der Lieferung beilagen: Ein wunderschön (in Englisch) gemachter Basic-Lehrgang für Anfänger, deutlich besser noch als das zu Recht oft gerühmte TRS-80-Begleitmaterial, ein ausführliches Applesoft-Handbuch dazu, sodann eine 150seitige Materialsammlung mit Schaltbildern, Bauteil-Stücklisten, ausführlicher Besprechung aller Ports, mit einer Bedienungsanweisung für Mini-Assembler und Monitor plus Betriebssoftware-Listing, mit Kurzaufsätzen verschiedener in den Staaten bekannter Hobby-Programmierer, die über spezielle Tricks berichten, mit denen sie ihren Apple zu Sonder- und Höchst-Leistungen anspornten. Dazu, um das Maß voll zu machen, noch gesonderte Betriebsanleitungen und Anwendungshinweise für die mitgelieferten Demonstrations- und Maschinenroutinen-Programme, für das Disk Operating System und das Drucker-Interface nochmal ein eigenes Vorab-Handbuch: PET-Käufer werden so was mit Wehmut hören! Nun zum Rechner selbst:

Beim Programmieren in Applesoft erweist sich als angenehm, daß ohne weiteres speichersparend Zeilen mit bis zu 239 Zeichen Länge eingesetzt werden können (*Bild 2*), andere Rechner streiken bereits bei einem Drittel dieses Wertes.

Die Hauptvorzüge des Euro-Apple liegen jedoch in der hohen Flexibilität seiner Ein- und Ausgabe-Möglichkeiten. Das Steckplatz-Konzept, die Farbgrafik und der Lautsprecher wurden bereits erwähnt, nicht jedoch, was man mit dem Lautsprecher alles machen kann: So erprobte der Autor zum Beispiel ein „Appletalker" genanntes Sprachausgabe-Programm von Mad Hatter Software, Dracut, Massachusetts, das über den Tonband-Anschluß eingegebene Geräusche und menschliche Worte (recht speicherintensiv allerdings) abspeichert und per Aufruf durch Benutzerprogramm über den Lautsprecher wiedergibt: Kratzig und verzerrt zwar, manche Worte bisweilen unverständlich, aber für markige Mitteilungen wie „Fehler" oder „Finger weg!" reicht's allemal. Die gleiche Firma bietet auch ein Spracherkennungs-Programm an, das der Autor allerdings noch nicht erprobte.

Eine feine Sache auch die Analog-Digital-Wandler im Euro-Apple, mit denen sich – über ein paar Bauteile wie Opto-Koppler oder FETs zum Beispiel – der 2020 bequem Informationen aus elektrischen Systemen beliebiger Art beschaffen kann. Faszinierend schließlich noch, wie der Euro-Apple zum „Software-Basteln" herausfordert: Die vorzügliche Dokumentation, die komfortable Unterstützung durch Monitor und Mini-Assembler verleiten förmlich dazu, sich eigene Mini-Spezial-Hochsprachen zu basteln oder Sonderwünsche, ins Betriebs-System gepflanzt, zu verwirklichen. Um's kurz zu machen: Nach zehn Wochen täglicher (Feierabend- und Wochenend-)Arbeit ist der Autor des Beitrages begeistert von dieser Maschine, die der Programmierer-Fantasie erst sehr viel später Grenzen setzt, als das die billigeren Mitbewerber des 2020 tun (*Bild 3*).

Empfehlenswert – aber für wen?

Nun, wer das Programmieren erst zum Steckenpferd machen will, noch nicht über nennenswerte Erfahrungen verfügt und in den kommenden Jahren nicht sehr viel mehr an Freizeit ins Hobby investieren kann, als einem Vierzig-Wochenstunden-Arbeitnehmer gemeinhin übrigbleibt, der wird – wenn überhaupt – erst sehr spät in die Lage kommen, die spezifischen 2020-Vorteile voll nutzen zu können. Der ist mit Maschinen wie dem PET oder TRS-80 besser bedient als mit dem rund zweimal teureren ITT-Rechner. Das gleiche gilt auch für professionelle Anwender, die mit dem Rechner busorientierte Systeme kontrollieren wollen – da ist der PET mit seinem IEC-Bus bisher unerreicht.

