

Herwig Feichtinger

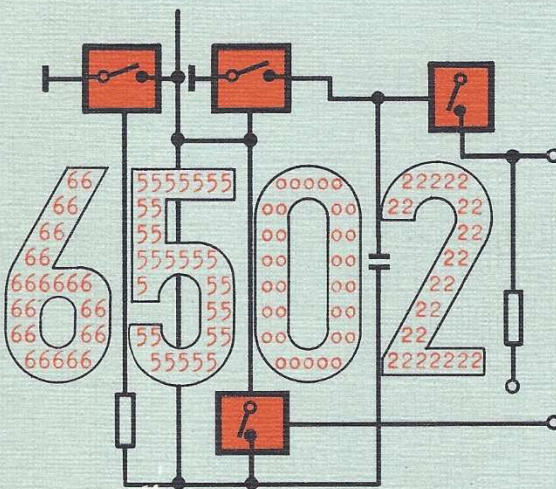
173

# Anwendungsbeispiele für den Mikroprozessor 6502

RPB

electronic-  
taschenbücher

Hardware-Tips und nützliche Programmbeispiele  
in Maschinensprache



**Franzis'**

Herwig Feichtinger

# Anwendungsbeispiele für den Mikroprozessor 6502

Hardware-Tips und nützliche  
Programmbeispiele in Maschinensprache

Mit 40 Abbildungen

**F'**

---

Franzis-Verlag München

Nr. 173 der RPB electronic-taschenbücher

CIP-Kurztitelaufnahme der Deutschen Bibliothek

**Feichtinger, Herwig:**

Anwendungsbeispiele für den Mikroprozessor 6502: Hardware-Tips u. nützl. Programmbeispiele in Maschinensprache/Herwig Feichtinger. — München: Franzis-Verlag, 1980.

([RPB-Elektronik-Taschenbücher] RPB-electronic-taschenbücher;  
Nr. 173)

ISBN 3-7723-1731-6

© 1980 Franzis-Verlag GmbH, München

Sämtliche Rechte — besonders das Übersetzungsrecht — an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2  
Printed in Germany. Imprimé en Allemagne.

ISBN 3-7723-1731-6

## Vorwort

Der Mikroprozessor 6502 ist etwa seit 1975 auf dem Markt und gehört heute zu den am weitesten verbreiteten 8-bit-Prozessoren. Ein besonderer Vorteil ist, daß eine Reihe preisgünstiger Systeme wie KIM, SYM und AIM-65 damit erhältlich ist, was besonders für den Hobbyisten interessant ist.

Das vorliegende Buch zeigt eine Reihe von Anwendungsbeispielen für diesen Mikroprozessor, wobei die Adressenbelegung für das KIM-System ausgelegt wurde; eine Adaption an andere 6502-Systeme ist jedoch leicht möglich. Wenn in vielen Fällen auch eine unveränderte Übernahme der hier abgedruckten Programme für den individuellen Verwendungszweck nicht optimal sein kann, so findet der Anwender doch wertvolle Anregungen und Ideen für die Verwirklichung „seiner“ Programme.

Der Abdruck der Programm-Auflistungen erfolgte möglichst immer vom laufenden Programm, so daß Fehler bei der Reproduktion weitgehend ausgeschlossen sind. Sollte ein Leser dennoch einmal einen „Wurm“ feststellen, so ist der Autor für eine entsprechende Mitteilung dankbar.

Herwig Feichtinger



## Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden\*).

Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag sieht sich deshalb gezwungen, darauf hinzuweisen, daß er weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann. Für Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

\*) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenzinhabers einzuholen.

# Inhalt

<b>1</b>	<b>Allgemeines</b>	<b>7</b>
1.1	Ratschläge für Computer-Neulinge	7
1.2	µP-Portrait	8
1.3	Die Mikrocomputer KIM-1, SYM-1, AIM-65 und PC-100	12
1.4	Gebrauchsanleitung für den KIM-Timer	15
1.5	Zusätzliche Befehle und Maskenfehler beim 6502	17
<b>2</b>	<b>Hardware-Tips</b>	<b>19</b>
2.1	RS-232/V-24-Interface für den KIM-1	19
2.2	„Automatische“ Hardware-Speicherverschiebung	19
2.3	Kassetten-Probleme beim KIM-1	22
2.4	FSK-Modem	23
2.4.1	Modulator	23
2.4.2	Demodulator	24
2.5	Norm für die ASCII-Übertragung im Amateurfunk	26
<b>3</b>	<b>Programme für den „rohen“ KIM-1</b>	<b>28</b>
3.1	Funktionsgenerator	28
3.2	Nf-Zähler	32
3.3	Speicheroszilloskop	34
3.4	Baudot-Ausgabeprogramm	37
3.5	Programm für einen Metallpapier-Drucker	41
3.6	Serielle ASCII-Eingabe per Interrupt	43
3.7	Binär-Dezimal-Umwandlung	49
3.8	Hypertape	51
3.9	Interrupt-Uhr	52
<b>4</b>	<b>Programme für ASCII-Terminals</b>	<b>56</b>
4.1	Debugger	56
4.2	Disassembler	58
4.3	Plotter für das Speicher-Oszilloskop	67
4.4	Datensuche – ein Karteiprogramm	69
4.5	Automatische Text-Formatierung	75

4.6	KIM versteht Pseudo-Befehle	78
4.7	Baudot-Disassembler	87
4.8	Ein Baudot-Fernschreibprogramm	90
4.8.1	Zweck des Programms	90
4.8.2	Notwendige Hardware	92
<b>5</b>	<b>Literatur</b>	94
	<b>Sachverzeichnis</b>	95

# 1 Allgemeines

## 1.1 Ratschläge für Computer-Neulinge

### *Während der Lieferzeit . . .*

. . . Ihres Mikrocomputers sollten Sie die Zeit nutzen, sich ein passendes Netzgerät zu besorgen oder selbst zu bauen. Vermeiden Sie unbedingt die Verwendung mehrerer, unglücklicherweise meist auch zu höheren Spannungswerten hin regelbarer Netzgeräte, da sonst nie sichergestellt werden kann, daß im richtigen Moment die richtige Spannung vorhanden ist. Ein spezielles Mikrocomputer-Netzteil ist billiger als eine neue Mikrocomputer-Platine!

### *Terminals mit serieller ASCII-Schnittstelle . . .*

. . . sollten entsprechend der Mikrocomputer-Schnittstelle programmiert werden. In den meisten Fällen bedeutet dies: Kein Parity-Bit, acht Datenbits, bei 110 Bd zwei Stopbits, darüber 1 Stopbit. Das achte Bit legt man zweckmäßig auf L-Pegel (null), das geschieht am UART-Anschluß D<sub>7</sub>.

### *Wenn der Computer schweigt*

Sollte auf dem Terminal-Bildschirm außer den eingetippten Zeichen nichts erscheinen, so programmieren Sie den Mikrocomputer für eine einfache Testschleife, die irgendein ASCII-Zeichen pausenlos ausgibt, z.B. den Buchstaben A. Beim KIM-1 kann dieses Programm etwa so aussehen:

```
A9 00 8D F3 17 A9 74 8D
F2 17 A9 41 20 A0 1E 18
90 F8
```

Schreiben Sie dieses Programm irgendwo in Ihren Speicher, z.B. ab der Adresse 0000 (dies kann mit dem KIM-

Display und der Tastatur auf der KIM-Platine geschehen) und starten Sie es. Es druckt pausenlos den Buchstaben A mit einer Geschwindigkeit von 600 Bd aus, was die Fehlersuche z.B. in den Interface-Schaltungen sehr erleichtert.

### *Endlich läuft das System*

Ist endlich die Kommunikation mit dem Computer möglich, so vergessen Sie nicht, auch einmal einen Blick in das Systemhandbuch zu werfen. (Software- und vor allem Hardware-Handbücher sind im Augenblick noch nicht wichtig.) Versuchen Sie, auch wenn Ihnen zunächst alles spanisch vorkommt, einfach zwei Adresseninhalte zu addieren und an eine dritte Adresse abzuspeichern; das KIM-Handbuch gibt dafür ein einfaches Beispiel. Nach diesem ersten Programm sieht alles schon ganz anders aus!

## **1.2 $\mu$ P-Portrait**

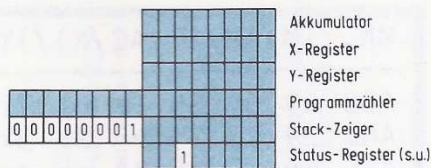
Der 6502 wurde von MOS Technology entwickelt; diese Firma ist mittlerweile von Commodore übernommen worden. Als Zweitlieferanten fungieren derzeit Synertek und Rockwell. Der Prozessor ist das Mitglied einer ganzen Familie von  $\mu$ Ps, zu der auch die Typen 6503, 6504, 6505 und 6506 gehören, die aber bisher weniger Verbreitung fanden.

Die 65XX-Familie ging ursprünglich durch Weiterentwicklung aus der Motorola-6800-Familie hervor. Die Entwicklungsphilosophie war dabei eine gänzlich andere als z.B. bei den Prozessoren 8080 oder Z 80, und ein Vergleich mit diesen beiden ist daher nur schwer möglich.

Das beginnt bereits bei dem internen „Timing“, dem Zeitablauf. Alle Befehle werden in ganzzahligen Vielfachen der Takt-Periodendauer ausgeführt, und viele Befehle brauchen nur ebenso viele Taktzyklen, wie sie Bytes benötigen!

Die CPU 6502 ist dadurch einer der schnellsten Prozessoren, die derzeit erhältlich sind, obwohl die übliche Taktfrequenz „nur“ 1 MHz, maximal 2 MHz beträgt.





### 1.2.1 Registerstruktur der CPU 6502

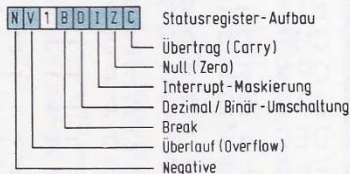


Abb. 1.2.1 zeigt die im Prozessor vorhandenen Register und ihre Wortlänge; addieren und subtrahieren kann man nur im Akku, und zwar – je nachdem, ob das Dezimal-Bit im Statusregister 1 oder 0 ist – im Dezimal- oder im Binärsystem. Dadurch ist der bei dem Prozessor 8080 vorhandene Befehl „Decimal Adjust“ hier nicht erforderlich.

Außer für Zwischenspeicherungen können die X- und Y-Register für die besonderen Adressierungsarten des 6502 verwendet werden.

Wie man leicht erkennt, erlaubt die äußerst flexible Adressierung des 6502 die Verwendung vieler Befehle, die nur zwei Byte lang sind. Dabei wird die Zero-Page-Adressierung verwendet, wobei das höherwertige Adressenbyte gleich Null gesetzt wird. Dem Anwender stehen damit praktisch 256 Zellen zur Verfügung, die mit nur zwei Bytes erreicht werden können.

Für die Speicherung der Rücksprung-Adressen bei Unterprogramm- oder Interrupt-Operationen wird normalerweise ein „Stack“ verwendet, der beim 6502 fest auf den Speicherbereich 0100 . . . 01FF programmiert ist und nicht verändert werden kann. Die Stacklänge beträgt also 256 Byte, so daß maximal 128 Rücksprungadressen gespeichert werden können.

Es existieren drei Interrupt-Ebenen: Reset, NMI und IRQ.

MN. /IM/AB/ZP /AC/X) /Y/ZX/ZY/AX/AY

ADC	69	6D	65	--	61	71	75	--	7D	79
AND	29	2D	25	--	21	31	35	--	3D	39
ASL	--	0E	06	0A	--	--	16	--	1E	--
BIT	--	2C	24	--	--	--	--	--	--	--
CMP	C9	CD	C5	--	C1	D1	D5	--	DD	D9
CPX	E0	EC	E4	--	--	--	--	--	--	--
CPY	C0	CC	C4	--	--	--	--	--	--	--
DEC	--	CE	C6	--	--	--	D6	--	DE	--
EOR	49	4D	45	--	41	51	55	--	5D	59
INC	--	EE	E6	--	--	--	F6	--	FE	--
LDA	A9	AD	A5	--	A1	B1	B5	--	BD	B9
LDX	A2	AE	A6	--	--	--	--	B6	--	BE
LDY	A0	AC	A4	--	--	--	B4	--	BC	--
LSR	--	4E	46	4A	--	--	56	--	5E	--
ORA	09	0D	05	--	01	11	15	--	1D	19
ROL	--	2E	26	2A	--	--	36	--	3E	--
ROR	--	6E	66	6A	--	--	76	--	7E	--
SBC	E9	ED	E5	--	E1	F1	F5	--	FD	F9
STA	--	8D	85	--	81	91	95	--	9D	99
STX	--	8E	86	--	--	--	--	96	--	--
STY	--	8C	84	--	--	--	94	--	--	--

BCC	90	BCS	B0	BEQ	F0	BMI	30
BNE	D0	BPL	10	BRK	00	BVC	50
BVS	70	CLC	18	CLD	D8	CLI	58
CLV	B8	DEX	CA	DEY	88	INX	E8
INY	C8	JMI	6C	JMP	4C	JSR	20
NOP	EA	PHA	48	PHP	08	PLA	68
PLP	28	RTI	40	RTS	60	SEC	38
SED	F8	SEI	78	TAX	AA	TAY	A8
TSX	BA	TXA	8A	TXS	9A	TYA	98

Adressierung	Beispiel	Wirkung im Beispiel
Immediate	A9 02	Akku wird mit dem Wert 02 geladen
Absolute	AD 00 17	Akku wird mit dem Inhalt der Zelle 1700 geladen
Zero Page	A5 08	Akku wird mit dem Inhalt der Zelle 0008 geladen
Indirect	6C 67 02	Sprung zu der Adresse, die in den Zellen 0267 und 0268 steht
Implied	CA	X-Register um 1 erniedrigen (der Befehl enthält die Adressierung selbst)
Abs.,indiziert	BD 00 02	Der Akku wird mit dem Inhalt der Adresse geladen, die sich aus der Summe der angegebenen (0200) Adresse und dem X-Register ergibt
Nach-indiziert	B1 FA	Der Akku wird mit dem Inhalt der Zelle geladen, die sich aus der in den Zellen 00FA und 00FB stehenden Adresse plus dem Inhalt des Y-Registers ergibt
Vor-indiziert	A1 12	Der Akku wird mit dem Inhalt der Zelle geladen, deren Adresse in der Zelle steht, deren Adresse sich aus der Summe von 0012 und dem X-Register ergibt (schrecklich kompliziert, nicht wahr? )
Relativ	F0 12	Wenn das Null-Flag (Z) gesetzt ist, erfolgt ein Sprung um 12 Byte nach vorn
Zero Page, X	B5 40	Der Akku wird mit dem Inhalt der Zelle geladen, deren Adresse sich aus der Summe von 0040 und dem Inhalt des X-Registers ergibt

Reset ist ein nicht maskierbarer Interrupt mit der höchsten Priorität. NMI ist ebenfalls ein „nicht maskierbarer Interrupt“, während IRQ „Interrupt Request“ bedeutet und nur dann tatsächlich bedient wird, wenn das Interrupt-Disable-Bit im Statusregister auf log. 0 liegt. Alle Interrupt-Arten führen zu einem Sprung an bestimmte Interrupt-Adressen, die vom Anwender in gewissen Zellen (FFF . . .) frei programmiert werden können. Der Interrupt IRQ ist auch innerhalb eines Programms durch den Break-Befehl zu simulieren (hexadezimal 00). NMI und IRQ führen zum Abspeichern der Rücksprungadressen und des Statusregisters im Stackbereich.

Da die CPU 6502 zu den preiswertesten und dennoch zu den leistungsfähigsten gehört, fand sie weite Verbreitung im Hobbycomputer-Bereich; so enthalten z.B. die Systeme KIM-1, SYM-1, AIM-65, CBM3001, PC100, PC1000, PET 2001 und Apple II bzw. ITT 2020 alle diese CPU. *Abb. 1.2.2* gibt die Operationscodes wieder.

### **1.3 Die Mikrocomputer KIM-1, SYM-1, AIM-65 und PC-100**

Bis heute sind es im wesentlichen nur vier 6502-Systeme, die speziell für maschinenorientierte Aufgaben weite Verbreitung fanden; der KIM-1 (Commodore) ist bei uns seit etwa 1976 lieferbar, und man schätzt, daß es allein in Deutschland etwa 10 000 Stück davon gibt.

„KIM“ bedeutet *Keyboard Input Monitor* und stellt einen Einplatinen-Computer mit sechsstelligem Siebensegment-Display, Hexadezimal-Tastatur, TTY-Schnittstelle (110 . . . 9600 Bd, ASCII) und Kassetten-Interface dar, der heute weniger als 500 DM kostet und dessen Auslegung zum Vorbild zahlreicher anderer Produkte wurde. Sein 2-KByte-Monitorprogramm gestattet die Anzeige und das Ändern beliebiger Adressen und Daten, den Programmstart und einen komfortablen Einzelschritt-Betrieb über die Hex-Tastatur oder ein ASCII-Terminal. Auf den KIM-1 beziehen sich die meisten in diesem Buch vorgestellten Programme, und sie rufen auch eine Reihe von Monitor-Unterprogrammen auf. Die wichtigsten sind:



Adresse	Funktion	Zerstörte Register
1E5A	GETCH	A, Y
1EA0	OUTCH	A, Y
1F9D	GETBYT	A, Y
1E3B	PRTBYT	A, Y
1E2F	CRLF	A, X, Y
1E9E	OUTSP	A, Y
1F63	INCPT	—
1F1F	SCANS	A, X, Y
1F6A	GETKEY	A, X, Y
1E1E	PRTPNT	A, Y

Das Monitorprogramm ist beim KIM-1 in zwei ICs (6530) fest gespeichert. Seine Aufgabe ist es, dem Benutzer die Kommunikation mit dem System zu erlauben; außerdem enthält es die Software für die Aufzeichnung von Programmen auf Kassette.



Am Ende eines Anwenderprogrammes erfolgt gewöhnlich ein Rücksprung zum Monitorprogramm, beim KIM-1 sinnvollerweise an die „Warmstartadresse“ 1C4F.

Das schriftliche Begleitmaterial bei den übrigen erwähnten 6502-Systemen ist wesentlich umfangreicher als beim KIM-1, so daß für die Mikrocomputer SYM-1, AIM-65 und PC-100 auf die Darstellung der Monitor-Unterprogramme verzichtet werden kann.

Der SYM-1 wird von Synertek hergestellt und ist praktisch eine verbesserte Version des KIM-1 mit 4 KByte Monitorprogramm und einem zweiten, schnelleren (allerdings unsichereren) Kassetten-Interface. Auch er besitzt eine Hex-Tastatur, ein Siebensegment-Display und eine Terminal-Schnittstelle, an die sich z.B. das Video-Interface KTM-2 anschließen läßt (es paßt natürlich auch zum KIM-1). Der SYM ist dem KIM gegenüber um etwa ein Drittel teurer.

Seit Ende 1978 ist ein 6502-System auf dem Markt, das derzeit wohl eines der interessantesten ist: Der AIM-65 von Rockwell. Er besitzt eine ASCII-Tastatur (ähnlich der einer Schreibmaschine), ein 20stelliges alphanumerisches Display, einen Thermodrucker – dessen Qualität allerdings für gehobene Ansprüche kaum ausreicht – und auch ein Kassetteninterface. Für weniger als 1000 DM steht somit ein System zur Verfügung, das als „Stand-Alone“-Version sogar das Programmieren mit einem Assembler oder – bei weniger zeitkritischen Programmen – in BASIC gestattet; letzteres allerdings nur mit EPROM-Optionen.

Mitte 1979 brachte Siemens mit dem PC-100 ein dem AIM-65 sehr ähnliches System heraus, das wahlweise als Fertiggerät im Gehäuse und mit BASIC-PROMs oder aber in Bausatzform und mit einem 8-KByte-Monitorprogramm (identisch mit dem AIM-65!) erhältlich ist.

Ein Umschreiben von KIM-Programmen, die die oben erwähnten KIM-Monitor-Unterprogramme verwenden, ist nur beim SYM-1 ohne weiteres möglich. Dagegen besitzen AIM-65 bzw. PC-100 (beide sind untereinander völlig kompatibel) völlig anders strukturierte Unterprogramme; auch ist, wie wir noch sehen werden, der Aufbau der Interrupt-Timer unterschiedlich.

Trotzdem dürfte es nach einiger Beschäftigung mit den Systemhandbüchern nicht schwerfallen, die in diesem Buch vorgestellten KIM-Programme auf andere 6502-Systeme umzuschreiben.

Selbstverständlich gibt es außer den erwähnten Mikrocomputern auch noch andere 6502-Systeme, die jedoch meist in erster Linie für das Programmieren in BASIC ausgelegt sind. Da sich dieses Buch jedoch speziell maschinenorientierten Problemen widmet, die sich zum großen Teil in BASIC nicht oder nur schwierig lösen lassen, wollen wir uns hier auf diese vier Mikrocomputer beschränken.

Bei der Auswahl eines Systems sollte man möglichst nicht nur Preis und Leistungsfähigkeit in Betracht ziehen, sondern auch auf die Erhältlichkeit von Software achten. Dies ist der Grund, warum auch heute noch der KIM-1 sehr beliebt ist. Ferner sollte man sich überlegen, ob man in naher Zukunft vorhat, den Mikrocomputer über ein ASCII-Terminal zu bedienen. Wenn dies der Fall ist, so wird man u.U. wiederum dem KIM-1 oder dem SYM-1 den Vorzug gegenüber dem AIM-65 oder dem PC-100 geben, da die „On-Board“-ASCII-Tastatur und das teure alphanumerische Display dann überflüssig werden.

#### **1.4    Gebrauchsanleitung für den KIM-Timer**

Auf der Platine des Mikrocomputers KIM-1 befinden sich, integriert in die ICs 6530, zwei programmierbare Intervall-Timer, die sich recht nützlich einsetzen lassen. Bedauerlicherweise enthalten die KIM-Handbücher nur recht vage Angaben über die Programmierung der Timer.

Die Timer bestehen jeweils aus einem Vorteiler, der den Systemtakt durch 1, 8, 64 oder 1024 teilt, und einem voreinstellbaren Zähler, der sich über den Datenbus mit einem bestimmten Wert zwischen hex 00 und hex FF laden läßt und dann von diesem Wert aus mit der bereits vorgeteilten Frequenz bis auf Null zählt.

Der Vorteiler-Faktor ergibt sich aus der Adresse, an die die Daten gespeichert werden, mit denen der eigentliche Timer

geladen werden soll:

<i>Timer 1</i>	<i>Timer 2</i>	<i>Vorteilung</i>
1704	1744	1
1705	1745	8
1706	1746	64
1707	1747	1024

Hat der Timer nach Ablauf der vorprogrammierten Zeit auf Null gezählt, so wird an der Adresse 1707 das N-Flag gesetzt, das dann z.B. mit dem BIT-Befehl abgefragt werden kann. Eine einfache Zeitschleife für  $255 \cdot 1024 \mu s = 261,12 \text{ ms}$  könnte dann z.B. so aussehen:

0050	A9 FF	LDA # FF
0052	8D 07 17	STA 1707
0055	2C 07 17	BIT 1707
0058	10 FB	BPL 0055

Der Timer 1 wird hier mit dem Wert FF geladen; der Vorteiler-Faktor ist 1024, da die Adresse 1707 angesprochen wird; und bis die Zeit abgelaufen ist, wartet der Prozessor in der Schleife 0055/0058. Zwischen die Zeilen 0052 und 0055 lassen sich bei entsprechender Korrektur der relativen Adressierung im BPL-Befehl noch andere Befehle unterbringen, z.B. ein Unterprogramm-Aufruf zur Anzeige von Daten auf dem KIM-Display.

Ob die Zeit abgelaufen ist, wird also durch Testen des N-Flags im Statusregister an der Adresse 1707 erkannt. Will man dagegen den aktuellen Zählerstand auslesen, so kann das z.B. mit dem Befehl LDA 1706 geschehen, also stets an der Adresse 1706.

Einer der beiden KIM-Timer, nämlich Timer 1, ist in der Lage, bei Ablauf der vorprogrammierten Zeit einen Interrupt auszulösen. Addiert man zu den oben angegebenen Adressen jeweils 0008, so werden die Vorteiler-Faktoren wie oben eingestellt, nach dem Ablauf der gewünschten Zeit wird jedoch am Anschluß PB 7 (der als Eingang programmiert sein muß) eine negative Flanke ausgelöst.

