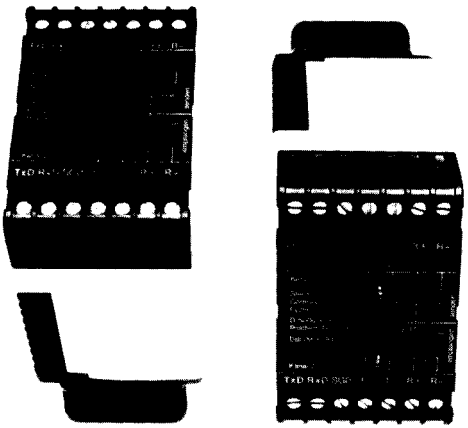
Mathe-ROM für 6502

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



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.

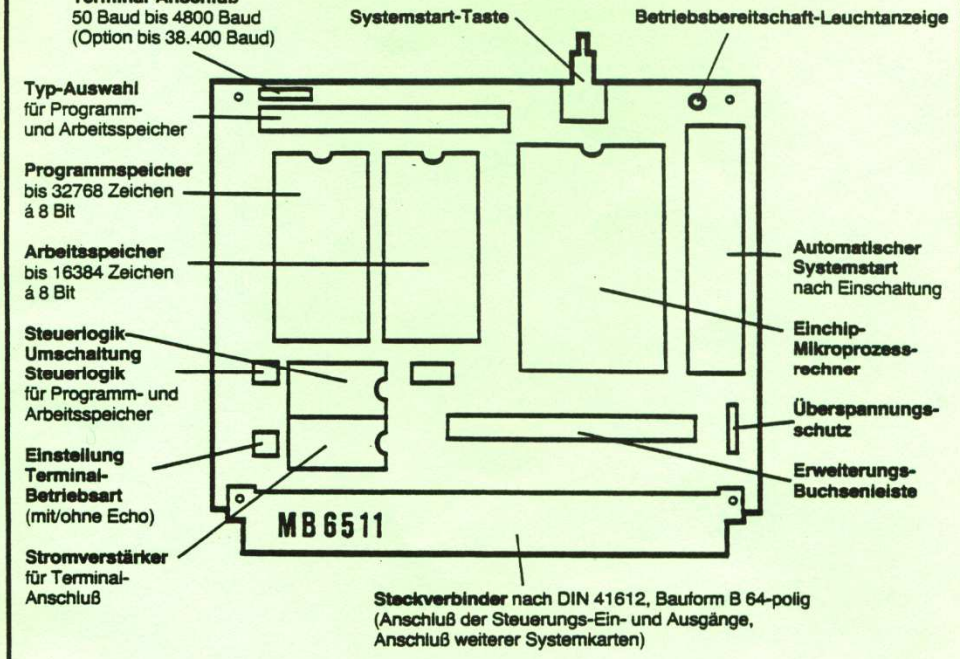
Getestet werden kann die Datenübertragungsstrecke dank der in den Stromschleifen liegenden Leuchtdioden auch ohne die angeschlossenen Datengeräte, da es sich um ein Ruhestromsystem handelt, d.h. im Ruhezustand wird die logische 1 übertragen. - Bei der Errichtung von Punkt zu Punkt Datennetzen mittels 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

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)

EINCHIP-MIKROPROZESSOR-STEUE

mit R6511Q

DM 350,– + Mwst

SPEZIAL-ANGEBOT für MICRO MAG-Leser:
DM 380,– incl. 8 kByte RAM
und AIM65-Terminal-Monitor
(Mit Assembler + Disassembler)

BRÜHL ELEKTRONIK ENTWICKLUNGS-GESELLSCHAFT mbH

8500 Nürnberg 10, Hegelstr.10 TEL. 0911-359088

MICRO MAG

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

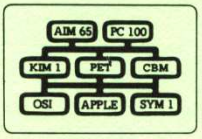
 Offsetdruck: L & L Druckservice, Hamburg 73

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 17-43 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,--