Wo jedoch fortgeschrittenere Stekkenpferd-Reiter am Werke sind oder wo man sich bei professionellen Anwendungen noch nicht auf den IEC-Bus festgelegt hat und dabei neben Basic weitere Hochsprachen zu benutzen wünscht, dort wird der Euro-Apple sicherlich das Rennen machen, wenn ihm nicht die Konkurrenz in die Quere kommt; in den USA bietet Microsoft nämlich für den TRS-80 bereits FORTRAN an.

Eine Morseschreibmaschine

Über den Sinn von Morsedecodern wollen wir uns hier nicht den Kopf zerbrechen; schließlich hat Telegrafie heute nur noch dort Sinn, wo es darum geht, aus einem hohen Störpegel noch ein schwaches Signal herauszuhören, und da kann der beste Morsedecoder nicht mehr mit. Sinnvoll ist es dagegen, Morsezeichen vom Terminal aus zu schreiben. Dazu dient hier der Mikrocomputer KIM-1 zusammen mit einem ASCII-Terminal.

Das hier beschriebene 6502-Maschinencode-Programm setzt jeden von einem ASCII-Terminal aus getippten Buchstaben in das entsprechende Morsezeichen um. Über einen I/O-Port des Mikrocomputers KIM-1 kann direkt z. B. ein Kurzwellensender angesteuert werden; wahlweise ist auch eine Ausgabe als Nf-Ton möglich. Die Eingabe kann sehr schnell erfolgen, da sich der Mikrocomputer einen 255 Zeichen umfassenden Zwischenspeicher aufbaut und diesen mit einer beliebig programmierbaren Geschwindigkeit als Telegrafie (CW) ausgibt.

Bild 1 zeigt das komplette Programm. Es verwendet die in Heft 7/1979, Seite 396, bereits näher besprochene Interrupt-Routine zum Empfang der ASCII-Zeichen, die hier nur innerhalb des „Hex-Dump" wiedergegeben ist. Bild 1 enthält auch die Tabelle zur Umwandlung der ASCII- in Morsezeichen (Adressen 006B...00AA). Es ist zu beachten, daß unbedingt die NMI-Leitung des KIM-1 mit dem Port PB 7 zu verbinden ist, da sonst kein Zeichen vom Terminal empfangen werden kann.

Das Kernstück der Morseschreibmaschine ist der in *Bild 2* gezeigte Programmteil. Er gibt ein im Akkumulator des Mikroprozessors 6502 stehendes

```
0000 A9 02 85 E6 85 E7 85 EB 85 EC 8D 0F 17 A9 00 8D
0010 FB 17 A9 AB 8D FA 17 A5 E6 38 E5 EB F0 F9 C6 E6
0020 A0 00 B1 E6 C9 20 30 EF 29 3F AA BD 6B 00 0A D0
0030 09 A5 E9 85 F4 20 54 00 F0 DD A2 01 90 02 A2 03
0040 86 F4 20 4F 00 A2 01 86 F4 20 54 00 18 90 DF A2
0050 01 8E 01 17 A6 EE A4 ED 88 D0 FD EE 00 17 CA D0
0060 F5 C6 F4 D0 EF A2 00 8E 01 17 60 4C 60 88 A8 90
0070 40 28 D0 08 20 78 B0 48 E0 A0 F0 68 D8 50 10 C0
0080 30 18 70 98 B8 C8 AC 80 54 80 80 80 56 80 80 80
0090 80 80 80 80 80 80 80 CE 8C 56 94 FC 7C 3C 1C 0C
00A0 04 84 C4 E4 F4 56 56 80 80 80 32 48 98 48 A5 E8
00B0 D0 16 2C 40 17 10 09 A9 40 8D 0E 17 68 A8 68 40
00C0 A9 09 85 E8 A9 A0 D0 F1 C6 E8 D0 0E A5 EA F0 06
00D0 C6 EB A0 00 91 EB A9 8E D0 DF 18 2C 40 17 10 01
00E0 38 66 EA 18 90 F0
```