Verbindet man nun PB 7 z.B. mit der NMI-Leitung, so läßt sich der Timer 1 so programmieren, daß nach Ablauf einer gewissen Zeit ein NMI-Interrupt ausgelöst wird. Die Befehlsfolge zum Starten des Timers kann hier z.B. lauten:

0050	A9 FF	LDA #FF
0052	8D 0F 17	STA 170F

In diesem Beispiel tritt nach 261,12 ms ein nicht maskierbarer Interrupt auf. Verwendet man die gleiche Befehlsfolge in der Interrupt-Routine selbst, so entsteht ein periodischer Interrupt. Dann ist jedoch das Initialisieren durch diese Befehlsfolge auch zu Beginn des Hauptprogramms erforderlich, um den Interrupt das erste Mal auszulösen.

Die Methode, den Timer als periodische Interrupt-Quelle zu verwenden, ist sehr universell anwendbar und ersetzt externe Interruptquellen wie etwa RC-Impulsgeneratoren. Sie bietet auch den Vorteil, daß die Interrupt-Frequenz praktisch quarzstabil ist, da der Timer mit dem quarzgesteuerten Systemtakt arbeitet. Es sei noch erwähnt, daß der AIM-65 im Gegensatz zum KIM-1 mit 16-bit-Timern (6522-VIA) arbeitet, die anders zu programmieren sind.

## 1.5 Zusätzliche Befehle und Maskenfehler beim 6502

Den Mikroprozessor 6502 haben Sie bisher vermutlich erheblich unterschätzt, was seinen Befehlsvorrat angeht. Wegen der Struktur der Befehlsdecodierung auf dem Chip werden manche Hexadezimalzahlen, die nicht im Programmierhandbuch enthalten sind, als u. U. recht nützliche Befehle interpretiert und ausgeführt. Sie stellen Bitmuster-Kombinationen „offizieller“ Operationscodes dar und sind in der *Tabelle* aufgelistet. Manche Hexadezimal-Zahlen führen – als Befehl interpretiert – allerdings dazu, daß der Prozessor nur noch mit einem Reset-Befehl „aus dem Weltraum“ zurückgeholt werden kann, wie z.B. 12 Befehle, deren letztes Halbbyte (niederwertiges „Nibble“) 2 ist.



## Zusätzliche 6502-Operationscodes

<i>Hex-Code</i>	<i>Wirkung</i>
A7 aa	Akku und X-Register werden mit dem Inhalt der Zero-Page-Adresse aa geladen
87 aa	Das Ergebnis einer UND-Funktion zwischen Akku und X-Register wird an die Zero-Page-Adresse aa gespeichert
97 aa	wie 87, jedoch wird das UND-Ergebnis an die Zero-Page-Adresse aa + Y gespeichert
9E aaaa	Das Ergebnis einer UND-Funktion zwischen dem X-Register und der Konstante 02 wird an die Adresse aaaa gespeichert
9F aaaa	Das Ergebnis einer UND-Funktion zwischen dem Akku, dem X-Register und der Konstante 02 wird an die Adresse aaaa gespeichert
E7 aa	Der Inhalt der Zero-Page-Adresse aa wird um eins erhöht; das Resultat wird dann vom Akkuinhalt subtrahiert
C7 aa	Der Inhalt der Zero-Page-Adresse aa wird um eins erniedrigt und anschließend mit dem Akkuinhalt verglichen; die Status-Flags werden wie bei CMP-Befehlen beeinflusst.

Kaum bekannt ist bisher ein Maskenfehler des 6502-Chips geworden: Folgt einem indirekten Sprung (hexadezimaler Operationscode 6C) das Byte FF, so wird der Sprung falsch ausgeführt. Lautet der Befehl z.B. 6C FF 03, so wird die Sprungadresse nicht etwa aus den Zellen 03FF und 0400 geholt, sondern aus den Zellen 03FF und 0300, was zu einem zunächst völlig unerklärlichen Verhalten des Programms führt. Dem Operationscode 6C sollte man also nicht das Byte FF folgen lassen. Bis Mitte 1979 trat dieser Effekt noch bei 6502-Chips aller Hersteller auf; eine Maskenänderung ist noch nicht abzu-  
sehen.



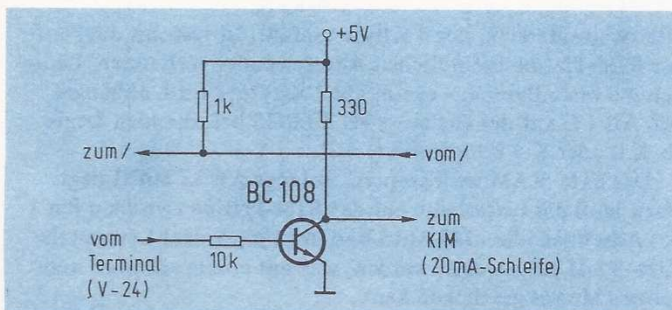
## 2 Hardware-Tips

### 2.1 RS-232/V-24-Interface für den KIM-1

Da der Mikrocomputer KIM-1 nur über eine 20-mA-Schnittstelle (Stromschleife) verfügt, kann er nicht direkt mit Terminals verbunden werden, die für V-24- bzw. RS-232-Schnittstellen ausgelegt sind. Zur Pegelanpassung ist in diesem Fall aber eine einfache Schaltung mit einem NPN-Transistor ausreichend (Abb. 2.1). Damit läßt sich dann z.B. ein CT-64-Terminal mit dem KIM-1 verbinden. Die angegebene Schaltung hält sich, was Ein- und Ausgangspegel betrifft, wegen ihrer Einfachheit zwar nicht ganz an die Norm, funktioniert aber in allen bisher erprobten Fällen anstandslos.

### 2.2 „Automatische“ Hardware-Speicherverschiebung

Daß selbst Mikrocomputer-Systeme mit dem gleichen Prozessor nicht unbedingt software-kompatibel sind, liegt oft



2.1 Anschluß eines RS-232-Terminals an die 20-mA-Schnittstelle des Mikrocomputers KIM-1

daran, daß unterschiedliche RAM-Speicheradressen verwendet werden.

Bei dem Mikrocomputer KIM-1 sind die verbreitetsten Erweiterungen die RAM-Adressen 0400 . . . 13FF oder 2000 . . . 2FFF. Beide Erweiterungsmöglichkeiten lassen sich mit einer schon recht preiswerten 4-KByte-RAM-Karte realisieren. Programme, die für den Adressenbereich 0400 . . . 13FF geschrieben wurden, laufen nun aber leider nicht ohne weiteres im Bereich 2000 . . . 2FFF, es sei denn, man ändert alle absoluten Adressen im Programm, was recht mühsam ist.

Um Ihnen nun die Arbeit mühsamer Programmverschiebungen zu ersparen, sei eine Möglichkeit aufgezeigt, die dieses Problem löst. Auf ähnliche Art und Weise läßt sich das natürlich auch bei anderen Systemen durchführen.

Abb. 2.2 zeigt die Schaltung der Hardware-Adressenduplizierung. Es ist keine manuelle Umschaltung erforderlich, in welchem Adressenbereich gearbeitet werden soll – das tut das Programm selbst durch Ansprechen der Adressenleitung AB 13. Ist diese Leitung H, so ist der Bereich 2000 . . . 2FFF gemeint, andernfalls der Bereich 0400 . . . 13FF.

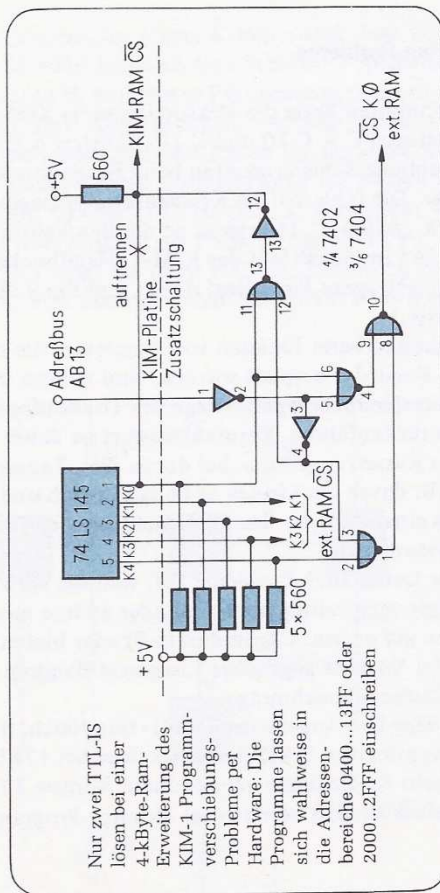
Je nach dem Zustand von AB 13 liegt das externe RAM also in folgenden Adressenbereichen:

2000 . . . 23FF    oder    1000 . . . 13FF    und  
2400 . . . 2FFF    oder    0400 . . . 0FFF.

Der zweite Bereich, der 3 KByte umfaßt, ist fest mit dem auf der KIM-Platine befindlichen Adreßdecoder verbunden. Lediglich der erste Bereich – er umfaßt 1 KByte – wird, abhängig von AB 13, mit der aus nur zwei TTL-IS bestehenden Logik als K 0 oder K 4 interpretiert.

Das KIM-RAM wird gesperrt, solange AB 13 auf H liegt. Dazu muß die Leiterbahn auf der KIM-Platine zwischen Pin 1 des Adreßdecoders 74LS145 und der Chip-Select-Leitung des KIM-RAM aufgetrennt werden, was mit einem scharfen und spitzen Messer geschehen kann.

Diese Hardware-Programmverschiebung läßt sich natürlich nur dann durchführen, wenn der höherwertige Adressenteil,



2.2 Duplizieren eines 4-K-RAM in zwei Adressbereichen

also AB 13 . . . 15, nicht von der externen RAM-Platine selbst decodiert wird.

### 2.3 Kassetten-Probleme

Bei einer bestimmten Serie des Mikrocomputers KIM-1 haben die Kondensatoren C 9, C 10 und C 11 (Sollwert 6,8 nF) nur 680 pF. Erhebliche Schwierigkeiten beim Einlesen von Kassetten sind die Folge. Die fehlerhaften Kondensatoren tragen meist den Aufdruck „680/10“. (Übrigens ist die Beschaltung des PLL-IC LM 565 im Schaltbild des KIM-1-Handbuches nicht vollständig eingetragen: Pin 1 liegt direkt und Pin 9 über 22 nF an Masse.)

Schwierigkeiten beim Einlesen von Kassetten, die nicht mit dem eigenen Recorder bespielt wurden, sind nahezu immer auf eine unterschiedliche Spaltjustage des Tonkopfes in den Recordern zurückzuführen. Empfehlenswert ist daher die Verwendung von Kassettengeräten, bei denen die „Taumelschraube“ von außen z.B. durch ein kleines Loch zugänglich und während des Betriebes einstellbar ist. Dies geschieht ganz einfach auf beste Höhenwiedergabe.

Ab und zu treten auch Probleme auf, weil die verwendeten Kassetten nicht verwindungssteif sind oder andere mechanische Unstabilitäten aufweisen. Chromdioxid-Bänder bieten elektrisch keinerlei Vorteile gegenüber Eisenoxid-Bändern – zumindest bei der Datenaufzeichnung.

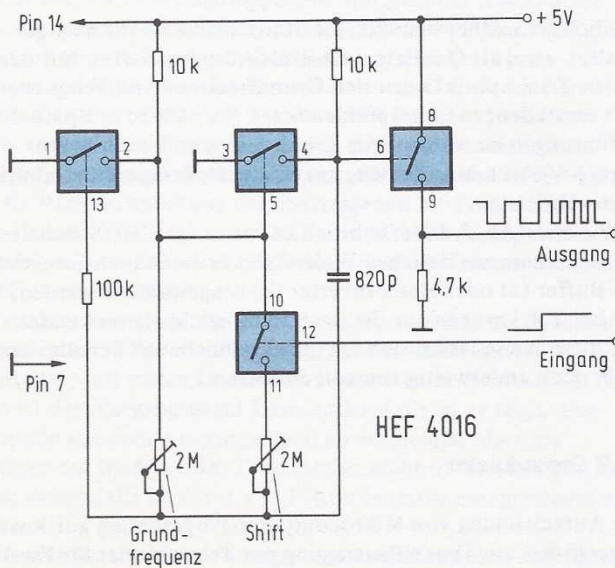
Noch ein Tip: Die Angabe im KIM-1-Handbuch, die obere Grenze des Applikation-RAM-Bereiches liege bei 17EB, ist falsch. Vielmehr ist die letzte verwendbare Adresse 17E6; die folgenden Zellen werden bereits vom Monitor-Programm verwendet.

## 2.4 FSK-Modem

### 2.4.1 Modulator

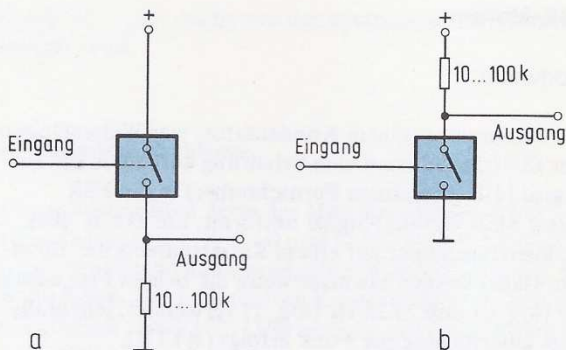
Aus zwei Trimpotenis, einem Kondensator, vier Widerständen und einer IS 4016 lässt sich eine Schaltung aufbauen, die ein Digitalsignal (z.B. von einem Fernschreiber) in ein FSK (Frequency Shift Keying)-Signal umformt, um es z.B. über Funk zu übertragen oder auf einem Kassettenrecorder aufzuzeichnen. Dabei werden normalerweise die beiden Frequenzen 1275 Hz (log. 0) und 2125 Hz (log. 1) verwendet, jedenfalls soweit die Übertragung per Funk erfolgt (RTTY).

Abb. 2.4.1 zeigt die recht einfache Schaltung des FSK-Oszillators. Von den vier Schaltern im 4016 wird nur einer



2.4.1 Schaltung eines CMOS-FSK-Modulators





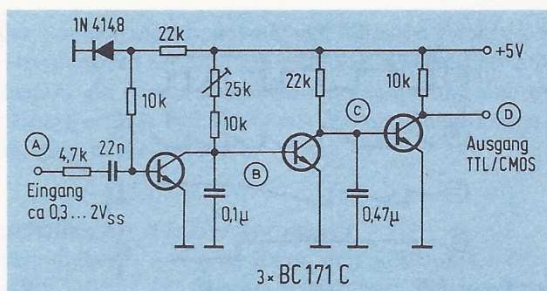
#### 2.4.2 Ein CMOS-Schalter lässt sich als Inverter oder Buffer einsetzen

wirklich als solcher benutzt; die übrigen sind als Inverter geschaltet, zwei als Oszillator, einer als Ausgangsbuffer. Mit den beiden Trimpotis lassen sich Grundfrequenz und Frequenzshift einstellen; es ist empfehlenswert, Wendel- bzw. Spindelausführungen zu wählen. Als Kondensator sollte ein Styroflex typ Verwendung finden, um eine gute Temperaturstabilität zu gewährleisten.

Wie aus Abb. 2.4.2 ersichtlich ist, kann ein CMOS-Schalter mit nur einem zusätzlichen Widerstand in einen nichtinvertierenden Buffer (a) oder einen Inverter (b) umgewandelt werden; in Abb. 2.4.1 wurde nur die letztere Möglichkeit verwendet. Auf diese Weise lassen sich oft „übriggebliebene“ Schalter eines 4016 noch anderweitig sinnvoll einsetzen.

#### 2.4.2 Demodulator

Zur Aufzeichnung von Mikrocomputer-Programmen auf Kassettenrecorder, zur Datenübertragung per Telefon oder für Funkfern schreiben verwendet man meist das sogenannte FSK-Verfahren, auf Neuhochdeutsch „Frequency Shift Keying“. Die digi-



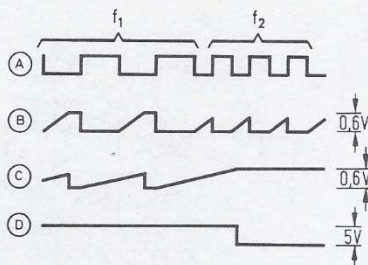
2.4.3 FSK-Demodulator mit drei Transistoren

talen Werte 0 und 1 werden dabei als zwei Tonfrequenzen übertragen, die um einige hundert Hertz differieren. Welcher Code verwendet wird, z.B. ASCII oder Baudot, spielt dabei zunächst keine Rolle; als Übertragungsgeschwindigkeit bei ASCII-Funkfern schreiben haben sich allerdings 110 bzw. 300 Baud (bit pro Sekunde) und im Baudot-Code 45,45 Baud eingebürgert.

Die nachfolgend beschriebene Schaltung *Abb. 2.4.3* eignet sich für Geschwindigkeiten bis etwa 600 Baud und zeichnet sich in erster Linie durch den extrem geringen Bauelementeaufwand aus. Sie ermöglicht die Demodulation des FSK-Signals, d.h. die Rückverwandlung der übertragenen Nf-Töne (z.B. 1275 Hz und 2125 Hz) in ein TTL- und CMOS-kompatibles Digitalsignal, das z.B. einem Fernschreiber oder einem Terminal zugeführt werden kann.

Die Anordnung mit nur drei Transistoren wirkt als Frequenzdiskriminator: Wenn die Eingangsfrequenz höher als ein bestimmter, mit einem Trimpoti abgleichbarer Wert (z.B. 1,7 kHz) ist, so ist der Ausgangspegel Low; andernfalls ist er High. Um Nf-Anteile auf dem Ausgangssignal zu vermeiden, darf die niedrigste der beiden FSK-Frequenzen nicht unter etwa 800 Hz liegen; andernfalls ist der 0,47-μF-Kondensator entsprechend zu erhöhen.

Ein praktischer Versuch zeigte erstaunlicherweise, daß diese einfache Schaltung bei der Aufnahme von Programmen mit 110 Baud auf einem üblichen Kassettenrecorder eine geringere



2.4.4 Impulsschema des FSK-Demodulators

Fehlerquote zeigte als eine PLL-Demodulatorschaltung mit dem IC NE 567, das auf die höhere der beiden FSK-Frequenzen abgestimmt war. Dies ist vor allem auf die wesentlich geringere Empfindlichkeit der Transistorschaltung gegenüber Phasenschwankungen des Nf-Signals zurückzuführen, die z.B. von dem unvermeidlichen Schlupf des Bandes hervorgerufen werden können.

Der Impulsplan (Abb. 2.4.4) macht deutlich, wie das Ganze funktioniert.

## 2.5 Norm für die ASCII-Übertragung im Amateurfunk

Mitte Januar 1979 trafen sich in München rund 30 Funkamateure, die sich zum Ziel gesetzt hatten, die Übertragung von ASCII- bzw. ISO-7-bit-Zeichen per Funk zu vereinheitlichen. Der Code ist seit 1976 von der Deutschen Bundespost für zulässig erklärt und erfreut sich speziell bei den Mikrocomputer-Anwendern unter den Funkamateuren steigender Beliebtheit, da er doppelt so viele mögliche Zeichen umfaßt wie der bisher übliche Baudot-5-bit-Code.

Die Übertragungsparameter wurden so gewählt, daß Inkompatibilitäten mit den üblichen Terminals und Mikrocomputern möglichst vermieden werden. Ferner gestattet es das Toleranz-

feld der Tonfrequenzen, sowohl mit üblichen Funkfernseh-  
FSK-Modulatoren als auch mit dem sog. Kansas-City-Standard  
zu arbeiten, der für die Magnetbandaufzeichnung eine weite  
Verbreitung erlangt hat. Geeignete Modem-Schaltungen finden  
sich übrigens in diesem Buch.

### *Eigenschaften der ASCII-Übertragung*

Geschwindigkeit	300 Baud, min. 1 Stopbit
Parity-Bit	konstant Null bzw. Zeichen-Invertierung
Modulationsart	F2; H = 2100 ... 2425 Hz, L = 1175 ... 1300 Hz
Sendeformat	2 ... 5 s H-Ton; X DE Y; Text; PSE K X, Y = Rufzeichen; je max. 7 Zeichen, ohne Leerraum vor und hinter der Ziffer)
Steuerzeichen	Line Feed (0A), Carriage Return (0D), Back Space (08), NUL (00, keine Wirkung)
Zeichensatz	Alle ASCII-Zeichen von hex 20 bis hex 7E; Rufzeichen und Systembefehle nur hex 20 ... 5A.



## 3 Programme für den „rohen“ KIM – 1

### 3.1 Funktionsgenerator

Der Software-Funktionsgenerator entstand ursprünglich aus dem Bedürfnis, die Resonanzfrequenzen von Nf-Filtern möglichst genau einzustellen. Für diesen Zweck ist es notwendig, ein Sinussignal exakt einstellbarer und reproduzierbarer Frequenz zu erzeugen. Das Vorhaben, einen solchen Sinusgenerator mit z.B. 2 Hz Genauigkeit per Hardware zu realisieren, wurde aus Aufwandsgründen schnell aufgegeben; stattdessen fiel die Wahl auf den Mikrocomputer KIM-1, der über einen an die I/O-Ports PA 0 . . . 7 angeschlossenen Digital-Analog-Wandler nicht nur Sinussignale, sondern ganz beliebige Kurvenformen erzeugen kann. Wegen seiner recht schnellen CPU 6502 ist er in der Lage, das Ausgangssignal nach der Sample-Methode zu erzeugen.

Das Sinussignal kann vom Mikrocomputer nicht kontinuierlich, sondern nur als Approximation in kleinen Amplitudenschritten erzeugt werden. Zu diesem Zweck ist im Adressenbereich 0200 . . . 02FF eine Amplitudentabelle für genau eine Sinusperiode gespeichert, die zyklisch abgetastet und an den D/A-Wandler ausgegeben wird. Selbstverständlich können auch andere Kurvenformen dort abgespeichert werden, z.B. Dreiecksfunktionen oder auch komplexere Dinge, deren Amplitudenwerte sinnvollerweise mit einem programmierbaren Taschenrechner oder auch einem BASIC-Computer ausgerechnet werden. *Abb. 3.1.1* enthält eine Tabelle für ein Viertel der Sinusperiode; das kleine Programm in *Abb. 3.1.2* errechnet daraus die übrigen drei Viertel und speichert sie an die richtigen Adressen ab. Es funktioniert unabhängig vom übrigen Programm und läßt sich bei der Tabellenerstellung aller doppelt symmetrischen Kurvenformen anwenden.

```

0200 FF FF FF FF FE FE FE FD FD FC FB FA FA F9 F8 F6
0210 F5 F4 F3 F1 F0 EE ED EB EA E8 E6 E4 E2 E0 DE DC
0220 DA D7 D5 D3 D0 CE CB C9 C6 C4 C1 BE BC B9 B6 B3
0230 B0 AD AA A7 A5 A2 9E 9B 98 95 92 8F 8C 89 86 83
0240 80 7C

```

### 3.1.1 Wertetabelle für ein Viertel einer Sinus-Periode

#### 3.1.2 Programm zum Errechnen der übrigen drei Viertel des Sinus

```

00B0 A2 40 LDX #40
00B2 A0 00 LDY #00
00B4 B9 00 02 LDA 0200,Y
00B7 9D C0 02 STA 02C0,X
00BA 49 FF EOR #FF
00BC 99 80 02 STA 0280,Y
00BF 9D 40 02 STA 0240,X
00C2 C8 INY
00C3 CA DEX
00C4 10 EE BPL 00B4
00C6 4C 4F 1C JMP 1C4F

```

#### 3.1.3 Erzeugung des Ausgangssignales durch Tabellenabfrage

```

0088 A9 FF LDA #FF
008A 8D 01 17 STA 1701
008D BD 00 02 LDA 0200,X
0090 8D 00 17 STA 1700
0093 98 TYA
0094 65 E3 ADC E3
0096 A8 TAY
0097 8A TXA
0098 65 E4 ADC E4
009A AA TAX
009B 18 CLC
009C EA NOP
009D EA NOP
009E 90 ED BCC 008D

```

```

0000 D8 A2 FF 9A E8 8E FA 17 8E FB 17 86 F9 86 FA 20
0010 37 00 D0 FB 20 37 00 F0 FB 20 6A 1F C9 0A 30 07
0020 38 E9 08 85 8F D0 1C 2A 2A 2A A0 04 2A 26 F9
0030 26 FA 88 D0 F8 F0 D8 A2 0D A0 02 A9 7F 8D 41 17
0040 4C 28 1F A9 00 85 E2 85 E1 A5 F9 D0 04 A5 FA F0
0050 19 38 F8 A5 F9 E9 01 85 F9 E6 E1 D0 02 E6 E2 B0
0060 E8 38 A5 FA E9 01 85 FA B0 F5 D8 A5 E2 85 E4 A5
0070 E1 A2 06 46 E4 6A CA D0 FA 18 65 E1 85 E3 A5 E4
0080 65 E2 85 E4 06 E3 26 E4

```

#### 3.1.4 Programmteil zur Eingabe der gewünschten Frequenz

Das nächste Problem ist nun, die zyklische Abfrage der Kurvenform-Tabelle so zu gestalten, daß dabei die gewünschte Frequenz herauskommt. Zu diesem Zweck wird nicht Byte für Byte der Tabelle nacheinander abgefragt, sondern es werden umso mehr Bytes ausgelassen, je höher die gewünschte Frequenz ist. Die Sollfrequenz wird dabei durch eine 16-bit-Zahl repräsentiert, deren niederwertiges Byte von dem Programm in *Abb. 3.1.3* in das Y-Register übernommen wird, das höherwertige in das X-Register. Letzteres wird als Index für die Kurvenformtabelle verwendet. Bei jedem Schleifendurchlauf wird die 16-bit-Zahl in den X- und Y-Registern nun um den 16-bit-Betrag erhöht, der in 00E4 und 00E3 steht.

Ein Schleifendurchlauf dauert genau  $31 \mu\text{s}$  (1 MHz Taktfrequenz des Mikrocomputers vorausgesetzt). Bezeichnet man die 16-bit-Zahl in den Zellen 00E3 und 00E4 mit F, so gilt die Beziehung

$$f = \frac{F}{31 \mu\text{s} \cdot 256^2}$$

so daß - wenn man die Frequenz in Hz eingeben möchte - ein Umrechnungsfaktor  $F/f = 2,031616$  erforderlich ist.

Neben den Routinen für Display und Tastatur-Abfrage enthält das in *Abb. 3.1.4* hexadezimal aufgelistete Programm die erforderliche Dezimal-Hexadezimal-Umwandlung (16 bit) und die Multiplikation mit dem Umrechnungsfaktor  $2,0316 \dots$ , der hier durch die Approximation

$$2,0316 \approx \left(1 + \frac{1}{64}\right) \cdot 2$$

gebildet wird. Der dabei entstehende Fehler ist  $180 \cdot 10^{-6}$ , was bei einer Frequenz von 1 kHz nur 0,18 Hz ausmacht und somit vernachlässigbar klein gegenüber der Quantisierungs-Unsicherheit von 1 Hz ist.

### *Die Tastenfunktionen des KIM*

Ist das Programm einmal an der Adresse 0000 gestartet, so haben die KIM-Tasten eine andere Bedeutung als im Programmiermodus. Auffallend ist zunächst, daß das Display nur vier Stellen anzeigt; die restlichen beiden sind dunkelgetastet. Jetzt kann mit den Tasten 0 . . . 9 eine vierstellige Frequenz (1 . . . 9999 Hz) dezimal eingegeben werden. Hat man sich einmal vertippt, so läßt sich mit der Stop-Taste (ST) das Display wieder löschen.

Nach einem Druck auf die Taste A wird die im Adressenbereich 0200 . . . 02FF gespeicherte Kurvenform, z.B. ein Sinus, mit der gewählten Frequenz über den D/A-Wandler ausgegeben. An PA 7 steht dabei gleichzeitig ein Rechteck zur Verfügung. Will man eine andere Frequenz eingeben, so braucht man nur die Stop-Taste zu drücken, und das Display, das während der Signalerzeugung dunkel ist, leuchtet wieder.

Die Tasten B . . . F haben die gleiche Bedeutung wie die Taste A, sprechen aber Kurvenform-Tabellen in anderen Speicherbereichen an: Taste B z.B. 0300 . . . 03FF. Auf diese Weise ist es möglich, mehrere unterschiedliche Wellenformen abzurufen. Ein kleiner Trick: Lädt man den Bereich 0300 . . . 03FF mit irgendeinem Programm, so erzeugt ein Druck auf die Taste B u.U. ein breitbandiges Rauschen, da die Programmbefehle als statistisch verteilte Amplituden interpretiert werden.

Für bestimmte Aufgaben, wie etwa das oben erwähnte Einstellen von Filter-Durchlaßkurven, ist dieser Software-Funktionsgenerator ideal geeignet, gestattet er doch eine exakte digitale Frequenzeingabe. Aber: Die Frequenz läßt sich eben nur in Schritten variieren, und zudem weist sie wegen der verwendeten Sample-Methode noch einen bestimmten, wenn auch sehr kleinen Störhub auf. Ferner bringt es das Sample-Prinzip mit sich, daß ein gewisses Quantisierungs-Geräusch entsteht, das dem Störabstand Grenzen setzt. Und schließlich hat das Ausgangssignal bei knapp 10 kHz mit einem Sinus kaum noch etwas zu tun, da die Sample-Frequenz ja „nur“  $1/31$  MHz = 32,26 kHz beträgt. Dieses Problem läßt sich jedoch mit einem 15-kHz-Tiefpaß leicht beseitigen.



Als D/A-Wandler eignet sich z.B. der Typ ZN 425 (s.a. Kapitel 3.3).

### 3.2 Nf-Zähler

Sowohl in der Analogtechnik als auch in der Mikrocomputer-Technik müssen manchmal Frequenzen gemessen werden. Was liegt näher, als den Mikrocomputer selbst damit zu beauftragen? So lassen sich z.B. Modem-Tonfrequenzen oder Kassetten-Interfaceschaltungen exakt abgleichen, ohne einen teuren Frequenzzähler zu benötigen.

Das in *Abb. 3.2* als KIM-Version aufgelistete Programm mißt 100 ms lang die Zahl der am I/O-Port PB 3 auftretenden Perioden, wobei jede aus einem positiven und einem negativen Nulldurchgang besteht. Die Aufsummierung erfolgt dezimal –beim 6502 ist das besonders einfach, da sich dieser Mikroprozessor mit dem Befehl SED (Set Decimal) in die dezimale Arithmetik umschalten läßt.

Nach diesen 100 ms tritt ein Interrupt des programmierbaren Timers auf. Das Programm springt an die Adresse 003B, setzt den Stackpointer auf seine Grundstellung zurück und zeigt etwa 260 ms lang das Zählergebnis an. Da die Auflösung 10 Hz beträgt, muß dafür gesorgt werden, daß –um eine Anzeige in Hz zu erreichen– noch genau eine Stelle folgt; d.h. die niederwertigste Stelle des Zählergebnisses muß in die vorletzte Anzeigenstelle geschoben werden. Dies erfolgt mit LSR- und ROR-Befehlen; zu beachten ist, daß die 6502-Chips, die vor dem Juni 1976 gefertigt wurden, den ROR-Befehl nicht kennen! In diesem Fall müßte das Programm geringfügig umgeschrieben werden, um die entsprechende Befehlsfolge durch eine andere, die kein ROR beinhaltet, zu ersetzen.

Das zu messende Eingangssignal kann auf zwei Arten dem Mikrocomputer zugeführt werden; entweder legt man es als TTL-Pegel direkt an den I/O-Port PB 3, oder man verwendet eine Schaltung, die im nächsten Kapitel vorgestellt wird. Dann ist nämlich über die KIM-Tastatur eine digitale Einstellung des Triggerpegels möglich, wobei die Taste 0 dem Pegel -2,55 V und die

0000 LDA #FF	0037 CLD	0006 A9 FF 8D 01
0002 STA 1701	0038 JMP 001D	0004 17 A9 00 85
0005 LDA #00	003B LDX #FF	0008 FA 85 FB 8D
0007 STA FA	003D TXS	000C FB 17 A9 80
0009 STA FB	003E CLD	0010 8D 00 17 A9
000B STA 17FB	003F STX 1707	0014 3B 8D FA 17
000E LDA #80	0042 LDA #00	0018 A9 62 8D 0F
0010 STA 1700	0044 STA F9	001C 17 A9 88 2C
0013 LDA #3B	0046 LDX #04	0020 02 17 00 FB
0015 STA 17FA	0048 LSR FB	0024 2C 02 17 F0
0018 LDA #62	004A ROR FA	0028 FB F8 A5 FA
001A STA 170F	004C ROR F9	002C 38 69 00 85
001D LDA #08	004E DEX	0030 FA A5 FB 69
001F BIT 1702	004F BNE 0048	0034 00 85 FB D8
0022 BNE 001F	0051 JSR 1F1F	0038 4C 1D 00 A2
0024 BIT 1702	0054 BIT 1707	003C FF 9A D8 8E
0027 BEQ 0024	0057 BPL 0051	0040 07 17 A9 00
0029 SED	0059 JSR 1F6A	0044 85 F9 A2 04
002A LDA FA	005C CMP #10	0048 46 FB 66 FA
002C SEC	005E BPL 0066	004C 66 F9 CA D0
002D ADC #00	0060 ASL A	0050 F7 20 1F 1F
002F STA FA	0061 ASL A	0054 2C 07 17 10
0031 LDA FB	0062 ASL A	0058 F8 20 6A 1F
0033 ADC #00	0063 ASL A	005C C9 10 10 06
0035 STA FB	0064 STA 0F	0060 0A 0A 0A 0A
	0066 JMP 0006	0064 85 0F 4C 00
		0068 00 00 00 00

### 3.2 KIM-Programm für die Messung von Frequenzen im Nf-Bereich

Taste F dem Pegel + 2,4 V entspricht. Selbstverständlich sind alle Zwischenwerte in Schritten von etwa 0,3 V einstellbar, und die Taste 8 legt den Triggerpegel auf 0 V.

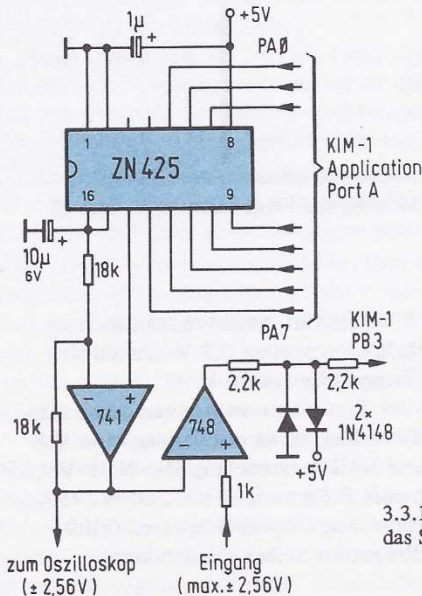
Um zu erreichen, daß der Timer-Interrupt einen nicht maskierbaren Interrupt (NMI) auslöst, ist es notwendig, eine Verbindung zwischen PB 7 und NMI herzustellen. Der NMI-Vektor (003B) wird automatisch vom Programm in die Zellen 17FA und 17FB gespeichert. Außer den Display-Registern 00F9, 00FA, 00FB sowie dem Programm selbst werden keine Speicherplätze belegt.

### 3.3 Speicheroszilloskop

Um langsam ablaufende Vorgänge, z.B. die Entladekurve eines Nickel-Cadmium-Akkus, darzustellen, ist die Verwendung eines Speichervorsatzes in Verbindung mit einem handelsüblichen Oszilloskop recht praktisch.

Das Abspeichern einer analogen Information ist leider nicht ohne weiteres möglich; ein Mikrocomputer kann schließlich nur digitale Bits oder Bytes speichern. Aus diesem Grunde ist zunächst einmal ein Analog-Digital-Wandler erforderlich, der die zu messende Eingangsspannung in ein digitales Wort, ein 8-bit-Byte, umsetzt.

Analog-Digital-Wandler sind nun leider recht teuer, und da man zur Wiedergabe der gespeicherten Spannungswerte ohnehin



3.3.1 Notwendige Hardware für das Speicheroszilloskop

einen Digital-Analog-Wandler benötigt, wurde eine preisgünstigere Lösung verwendet.

Sie besteht darin, den Digital-Analog-Wandler durch zwei Operationsverstärker und etwas Software auch als Analog-Digital-Umsetzer anzuwenden. Dazu dient die in *Abb. 3.3.1* gezeigte Schaltung. Der Mikrocomputer ändert gezielt die am D/A-Wandler anstehenden Daten (KIM-Port A) solange, bis

a2 ff	0061	ao oo	00c5	a9 ff
9a		84 fa		8d o1 17
a9 oo		a9 o2		8d oo 17
8d fa 17		85 fb		a9 8o
8d fb 17		a9 ff		aa
85 f9		8d o1 17		4d oo 17
85 fa		8d oo 17		8d oo 17
85 fb		a9 oo		a9 o8
85 f6		8d oo 17		2c o2 17
a9 o2		b1 fa		fo o7
85 f7		8d oo 17		8a
2o 1f 1f		2o 63 1f		4d oo 17
do fb		a5 fb		8d oo 17
2o 1f 1f		c9 o4		8a
fo fb		do f2		4a
2o 1f 1f		fo db		9o e7
2o 6a 1f		a9 oo	00e8	4c oo o1
c9 1o		85 f3		
fo 5b		85 f4	o1oo	ad oo 17
c9 11		85 f5		ao oo
fo 32		d8		91 f6
c9 oc		a5 f3		e6 f6
fo 16		c5 f9		do ob
c9 oa		do oc		e6 f7
1o eo		a5 f4		a5 f7
2a		c5 fa		c9 o4
2a		do o6		do o3
2a		a5 f5		4c oo oo
2a		c5 fb		4c 86 oo
ao o4		fo 24	o119	ende
2a		a9 1e		
26 f9		8d o6 17		
26 fa		2c o7 17		
26 fb		1o fb		
88		f8		
do f6		18		
fo ce		a5 f3		
a9 o2		69 o1		
85 fb		85 f3		
a9 oo		9o d9		
85 fa		a5 f4		
ao oo		69 oo		
91 fa		85 f4		
2o 63 1f		9o f6		
a6 fb		a5 f5		
eo o4		69 oo		
do f5		85 f5		
4c oo oo		9o ee		

### 3.3.2 6502-Maschinencode für das KIM-Speicheroszilloskop



dieser eine der zu messenden Eingangsspannung entsprechende Spannung liefert. Dies geschieht nach einem Algorithmus, der ähnlich wie das bekannte Zahlenratespiel Hi-Lo arbeitet und eine Umsetzzeit von nur 200 . . . 300  $\mu$ s für ein 8-bit-Wort ermöglicht. *Abb. 3.3.2* zeigt das gesamte Programm in hexadezimaler Form; der Teil im Adressenbereich 00C5 . . . 00E8 dient der A/D-Umsetzung. (Zur Orientierung enthält *Abb. 3.3.2* einige Adressenangaben; Startadresse ist 0000.)

Die Bedienung dieses Mikrocomputer-Speicherzusatzes ist denkbar einfach. Sobald das Programm an der Adresse 0000 durch Drücken der Taste GO gestartet wurde, zeigen alle sechs Siebensegment-Displays des KIM-1 eine Null. Mit den Tasten 0 . . . 9 läßt sich nun eine beliebige Aufzeichnungsdauer eingeben; hat man sich vertippt, kann man mit der Stop-Taste (ST) die Anzeige wieder löschen. Es lassen sich Zeiten zwischen 1 s und 999 999 s eingeben.

Die Aufzeichnung des analogen Spannungsverlaufes am nichtinvertierenden Eingang des Operationsverstärkers 748 beginnt, sobald nun die Taste AD („Analog/Digital“) gedrückt wird. Für die gerade gewählte Aufzeichnungszeit bleibt das Display dunkel und zeigt dann wieder 000 000 an. Mit der Taste DA (Digital/Analog) lassen sich nun die 512 Bytes des Speicherbereiches 0200 . . . 03FF wieder abrufen und mit etwa 18 ms Wiederholzeit auf dem angeschlossenen Oszilloskop darstellen. Das Programm sorgt auch automatisch dafür, daß zu Beginn jedes Zyklus ein Triggerimpuls zur Synchronisation des Oszilloskops mit vollem Spannungshub ( $U_{ss} = 5,12$  V) erzeugt wird.

Aus dem Darstellungs-Modus kann durch Drücken von ST wieder herausgesprungen werden. Diese Taste ermöglicht auch ein vorzeitiges Abbrechen einer Aufzeichnung. Während das Display leuchtet, kann schließlich mit der Taste C der gesamte Speicherinhalt (Adressenbereich 0200 . . . 03FF) gelöscht werden.

Die Auflösung dieses „Speicheroszilloskops“ beträgt 256 Amplitudenschritte (vertikal) und 512 Zeitschritte (horizontal). Für die beiden Operationsverstärker empfiehlt sich

eine Betriebsspannung von etwa  $\pm 12$  V, die nicht stabilisiert zu sein braucht.

### 3.4 Baudot-Ausgabeprogramm

Die beiden am weitesten verbreiteten Codes zur seriellen Datenübertragung sind der Baudot- und ASCII-Code. Unglücklicherweise arbeiten praktisch alle Mikrocomputer im ASCII- und alle Fernschreiber im Baudot-Code; doch stellt dies keinen Grund dar, ein Programm nicht auch einmal von einem preiswert erhältlichen, gebrauchten Fernschreiber ausdrucken zu lassen.

#### *Die Codes*

Der Baudot-Code besteht aus fünf zu übertragenden Datenbits, die das jeweilige Zeichen in serieller Form als Nullen und Einsen enthalten. Das niederwertigste Datenbit wird zuerst, das höchstwertige zuletzt gesendet. Vor dem ersten Datenbit wird log. 0 als Startbit ausgegeben, nach dem letzten Datenbit folgen 1,5 Stopbits mit log. 1. (1,5 bezieht sich hier auf die Länge; ein Bit dauert bei der üblichen Fernschreibgeschwindigkeit von 50 Baud genau 20 ms.) Ein besonderes Problem des Baudot-Codes ist, daß die einzelnen Zeichen in sich nicht eindeutig sind; so kann eine bestimmte 5-bit-Folge sowohl einen Buchstaben als auch eine Ziffer oder ein Zeichen darstellen – je nachdem, ob irgendwann vorher einmal ein Zeichen, „Buchstabe“ oder „Ziffer“, gesendet wurde. Der Baudot-Zeichenvorrat ist also zunächst einmal in zwei Teilbereiche getrennt, nämlich die Buchstaben von A bis Z und die Ziffern und Zeichen. Wurde z.B. als letztes Zeichen ein Buchstabe gedruckt, so muß vor der Ausgabe einer Ziffer oder eines Satzzeichens erst einmal das Zeichen „ZI“ ausgegeben werden.

Ganz anders beim ASCII-Code (American Standard Code for Information Interchange): Jedes Zeichen ist für sich eindeutig definiert; außerdem handelt es sich bei diesem Code normalerweise um ein 7-bit-Format, das  $2^7 = 128$  verschiedene

```

UNTERPROGRAMM ZUR BAUDOT-AUSGABE
0352 86 F5      STX F5      X RETTEN
0354 A2 80      LDX #80     PA 7 =
0356 8E 01 17   STX 1701   AUSGANG
0359 C9 0D      CMP #0D     C.R.
035B D0 04      BNE 0361
035D A9 08      LDA #08
035F D0 0E      BNE 036F
0361 C9 0A      CMP #0A     L.F.
0363 D0 04      BNE 0369
0365 A9 02      LDA #02
0367 D0 06      BNE 036F
0369 29 3F      AND #3F     6 BIT ASCII
036B AA         TAX         ALS INDEX
036C BD C0 03   LDA 03C0,X
036F 85 F4      STA F4
0371 29 20      AND #20     MODUS-TEST
0373 C5 F3      CMP F3      (BU./ZI.)
0375 F0 0E      BEQ 0385
0377 85 F3      STA F3      NEUER MODUS
0379 A8         TAY         NULL-TEST
037A F0 04      BEQ 0380
037C A9 1B      LDA #1B
037E D0 02      BNE 0382   MODUS-
0380 A9 1F      LDA #1F     UMSCHALTUNG
0382 20 8D 03   JSR 038D   AUSGEBEN
0385 A5 F4      LDA F4      UND ZEICHEN
0387 20 8D 03   JSR 038D   DRUCKEN
038A A6 F5      LDX F5
038C 60         RTS
038D A2 7F      LDX #7F     STARTBIT
038F 8E 00 17   STX 1700
0392 20 B4 03   JSR 03B4
0395 A0 04      LDY #04     5 BITS
0397 4A         LSR
0398 90 04      BCC 039E
039A A2 FF      LDX #FF     '1'
039C D0 02      BNE 03A0
039E A2 7F      LDX #7F     '0'
03A0 8E 00 17   STX 1700
03A3 20 B4 03   JSR 03B4
03A6 88         DEY
03A7 10 EE      BPL 0397
03A9 A2 FF      LDX #FF     STOPBIT
03AB 8E 00 17   STX 1700   (LAENGE =
03AE A2 1E      LDX #1E     1,5 BIT)
03B0 20 B6 03   JSR 03B6
03B3 60         RTS
03B4 A2 14      LDX #14     VERZOEGL. 1 BIT
03B6 8E 07 17   STX 1707
03B9 2C 07 17   BIT 1707
03BC 10 FB      BPL 03B9
03BE 60         RTS
03BF 00

```

ENDE

3.4.1 Unterprogramm zur Ausgabe eines im Akku stehenden ASCII-Zeichens auf einem Baudot-Fernschreiber



### 3.4.2 ASCII-Baudot-Umwandlungstabelle

TABELLE 6-BIT-ASCII ZU BAUDOT

03C0	00	03	19	0E	09	01	0D	1A
03C8	14	06	0B	0F	12	1C	0C	18
03D0	16	17	0A	05	10	07	1E	13
03D8	1D	15	11	2D	00	3A	00	00
03E0	24	34	00	29	00	24	31	25
03E8	2F	32	39	31	2C	23	3C	3D
03F0	36	37	33	21	2A	30	35	27
03F8	26	38	2E	2B	24	3E	24	39
0400	ENDE							

Zeichen darstellen kann, wesentlich mehr, als das beim Baudot-Code ( $2 \cdot 2^5 = 64$  Zeichen) möglich ist. So sind im ASCII-Code auch Kleinbuchstaben sowie eine ganze Reihe von Steuerzeichen möglich. Aus diesen Gründen setzte sich ASCII in der Mikrocomputertechnik praktisch hundertprozentig durch.

#### Das Programm

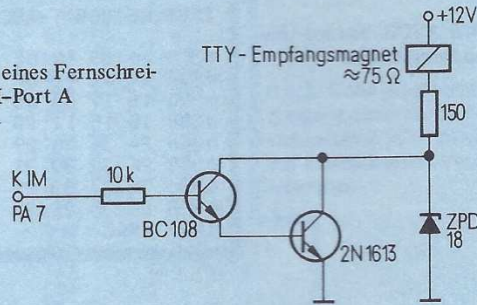
Aufgabe des in Abb. 3.4.1 dargestellten Unterprogramms ist es, ein im Akku stehendes ASCII-Zeichen auf einem Baudot-Fernschreiber auszudrucken. Um Platz für die notwendige Umwandlungstabelle (Abb. 3.4.2) zu sparen, werden nur 6 bit des 7-bit-ASCII-Zeichens als Tabellenindex verwendet; die beiden Steuerzeichen CR (Wagenrücklauf) und LF (neue Zeile) werden vom Programm getrennt decodiert, da sie bei der Einschränkung auf 6 bit verlorengehen. Der 6-bit-Maskierung fallen auch die Kleinbuchstaben zum Opfer; sie können aber auf dem Fernschreiber ohnehin nicht ausgegeben werden.

Das „Haupt-Unterprogramm“ ruft zwei weitere Unterprogramme auf, nämlich eines zur Ausgabe eines im Akku stehenden Baudot-Zeichens (Adresse 038D) und eines zur Verzögerung um eine der jeweiligen Baud-Rate entsprechende Bit-Dauer, hier für 50 Baud dimensioniert.

Es wurde darauf geachtet, daß die Wirkung dieses Programms auf die Prozessor-Register genau identisch ist mit dem im KIM-Monitor-ROM stehenden Unterprogramm „OUTCH“, d.h. das X-Register wird „gerettet“. Y ist nach der Rückkehr vom



### 3.4.3 Anschluß eines Fernschreibers an den KIM-Port A



Unterprogramm hexadezimal FF, und der Akkuinhalt ist verloren.

Es muß noch erwähnt werden, daß die Tabelle außer der 5-bit-Baudot-Information als sechstes Bit noch die Aussage enthält, ob es sich um den Buchstaben- oder um den Ziffernmodus handelt. Ändert sich der Modus gegenüber dem zuletzt ausgegebenen Zeichen, so wird ein entsprechendes Umschaltzeichen an den Fernschreiber ausgegeben. Das Programm benutzt die Zero-Page-Adressen 00F3 (Modus), 00F4 (Baudot-Zeichen) und 00F5 (X-Register).