Bild 1. Mit diesem kleinen Programm wird aus dem Mikrocomputer KIM-1 eine „Morseschreibmaschine"

```
0028 29 3F     AND #3F
002A AA        TAX
002B BD 6B 00  LDA 006B,X
002E 0A        ASL A
002F D0 09     BNE 003A
0031 A5 E9     LDA E9
0033 85 F4     STA F4
0035 20 54 00  JSR 0054
0038 F0 DD     BEQ 0017
003A A2 01     LDX #01
003C 90 02     BCC 0040
003E A2 03     LDX #03
0040 86 F4     STX F4
0042 20 4F 00  JSR 004F
0045 A2 01     LDX #01
0047 86 F4     STX F4
0049 20 54 00  JSR 0054
004C 18        CLC
004D 90 DF     BCC 002E
004F A2 01     LDX #01
0051 8E 01 17  STX 1701
0054 A6 EE     LDX EE
0056 A0 80     LDY #80
0058 8C 00 17  STY 1700
005B 88        DEY
005C D0 FD     BNE 005B
005E CA        DEX
005F D0 F5     BNE 0056
0061 C6 F4     DEC F4
0063 D0 EF     BNE 0054
0065 A2 00     LDX #00
0067 8E 01 17  STX 1701
006A 60        RTS

00E9 03 PAUSE
00ED 80 TONFREQUENZ
00EE 60 GESCHWINDIGKEIT
```

Bild 2. Disassemblierte Auflistung des Programmteils für die Ausgabe eines CW-Zeichens zum Tasten eines Senders. Das im Akku stehende ASCII-Zeichen wird in den Morsecode umgesetzt

ASCII-Zeichen in Telegrafie am Port PA 0 aus; dies geschieht als digitales Signal, wobei H-Pegel an PA 0 dem Zustand „Sender aus" und L-Pegel dem Zustand „Sender ein" entspricht.

Für Test- und Übungszwecke kann es sinnvoll sein, nicht ein digitales Signal zum Tasten des Senders zu erzeugen, sondern eine Tonfrequenz, die dann z. B. mit Hilfe der im KIM-1-Handbuch beschriebenen Schaltung (PNP-Transistor als Lautsprecher-Treiber) hörbar gemacht werden kann. Dazu sind die in *Bild 3* angegebenen Adressen zu ändern.

```
0056 A4 ED     LDY ED
0058 88        DEY
0059 D0 FD     BNE 0058
005B EE 00 17  INC 1700
```

Bild 3. Programmänderung zum Erzeugen eines Nf-Tones am Port PA 0

Die Eigenschaften der abgegebenen CW-Zeichen können durch Ändern bestimmter Zero-Page-Adressen den individuellen Erfordernissen angepaßt werden. Die Standard-Inhalte dieser Zellen sind ebenfalls in Bild 2 angegeben. Zu bedenken ist, daß eine Änderung der Tonfrequenz auch die CW-Geschwindigkeit beeinflußt: Höhere Zahlenwerte ergeben niedrigere Frequenzen und Geschwindigkeiten.

Für Übungszwecke wird manchmal der Zeichenabstand über die Norm von drei Punktlängen hinaus verlängert. Dies kann hier durch Einschreiben der gewünschten Punktlängen in die Zelle 00E9 geschehen.

Der von dem Programmteil in Bild 3 erzeugte Nf-Ton hört sich leider ein wenig rauh an. Dies hängt damit zusammen, daß er im Abstand von einigen Millisekunden jeweils für die Dauer einiger Mikrosekunden von der ASCII-Eingabe-Interruptroutine unterbrochen wird. Ein absolut sauberer Ton ist daher nur zu erzeugen, indem man wie in Bild 2 ein rein digitales Signal erzeugt und damit einen Tongenerator steuert, wenn auch die „Unsauberkeit" des Software-Tones nur gering ist.