### Notwendige Hardware

Die Timer-Adressen (1707) und die Adressen der I/O-Register beziehen sich auf den Mikrocomputer KIM-1. Das Baudot-Zeichen wird am Anschluß PA 7 ausgegeben; das Programm gestattet es aber, auch einen beliebigen anderen Port als Ausgang zu verwenden. Zum Anschluß des Baudot-Fernschreibers genügt eine Schaltung nach Abb. 3.4.3. Die Z-Diode dient dazu, Schaltspitzen zu dämpfen, die durch die Selbstinduktion des Empfangsmagneten auftreten.

Abb. 3.4.4 zeigt schließlich ein einfaches Programm, das das direkte Schreiben auf dem Drucker des Fernschreibers mit einem ASCII-Terminal gestattet; es besteht nur aus zwei Unterprogramm-Aufrufen und einem Rücksprungbefehl. Sollte sich

#### ASCII-EINGABE BAUDOT-AUSGABE

```

0000 20 5A 1E      JSR 1E5A  GETCH(KIM)
0003 20 52 03      JSR 0352  BAUDOT-AUSG.
0006 4C 00 00      JMP 0000  NOCHMAL

```

#### 3.4.4 Programm zur Bedienung des Fernschreibers mit einer ASCII-Tastatur

herausstellen, daß die Fernschreibergeschwindigkeit nicht 50 Baud beträgt, so sind die hexadezimalen Inhalte der Zellen 03AF und 03B5 zu ändern.

Ein Verlegen des Baudot-Ausgabe-Unterprogramms in eine andere „Page“, also in eine andere Speicherseite, ist sehr einfach: Man braucht nur bei allen 3-Byte-Befehlen, deren drittes Byte 03 lautet, dieses Byte auf die neue Page ändern.

### 3.5 Programm für einen Metallpapier-Drucker

Für weniger als 400 DM bekommt man heute schon einfache Matrixdrucker, die mit aluminiumbeschichtetem Papier arbeiten; ein Beispiel ist der von Matsushita hergestellte KE-100 (Vertrieb: Logitec Byte-Shop, München). Er benötigt zwei Versorgungsspannungen (- 24 V für die Druckköpfe und + 5 V für das IC) und arbeitet im 6-bit-ASCII-Code; Kleinbuchstaben können daher nicht dargestellt werden, aber für die Anfertigung von Programmauflistungen oder ähnliche Aufgaben spielt das ja auch keine Rolle.

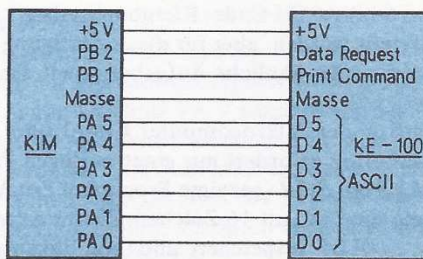
Der Anschluß an den Mikrocomputer KIM-1 oder ein anderes System auf 6502-Basis erfordert nur einen geringen Programmaufwand. Das in *Abb. 3.5.1* gezeigte Programm druckt genau eine Zeile, in diesem Fall mit 16 Zeichen, die im Adressenbereich 01D0 . . . 01DF gespeichert sind (von links nach rechts). Der notwendige Hardware-Aufwand ist minimal; im vorliegenden Fall wurden sechs Leitungen des I/O-Ports A für das ASCII-Zeichen und zwei Anschlüsse des Ports B für den Druckbefehl und die Zeichenanforderung verwendet *Abb. 3.5.2*.

```

6502-DRUCKERPROGRAMM
0000 A9 FF  LDA #$FF
0002 8D 01 17  STA $1701
0005 A9 02  LDA #$02
0007 8D 03 17  STA $1703
000A A2 00  LDX #$00
000C 8E 02 17  STX $1702
000F 8D 02 17  STA $1702
0012 F0 0F  BEQ 0023
0014 E8  INX
0015 E0 10  CPX #$10
0017 D0 03  BNE 001C
0019 4C 4F 1C  JWP $1C4F
001C AD 02 17  LDA $1702
001F 29 04  AND #$04
0021 F0 F9  BEQ 001C
0023 B0 D0 01  LDA $01D0,X
0026 8D 00 17  STA $1700
0029 AD 02 17  LDA $1702
002C 29 04  AND #$04
002E D0 F9  BNE 0029
0030 F0 E2  BEQ 0014

```

### 3.5.1 Programm zur Ausgabe einer Zeile auf einem Metallpapier-Drucker



### 3.5.2 Anschluß des Druckers KE-100 an den KIM-1

Das angegebene Programm ist beliebig verschiebbar, die angegebenen Adressen sind daher nur als relativ zu betrachten.

Da das vorliegende Programm die am Drucker ebenfalls vorhandene „Busy“-Leitung nicht abfragt, muß zwischen dem Ausdruck zweier Zeilen genügend Zeit „verschwendet“ werden, um dem Motor des Druckers das Zurücklaufen zu gestatten. Diese Zeit ergibt sich allerdings automatisch, wenn man z.B. gleichzeitig das Auszudruckende auch noch auf einem Terminal mit serieller Schnittstelle und Geschwindigkeiten bis zu 1200 Bd ausgibt; andernfalls ist eine Verzögerungsschleife z.B. mit dem im KIM vorhandenen Timer vorzusehen.

### 3.6 Serielle ASCII-Eingabe per Interrupt

Mikrocomputer sind leider die meiste Zeit ihres Daseins damit beschäftigt, darauf zu warten, bis irgendein verwertungswürdiges Ereignis eintrifft, z.B. bis der Benutzer ein paar Buchstaben auf seiner Terminal-Tastatur eintippt. Daß man diese Eingabe von ASCII-Zeichen nicht zur Hauptbeschäftigung eines Mikroprozessors machen muß, zeigt das folgende Beispiel.

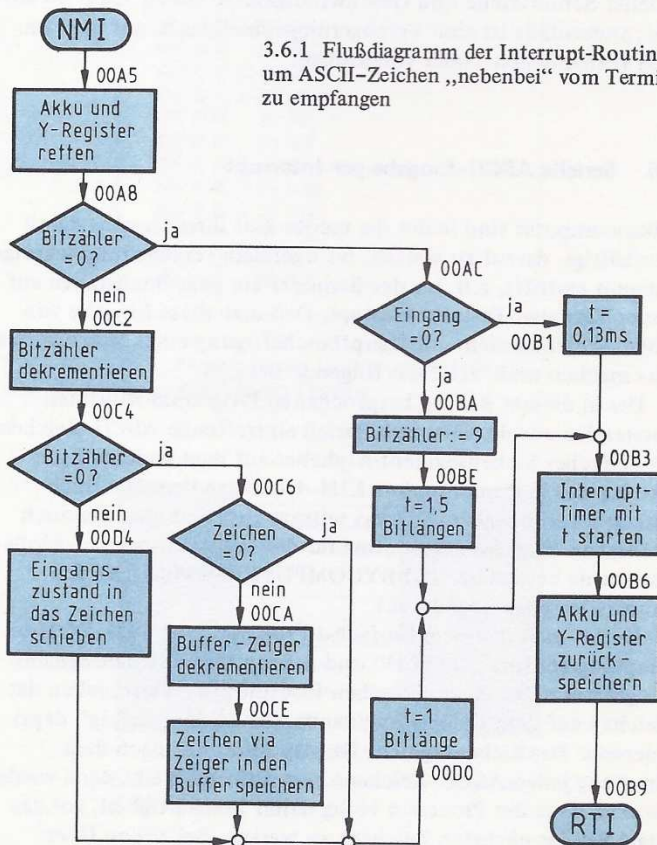
Die in diesem Beitrag besprochenen Programm-Routinen entstanden aus dem Wunsch, seriell eintreffende ASCII-Zeichen als einfaches Siebensegment-Alphabet auf dem sechsstelligen Display des Mikrocomputers KIM-1 darzustellen. Die Buchstaben sehen dabei zwar etwas seltsam aus, sind aber dennoch gut lesbar. Eine Anzeigeroutine für dieses Siebensegment-Alphabet wurde bereits im HOBBYCOMPUTER-Sonderheft des Franzis-Verlages abgedruckt.

Kettet man in einem Einfachst-Programm die KIM-ROM-Empfangsroutine „GETCH“ und das erwähnte Siebensegment-Programm nebst einer einfachen Routine zum Verschieben der Zeichen auf dem Display aneinander, so ist der „Erfolg“ deprimierend: Das Siebensegment-Display blitzt nur nach dem Empfang jeden ASCII-Zeichens kurz auf und bleibt dann wieder dunkel, da ja der Prozessor völlig damit beschäftigt ist, auf das Start-Bit des nächsten Zeichens zu warten. Bei hohen Über-



tragungsgeschwindigkeiten, wie etwa 600 Baud (bit/s), ist auf dem Display zwar schon etwas zu erkennen, aber erstens laufen die Buchstaben dann so schnell durch, daß man nicht mehr mitlesen kann, und zweitens treten auch noch Übertragungsfehler auf, weil die Display-Routine so lange dauert, daß bis zu ihrem Ende der Anfang des nächsten Zeichens bereits verpaßt wurde. So funktioniert es also nicht!

3.6.1 Flußdiagramm der Interrupt-Routine, um ASCII-Zeichen „nebenbei“ vom Terminal zu empfangen



### *Die Lösung: Der Timer-Interrupt*

Zum Glück verfügen viele Mikrocomputersysteme, auch der KIM-1, über einen taktgesteuerten Timer, der den Prozessor veranlassen kann, nach Ablauf einer bestimmten, programmierbaren Zeit eine Interrupt-Routine abzuarbeiten und danach in das Hauptprogramm zurückzukehren, als wäre nichts geschehen. Beim KIM-1 ist dazu die Leitung PB7 (sie ist der Interrupt-Ausgang des Timers im IC 6530-003) mit der NMI-Leitung zu verbinden.

*Abb. 3.6.1* zeigt das Flußdiagramm für eine Interrupt-Routine, deren Aufgabe es ist, ein ASCII-Zeichen unabhängig von einem beliebigen parallellaufenden Hauptprogramm zu empfangen und in einen Buffer abzuspeichern. Dieser Buffer wird indirekt adressiert, nämlich durch die Zellen 00EB und 00EC, wobei 00EC den höherwertigen Adreßteil enthält und 00EB den niederwertigen. Ist ein Zeichen vollständig empfangen und ist sein Wert nicht 00, so wird der niederwertige Adreßteil um 1 erniedrigt und das Zeichen an die sich dadurch ergebende neue Buffer-Adresse abgespeichert.

Diese Methode ermöglicht es, bis zu 255 Zeichen vor einer weiteren Verarbeitung durch das Hauptprogramm zwischenspeichern; denn schließlich werden die Zeichen u.U. langsamer ausgewertet, als sie empfangen werden, wie dies auch in unserem Anwendungsbeispiel der Fall ist.

*Abb. 3.6.2* zeigt den disassemblierten 6502-Maschinencode für die Interrupt-Routine. Es muß noch erwähnt werden, daß sie den Interrupt-Timer selbst wieder startet; zu Beginn des Hauptprogramms muß jedoch „künstlich“ ein NMI-Interrupt erzeugt werden, um die Interrupt-Routine das erste Mal anzuspringen. Das Programm in *Abb. 3.6.3* tut dies; und es tut noch einiges mehr. Da, wie schon erwähnt, bei hohen Übertragungsgeschwindigkeiten die auf dem Display von rechts nach links durchlaufenden Zeichen nicht mehr mitgelesen werden könnten, wird der Zeichenbuffer recht langsam ausgelesen, nämlich mit einer Geschwindigkeit von nur etwa 3 Zeichen pro Sekunde (die genaue Zeit läßt sich in der Zelle 005A in weiten Grenzen wählen). Der „Vorlauf“ der Interrupt-Eingabe-Routine kann

```

00A5 48      PHA
00A6 98      TYA
00A7 48      PHA
00A8 A5 E8   LDA E8
00AA D0 16   BNE 00C2
00AC 2C 40 17 BIT 1740
00AF 10 09   BPL 00BA
00B1 A9 02   LDA #02
00B3 8D 0E 17 STA 170E
00B6 68      PLA
00B7 A8      TAY
00B8 68      PLA
00B9 40      RTI
00BA A9 09   LDA #09
00BC 85 E8   STA E8
00BE A9 4D   LDA #4D
00C0 D0 F1   BNE 00B3
00C2 C6 E8   DEC E8
00C4 D0 0E   BNE 00D4
00C6 A5 EA   LDA EA
00C8 F0 06   BEQ 00D0
00CA C6 EB   DEC EB
00CC A0 00   LDY #00
00CE 91 EB   STA (EB),Y
00D0 A9 34   LDA #34
00D2 D0 DF   BNE 00B3
00D4 18      CLC
00D5 2C 40 17 BIT 1740
00D8 10 01   BPL 00DB
00DA 38      SEC
00DB 66 EA   ROR EA
00DD 18      CLC
00DE 90 F0   BCC 00D0

```

3.6.2 Die fertige Interrupt-Routine.  
Sie ist hier für 300 Baud dimensioniert

Unten:

3.6.3 Dieses KIM-Programm stellt die per Interrupt empfangenen ASCII-Zeichen als Siebensegment-Alphabet auf dem Display dar und gestattet auch das Senden von vorprogrammierten ASCII-Zeichenfolgen

```

0000 A9 00 8D FB 17 A9 A5 8D FA 17 A2 08 A9 03 95 E6
0010 CA 10 FB 8E 0E 17 A9 7F 8D 41 17 A2 09 A0 06 84
0020 FC 98 18 65 ED 85 E6 A0 00 B1 E6 A8 C0 30 90 04
0030 C0 5B 90 02 A0 2F B9 4A 00 A0 00 8C 40 17 8E 42
0040 17 8D 40 17 A0 7F 88 D0 FD E8 E8 A4 FC 88 D0 CF
0050 20 3D 1F D0 16 C6 E9 D0 BD A9 30 85 E9 A5 ED 38
0060 E5 EB C9 FF F0 B0 C6 ED 18 90 AB 20 6A 1F C9 15
0070 10 A4 4C 00 02 00 00 00 00 00 3F 06 5B 4F 66 6D
0080 7D 07 7F 6F 00 00 00 41 00 53 00 77 7C 58 5E 79
0090 71 3D 74 05 1E 78 38 37 54 5C 73 67 50 6D 31 3E
00A0 1C 7E 76 6E 5B

0200 A2 00 8E F3 17 8E 04 17 A2 EB 8E F2 17 09 80 A2
0210 29 DD 00 02 F0 03 E8 D0 F8 E8 BD 00 02 10 03 4C
0220 00 00 20 A0 1E D0 F2 F0 F6

```

bis zu 255 Zeichen betragen. Der Zeichenbuffer wird dabei im Adressenbereich 0300 . . . 03FF aufgebaut. Da große Teile des Hauptprogramms dem schon erwähnten Programm zur Sieben-segment-Anzeige von Buchstaben sehr ähneln, wurde es lediglich in hexadezimaler Form abgedruckt. (Ausgewertet werden alle ASCII-Zeichen mit Hex-Codes zwischen 30 und 5A.)

Neben seinem ursprünglichen Verwendungszweck, nämlich für ASCII-Amateurfunkfern-schreiben, läßt sich das Programm auch recht gut für den Test von Terminals und UART-Schal-tungen einsetzen. Dabei lassen sich praktisch alle üblichen Übertragungsgeschwindigkeiten einstellen.

Jetzt fehlt uns für diesen Verwendungszweck also nur noch eine einfache Routine, die vorher programmierte Texte und Zeichenfolgen seriell ausgeben kann.

## Wichtige Adressen im Programm

### *Zero-Page-Adressen*

E6	Display-Zeiger L
E7	Display-Zeiger H
E8	Bit-Zähler
E9	Display-Verzögerung
EA	Empfangenes Zeichen
EB	NMI-Zeiger L
EC	NMI-Zeiger H
ED	Displ.-Zeiger-Index
FC	Y-Zwischenspeicher
(Werden vom Programm gesetzt)	

### *Geschwindigkeits-Einstellung*

Baud ►	110	300	600
00BF	D4	4D	26
00D1	8E	34	1A
0201	02	00	00
0209	7D	EB	73



## Hex-Code der ASCII-Zeichen

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	CS	RS	VS
2	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	,	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	}	~	DEL	

Die waagrechtten Ziffern stellen das niederwertige, die senkrechten das höherwertige Halbbyte dar; das Zeichen „e“ ist z.B. hex 65. Die wichtigsten Steuerzeichen sind:

CR = Wagenrücklauf (Carriage Return), LF = neue Zeile (Line Feed), BS = Cursor rückwärts (Back Space), SP = Leerraum (Space).

### *Das Sendeprogramm*

Um auch fest programmierte Zeichenfolgen per Tastendruck über den seriellen ASCII-Anschluß des KIM senden zu können, ist im Adressenbereich 0200 . . . 0228 ein kleines Programm untergebracht, das mit Hilfe des KIM-ROM-Unterprogramms „OUTCH“ Standardtexte ausgeben kann, die zwischen 0229 und 02FF stehen. Diese Standardtexte haben das Format *ID Text ID Text ID Text . . . FF*.

ID ist dabei eine Hexzahl zwischen 80 und FE. Sie errechnet sich aus der Summe von 80 und dem Hex-Code derjenigen KIM-Taste, der der jeweilige Text zugeordnet werden soll. Das Byte FF sagt dem Computer, daß die Texttabelle hier zu Ende ist. Der Text selbst ist mit der ASCII-Tabelle zu programmieren.

Das gewählte Format hat den Vorzug, daß zum Aufrufen des gewünschten Textes nicht die absolute Adresse dieses Textes bekannt sein muß. Das Auffinden geschieht vielmehr über eine Marke ID (Label). Eventuelle Änderungen der Texttabelle werden damit besonders einfach.

## **3.7 Binär-Dezimal-Umwandlung**

Für die Umwandlung einer Zahl in ein anderes Zahlensystem, hier vom Binärsystem in das Dezimalsystem, gibt es prinzipiell zwei Lösungsmöglichkeiten: Entweder verwendet man die Zählmethode, wobei die in einer bestimmten Speicherzelle stehende Zahl binär abwärts bis auf Null und eine andere Speicherzelle von Null dezimal aufwärts gezählt wird. Letztere Zelle enthält dann das dezimale Ergebnis. Die zweite Möglichkeit ist die Verwendung eines geeigneten Algorithmus; dazu ist zwar erheblich weniger Rechenzeit, aber meist ein deutlich längeres Programm erforderlich.

Das in Abb. 3.7 aufgelistete Programm verwendet zwar auch einen bestimmten Umwandlungs-Algorithmus, bedient sich

### BINAER-DEZIMAL-UMWANDLUNG MIT DEM 6502

```

0000 F8      SED
0001 A0 08    LDY #08
0003 A9 00    LDA #00
0005 85 F4    STA F4
0007 85 F3    STA F3
0009 06 F5    ASL F5
000B A5 F3    LDA F3
000D 65 F3    ADC F3
000F 85 F3    STA F3
0011 26 F4    ROL F4
0013 88      DEY
0014 D0 F3    BNE 0009
0016 D8      CLD
0017 60      RTS

```

```

ARGUMENT: 00F5
ERGEBNIS: 00F4,00F3

```

### 3.7 Umwandlung eines binären Bytes in eine Dezimalzahl

```

0100 A9 AD 8D EC 17 20 32 19
0108 A9 27 85 F5 A9 BF 8D 43
0110 17 A2 64 A9 16 20 61 01
0118 A9 2A 20 88 01 AD F9 17
0120 20 70 01 AD F5 17 20 6D
0128 01 AD F6 17 20 6D 01 20
0130 EC 17 20 6D 01 20 EA 19
0138 AD ED 17 CD F7 17 AD EE
0140 17 ED F8 17 90 E9 A9 2F
0148 20 88 01 AD E7 17 20 70
0150 01 AD E8 17 20 70 01 A2
0158 02 A9 04 20 61 01 4C 5C
0160 18 86 F1 48 20 88 01 68
0168 C6 F1 D0 F7 60 20 4C 19
0170 48 4A 4A 4A 4A 20 7D 01
0178 68 20 7D 01 60 29 0F C9
0180 0A 18 30 02 69 07 69 30
0188 A0 07 84 F2 A0 02 84 F3
0190 BE BE 01 48 2C 47 17 10
0198 FB B9 BF 01 8D 44 17 A5
01A0 F5 49 80 8D 42 17 85 F5
01A8 CA D0 E9 68 C6 F3 F0 05
01B0 30 07 4A 90 DB A0 00 F0
01B8 D7 C6 F2 10 CF 60 02 C3
01C0 03 7E

```

### 3.8 Das Hypertape-Programm von Jim Butterfield

aber einer recht schnellen Methode zur Konversion. Dabei werden die Bitwertigkeiten dezimal Schritt für Schritt addiert, was das dezimale Byte-Äquivalent liefert.

Eine in der Zelle 00F5 stehende Binärzahl (00 . . . FF) wird vom Programm in eine Dezimalzahl umgewandelt; das niederwertige Byte des Ergebnisses (LSB), erscheint an der Adresse 00F3, das höherwertige Byte (es kann 0, 1 oder 2 sein) an der Adresse 00F4 (MSB). Die Umwandlung ist etwas von der jeweiligen Zahl abhängig, dauert aber durchschnittlich nur etwa 150  $\mu$ s, bezogen auf einen Mikroprozessor 6502 mit 1 MHz Taktfrequenz. Nach dem Durchlaufen des Programms sind Akku, X-Register und der Inhalt der Zelle 00F5 verloren.

Zusammen mit dem unter 3.3 beschriebenen Programm zur Analog/Digital-Umsetzung (KIM als Speicheroszilloskop) läßt sich aus dem Mikrocomputer KIM-1 ein kleines Digitalvoltmeter machen - aber das ist natürlich nicht die einzige Anwendungsmöglichkeit des hier beschriebenen Programms. Seine Struktur eignet sich auch für andere Zahlensysteme.

### 3.8 Hypertape

Hypertape, früher auch Supertape genannt, ist ein Verfahren zur Bandaufzeichnung mit dem KIM-1, das rund sechsmal schneller funktioniert als das „normale“ KIM-Format. Es wurde 1976 von Jim Butterfield entwickelt.

Der besondere Vorteil von Hypertape ist, daß zum Wiedereinlesen des Bandes keinerlei Kunstgriffe notwendig sind; Hypertape ist also völlig kompatibel mit der KIM-Monitor-Routine (Adresse 1873). Lediglich für den Aufzeichnungsvorgang selbst wird das in Page 1 des KIM-Speichers stehende Hypertape-Programm benötigt. Anfangs- und Endadresse sowie Band-ID werden analog zur KIM-Routine (Adresse 1800) gesetzt, und die Startadresse ist 0100 (*Abb. 3.8*).

Es ist ein gewisser Nachteil von Hypertape, daß u.U. Probleme mit schlechten Kassetten oder Recordern auftreten. Trotzdem führt in den meisten Fällen das Justieren der Tonkopf-Taumel-



schraube beim Einlesen von Fremdkassetten zum Erfolg. Die Einstellung der Lautstärke am Recorder ist u. U. etwas kritischer als beim KIM-Normalformat.

Die Aufzeichnungsgeschwindigkeit beträgt bei Hypertape rund 800 bit/s, beim KIM-Format dagegen nur etwa 135 bit/s. Da jedes Byte aber erst in zwei Halbbytes zerlegt werden muß und als Folge von zwei ASCII-Zeichen auf Band gebracht wird, dauert auch in Hypertape die Übertragung von 1 KByte immerhin noch rund 22 Sekunden.

Obwohl Hypertape nicht ganz so unkritisch funktioniert wie das KIM-Format, wurde es inzwischen doch zu dem beliebtesten Aufzeichnungsformat für die KIM-Besitzer. Wie sich das Prinzip für die schnelle Textaufzeichnung einsetzen läßt, wird weiter unten noch beschrieben; die Geschwindigkeit läßt sich dabei nochmals verdoppeln.

Das Hypertape-Format wird - im Gegensatz vom „normalen“ KIM-Format - vom AIM-65 nicht richtig verarbeitet. Es kann daher für den Austausch von Programmen nicht empfohlen werden.

### 3.9 Interrupt-Uhr

Mikrocomputer werden oft zur automatischen Meßwertverarbeitung herangezogen, und außer der Messung analoger Eingangsgrößen spielt die Zeit dabei eine wichtige Rolle. Versteht man unter „Zeit“ die tatsächliche Uhrzeit, so spricht man auch von „Echtzeit“. Sinnvollerweise beauftragt man den Mikrocomputer selbst damit, die aktuelle Uhrzeit bereitzustellen.

Zur Realisation einer reinen Software-Uhr gibt es mehrere Möglichkeiten. Einmal könnte man eine Verzögerungsschleife so programmieren, daß sie einige Speicherzellen für Stunden, Minuten usw. fortlaufend weiterstellt, so daß diese die Uhrzeit enthalten. Der Nachteil eines solchen Verfahrens ist so groß, daß es in der Praxis kaum angewendet wird: Der Mikrocomputer ist vollständig damit beschäftigt, die Uhr weiterlaufen zu lassen und hat z.B. für die Verarbeitung parallel laufender Meßvorgänge keine Zeit mehr.

```

03A0 A9 03 8D FB 17 A9 D8 8D
03A8 FA 17 8D 0E 17 A5 DC 85
03B0 FB A5 DD 85 FA A5 DE 85
03B8 F9 20 1F 1F 20 6A 1F C9
03C0 00 D0 EA 4C 25 19

```

```

03D3 FF TAGE
03D4 24 STUNDEN
03D5 60 MINUTEN
03D6 60 SEKUNDEN
03D7 04 1/4 SEK.

```

```

03D8 48 PHA
03D9 8A TXA
03DA 48 PHA
03DB A2 10 LDX #10
03DD CA DEX
03DE D0 FD BNE 03DD
03E0 A9 F4 LDA #F4
03E2 8D 0F 17 STA 170F
03E5 F8 SED
03E6 A2 05 LDX #05
03E8 CA DEX
03E9 30 10 BMI 03FB
03EB B5 DB LDA DB,X
03ED 18 CLC
03EE 69 01 ADC #01
03F0 DD D3 03 CMP 03D3,X
03F3 90 02 BCC 03F7
03F5 A9 00 LDA #00
03F7 95 DB STA DB,X
03F9 F0 ED BEQ 03E8
03FB D8 CLD
03FC 68 PLA
03FD AA TAX
03FE 68 PLA
03FF 40 RTI

```

```

00DB 04 TAG
00DC 19 STUNDE
00DD 01 MINUTE
00DE 35 SEKUNDE
00DF 02 1/4 SEK.

```

3.9 Anzeige der Uhrzeit auf dem KIM-Display (Kaltstart: 03A0; Warmstart: 03AD; Taste 0: Sprung zum Monitorprogramm)

Die Alternative ist die Ausnutzung des Prozessor-Interrupts. Es ist z.B. möglich, jede volle Sekunde einen Interrupt auszulösen; der Prozessor springt aus dem parallel laufenden Hauptprogramm in eine Interrupt-Routine, die die Uhr um eine Sekunde weiterstellt und kehrt dann wieder in das Hauptprogramm zurück. Da der relative Zeitbedarf für die Interrupt-Routine minimal ist, kann das Hauptprogramm ungestört z.B. Meßwerte verarbeiten oder mit einem Terminal kommunizieren; es kann aber auch die Uhrzeit auf einem Siebensegment-Display darstellen. Genau dies tut das in *Abb. 3.9* aufgelistete 6502-Programm.

Das (KIM-spezifische) Hauptprogramm (03A0 . . . 03C5) dient lediglich dazu, den Inhalt der Zellen 00DC, 00DD und 00DE auf dem sechsstelligen KIM-Display als Stunden, Minuten und Sekunden darzustellen. Der Tag-Zähler (00DB) wird hier nicht angezeigt, ebenso nicht der Viertel-Sekunden-Zähler (00DF).

Warum eigentlich ein Viertel-Sekunden-Zähler? Nun, der Mikrocomputer KIM-1 besitzt einen wunderschönen 8-bit-Timer, mit dem man nach einer vorprogrammierten Zeit einen Interrupt auslösen kann (vgl. „Gebrauchsanleitung für den KIM-Timer“). Er läßt sich maximal aber nur mit einer Zeit von  $255 \times 1024 \mu\text{s}$  voreinstellen, so daß der Interrupt nicht jede Sekunde, sondern bereits nach einer Viertelsekunde stattfinden muß. Deswegen fungiert eine Zelle im Zero-Page-Bereich (00DF) als Viertelsekunden-Zähler; ihr Inhalt wird normalerweise nicht weiterverwendet.

Ein ähnliches Programm („Clock“) wurde bereits im „First Book of KIM“ von Charles Parsons veröffentlicht; die Interrupt-Routine umfaßt dabei hexadezimal 5B Bytes. Wendet man dagegen die hervorragenden Möglichkeiten der indizierten Adressierung an, so läßt sich diese Routine auf (hex) 28 Bytes verkürzen, also um mehr als die Hälfte. Das Weiterzählen der aktuellen Zeit geschieht nicht für jede Zelle (00DB . . . 00DF) einzeln, sondern in einer Schleife, bei der das X-Register dazu dient, die Zero-Page-Adressen zu inkrementieren und mit den im Bereich 03D3 . . . 03D7 abgelegten Grenzwerten zu ver-

gleichen. So werden etwa die Tage nur dann weitergezählt, wenn bei den Stunden der Grenzwert 24 erreicht ist; der Stunden-zähler wird dann gleichzeitig auf Null rückgesetzt.

Der Grenzwert für die Tage ist im Beispielprogramm FF, was dazu führt, daß - weil diese Zahl im Dezimal-Modus des 6502 nie erreicht wird - bis 99 Tage gezählt werden kann. Selbstverständlich lassen sich hier auch Grenzwerte wie 07 (Wochentag) programmieren.

Es sei erwähnt, daß hier natürlich eine Verbindung zwischen PB 7 und NMI beim KIM-1 notwendig ist, damit der Timer den Interrupt auslösen kann. Er wird dann in der Interrupt-Routine selbst wieder neu gestartet. Da der 1-MHz-Quarz etwas neben der Frequenz liegen kann, ist mit dem Inhalt der Adresse 03DC eine Feinjustierung der Uhr möglich. Für die Verwendung im AIM-65 muß das Programm der veränderten Timer-Struktur des VIA 6522 gegenüber dem KIM-1 angepaßt werden.



## 4 Programme für ASCII-Terminals

### 4.1 Debugger

Mit wenig Aufwand ist es möglich, einen recht komfortablen Einzelschritt-Betrieb oder Breakpoint-Routinen auf dem KIM-1 zu realisieren. Auf einem angeschlossenen Terminal oder Fernschreiber wird dabei der Inhalt aller internen CPU-Register ausgedruckt, und zwar in folgendem Format:

X	Y	A	SP	NV	B	D	I	Z	C	
01	02	03	FF	0	1	1	0	0	1	0

KIM

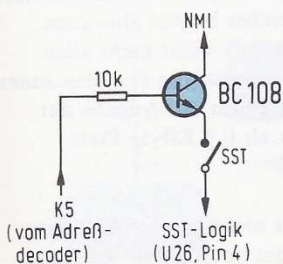
0204 A9

Die angegebenen Zahlen mögen hier nur als Beispiel dienen; es ist aber zu erkennen, daß das Statusregister nicht in Form zweier hexadezimaler Ziffern, sondern wirklich binär ausgegeben wird, so daß sofort der Zustand jedes einzelnen Bit zu sehen ist. Die Titelzeile (X, Y, A . . .) wird ebenfalls ausgedruckt.

Das für diesen Ausdruck erforderliche kleine Programm findet leicht in dem etwas abseits liegenden RAM-Bereich des KIM-1 (Adressen ab 1780) Platz, so daß der normalerweise für Anwenderprogramme benutzte Raum (0000 . . . 03FF) dadurch nicht eingeschränkt wird.

Nun muß allerdings dafür gesorgt werden, daß das in *Abb. 4.1.1* dargestellte Programm nicht selbst auch im Einzelschritt-Betrieb durchlaufen wird, denn dies soll ja nur im Anwenderprogramm geschehen. Wenn also der für den Einzelschritt maßgebende NMI-Vektor (Adressen 17FA und 17FB) auf die Programm-Startadresse 1780 zeigt, so soll das Register-Ausdrucken nicht im Single-Step-Modus erfolgen. Dazu ist nur eine winzige Hardware-Änderung erforderlich (*Abb. 4.1.2*). Ein in

#### 4.1.1 Debugging-Programm als Interrupt-Routine



#### 4.1.2 Ändern der Single-Step-Schaltung für das Debugging-Programm

```

START
(1780)
85 f3      Akku retten
68
85 f1      Statusreg.
68          retten
85 fa      Programm-
68          zähler
85 fb      retten
84 f4      Y-Index u.
86 f5      X-Index
ba
86 f2      Stackpointer
20 88 1e   I/O initial.
20 2f 1e   Neue Zeile
a2 14
bd cc 17   Überschrift
20 ao 1e   ausgeben
ca
10 f7
20 2f 1e   Neue Zeile
a2 04
b5 f1      X,Y,A,SP
20 3b 1e   ausgeben
20 9e 1e
ca
do f5
a2 08      Statusreg.
a5 f1      binär aus-
85 f9      geben
o6 f9
bo 04
a9 30
90 02
a9 31
20 ao 1e
ca
do fo
4c 4f 1c   Spr.z.KIM
43 5a 49   ASCII-
44 42 2d   Tabelle
56 4e 20   für die
50 53 20   Überschrift
20 41 20
20 59 20
20 58
END (17eo)

```

Serie mit dem Single-Step-Schalter am KIM-1 liegender Transistor verhindert diese Betriebsart, solange der Adressenbereich des RAM ab 1780 angesprochen wird. Das dazu erforderliche Signal wird vom Adressen-Decoder auf der KIM-Platine gewonnen.

Bei längeren Anwenderprogrammen ist der Einzelschritt-Modus etwas zeitaufwendig. Es ist dann günstiger, sogenannte „Breakpoints“ zu setzen, die beim 6502 dem Hex-Code 00 entsprechen und an die Adresse geschrieben werden, so das Programm anhalten soll. Schreibt man in die Zellen 17EE und 17FF als IRQ-Vektor die Daten 80 und 17, so werden auch beim Erreichen eines solchen Breakpoints die CPU-Register ausgedruckt. Zu beachten ist jedoch, daß die CPU den Break-Befehl als Zwei-Byte-Befehl interpretiert und der Programmzähler (unter dem Ausdruck „KIM“) nicht die nächste, sondern die übernächste Adresse anzeigt.

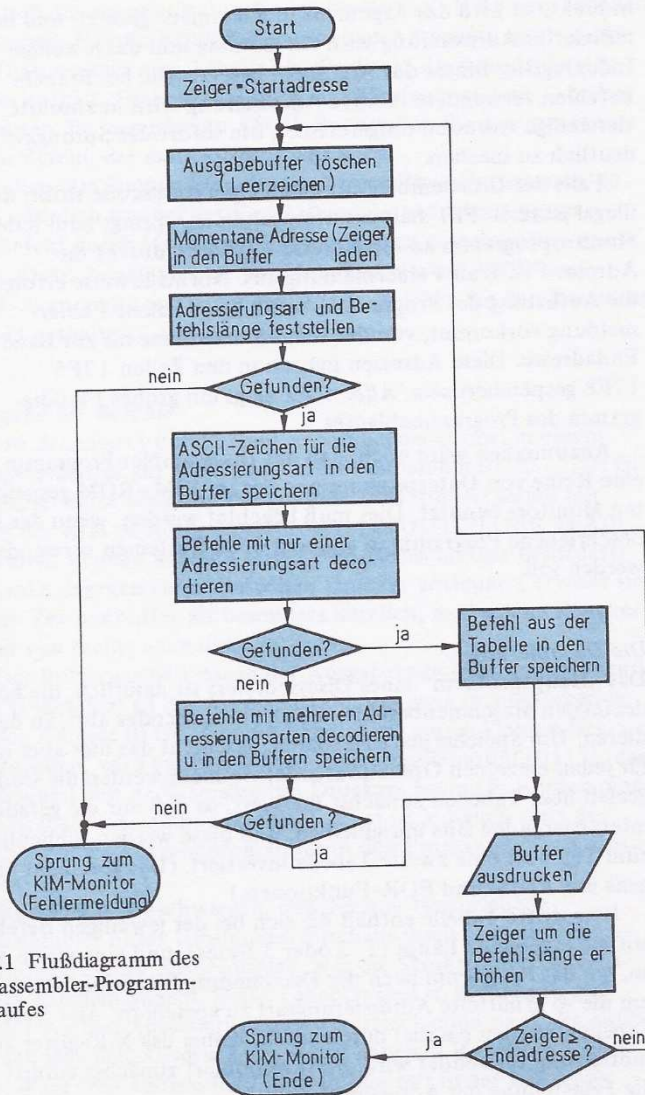
## 4.2 Disassembler

Wer einen KIM-1 und einen ASCII-Fernschreiber, einen Drucker oder ein Terminal besitzt, kann das hier vorgestellte Disassembler-Programm auch ohne zusätzlichen Speicher bereits einsetzen, wenn auch der verbleibende Programmplatz nicht mehr allzu groß ist. Der Disassembler erzeugt eine Auflistung von Maschinensprache-Programmen in mnemonischer Form mit Angabe der Adressierungsart und findet in weniger als 0,8 KByte Platz.

### *Wirkungsweise*

Beim Durchprüfen von Programmen ist nicht nur für Anfänger, die mit hexadezimalen Operations-Codes noch nichts anfangen können, ein Disassembler sehr hilfreich. Die mnemonischen Befehlsbezeichnungen, die er für jeden Operationscode liefert, sind normalerweise sofort verständlich, z.B. bedeutet LDA soviel wie „Load Accu“.

Eine weitere Hilfestellung bietet der hier beschriebene Disassembler bei der Angabe der Adressierungsart. Ist sie z.B.



4.2.1 Flußdiagramm des Disassembler-Programmablaufes



indirekt, so wird das Argument in Klammern gesetzt, und bei indizierter Adressierung wird ein Komma und das jeweilige Indexregister hinter das Argument gesetzt. Die bei Branch-Befehlen verwendete relative Adressierung wird in absolute, vierstellige Adressen umgerechnet, um sofort das Sprungziel deutlich zu machen.

Falls der Disassembler auf einen Operationscode stößt, der illegal ist (z.B. FF), hält er automatisch an, springt zum KIM-Monitorprogramm an die Adresse 1929 und druckt die Adresse FFFF als Fehlermeldung aus. Normalerweise erfolgt die Auflistung des Programms, wenn keine solche Fehlermeldung vorkommt, von der Band-Startadresse bis zur Band-Endadresse. Diese Adressen müssen in den Zellen 17F5 . . . 17F8 gespeichert sein. *Abb. 4.2.1* zeigt ein grobes Flußdiagramm des Programmablaufes.

Anzumerken wäre noch, daß das Disassembler-Programm eine Reihe von Unterprogrammen des im KIM-ROM gespeicherten Monitors benutzt. Dies muß beachtet werden, wenn das hier beschriebene Programm in anderen 6502-Systemen verwendet werden soll.

### *Die Decodierung*

Das „Hauptproblem“ eines Disassemblers ist natürlich, die hexadezimalen Maschinenbefehle, die Operationscodes also, zu decodieren. Um Speicherplatz zu sparen, geschieht das hier aber nicht für jeden einzelnen Operationscode; vielmehr werden die Codes gezielt über Tabellen zunächst maskiert, so daß nur die gerade interessierenden Bits übrigbleiben, und diese werden schließlich zum Teil über eine zweite Tabelle invertiert. (Das geschieht übrigens mit AND- und EOR-Funktionen.)

Eine dritte Tabelle enthält die sich bei der jeweiligen Befehlsgruppe ergebende Länge (1, 2 oder 3 Bytes), und eine vierte gibt an, wo das Programm nach der Decodierung hinspringen soll, um die so ermittelte Adressierungsart zu speichern. Alle diese vier Tabellen werden parallel durchlaufen, wobei das X-Register zur Indizierung verwendet wird. So funktioniert zunächst einmal die Feststellung der Adressierungsart.

Auch die mnemonischen Codes werden mit Hilfe von Tabellen gewonnen. Für die Befehle mit nur einer Adressierungsart sucht der Disassembler eine Tabelle ab, die direkt die hexadezimalen Codes enthält. Drei parallel dazu durchlaufene weitere Tabellen enthalten die zugehörigen ASCII-Buchstaben für den mnemonischen Befehl, der dann ausgedruckt wird.

Führte das Suchen hier aber zu keiner Übereinstimmung, so wird - ähnlich wie bei der Feststellung der Adressierungsart - der Befehl durch Maskieren und Invertieren einzelner Bits decodiert. Dazu dienen vier weitere Tabellen, die die AND- und EOR-Argumente sowie die drei ASCII-Zeichen für den jeweiligen Befehl enthalten.

#### *Ausgabe der Befehle*

Der so decodierte Befehl steht nun zunächst einmal in einem Ausgabe-Buffer zur Verfügung (Adressenbereich 01D0 . . . 01DF). Normalerweise wird nun der gesamte Bufferinhalt seriell mit Hilfe des KIM-Monitor-Unterprogramms OUTCH (JSR 1EA0) und einer kleinen Schleife seriell zum Fernschreiber befördert; will man dagegen einen einfachen Drucker ansteuern, erweist sich dieser Zeichenbuffer als besonders nützlich, weil solche Drucker meist von rechts nach links ausgeben.

Der Buffer enthält das in der Ausgabezeile ganz links stehende ASCII-Zeichen an der Adresse 01D0, das ganz rechts stehende an der Adresse 01DF. Um die Befehle auf einem kleinen Drucker auszugeben, wird ein besonderes Programm benötigt (0112), das individuell den Bedürfnissen des Druckers angepaßt werden muß und deshalb hier nicht aufgelistet ist; es ist lediglich statt des Fernschreiber-Ausgabeprogramms in den Programmablauf einzufügen.

Es sei nicht verschwiegen, daß der hier beschriebene Disassembler unter Umständen durchaus noch gewisse Vereinfachungen zuläßt, insbesondere, was die Struktur der Decodierungstabellen angeht. Diese Tabellen sind so aufgebaut, daß auch zwischen Adressierungsarten unterschieden wird, die zufällig den gleichen Ausdruck zur Folge haben, z.B. Absolute und Zero-Page (der Unterschied besteht hier nur in der Länge des

Arguments). Dies ist zwar nicht unbedingt erforderlich, vereinfacht jedoch spätere Erweiterungen oder die Umstellung auf andere Darstellungsarten erheblich.

### *Belegter Speicherplatz*

Das Disassembler-Programm (Abb. 4.2.2) wurde so ausgelegt, daß es in den im KIM-1 in der Grundstufe vorhandenen Speicherplatz paßt. Dabei wurde darauf geachtet, daß auch solche Anwenderprogramme disassembliert werden können,

#### 4.2.2 Hier hat sich der Disassembler selbst ausgedruckt

```
DISASSEMBLER
BRANCH-ADR. ABS.
```

```
START: 0000
0000 LDA 17F5
0003 STA FA
0005 LDA 17F6
0008 STA FB
000A LDX #10
000C LDA #120
000E STA 01CF,X
0011 DEX
0012 BNE 000C
0014 LDA FB
0016 JSR 013D
0019 STY 01D0
001C STA 01D1
001F LDA FA
0021 JSR 013D
0024 STY 01D2
0027 STA 01D3
002A NOP
002B NOP
002C NOP
002D LDX #00
002F LDY #00
0031 LDA (FA),Y
0033 AND 01B0,X
0036 EOR 01C0,X
```

```
0039 BEQ 0048
003B INX
003C CPX #10
003E BNE 002F
0040 LDA #3F
0042 STA 01DF
0045 JMP 1929
0048 LDA 03D8,X
004B STA F4
004D LDA #00
004F STA F0
0051 LDA 03E8,X
0054 STA EF
0056 LDX F4
0058 DEX
0059 BEQ 0078
005B DEX
005C BEQ 006B
005E LDY #02
0060 LDA (FA),Y
0062 JSR 013D
0065 STY 01D9
0068 STA 01DA
006B LDY #01
006D LDA (FA),Y
006F JSR 013D
```

0072	STY 01DB	00D1	DEX
0075	STA 01DC	00D2	BNE 00CC
0078	JMP(00EF)	00D4	JMP 0185
007B	LDA #23	00D7	JMP 0100
007D	STA 01D9	00DA	JMP 0157
0080	LDA #24	0100	LDA 02FF,X
0082	BNE 0086	0103	STA 01D5
0084	LDA #41	0106	LDA 031F,X
0086	BNE 00A7	0109	STA 01D6
0088	LDA #29	010C	LDA 033F,X
008A	STA 01DD	010F	STA 01D7
008D	LDA #2C	0112	JSR 1E2F
008F	STA 01DE	0115	LDX #00
0092	LDA #59	0117	LDA 01D0,X
0094	BNE 00A2	011A	JSR 1EA0
0096	LDA #2C	011D	INX
0098	STA 01DD	011E	CPX #10
009B	LDA #58	0120	BNE 0117
009D	STA 01DE	0122	JSR 1F63
00A0	LDA #29	0125	LDA FA
00A2	STA 01DF	0127	CMP 17F7
00A5	LDA #28	012A	BNE 0133
00A7	STA 01DA	012C	LDA FB
00AA	BNE 00C3	012E	CMP 17F8
00AC	LDA #58	0131	BEQ 013A
00AE	BNE 00B2	0133	DEC F4
00B0	LDA #59	0135	BNE 0122
00B2	STA 01DE	0137	JMP 000A
00B5	LDA #2C	013A	JMP 1C4F
00B7	BNE 00C0	013D	TAX
00B9	LDA #28	013E	LSR A
00BB	STA 01D8	013F	LSR A
00BE	LDA #29	0140	LSR A
00C0	STA 01DD	0141	LSR A
00C3	NOP	0142	JSR 014B
00C4	NOP	0145	TAY
00C5	NOP	0146	TXA
00C6	LDY #00	0147	JSR 014B
00C8	LDA (FA),Y	014A	RTS
00CA	LDX #20	014B	AND #0F
00CC	CMP 02DF,X		
00CF	BEQ 00D7		

zu Abb. 4.22



014D CMP #0A	0177 STY 01DB
014F CLC	017A LDA F9
0150 BMI 0154	017C JSR 013D
0152 ADC #07	017F STY 01D9
0154 ADC #30	0182 JMP 00A7
0156 RTS	0185 LDX #18
0157 LDA FB	0187 LDY #00
0159 STA F9	0189 LDA (FA),Y
015B LDY #01	018B AND 03A7,X
015D LDA (FA),Y	018E EOR 03BF,X
015F BPL 0163	0191 BEQ 0199
0161 DEC F9	0193 DEX
0163 CLC	0194 BNE 0189
0164 ADC #02	0196 JMP 1929
0166 BCC 016A	0199 LDA 035F,X
0168 INC F9	019C STA 01D5
016A CLC	019F LDA 0377,X
016B ADC FA	01A2 STA 01D6
016D BCC 0171	01A5 LDA 038F,X
016F INC F9	01A8 STA 01D7
0171 JSR 013D	01AB JMP 0112
0174 STA 01DC	

zu Abb. 4.22

die an der meist üblichen Startadresse 0200 beginnen. Die Länge dieser Anwenderprogramme ist hier allerdings bis zur Adresse 02DF begrenzt, denn dort beginnen einige für den Disassembler erforderliche Code-Tabellen. *Abb. 4.2.3* zeigt die hexadezimale Programmauflistung mit diesen Tabellen.

Eine Verlegung des Disassemblers in einen anderen Speicherbereich ist bei erweiterten Systemen durchaus möglich. Dabei ist zu beachten, daß außer den Adressen auch der LDA-00-Befehl, der die für einen indirekten Sprung während der Adressierart-Decodierung nötige Zieladresse als höherwertige Byte in die Zelle 00F0 speichert (Page der Zieladresse), geändert werden muß.