Der Vorteil einer Morseschreibmaschine wird beim praktischen Betrieb schnell deutlich: Man kann so schnell geben wie bei RTTY, und die Zeichen werden mit konstant exaktem Punkt-Strich- und Punkt-Pausen-Verhältnis gesendet; und das ist wiederum Voraussetzung für das Funktionieren von Morsedecodern – siehe oben.

Herwig Feichtinger

Ist Maschinensprache überholt?

Ganz grob lassen sich die heutigen Mikrocomputer in zwei Klassen einteilen: Auf der einen Seite die BASIC-Rechner, die sich für mathematische Aufgaben und auch ein wenig für kleine Textverarbeitungsprobleme einsetzen lassen, und auf der anderen Seite die „maschinenorientierten" Mikrocomputer, die in erster Linie als Hardware-Ersatz gedacht sind.

In letzter Zeit hörte man häufig die Ansicht, daß letztere Gruppe, die sich „nur" in Maschinensprache bzw. in der Assembler-Sprache programmieren läßt, überholt sei. Ist das wirklich so?

Tatsächlich ist die Programmierung in Maschinensprache mit einigen Nachteilen verbunden. Erstens ist sie systembezogen, d. h. ein Programm, das auf einem bestimmten Mikrocomputer-System läuft, muß für ein anderes erst umgeschrieben werden; wenn dabei unterschiedliche Mikroprozessortypen verwendet werden, ist dies oft sogar unmöglich. Zweitens enthält die Maschinensprache, die der Mikroprozessor ja direkt verarbeitet, nur sehr primitive arithmetische Befehle, z. B. für das Addieren und Subtrahieren zweier Zahlen zwischen 0 und 255. Drittens ist das Erlernen der Maschinensprache ebenso wie die Programmierung zeitraubender als in einer höheren Programmiersprache wie BASIC.

Also wäre es doch besser, wenn wir uns nur noch der höheren Programmiersprachen bedienten?! Nein, ganz so einfach ist das auch wieder nicht. Hier ein Beispiel, bei dem kein vernünftiger Mensch auf die Idee kommen wird, die Maschinensprache durch eine höhere Sprache zu ersetzen.

Die in der FUNKSCHAU beschriebene Mikrocomputer-Uhr zum Decodieren der vom Sender DCF 77 ausgestrahlten Zeitinformation muß natürlich in Echtzeitbetrieb arbeiten; d. h. das Programm muß die empfangene Information sofort in den zur Anzeige notwendigen Code umformen. Dies ist eine ausgesprochen zeitkritische Aufgabe. Ein BASIC-Interpreter, der allein für einen Programmbefehl mehr als tausendmal länger braucht als ein Maschinenbefehl, wäre hier fehl am Platze – ganz abgesehen davon, daß der Hardware-Aufwand wegen der dann notwendigen PROMs mit dem BASIC-Interpreter um mindestens 100 DM steigen würde. Ähnlich ist es bei den jetzt recht beliebten Schachcomputern.

Viele andere auch in der FUNKSCHAU veröffentlichte Programme sind Beispiele dafür, daß man mit BASIC zwar besser rechnen kann, aber daß die Maschinensprache für bestimmte Zwecke unumgänglich ist.

Der oft heraufbeschworene Gegensatz zwischen BASIC- und Assembler-Computern existiert meist nicht; denn Computer wie der PET 2001 oder der TRS-80 lassen sich – wenn auch im Urzustand extrem umständlich – in Maschinensprache programmieren, und Mikrocomputer wie der AIM-65, der SYM-1 oder der KIM-1 lassen sich problemlos mit einem BASIC-Interpreter erweitern, wodurch allerdings ihr Preisvorteil gegenüber BASIC-Computern verlorengeht.