0000	AD	F5	17	85	00A0	A9	29	8D	DF
0004	FA	AD	F6	17	00A4	01	A9	28	8D
0008	85	FB	A2	10	00A8	DA	01	D0	17
000C	A9	20	9D	CF	00AC	A9	58	D0	02
0010	01	CA	D0	F8	00B0	A9	59	8D	DE
0014	A5	FB	20	3D	00B4	01	A9	2C	D0
0018	01	8C	D0	01	00B8	07	A9	28	8D
001C	8D	D1	01	A5	00BC	D8	01	A9	29
0020	FA	20	3D	01	00C0	8D	DD	01	EA
0024	8C	D2	01	8D	00C4	EA	EA	A0	00
0028	D3	01	EA	EA	00C8	B1	FA	A2	20
002C	EA	A2	00	A0	00CC	DD	DF	02	F0
0030	00	B1	FA	3D	00D0	06	CA	D0	F8
0034	B0	01	5D	C0	00D4	4C	85	01	4C
0038	01	F0	0D	E8	00D8	00	01	4C	57
003C	E0	10	D0	EF	00DC	01			
0040	A9	3F	8D	DF	0100	BD	FF	02	8D
0044	01	4C	29	19	0104	D5	01	BD	1F
0048	BD	D8	03	85	0108	03	8D	D6	01
004C	F4	A9	00	85	010C	BD	3F	03	8D
0050	F0	BD	E8	03	0110	D7	01	20	2F
0054	85	EF	A6	F4	0114	1E	A2	00	BD
0058	CA	F0	1D	CA	0118	D0	01	20	A0
005C	F0	0D	A0	02	011C	1E	E8	E0	10
0060	B1	FA	20	3D	0120	D0	F5	20	63
0064	01	8C	D9	01	0124	1F	A5	FA	CD
0068	8D	DA	01	A0	0128	F7	17	D0	07
006C	01	B1	FA	20	012C	A5	FB	CD	F8
0070	3D	01	8C	DB	0130	17	F0	07	C6
0074	01	8D	DC	01	0134	F4	D0	EB	4C
0078	6C	EF	00	A9	0138	0A	00	4C	4F
007C	23	8D	D9	01	013C	1C	AA	4A	4A
0080	A9	24	D0	02	0140	4A	4A	20	4B
0084	A9	41	D0	1F	0144	01	A8	0A	20
0088	A9	29	8D	DD	0148	4B	01	60	29
008C	01	A9	2C	8D	014C	0F	C9	0A	18
0090	DE	01	A9	59	0150	30	02	69	07
0094	D0	0C	A9	2C	0154	69	30	60	A5
0098	8D	DD	01	A9	0158	FB	85	F9	A0
009C	58	8D	DE	01	015C	01	B1	FA	10

#### 4.2.3 Hexadezimale Programmliste des Disassemblers

0160	02	C6	F9	18	0310	49	4E	50	50
0164	69	02	90	02	0314	50	50	52	52
0168	E6	F9	18	65	0318	54	54	54	54
016C	FA	90	02	E6	031C	54	54	53	53
0170	F9	20	3D	01	0320	52	43	43	45
0174	8D	DC	01	8C	0324	4D	4E	50	56
0178	DB	01	A5	F9	0328	56	4C	4C	4C
017C	20	3D	01	8C	032C	4C	45	45	4E
0180	D9	01	4C	A7	0330	4E	4F	48	48
0184	00	A2	18	A0	0334	4C	4C	54	54
0188	00	B1	FA	3D	0338	41	41	53	58
018C	A7	03	5D	BF	033C	58	59	45	45
0190	03	F0	06	CA	0340	4B	43	53	51
0194	D0	F3	4C	29	0344	49	45	4C	43
0198	19	BD	5F	03	0348	53	43	44	49
019C	8D	D5	01	BD	034C	56	58	59	58
01A0	77	03	8D	D6	0350	59	50	41	50
01A4	01	BD	8F	03	0354	41	50	49	53
01A8	8D	D7	01	4C	0358	50	59	58	41
01AC	12	01	00	00	035C	53	41	43	44
01B0	FF	FF	FF	DF	0360	45	53	53	52
01B4	9F	1F	8D	1F	0364	4F	4C	4C	4C
01B8	1F	1C	1C	1C	0368	4C	43	41	41
01BC	1C	19	05	14	036C	41	4A	53	53
01C0	20	6C	BE	96	0370	52	49	44	43
01C4	0A	10	80	11	0374	43	42	4A	53
01C8	01	04	0C	14	0378	4F	54	42	4F
01CC	1C	19	00	00	037C	52	53	44	44
02E0	00	90	B0	F0	0380	44	4D	53	4E
02E4	30	D0	10	50	0384	44	4D	54	54
02E8	70	18	D8	58	0388	4F	4E	45	50
02EC	B8	CA	88	E8	038C	50	49	53	45
02F0	C8	EA	48	08	0390	52	41	43	4C
02F4	68	28	40	60	0394	41	52	59	58
02F8	AA	A8	BA	8A	0398	41	50	4C	44
02FC	9A	98	38	F8	039C	43	50	59	58
0300	42	42	42	42	03A0	52	43	43	58
0304	42	42	42	42	03A4	59	54	52	49
0308	42	43	43	43	03A8	E3	E3	E3	E3
030C	43	44	44	49	03AC	E3	E3	E3	E3

zu Abb. 4.2.3

03B0	E3	E3	E3	E3	03D8	03	03	03	02
03B4	E3	DF	E7	E7	03DC	01	02	02	02
03B8	E7	E7	E7	F3	03E0	02	02	03	02
03BC	F3	F7	FF	FF	03E4	03	03	01	02
03C0	41	81	E1	22	03E8	C6	B9	B0	B0
03C4	01	42	A0	A2	03EC	84	DA	7B	88
03C8	A1	C1	02	21	03F0	96	C6	C6	AC
03CC	61	4C	84	86	03F4	AC	B0	C6	7B
03D0	66	E6	C6	E0	03F8	E9	ED	ED	ED
03D4	C0	24	20	78					

zu Abb. 4.2.3

### 4.3 Plotter für das Speicher-Oszilloskop

Weiter oben war beschrieben, wie man ein gewöhnliches Oszilloskop mit Hilfe des Mikrocomputers KIM-1 zu einem Speicheroszilloskop umfunktionieren kann. Dieses Programm bedient sich eines Analog-Digital-Wandlers, der die momentanen Amplitudenwerte in die Speicherzellen 0200 ... 03FF ablegt.

Das in *Abb. 4.3* aufgelistete KIM-Maschinencode-Programm (Mikroprozessor 6502) ist nun in der Lage, den Speicherinhalt, wenn auch nur in recht grober Auflösung, auf ein Terminal bzw. einen ASCII-Fernschreiber als Kurve auszugeben. Diesen Vorgang nennt man „Plotten“.

Startadresse des Programms ist 0120. Die Auslegung erfolgte für ein SWTPC-CT-64-Terminal, das 16 Zeilen mit je 64 Zeichen darstellen kann. Um vor dem Ausgeben der Kurve den Bildschirm zu löschen, enthält die Zeile 0119 einige ASCII-Zeichen, die folgender Terminal-Tastenbelegung entsprechen: Home Cursor = CTRL F; Erase Sreen = CTRL E; Page/Scroll Mode = CTRL N. Die Umschaltung Page/Scroll und zurück wird deshalb vorgenommen, weil in der Scroll-Betriebsart die Funktion „Home Cursor“ nicht definiert ist.



```

0119 00 05 06 06 0E 0E 00
0120 A2 07 LDX #07 CURSOR
0122 BD 18 01 LDA 0118,X HOME UND
0125 20 A0 1E JSR 1EA0 BILDSCHIRM
0128 CA DEX LOESCHEN
0129 D0 F7 BNE 0122
012B A9 10 LDA #10 MAX.AMPL.
012D 85 F3 STA F3
012F C6 F3 DEC F3 AMPL.DEKR.
0131 10 03 BPL 0136
0133 4C 64 1C JMP 1C64 ZUM KIM
0136 A9 00 LDA #00
0138 85 FA STA FA ZEIGER=
013A A9 02 LDA #02 0200
013C 85 FB STA FB
013E A2 3F LDX #3F 64 ZEICHEN
0140 A0 00 LDY #00 PRO ZEILE
0142 B1 FA LDA (FA),Y
0144 4A LSR A AMPLITUDE
0145 4A LSR A DURCH
0146 4A LSR A 8 DIVID.
0147 EA NOP
0148 38 SEC NULLPUNKT
0149 E9 0F SBC #0F VERSCHIEBEN
014B EA NOP
014C C5 F3 CMP F3 VGL.ZEILE
014E F0 06 BEQ 0156 UND AMPL.
0150 20 9E 1E JSR 1E9E SPACE
0153 18 CLC
0154 90 05 BCC 015B
0156 A9 2E LDA #2E PUNKT
0158 20 A0 1E JSR 1EA0 DRUCKEN
015B A0 08 LDY #08 ZEIGER
015D 20 63 1F JSR 1F63 UM 8
0160 88 DEY BYTE VOR-
0161 D0 FA BNE 015D RUECKEN
0163 CA DEX
0164 D0 DA BNE 0140
0166 20 2F 1E JSR 1E2F CRLF
0169 4C 2F 01 JMP 012F
016C 00 BRK ENDE

```

#### 4.3 Ausgabe eines aufgezeichneten Spannungs- verlaufes auf einem Terminal

Die gespeicherte Amplitude wird vor der Darstellung durch 8 dividiert; anschließend wird hexadezimal 0F abgezogen. Dies dient hier dazu, um ausschließlich positive Spannungen (hex 80 ... FF) darzustellen, diese dann aber über die gesamte Bildschirmhöhe zu spreizen. Will man auch negative Spannungswerte plotten, so ist Zeile 0147 durch 4A und Zeile 0149 durch E900 zu ersetzen.

#### 4.4 Datensuche — ein Karteiprogramm

Nehmen wir an, Sie hätten eine Kundenkartei mit hundert oder mehr Namen, Adressen und Telefonnummern. Wenn Sie jetzt herausfinden wollen, wer Ihrer Kunden z.B. in Frankfurt wohnt, wer mit Vornamen Max heißt oder wer die Telefonnummer mit den drei Vieren am Ende hat, dann kann die Sucharbeit Stunden dauern — es sei denn, Sie bedienen sich des hier vorgestellten Programms, das für den Mikrocomputer KIM-1 entwickelt wurde.

##### *Das Prinzip der Stichwort-Suche*

In einem bestimmten RAM-Speicherbereich, z.B. ab der Adresse 0200, stehen Namen, Adressen oder sonstige Textdaten in ASCII-Form, also ein Zeichen pro Byte. Die zusammenhängenden Daten, d.h. alles, was z.B. zu einem Namen gehört, sind jeweils durch 0D (Hex), also ein Wagenrücklauf-Zeichen abgegrenzt.

Wird das Programm gestartet, so erscheint ein Fragezeichen am Beginn einer neuen Zeile. Wenn man nun ein Stichwort eingibt, z.B. MAX, gefolgt von der Return-Taste, so beginnt sofort der Mikrocomputer alle gespeicherten Texte nach diesem Stichwort, besser gesagt, nach der gerade eingegebenen ASCII-Zeichenfolge, abzusuchen. Dann werden alle Texte, die das Stichwort enthalten, nacheinander ausgedruckt.

Enthält der Speicher z.B. einen Text „MAX MEIER, FRANKFURT, 061112345“, so wird dieser Text ausgegeben, wenn das Such-Stichwort z.B. MAX, FRANKFURT, FRANK oder 0611 lautete. Dadurch ist ein Suchen praktisch nach beliebigen Kriterien möglich.

### *Tastatur-Steuerbefehle*

Das Programm wurde für den Mikrocomputer KIM-1 mit einem seriell arbeitenden ASCII-Fernschreiber oder -Terminal entwickelt. Eine Reihe von ASCII-Zeichen bzw. Terminal-Tasten sind für spezielle Steuerfunktionen reserviert. Im einzelnen sind dies folgende Tasten:

#### *~ (n)*

Daten vom Band laden; dabei wird das ASCII-Zeichen (n), z.B. A, B, a, b, 1, 2 usw. als Identifikation benutzt, um einen bestimmten Datenblock ausfindig zu machen. Werden Datenblöcke gefunden, die nicht gesucht werden, so wird deren Identifikation mit ausgedruckt.

#### *ESC*

Die Escape-Taste dient zum Umschalten in den Eingabemodus. Soll vor der Eingabe ein alter Text gelöscht werden, so ist dieser Taste ein Stichwort nachzustellen, das diesen Text eindeutig definiert. Ist die Löschung vollzogen, so wird das Fragezeichen am Beginn der Zeile durch einen Stern ersetzt. (Das funktioniert natürlich nur bei Sichtgeräten, nicht bei Druckern!). Jetzt können neue Texte eingegeben werden, die jeweils mit der Return-Taste beendet werden müssen. Ist die Eingabe beendet, muß wieder ESC gedrückt werden.

#### *CTRL H*

Diese Taste dient – wie üblich – als Back Space und setzt den Cursor um eine Schreibstelle rückwärts. Im Eingabemodus wird dabei gleichzeitig der Speicher-Zeiger um Eins erniedrigt, so daß damit Korrekturen möglich sind, wenn man sich einmal vertippt hat.

#### *↑ (n)*

Daten mit (n) als Identifikations-Zeichen auf Band speichern; wie beim Laden des Bandes kann (n) ein beliebiges ASCII-Zeichen sein. Das verwendete Kassetten-Format benützt die auf der KIM-Platine verwendete Hardware, ist aber rund 12mal schneller als die KIM-Monitor-Routine zur Bandaufzeichnung.

1 KByte wird in nur etwa 11 s auf Band gespeichert!

Soll ein neuer Text eingegeben werden, ohne einen vorhandenen zu löschen, so ist das Zeichen > statt des Stichwortes nach der ESC-Taste zu drücken. Nach einem weiteren Tastendruck, nämlich „Return“, kann der neue Text eingetippt werden, wie dies unter ESC beschrieben ist.

Zu beachten ist dabei, daß ein von zwei Wagenrücklaufzeichen eingeschlossener Text nicht länger als 255 Zeichen sein darf. Wenn einmal versehentlich ESC gedrückt wurde, obwohl gar keine Eingabe beabsichtigt ist, drückt man einfach etwa 10mal die Leertaste und dann Return. Das zu suchende Stichwort darf übrigens nie mehr als 20 Buchstaben oder Ziffern umfassen.

S. 74 zeigt ein Beispiel für den Umgang mit diesem recht universell verwendbaren Programm. Die Benutzereingaben sind dabei unterstrichen. Zu erwähnen wäre noch, daß das Funktionieren der Kassetten-Lesefunktion auf der rechten Stelle des KIM-Display überwacht werden kann; das anfängliche Synchronisieren und das Erkennen des Datenstarts sind deutlich zu sehen.

Das Programm belegt den Adressenbereich 0000 . . . 01F0; die Text-Startadresse ist normalerweise 0200. Wer diese Textadresse in einen anderen Speicherbereich legen möchte, muß die entsprechende Page in die Zellen 000C, 01E6, 0105 und 015C schreiben (normalerweise 02). Bei der erstmaligen Eingabe eines Textes muß der Speicher wie folgt initialisiert werden:

0201	0D	(Carriage Return)
0202	3E	(,, > “)
0203	00	(End-Zeichen)

Abb. 4.4 zeigt eine hexadezimale Auflistung des Programms, wie sie mit dem in diesem Buch beschriebenen Super-KIM-Monitor zustandekommt. Es dürfte damit keine Schwierigkeiten bereiten, es auf Anhieb laufen zu lassen.



0000	20 2F	1E	0051	E8		00A0	38	
0003	A9 3F		0052	D0 EE		00A1	A5 FA	
0005	20 A0	1E	0054	A5 DF		00A3	E9 01	
0008	20 9E	1E	0056	D0 19		00A5	85 FA	
000B	A9 02		0058	20 2F	1E	00A7	B0 02	
000D	85 FB		005B	A0 01		00A9	C6 FB	
000F	A9 00		005D	B1 FA		00AB	18	
0011	85 FA		005F	F0 0E		00AC	90 EB	
0013	A2 18		0061	C9 0D		00AE	C9 18	
0015	95 DE		0063	F0 CE		00B0	F0 12	
0017	CA		0065	84 F9		00B2	C9 0D	
0018	D0 FB		0067	20 A0	1E	00B4	D0 05	
001A	4C C8	01	006A	A4 F9		00B6	20 2F	1E
001D	C9 18		006C	C8		00B9	A9 0D	
001F	D0 07		006D	D0 EE		00BB	C8	
0021	A9 FF		006F	F0 8F		00BC	91 FA	
0023	85 DF		0071	A0 02		00BE	20 63	1F
0025	20 5A	1E	0073	B1 FA		00C1	18	
0028	C9 0D		0075	F0 19		00C2	90 D5	
002A	F0 07		0077	C9 0D		00C4	C8	
002C	95 E0		0079	F0 03		00C5	A9 3E	
002E	E8		007B	C8		00C7	91 FA	
002F	E0 15		007C	D0 F5		00C9	20 63	1F
0031	D0 F2		007E	84 DF		00CC	98	
0033	20 63	1F	0080	A4 DF		00CD	91 FA	
0036	A0 00		0082	B1 FA		00CF	20 63	1F
0038	B1 FA		0084	F0 0A		00D2	A5 FA	
003A	F0 C4		0086	A0 00		00D4	8D F7	17
003C	C9 0D		0088	91 FA		00D7	A5 FB	
003E	D0 F3		008A	20 63	1F	00D9	8D F8	17
0040	A2 00		008D	18		00DC	4C 00	00
0042	C8		008E	90 F0		0100	A9 00	
0043	B5 E0		0090	20 C2	01	0102	85 FA	
0045	F0 0D		0093	E6 DF		0104	A9 02	
0047	B1 FA		0095	D0 09		0106	85 FB	
0049	C9 0D		0097	F0 25		0108	A9 7F	
004B	F0 E6		0099	20 5A	1E	010A	8D 41	17
004D	D5 E0		009C	C9 08		010D	A9 13	
004F	D0 EF		009E	D0 0E		010F	8D 42	17

#### 4.4 Das Karteiprogramm gestattet das Durchsuchen von Textblöcken nach beliebigen Stichworten

0112	20 41 1A	015F	A9 27	01A8	CA
0115	46 F3	0161	85 F5	01A9	D0 E9
0117	05 F3	0163	85 F4	01AB	68
0119	85 F3	0165	A9 BF	01AC	C6 F3
011B	8D 40 17	0167	8D 43 17	01AE	F0 05
011E	C9 16	016A	A9 16	01B0	30 07
0120	D0 F0	016C	20 88 01	01B2	4A
0122	20 24 1A	016F	C6 F4	01B3	90 DB
0125	8D 40 17	0171	D0 F7	01B5	A0 00
0128	C9 2A	0173	A9 2A	01B7	F0 D7
012A	D0 F2	0175	20 88 01	01B9	C6 F2
012C	20 24 1A	0178	A0 00	01BB	10 CF
012F	C5 F4	017A	B1 FA	01BD	60
0131	F0 0C	017C	20 63 1F	01BE	02 C3
0133	20 8C 1E	017F	48	01C0	03
0136	20 A0 1E	0180	20 88 01	01C1	7E A9 2A
0139	20 9E 1E	0183	68	01C4	20 A0 1E
013C	4C 00 01	0184	D0 F2	01C7	20 2F 1E
013F	91 FA	0186	F0 C4	01CA	60
0141	20 63 1F	0188	A0 07	01CB	20 5A 1E
0144	20 24 1A	018A	84 F2	01CE	C9 7E
0147	AA	018C	A0 02	01D0	D0 0B
0140	D0 F5	018E	84 F3	01D2	20 5A 1E
014A	91 FA	0190	BE BE 01	01D5	85 F4
014C	20 8C 1E	0193	48	01D7	20 2F 1E
014F	20 1E 1E	0194	2C 47 17	01DA	4C 00 01
0152	4C 00 00	0197	10 FB	01DD	C9 5E
0155	00	0199	B9 BF 01	01DF	D0 0C
0156	00	019C	8D 44 17	01E1	20 5A 1E
0157	A9 00	019F	A5 F5	01E4	8D 00 02
0159	85 FA	01A1	49 80	01E7	20 9E 1E
015B	A9 02	01A3	8D 42 17	01EA	4C 57 01
015D	85 FB	01A6	85 F5	01ED	4C 1D 00

zu Abb. 4.4

## Beispiel für den Umgang mit dem Suchprogramm (r = Return-Taste, e = Escape)

1873 A9 G  
 KIM  
 0000 20 G  
 ? ~ P

0D2B

? INHALT r

INHALT: PRAXIS & HOBBY 1978

? UHR r

MODUL-UHR MIT WECKSCHALTUNG 1/33

TASCHENRECHNER ALS SCHALTUHR

UND ZÄHLER 4/157

DIGITALUHR MIT MN 5316 16/781

? e MN 5316 r

DIGITALUHR MIT MM 5316 r

e

? 5316 r

DIGITALUHR MIT MM 5316

? e > r

STEREO-NACHBRENNER 19/945 r

e

? ↑ P

Das Steuerprogramm wird von  
 der Kassette geladen ...  
 ... und gestartet.  
 Daten mit dem Kennbuchstaben P werden von der  
 Kassette geladen.  
 Fertig, Text-Endadresse ist 0D2B.  
 „INHALT“ ist hier der Suchbegriff.  
 Aha! Sehen wir weiter.  
 Gesucht: Uhren aller Art.

Halt! Es muß heißen MM 5316!  
 Der richtige Text wird eingegeben.  
 Eingabe-Modus verlassen!  
 Zur Kontrolle ...  
 ... sehen wir es uns nochmal an.  
 Ein neuer Text ...  
 ... soll hinzugefügt werden.  
 Das ist alles!  
 Der korrigierte Inhalt wird auf die Kassette geladen,  
 Kennbuchstabe ist wieder P.



#### 4.5 Automatische Text-Formatierung

Wenn Sie eine Zeitschrift lesen, machen Sie sich wahrscheinlich kaum Gedanken darüber, warum alle Zeilen immer genau gleich lang sind und welcher technische Aufwand dahinter steckt, diese Formatierung und auch die richtige Trennung von Worten zu erreichen. Früher war dies Aufgabe der Schriftsetzer; heute wird oft mit einem Computer gesetzt, dem ein fortlaufender Text (also ohne Zeilenwechsel- oder Trennungszeichen) eingetippt wird. Der Computer sorgt dann von selbst dafür, daß genau die gewünschte Zeilenlänge eingehalten wird, und dank eines umfangreichen Programms ist er sogar schlau genug, Worte an der richtigen Stelle zu trennen.

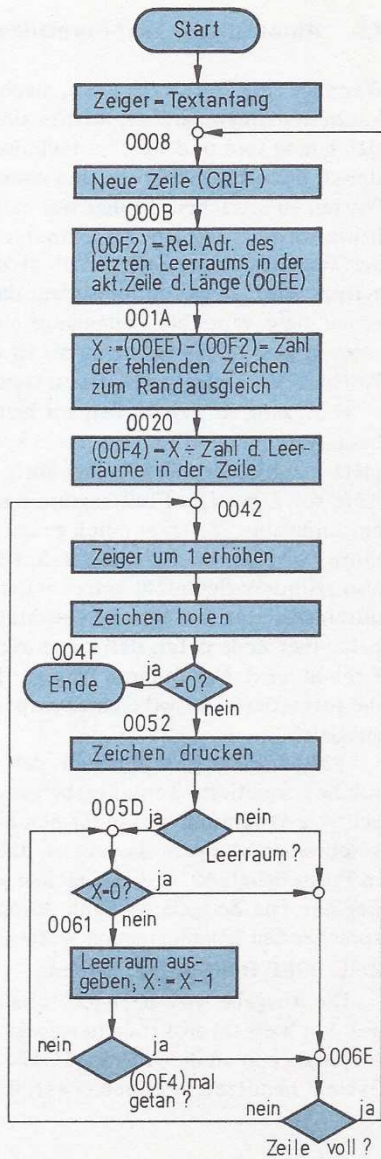
Nun, ganz so weit wollen wir hier nicht gehen, sonst bleibt bei den üblichen Mikrocomputer-Systemen ja kein Speicherplatz mehr für den Text selbst übrig. Zumindest erlaubt das in *Abb. 4.5.1* gezeigte Flußdiagramm aber die richtige Trennung am Ende einer Zeile, nämlich genau an der Stelle, wo ein Leerzeichen zwischen zwei Worten steht (die Worte selbst werden also grundsätzlich nicht getrennt); und nicht zuletzt sorgt es durch gezieltes Einfügen von zusätzlichen Leerräumen innerhalb einer Zeile dafür, daß die gewünschte Zeilenlänge exakt erreicht wird. Mit anderen Worten: Das Programm besorgt die formatierte Ausgabe eines gespeicherten Textes mit Randausgleich.

*Abb. 4.5.2* zeigt schließlich, daß der Aufwand für eine solche formatierte Textausgabe gar nicht sehr groß ist. Das aufgelistete Programm wurde für den Mikrocomputer KIM-1 geschrieben und geht davon aus, daß der auszudruckende Text in Form beliebiger ASCII-Zeichen an der Adresse 0200 beginnt. Die Zeilenlänge kann durch Einschreiben des entsprechenden hexadezimalen Wertes (max. FF = 255) in die Zelle 00EE frei gewählt werden.

Die Ausgabe wird beendet, sobald das Programm im Text auf den Wert 00 stößt; dann erfolgt ein Sprung zum KIM-Monitorprogramm an die Adresse 1C4F. Für alle, die ein anderes System benutzen, sei noch erwähnt, welche Bedeutung die vor-



#### 4.5.1 Ausgabe eines Textes mit automatischem Randausgleich



0000 A9 00	LDA #\$00	0037 E4 F3	CPX \$F3
0002 85 FA	STA \$FA	0039 30 04	BNI 003F
0004 A9 02	LDA #\$02	003B E6 F4	INC \$F4
0006 85 FB	STA \$FB	003D 00 EA	BNE 0029
0008 20 2F 1E	JSR \$1E2F	003F EA	NOP
000A A4 EE	LDV \$EE	0040 EA	NOP
000C B1 FA	LDA (\$FA),Y	0041 EA	NOP
000E C9 20	CMP #\$20	0042 A5 F4	LDA \$F4
0010 F0 05	BEQ 0018	0044 85 F3	STA \$F3
0012 88	DEY	0046 20 63 1F	JSR \$1F63
0014 D0 F7	BNE 000D	0049 A0 00	LDV #\$00
0016 A4 EE	LDV \$EE	004B B1 FA	LDA (\$FA),Y
0018 84 F2	STV \$F2	004D D0 03	BNE 0052
001A A5 EE	LDA \$EE	004F 4C 4F 1C	JMP \$1C4F
001C 38	SEC	0052 85 F9	STA \$F9
001E E5 F2	SBC \$F2	0054 20 A0 1E	JSR \$1EA0
001F AA	TAX	0057 A5 F9	LDA \$F9
0020 A9 00	LDA #\$00	0059 C9 20	CMP #\$20
0022 85 F3	STA \$F3	005B D0 11	BNE 006E
0024 85 F4	STA \$F4	005D E0 00	CPX #\$00
0026 EA	NOP	005F F0 06	BEQ 0067
0027 EA	NOP	0061 20 9E 1E	JSR \$1E9E
0028 EA	NOP	0064 CA	DEX
0029 A4 F2	LDV \$F2	0065 A5 F3	LDA \$F3
002B 88	DEY	0067 F0 05	BEQ 006E
002C B1 FA	LDA (\$FA),Y	0069 C6 F3	DEC \$F3
002E C9 20	CMP #\$20	006B 18	CLC
0030 D0 02	BNE 0034	006C 90 EF	BCC 005D
0032 E6 F3	INC \$F3	006E C6 F2	DEC \$F2
0034 88	DEY	0070 D0 D0	BNE 0042
0035 D0 F5	BNE 002C	0072 4C 00 00	JMP \$0000

#### 4.5.2 Disassembliertes Textausgabe-Programm

kommenden Unterprogramm-Sprünge in das Monitorprogramm haben:

<i>Unterprogramm</i>	<i>Adresse</i>
Ausdruck eines Leerraums	1E9E
Ausdruck eines Zeichens	1EAO
Erhöhen des Zeigers (FA, FB)	1F63

Zum Schluß sei noch erwähnt, daß das Programm noch einige Variationsmöglichkeiten bietet. Z.B. wäre es ohne weiteres möglich, bei Erkennen eines Wagenrücklaufzeichens im gespeicherten Text die aktuelle Zeile gewaltsam zu beenden und eine neue zu beginnen, um die Möglichkeit zu schaffen, Absätze in den Text einzufügen. Wie bei allen Programmvorschlägen, so ist auch hier das Können des Computers in erster Linie durch den Zeitaufwand begrenzt, den man in die notwendigen Programme investiert!

#### 4.6 KIM versteht Pseudo-Befehle

Das nachfolgende Kapitel zeigt, daß es mit recht geringem Software-Aufwand schon möglich ist, den Mikrocomputer KIM-1 dazu zu bewegen, übliche aus der Programmiersprache BASIC gewohnte Worte zu „verstehen“ und als Steuerbefehle zu interpretieren. Dies erleichtert den Umgang mit dem System ungemein; die direkte Eingabe von Steuerworten erspart oft eine Unzahl mühsamer Einzeloperationen, z.B. das Speichern von Daten auf eine bestimmte Adresse zur Definition des auf Band aufzuzeichnenden Speicherbereichs, das Verschieben von Programmteilen u. a.

Die hier gezeigte Programmversion decodiert nur Steuerbefehle, stellt also keine Programmiersprache im üblichen Sinne dar, da die Befehle nicht als Programmbefehle gespeichert, sondern sofort ausgeführt werden. Die verwendete Methode, wie die Worte decodiert werden, läßt sich aber selbstverständlich auch bei der Verwirklichung von Compilern z.B. zum Aufruf und Transfer ganzer Programm-Moduln einsetzen.

Abb. 4.6.1 zeigt, wie die einzelnen Worte decodiert werden. Alle möglichen Steuerbefehle sind als Folge von ASCII-Zeichen in einer Tabelle gespeichert. Das Format dieser Tabelle ist:

FF XXXX ABCDEFG FF

Die Daten FF kennzeichnen dabei den Beginn und das Ende eines Tabellenteils, wobei die Enddaten FF gleichzeitig wieder





den Beginn eines neuen Wortes darstellen. XXXX steht hier für eine vierstellige Adresse; zu dieser Adresse springt das Programm, wenn das nachfolgende Wort (hier ABCDEFG) decodiert wurde. Das an dieser Adresse dann stehende Programm bewirkt die Ausführung des gewünschten Steuerbefehls.

In der vorliegenden Programmversion darf jedes Wort maximal aus 10 Buchstaben bestehen; ASCII-Zeichen von 00 . . . 20 werden ignoriert. Will man z.B. START eingeben, tippt aber versehentlich STURT ein (keine Reaktion), so braucht man nur z.B. die Leertaste zu drücken (ASCII 20), um das Wort richtig einzugehen. (Das Drücken der Return-Taste ist übrigens nicht erforderlich.)

### *Programmieren in Maschinensprache*

Nehmen wir einmal an, wir wollen ein Programm zur Anzeige von FFFFFFFF auf dem KIM-Display realisieren. Dieses Programm kann z.B. die Form haben:

```
0200   A9 FF
0202   85 F9
0204   85 FA
0206   85 FB
0208   20 1F 1F
020B   4C 00 02
```

Die Befehle sind hier bereits in der richtigen Länge pro Zeile geschrieben, und vorne steht eine vierstellige Adresse. Diese Formatierung geschieht bei Verwendung des Pseudo-Befehles „HEX“ automatisch; auch „LIST“ liefert dieses Format. Wollen wir obiges Programm eingeben, so tun wir einfach folgendes:

Zunächst einmal starten wir das Decodierprogramm an der Adresse 1200; es erfolgt sofort der Ausdruck „READY“. Dann geben wir „NEW“ ein; wir wollen ja ein neues Programm realisieren. Ein Leerraum (Space) wird nach NEW automatisch ausgedruckt, sobald das Wort decodiert wurde; jetzt müssen wir noch eine Kennnummer eingeben, z.B. 01, die später bei der

Bandaufzeichnung als Identifikation für das Anwenderprogramm dient. Jetzt wird sofort am linken Rand der nächsten Zeile die Adresse 0200 ausgedruckt, und das System wartet nun auf eine Eingabe. (NEW geht automatisch immer zur Adresse 0200, da dies für das KIM-System eine sinnvolle Programm-Startadresse ist.)

Da wir das Programm in hexadezimaler Form eingeben müssen, beginnen wir diese Eingabe mit dem Steuerbefehl HEX. Er hat zur Folge, daß sofort in der nächsten Zeile nochmals die Adresse 0200 geschrieben wird; wenn man jetzt die Hexadezimal-Daten bzw. -Befehle eingibt (A9FF85F985FA usw.), so erkennt das System selbständig die richtige Befehlslänge und liefert ein Format, wie es bei der vorher dargestellten Programm-Auflistung bereits zu sehen ist.

Haben wir alles eingegeben, so steht in der nächsten Zeile die Adresse 020E. Um den Hexadezimal-Modus zu beenden, tippen wir noch FF ein; und wieder wird eine neue Zeile mit der Adresse 020E eröffnet. An diese Stelle schreiben wir einfach END. Daraufhin springt das System sofort zur Startadresse (hier 0200). Um die richtige Eingabe zu kontrollieren, kann LIST verwendet werden. Dieser Befehl listet das Programm zwischen Start- und Endadresse wiederum in richtiger Formatierung (wie oben) auf, druckt READY aus und springt wieder an die Startadresse zurück.

### *Einige Steuerbefehle*

Mit dem Befehl SAVE kann das Programm jetzt auf Band aufgenommen werden; dabei wird das Hypertape-Programm verwendet, das eine Geschwindigkeit von etwa 800 Bd ermöglicht. Der Befehl OLD, gefolgt von der gewünschten Identifikations-Kennzahl (hier 01), sorgt für das Wiedereinlesen vom Band. Danach erfolgt allerdings ein Sprung zum KIM-Monitor auf die Adresse 0000 bzw. FFFF (letzteres bedeutet eine Fehlermeldung). An der Adresse 1200 kann das Programm dann wieder gestartet werden.

*Tabelle des Befehlsvorrats*

Befehl	Argument	Wirkung
BYE	—	Sprung zum KIM-Monitor auf die aktuelle Adresse
RUN	—	Ausführung des Anwenderprogramms ab der aktuellen Adresse
NEW	Progr.- Kenn- ziffer (ID)	Speicherung der Kennziffer an der Adresse 17F9 und Sprung an die Adresse 0200
LINE	Neue Adresse	Sprung zu einer neuen Adresse
START	—	Aktuelle Adresse wird als Startadresse deklariert
END	—	Aktuelle Adresse wird als Endadresse deklariert
OLD	Progr.- Kenn- ziffer	Laden eines Programms mit der def. Kennziffer, anschließend Sprung zum KIM-Monitor
SAVE	—	Aufzeichnen des Programms zwischen START und END auf Band (Hypertape)
LIST	—	Auflisten des Programms von der aktuellen Adresse bis zu END; Formatierung mit richtiger Befehlslänge
STRING	ASCII- Zeichen	Speichern von ASCII-Zeichen ab der aktuellen Adresse bis „Escape“ (1B); letzteres wird nicht mitgespeichert
INSERT	n	Einfügen von n Bytes an der aktuellen Adresse und entsprechende Korrektur der Endadresse ( $0 \leq n \leq \text{FF}$ )
DELÉTE	n	Löschen von n Bytes, sonst wie INSERT
HEX	Hexadez.- Bytes	Eingabe eines Programms in Maschinensprache, beendet mit FF; automatische Formatierung wie bei LIST
?	—	Ausdruck bzw. Anzeige des Bytes an der aktuellen Adresse



Will man das zuvor eingegebene Programm starten, so braucht man nur (an der Adresse 0200) RUN einzugeben. Um dieses Programm allerdings wieder anzuhalten, muß ein RST- oder NMI-Interrupt verwendet werden.

In diesem Zusammenhang hat es sich als praktisch erwiesen, die Decodier-Startadresse als NMI-Vektor zu verwenden (17FA = 00 und 17 FB = 12) und eine nicht belegte Taste des Terminals oder Fernschreibers mit der Stop-Taste des KIM zu verbinden. Dadurch kann man ein laufendes Anwenderprogramm jederzeit anhalten, indem man diese Taste drückt, und es erfolgt dann der Ausdruck von READY; das System erwartet die Eingabe neuer Steuerbefehle.

Die übrigen Steuerbefehle und ihre Wirkung gehen aus der *Tabelle* hervor. Jeder richtig decodierte Steuerbefehl, dem ein Argument folgen muß, z.B. die Band-Identifikationszahl oder die gewünschte Adresse, erzeugt hinter dem letzten Buchstaben sofort einen Leerraum, dem das Argument folgen muß. Dieser Leerraum wird auch bei SAVE erzeugt, um anzuzeigen, daß der Befehl richtig decodiert wurde. Nach der Aufzeichnungszeit meldet sich das System mit READY wieder.

#### *Mögliche Änderungen*

Die in *Abb. 4.6.2* aufgelistete Programmversion ist für ein KIM-System mit 4 KByte externer RAM-Erweiterung gedacht; daher liegt es im Adressenbereich 1100 . . . 13FF. Wenn man bei allen 3-Byte-Befehlen, bei denen das dritte Byte 11, 12 oder 13 lautet, eine entsprechende Änderung vornimmt, läßt es sich in beliebige andere „Pages“ (Speicher-Seiten) transferieren. Will man den Befehlsvorrat erweitern, so muß man einige Programmteile aus der Page 13 in einen anderen Speicher- teil legen und die Adressen entsprechend korrigieren. Damit wird in dieser Page 13 mehr Platz für die Wortetabelle frei, die nach dem oben erwähnten Format beliebig erweitert werden kann.



1100	A9	AD	LDA #AD	1170	48	PHA
1102	8D	EC	17 STA 17EC	1171	4A	LSR
1105	20	32	19 JSR 1932	1172	4A	LSR
1108	A9	27	LDA #27	1173	4A	LSR
110A	85	F5	STA F5	1174	4A	LSR
110C	A9	BF	LDA #BF	1175	20	7D 11 JSR 117D
110E	8D	43	17 STA 1743	1178	68	PLA
1111	A2	64	LDX #64	1179	20	7D 11 JSR 117D
1113	A9	16	LDA #16	117C	60	RTS
1115	20	61	11 JSR 1161	117D	29	oF AND #oF
1118	A9	2A	LDA #2A	117F	C9	oA CMP #oA
111A	20	88	11 JSR 1188	1181	18	CLC
111D	AD	F9	17 LDA 17F9	1182	30	o2 BMI 1186
1120	20	70	11 JSR 1170	1184	69	o7 ADC #o7
1123	AD	F5	17 LDA 17F5	1186	69	30 ADC #30
1126	20	6D	11 JSR 116D	1188	A0	o7 LDY #o7
1129	AD	F6	17 LDA 17F6	118A	84	F2 STY F2
112C	20	6D	11 JSR 116D	118C	A0	o2 LDY #o2
112F	20	EC	17 JSR 17EC	118E	84	F3 STY F3
1132	20	6D	11 JSR 116D	1190	BE	BE 11 LDX 11BE,Y
1135	20	EA	19 JSR 19EA	1193	48	PHA
1138	AD	ED	17 LDA 17ED	1194	2C	47 17 BIT 1747
113B	CD	F7	17 CMP 17F7	1197	10	FB BPL 1194
113E	AD	EE	17 LDA 17EE	1199	B9	BF 11 LDA 11BF,Y
1141	ED	F8	17 SBC 17F8	119C	8D	44 17 STA 1744
1144	90	E9	BCC 112F	119F	A5	F5 LDA F5
1146	A9	2F	LDA #2F	11A1	49	80 EOR #80
1148	20	88	11 JSR 1188	11A3	8D	42 17 STA 1742
114B	AD	E7	17 LDA 17E7	11A6	85	F5 STA F5
114E	20	70	11 JSR 1170	11A8	CA	DEX
1151	AD	E8	17 LDA 17E8	11A9	Do	E9 BNE 1194
1154	20	70	11 JSR 1170	11AB	68	PLA
1157	A2	02	LDX #02	11AC	C6	F3 DEC F3
1159	A9	04	LDA #04	11AE	F0	05 BEQ 11B5
115B	20	61	11 JSR 1161	11B0	30	07 BMI 11B9
115E	4C	F9	11 JMP 11F9	11B2	4A	LSR
1161	86	F1	STX F1	11B3	90	DB BCC 1190
1163	48		PHA	11B5	A0	00 LDY #00
1164	20	88	11 JSR 1188	11B7	F0	D7 BEQ 1190
1167	68		PLA	11B9	C6	F2 DEC F2
1168	C6	F1	DEC F1	11BB	10	CF BPL 118C
116A	Do	F7	BNE 1163	11BD	60	RTS
116C	60		RTS	11BE	02	***
116D	20	4C	19 JSR 194C	11BF	C3	***

#### 4.6.2 Disassembliertes Listing des KIM-Supermonitors

11C0	03	***	122E	30	EE	BMI	121E
11C1	7E	***	1230	95	E0	STA	E0, X
11C2	20	9F 12	1232	E0	0E	CP X	#0E
11C5	8D	F9 17	1234	10	E8	BPL	121E
11C8	20	9E 1E	1236	E8		INX	
11CB	4C	73 18	1237	A9	9E	LDA	#9E
11CE	20	2F 1E	1239	85	F4	STA	F4
11D1	20	1E 1E	123B	A9	13	LDA	#13
11D4	C8	INY	123D	85	F5	STA	F5
11D5	B1	FA	123F	A0	00	LDY	#00
11D7	20	76 12	1241	B1	F4	LDA	(F4), Y
11DA	20	9E 1E	1243	C9	FF	CMP	#FF
11DD	C8	INY	1245	D0	10	BNE	1257
11DE	B1	FA	1247	A0	03	LDY	#03
11E0	20	3B 1E	1249	B1	F4	LDA	(F4), Y
11E3	20	63 1F	124B	D9	E0 00	CMP	00E0, Y
11E6	AD	F7 17	124E	D0	03	BNE	1253
11E9	C5	FA	1250	C8		INY	
11EB	D0	07	1251	D0	F6	BNE	1249
11ED	AD	F8 17	1253	C9	FF	CMP	#FF
11F0	C5	FB	1255	F0	06	BEQ	125D
11F2	F0	05	1257	E6	F4	INC	F4
11F4	CA	DEX	1259	D0	E4	BNE	123F
11F5	D0	E3	125B	F0	CC	BEQ	1229
11F7	F0	D5	125D	A0	02	LDY	#02
11F9	20	8C 1E	125F	B1	F4	LDA	(F4), Y
11FC	4C	00 12	1261	85	EF	STA	EF
11FF	00	BRK	1263	88		DEY	
1200	A2	07	1264	B1	F4	LDA	(F4), Y
1202	BD	6E 12	1266	85	F0	STA	F0
1205	20	A0 1E	1268	20	9E 1E	JSR	1E9E
1208	CA	DEX	126B	6C	EF 00	JMP	(00EF)
1209	10	F7	126E	59	44 41	EOR	4144, Y
120B	AD	F5 17	1271	45	52	EOR	52
120E	85	FA	1273	20	20 20	JSR	2020
1210	AD	F6 17	1276	AA		TAX	
1213	85	FB	1277	A0	07	LDY	#07
1215	20	2F 1E	1279	8A		TXA	
1218	20	1E 1E	127A	39	88 12	AND	1288, Y
121B	20	9E 1E	127D	59	8F 12	EOR	128F, Y
121E	A2	0B	1280	F0	03	BEQ	1285
1220	A9	00	1282	88		DEY	
1222	95	E2	1283	D0	F4	BNE	1279
1224	CA	DEX	1285	BE	97 12	LDX	1297, Y
1225	D0	FB	1288	60		RTS	
1227	A2	03					
1229	20	5A 1E	1289	0C			
122C	C9	21	128A	1F			
			128B	0D	87 1F		

zu Abb. 4.6.2

128E FF				12F1 4C 15 12	JMP 1215
128F 03				12F4 20 2F 1E	JSR 1E2F
1290 0C				12F7 20 1E 1E	JSR 1E1E
1291 19 08 00				12FA 20 9E 1E	JSR 1E9E
1294 10 20				12FD 20 9F 12	JSR 129F
1296 03				1300 C9 FF	CMP #FF
1297 02				1302 D0 03	BNE 1307
1298 03				1304 4C 15 12	JMP 1215
1299 03				1307 91 FA	STA (FA),Y
129A 01 01				1309 20 76 12	JSR 1276
129C 02				130C 20 63 1F	JSR 1F63
129D 03				130F CA	DEX
129E 01				1310 F0 E2	BEQ 12F4
129F 20 5A 1E	JSR 1E5A			1312 20 9E 1E	JSR 1E9E
12A2 20 AC 1F	JSR 1FAC			1315 20 9F 12	JSR 129F
12A5 D0 F8	BNE 129F			1318 91 FA	STA (FA),Y
12A7 4C A3 1F	JMP 1FA3			131A 4C 0C 13	JMP 130C
12AA 20 9F 12	JSR 129F			131D 20 9F 12	JSR 129F
12AD 85 FB	STA FB			1320 AA	TAX
12AF 20 9F 12	JSR 129F			1321 AD F7 17	LDA 17F7
12B2 85 FA	STA FA			1324 85 EF	STA EF
12B4 4C 15 12	JMP 1215			1326 AD F8 17	LDA 17F8
12B7 20 9F 12	JSR 129F			1329 85 F0	STA F0
12BA 8D F9 17	STA 17F9			132B 38	SEC
12BD A9 02	LDA #02			132C A5 EF	LDA EF
12BF 8D F6 17	STA 17F6			132E E9 01	SBC #01
12C2 A9 00	LDA #00			1330 85 EF	STA EF
12C4 4C CE 12	JMP 12CE			1332 B0 02	BCS 1336
12C7 A5 FB	LDA FB			1334 C6 F0	DEC F0
12C9 8D F6 17	STA 17F6			1336 A0 00	LDY #00
12CC A5 FA	LDA FA			1338 B1 EF	LDA (EF),Y
12CE 8D F5 17	STA 17F5			133A 86 ED	STX ED
12D1 4C 0B 12	JMP 120B			133C A4 ED	LDY ED
12D4 A5 FB	LDA FB			133E 91 EF	STA (EF),Y
12D6 8D F8 17	STA 17F8			1340 A5 EF	LDA EF
12D9 A5 FA	LDA FA			1342 C5 FA	CMP FA
12DB 8D F7 17	STA 17F7			1344 D0 E5	BNE 132B
12DE 4C 0B 12	JMP 120B			1346 A5 F0	LDA F0
12E1 20 5A 1E	JSR 1E5A			1348 C5 FB	CMP FB
12E4 C8	INY			134A D0 F8	BNE 1344
12E5 C9 1B	CMP #1B			134C 18	CLC
12E7 F0 08	BEQ 12F1			134D AD F7 17	LDA 17F7
12E9 91 FA	STA (FA),Y			1350 65 ED	ADC ED
12EB 20 63 1F	JSR 1F63			1352 8D F7 17	STA 17F7
12EE 4C E1 12	JMP 12E1			1355 90 03	BCC 135A
				1357 EE F8 17	INC 17F8
				135A 4C 15 12	JMP 1215

135D 20 9F 12	JSR 129F	138D B0 03	BCS 1392
1360 AA	TAX	138F CE F8 17	DEC 17F8
1361 A5 FA	LDA FA	1392 4C 15 12	JMP 1215
1363 85 EF	STA EF		
1365 A5 FB	LDA FB	1395 C8	INY
1367 85 F0	STA F0	1396 B1 FA	LDA (FA),Y
1369 86 ED	STX ED	1398 20 3B 1E	JSR 1E3B
136B A4 ED	LDY ED	139B 4C 15 12	JMP 1215
136D B1 EF	LDA (EF),Y		
136F A0 00	LDY 000		
1371 91 EF	STA (EF),Y	139E FF FF	
1373 E6 EF	INC EF	13A0 1C 4F 42 59 45 FF 1D C8	
1375 D0 02	BNE 1379	13A8 52 55 4E FF 11 00 53 41	
1377 E6 FB	INC FB	13B0 56 45 FF 11 C2 4F 4C 44	
1379 A5 F0	LDA F0	13B8 FF 12 B7 4E 45 57 FF 12	
137B CD F8 17	CMP 17F8	13C0 AA 4C 49 4E 45 FF 13 1D	
137E D0 E9	BNE 1369	13C8 49 4E 53 45 52 54 FF 13	
1380 A5 EF	LDA EF	13D0 5D 44 45 4C 45 54 45 FF	
1382 CD F7 17	CMP 17F7	13D8 12 C7 53 54 41 52 54 FF	
1385 D0 F7	BNE 137E	13E0 12 D4 45 4E 44 FF 12 F4	
1387 38	SEC	13E8 48 45 58 FF 12 E1 53 54	
1388 E5 ED	SBC ED	13F0 52 49 4E 47 FF 11 CE 4C	
138A 8D F7 17	STA 17F7	13F8 49 53 54 FF 13 95 3F FF	
		1400 14 ENDE	

zu Abb. 4.6.2

## 4.7 Baudot-Disassembler

Gewöhnliche Baudot-Fernschreiber, wie sie gebraucht recht preisgünstig zu haben sind, eignen sich ideal zum Erstellen einer „Hard Copy“ - also einer Programmauflistung auf Papier. Die hier abgedruckten Programme sind zum großen Teil ebenfalls so entstanden, und der Leser wird zugeben, daß das Schriftbild eines solchen Fernschreibers deutlich besser aussieht als das eines u.U. viel teureren Matrix-Druckers.

Ein Baudot-Ausgabe-Programm wurde hier bereits beschrieben; das nun vorgestellte Programm benutzt es ebenfalls, enthält darüber hinaus jedoch auch einen sehr komfortablen Disassembler und bietet die Möglichkeit, vom ASCII-Terminal aus die ausgedruckten Programmzeilen zu kommentieren. Der Programmausdruck erfolgt hierbei synchron auf dem Baudot-Fernschreiber und auf dem Terminal-Bildschirm.



Das Programm besitzt drei Steuerfunktionen, die mit bestimmten ASCII-Zeichen eingeleitet werden:

- \* 1234      Springe zur Adresse 1234
- ↑ 5          Disassembliere 5 Zeilen (nicht 5 Bytes!)
- #8          Drucke eine Tabellenzeile mit 8 Bytes

Die hier angegebenen Zahlenwerte sind natürlich nur Beispiele. Die Byte- oder Zeilenzahl entsteht durch Abziehen von (hex) 30 von dem jeweiligen ASCII-Zeichen; dadurch kann man z.B. auch eine Tabellenzeile mit 16 Bytes ausgeben, indem man einfach # @ drückt.

Zum Hinzufügen von Kommentaren nach jeder disassemblierten Zeile besitzt das Programm eine Formatierungsfunktion, die dafür sorgt, daß die Kommentare immer genau an der gleichen Stelle beginnen, d.h. nach dem mnemonischen Befehl werden noch entsprechend viele Leerräume bis zum Kommentarfeld erzeugt.

Es sei noch erwähnt, daß der Baudot-Fernschreiber hier ebenso an den KIM-1 anzuschließen ist, wie das bereits bei dem Baudot-Ausgabe-Unterprogramm dargestellt wurde. Abb. 4.7 gibt das Programm als „Hex-Dump“ wieder.

MINI BAUDOT DISASSEMBLER																							
0400	20	8C	1E	A9	1A	85	E9	20	1B	04	20	F6	04	85	E4	84							
0410	E5	20	2C	07	A5	E9	10	F9	4C	F8	06	20	DC	04	A1	E4							
0420	A8	4A	90	0B	4A	B0	17	C9	22	F0	13	29	07	09	80	4A							
0430	AA	BD	20	05	B0	04	4A	4A	4A	4A	29	0F	Do	04	A0	80							
0440	A9	00	AA	BD	64	05	85	E0	29	03	85	E1	98	29	8F	AA							
0450	98	A0	03	E0	8A	F0	0B	4A	90	08	4A	4A	09	20	88	Do							
0460	FA	C8	88	Do	F2	48	B1	E4	20	05	05	A2	01	20	ED	04							
0470	C4	E1	C8	90	F1	A2	03	Co	03	90	F2	68	A8	B9	7E	05							
0480	85	E2	B9	BE	05	85	E3	A9	00	A0	05	06	E3	26	E2	2A							
0490	88	Do	F8	69	3F	20	0D	05	CA	Do	EC	20	EB	04	A2	06							
04A0	E0	03	Do	12	A4	E1	F0	0E	A5	E0	C9	E8	B1	E4	B0	1C							
04B0	20	05	05	88	Do	F2	06	E0	90	0E	BD	71	05	20	0D	05							
04C0	BD	77	05	F0	03	20	0D	05	CA	Do	D5	60	20	F9	04	AA							
04D0	E8	Do	01	C8	98	20	05	05	8A	4C	00	07	20	20	07	A5							
04E0	E5	A6	E4	20	D5	04	A2	01	4C	ED	04	A2	01	A9	20	20							
04F0	0D	05	CA	Do	F8	60	A5	E1	38	A4	E5	AA	10	01	88	65							

zu Abb. 4.7

```

0500 E4 90 01 C8 60 84 E8 20 00 07 A4 EB 60 84 E8 20
0510 45 07 A4 E8 60 60 B2 32 B3 33 32 B3 B3 A3 36 A2
0520 40 02 45 03 D0 08 40 09 30 22 45 33 D0 08 40 09
0530 40 02 45 33 D0 08 40 09 40 02 45 B3 D0 08 40 09
0540 00 22 44 33 D0 8C 44 00 11 22 44 33 D0 8C 44 9A
0550 10 22 44 33 D0 08 40 09 10 22 44 33 D0 08 40 09
0560 62 13 78 A9 00 21 01 02 00 80 59 4D 11 12 06 4A
0570 05 1D 2C 29 2C 23 28 41 59 00 58 00 00 00 1C 8A
0580 1C 23 5D 8B 1B A1 9D 8A 1D 23 9D 8B 1D A1 00 29
0590 19 AE 69 A8 19 23 24 53 1B 23 24 53 19 A1 00 1A
05A0 5B 5B A5 69 24 24 AE AE A8 AD 29 00 7C 00 15 9C
05B0 6D 9C A5 69 29 53 84 13 34 11 A5 69 23 A0 D8 62
05C0 5A 48 26 62 94 88 54 44 C8 54 68 44 E8 94 00 B4
05D0 08 84 74 B4 28 6E 74 F4 CC 4A 72 F2 A4 8A 00 AA
05E0 A2 A2 74 74 72 44 68 B2 32 B2 00 22 00 1A 1A
05F0 26 26 72 72 88 C8 C4 CA 26 48 44 44 A2 C8 00 00

0600 D8 78 20 60 06 20 5A 1E C9 5E D0 03 4C ED 06 C9
0610 23 F0 04 4C 49 06 00 20 5A 1E 3B E9 31 48 20 20
0620 07 68 AA A5 E5 20 00 07 A5 E4 20 00 07 A9 20 20
0630 45 07 A0 00 B1 E4 20 00 07 20 42 06 CA 10 EE 4C
0640 05 06 E6 E4 D0 02 E6 E5 60 C9 2A F0 06 20 52 07
0650 4C 05 06 20 9D 1F 85 E5 20 9D 1F 85 E4 4C 32 07
0660 A2 00 BD 70 06 D0 01 60 20 A0 1E E8 4C 62 06 00
0670 0D 0D 0A 0A 53 74 65 75 65 72 62 65 66 65 68 6C
0680 65 3A 0D 0D 0A 0A 2A 31 32 33 34 20 20 20 41 6B
0690 74 75 65 6C 6C 65 20 41 64 72 65 73 73 65 20 3D
06A0 20 31 32 33 34 0D 0D 0A 23 4E 20 20 20 20 20 20
06B0 54 61 62 65 6C 6C 65 6E 7A 65 69 6C 65 20 6D 69
06C0 74 20 4E 20 42 79 74 65 73 0D 0D 0A 5E 4E 20 20
06D0 20 20 20 20 44 69 73 61 73 73 65 6D 62 6C 69 65
06E0 72 65 20 4E 20 5A 65 69 6C 65 6E 20 00 20 5A 1E
06F0 38 E9 31 85 F8 4C 00 04 C6 F8 10 F9 4C 05 06 00

0700 85 FC 4A 4A 4A 4A 20 11 07 A5 FC 20 11 07 A5 FC
0710 60 29 0F C9 0A 18 30 02 69 07 69 30 4C 45 07 00
0720 A2 07 BD D5 1F 20 45 07 CA 10 F7 60 A9 20 20 0D
0730 05 60 20 2F 1E A5 E5 20 3B 1E A5 E4 20 3B 1E 20
0740 2F 1E 4C 05 06 86 F5 85 FF 20 A0 1E A5 FF F0 02
0750 C6 E9 A2 FF 8E 01 17 C9 0D D0 04 A9 08 D0 0E C9
0760 0A D0 04 A9 02 D0 06 29 3F AA BD C0 07 85 F4 29
0770 20 C5 F3 F0 0E 85 F3 A8 F0 04 A9 1B D0 02 A9 1F
0780 20 8D 07 A5 F4 20 8D 07 A0 FF A6 F5 60 A2 00 8E
0790 00 17 20 B4 07 A0 05 4A 90 04 A2 FF D0 02 A2 00
07A0 8E 00 17 20 B4 07 88 D0 EE A2 FF 8E 00 17 A2 1E
07B0 20 B6 07 60 A2 14 8E 07 17 2C 07 17 10 FB 60 00
07C0 00 03 19 0E 09 01 0D 1A 14 06 0B 0F 12 1C 0C 18
07D0 16 17 0A 05 10 07 1E 13 1D 15 11 2D 04 3A 04 04
07E0 04 34 04 29 00 24 31 25 2F 32 39 31 2C 23 3C 3D
07F0 36 37 33 21 2A 30 35 27 26 38 2E 2B 24 3E 24 39

START: 0600
BAUDOT OUT: PA 0...7

```

4.7 Programm zur Software-Dokumentation mit einem Baudot-Fernschreiber. Zur Eingabe der Steuerbefehle wird ein ASCII-Terminal benötigt

## 4.8 Ein Baudot-Fernschreibprogramm

### 4.8.1 Zweck des Programms

Das Programm, das als letztes Anwendungsbeispiel in *Abb. 4.8* vorgestellt wird, ist zwar nicht besonders umfangreich - es paßt immerhin in den Mikrocomputer KIM-1, ohne eine Speichererweiterung vorauszusetzen - lastet den Prozessor aber bis dicht an die Grenze seiner Leistungsfähigkeit aus. Leider läßt es sich nicht ohne weiteres auf den AIM-65 übertragen.<sup>1)</sup>

Das Baudot-Fernschreibprogramm dient dazu, Fernschreibzeichen als fertiges Nf-FSK-Signal auszugeben (FSK = Frequency Shift Keying) und so die Übertragung z.B. über ein gewöhnliches Sprechfunkgerät zu ermöglichen. In Amateurfunk-Kreisen wird dieses Verfahren als „RTTY“ bzw. „Radio Teletype“ bezeichnet. Den Signalwerten 0 und 1 werden hierbei die beiden Tonfrequenzen 1275 Hz und 2125 Hz zugeordnet. Die Übertragung geschieht im 5-bit-Baudot-Code.

Der Grund für die Auslastung des Prozessors ergibt sich aus den drei Aufgaben, die er hier mehr oder weniger gleichzeitig erledigen muß:

1. Vom ASCII-Terminal sind asynchron Zeichen mit 600 Bd Übertragungsgeschwindigkeit zu holen.
2. Der ASCII- ist in den Baudot-Code umzuwandeln und in einen Zwischenspeicher einzubringen.
3. Die Baudot-Zeichen sollen als fertiges Nf-FSK-Signal über einen I/O-Port ausgegeben werden.

Das vorliegende Programm benützt wieder einen periodischen Timer-Interrupt, der die doppelte Frequenz des zu erzeugenden Nf-Signales aufweist. Bei jedem Interrupt wird der Zustand des Nf-Ausgangs-Ports umgeschaltet. Die komplette Baudot-Ausgabe-Routine ist innerhalb des Interrupt-Programms untergebracht. Sie holt Zeichen für Zeichen aus dem Pufferspeicher, schiebt es

1) Die Zeitschrift FUNKSCHAU veröffentlichte in den Heften 15 und 16/1979 eine an den AIM-65 adaptierte Programmversion.



```

                                BAUDOT-RTTY
0000 A9 05 8D FA 17 A9 01 8D FB 17 8D 01 17 A9 00 D8
0010 78 8D 48 02 85 E5 85 E3 A9 00 8D F3 17 A9 61 8D
0020 F2 17 A9 2E 85 E2 8D 0D 17 20 5A 1E C9 1B Do 03
0030 4C 00 02 20 39 00 4C 29 00 C9 0D Do 04 A9 08 Do
0040 0E C9 0A Do 04 A9 02 Do 06 29 3F AA BD 70 01 85
0050 F4 29 20 C5 F3 Fo 0E 85 F3 A8 Fo 04 A9 1B Do 02
0060 A9 1F 20 45 01 A5 F4 20 45 01 60 4C 00 00 A9 00
0070 85 F3 A0 04 A9 08 2C 02 17 Do FB A2 1E 20 5C 01
0080 2C 02 17 18 Fo 01 38 66 FE 20 5A 01 88 10 F1 A5
0090 FE 4A 4A 4A Fo D5 C9 04 Do 04 A9 20 Do 06 C9 02
00A0 Do 04 A9 0A Do 06 C9 08 Do 04 A9 0D Do 1F C9 1B
00B0 Do 04 A9 20 Do BA C9 1F Fo B4 A2 40 05 F3 DD 70
00C0 01 Fo 03 CA Do F8 8A C9 20 10 02 09 40 20 A0 1E
00D0 4C 72 00
0105 48 A5 E2 8D 0D 17 EE 00 17 2C 47 17 30 02 68 40
0115 A5 E0 Fo 14 38 66 E1 A9 1A B0 02 A9 2E 85 E2 A9
0125 15 8D 47 17 C6 E0 10 E6 A9 08 85 E0 86 E4 A6 E3
0135 BD 48 02 Fo 05 E8 86 E3 85 E1 A6 E4 4C 13 01 00
0145 86 F5 A6 E5 0A 09 C0 9D 48 02 E8 A9 00 9D 48 02
0155 86 E5 A6 F5 60 A2 15 8E 07 17 A2 74 8E F2 17 A2
0165 FF 8E 00 17 2C 07 17 10 FB 60 00 00 03 19 0E 09
0175 01 0D 1A 14 06 0B 0F 12 1C 0C 18 16 17 0A 05 10
0185 07 1E 13 1D 15 11 2D 00 3A 00 00 04 34 00 2C 00
0195 24 31 25 2F 32 39 31 2C 23 3C 3D 36 37 33 21 2A
01A5 30 35 27 26 38 2E 2B 24 3E 24 39 48 20 A0 1E 68
01B5 20 39 00 A6 FD A5 E3 18 E5 E5 4A Fo F8 60 00 00
0200 20 5A 1E 85 F9 A9 48 85 FA A9 03 85 FB A0 00 B1
0210 FA Do 03 4C 6E 00 C9 5E Fo 06 20 63 1F 18 90 ED
0220 20 63 1F B1 FA C5 F9 Do E4 20 63 1F A0 00 B1 FA
0230 Fo 02 C9 5E Fo 0F C9 0D Do 05 20 B0 01 A9 0A 20
0240 B0 01 18 90 E4 4C 29 00
TEXTLADE-PROGRAMM
0000 A9 48 85 FA A9 03 85 FB 20 2F 1E 20 5A 1E C9 0D
0010 Do 05 48 20 2F 1E 68 C9 1B Do 15 A9 00 A8 91 FA
0020 20 63 1F A5 FA 8D F7 17 A5 FB 8D F8 17 4C 4F 1C
0030 C9 08 Do 0E 38 A5 FA E9 01 85 FA B0 02 C6 FB 4C
0040 0B 00 A0 00 91 FA D1 FA Do 06 20 63 1F 4C 0B 00
0050 A2 00 BD 60 00 Do 03 4C 4F 1C 20 A0 1E E8 Do F2
0060 0D 0A 07 4D 45 4D 4F 52 59 20 4F 56 45 52 46 4C
0070 4F 57 00

```

#### 4.8 Programm für Funkfern schreiben im Baudot-Code. Die NMI-Leitung des KIM-1 ist mit dem Port PB 7 zu verbinden

Bit für Bit ins Carry und gibt je nach dem Carry-Zustand 1275 Hz oder 2125 Hz aus. Dafür werden beide KIM-Timer verwendet: Einer zur Erzeugung des Interrupts mit der doppelten Tonfrequenz und einer zur Bestimmung der Bitlänge des auszugebenden Baudot-Zeichens.

Eine besondere Schwierigkeit des Interrupt-Betriebes ist, daß sich die Zeitschleife für die ASCII-Eingabe-Routine (1E5A)



im Monitorprogramm ändert, da der periodische Interrupt dem parallel ablaufenden Hauptprogramm, das die vom Terminal kommenden Zeichen im Puffer speichert, ein wenig Zeit „stiehlt“. Dies erfordert eine Korrektur des in den Zellen 17F2 und 17F3 stehenden Wertes zur Verzögerung innerhalb der ASCII-Eingabe-Routine, solange ein periodischer Interrupt auftritt. Aus diesem Grunde erfordert die hier aufgelistete Programmversion eine Terminal-Geschwindigkeit von 600 Bd.

Die Baudot-Ausgabe erfolgt dagegen mit etwa 48 Bd, was einen Kompromiß zwischen 50 Bd (kommerzieller Verkehr) und 45,45 Bd (Amateurfunkverkehr) darstellt.

Der periodische Interrupt wird unterbrochen, sobald das Programm in den Empfangsmodus schaltet. Dies geschieht, sobald länger als etwa 132 ms kein „Mark“-Ton (log. 1, 2125 Hz) empfangen wird. Hierbei wird die Verzögerungszeit in den Zellen 17F2/17F3 erneut korrigiert und an den „normalen“ Wert für 600 Bd angepaßt.

#### 4.8.2 Notwendige Hardware

Um den Betrieb des RTTY-Programms zu ermöglichen, sind einige Hardware-Voraussetzungen zu erfüllen. Zunächst muß eine Verbindung zwischen der NMI-Leitung und dem Timer-Ausgang PB7 hergestellt werden, um den periodischen Timer-Interrupt zu ermöglichen. Der Nf-Ausgang, der z.B. mit dem Modulationseingang eines Senders verbunden werden kann, ist PA 0; und PB 3 ist schließlich der Baudot-Eingang, an den z.B. der Ausgang eines RTTY-Konverters angeschlossen werden kann, der aus dem FSK-Signal empfangsseitig wieder ein Digital-Signal rückgewinnt (vgl. FUNKSCHAU 1979, Heft 9).

#### 4.8.3 Inbetriebnahme des RTTY-Programms

Vor der Inbetriebnahme müssen noch die Standard-Texte geladen werden, die folgendes Format aufweisen:

↑A Text a    ↑B Text b    ↑C Text c . . . 00

Der nach oben weisende Pfeil ist das ASCII-Zeichen hex 5E; es dient zur Trennung der einzelnen Texte voneinander. A, B, C usw. sind Kennbuchstaben, mit denen später der gewünschte Text abzurufen ist. Zur Eingabe kann man das kleine Programm verwenden, das hier ebenfalls aufgelistet ist. Es speichert die Texte ab der Adresse 0348 ab, bis die Escape-Taste gedrückt wird; dann wird die Endadresse in die Zellen 17F7 und 17F8 gespeichert. Ebenso wird automatisch 00 an das Textende gesetzt. Leider paßt das Eingabe-Programm nicht gleichzeitig mit dem RTTY-Programm in den KIM-Speicher, es ist also erforderlich, die Texte (0348 . . . Endadresse) auf eine Kassette zu „retten“. Es sei noch erwähnt, daß man sich vorher überlegen sollte, ob am Anfang oder Ende eines Standardtextes ein CR/LF stehen sollte. Zur Eingabe dieses Doppelzeichens genügt es, nur die Return-Taste zu drücken; das System fügt dann selbst „Line Feed“ hinzu, und zwar sowohl beim Eingabe- als auch beim RTTY-Programm.

Dann kann man das RTTY-Programm und ggf. die Texte laden und an der Adresse 0000 starten. Sofort muß ein 2125-Hz-Ton an PA 0 erscheinen. Schreibt man auf der Tastatur, so entsteht ein FSK-Signal an PA 0. Standardtexte lassen sich durch Drücken von Escape, gefolgt von dem jeweiligen Kennbuchstaben, aussenden; sie werden zunächst mit 600 Bd in den Ausgabepuffer transferiert. Ist der Puffer voll, so reduziert sich die Transfargeschwindigkeit automatisch auf die Baudot-Ausgabegeschwindigkeit. (Das Terminal muß auf 600 Bd geschaltet sein.)

Drückt man zweimal nacheinander Escape, so schaltet sich das System in den Empfangsmodus und bleibt in diesem, bis länger als 132 ms kein Mark-Ton empfangen wird. Während des Empfangsmodus wird kein Ton an PA 0 erzeugt.

Ein besonderer Vorteil ist, daß man sofort an der Tastatur weiterschreiben kann, wenn ein Standard-Text mit 600 Bd in den Ausgabepuffer (und auf den Bildschirm des Terminals) transferiert wurde. Bei der Gegenstation entsteht so nämlich der Eindruck, daß sie es mit dem Weltmeister im Maschinenschreiben zu tun habe . . .

## 5 Literatur

### *Zur Einführung in die Mikrocomputer-Technik*

- Werner, H.: Mikrocomputer – Eine kleine Einführung.  
Serie in der FUNKSCHAU, beginnend in Heft 21/1978.
- Gößler, R.: Dem Mikrocomputer auf's Bit géschaut.  
Serie in der ELO, beginnend in Heft 6/1978, aufbauend  
auf einem Selbstbau-System mit dem Prozessor 2650.
- Pelka, H.: Was ist ein Mikroprozessor?  
RPB 82, Franzis-Verlag, München
- Forster, C.C.: Programming A Microcomputer (6502).  
Addison-Wesley Publications, M. Nedela, Markdorf/Bodensee.  
ISBN 0-201-01995-7.

### *Weiterführende Literatur*

- 65XX Micro Mag. Vierteljährlich erscheinendes Heft für  
6502-Benutzer, Hrsg. R. Löhr, Hansdorfer Str. 4,  
2070 Ahrensburg.
- FUNKSCHAU, Mikrocomputer-Rubrik. Erscheint 14tägig und  
bringt interessante Applikations-Programme, vor allem für  
den 6502; z.B. ein Assembler in Heft 5/1979.
- HOBBYCOMPUTER-Sonderhefte des Franzis-Verlages.  
Zahlreiche Systembeschreibungen, Hardware-Hinweise und  
Programme auch für den 6502.
- ASCI, Amateurfunk-Selbstbau-Computer-Information. Viertel-  
jährlich erscheinendes Heft für Mikrocomputer-interessierte  
Funkamateure.
- The First Book of KIM. Zahlreiche Spielprogramme für den  
„rohen“ KIM-1, Hrsg. Jim Butterfield, erhältlich bei MCDS  
Computersysteme, Luisenplatz 4, 6100 Darmstadt.

# Sachverzeichnis

## A

Adressenverschiebung 19  
Adressierungsarten 11  
A/D-Wandler 34  
AIM-65 14  
Amateurfunk 26, 90  
ASCII 7, 26, 48  
ASCII-Baudot-Tabelle 39  
ASCII-Eingabe 44

## B

Baudot-Ausgabe 37, 87  
Baudot-Fernschreibprogramm 90  
Binär-Dezimal-Umwandlung 49

## C

CMOS-Schalter 23

## D

Datensuche 69  
D/A-Wandler 34  
Debugger 56  
Demodulator 24  
Disassembler 58, 87  
Druckeransteuerung 41

## F

Fernschreiber-Ansteuerung 40  
Fernschreibprogramm 90  
Frequenzzähler 32  
FSK-Modem 23  
Funktionsgenerator 28

## H

Hardware-Relocation 19  
Hex-Code von ASCII-Zeichen 48  
Hypertape 51

## I

Indirekter Sprung 18  
Interrupt 12, 32, 43, 52, 90

## K

Karteiprogramm 69  
Kassettenaufzeichnung 22, 51, 70  
KIM-1 12

## M

Maskenfehler 18  
Metallpapier-Drucker 41  
Modulator 23  
Monitorprogramm 13, 78

## O

Operationscodes 10, 17

## P

Parity-Bit 7, 26  
PC-100 14  
Plotter 67  
Pseudo-Befehle 78

## R

Randausgleich 75  
Register 9, 56  
RS-232-Schnittstelle 19  
RTTY-Programm 90

## S

Single-Step 56  
Sinus-Wertetabelle 29  
Speicher-Oszilloskop 34, 67  
Speicherverschiebung 19  
Stackbereich 9  
Stopbits 7, 26, 37



Supermonitor 78  
SYM-1 14

**T**  
Text-Formatierung 75  
Timer 15, 45, 52, 91

**U**  
Uhrzeit-Anzeige 52

**V**  
V-24-Schnittstelle 19

**Z**  
Zusätzliche Befehle 17

## Feichtinger, Anwendungsbeispiele für den Mikroprozessor 6502

# 173

Doppelband

Mit einem Mikroprozessor läßt sich mehr anfangen, als nur komplizierte Spiele ablaufen zu lassen oder Waschmaschinen zu steuern. Das beweist eindeutig dieser Band. Dazu bietet er aber noch mehr: nämlich handfeste Programmieranweisungen für ein Programm nach eigenen Vorstellungen und Wünschen, aber auch Fertiges, das nur eingegeben werden muß.

Der Benutzer des Bandes gewinnt Verständnis für die Systematik des Programmierens in der 6502-Maschinensprache und erkennt wie viele Anwendungen, die bisher mit der Hardware ausgeführt wurden, elegant mit dem Mikroprozessor gelöst werden können.

Der Autor ist FUNKSCHAU-Redakteur. Seit etwa 1945 beschäftigt er sich mit der Mikrocomputer-Technik, speziell mit dem Prozessor 6502. Zu seinen Leidenschaften gehört es, auf kleinen, preiswerten Systemen umfangreiche und verwickelte Aufgabenstellungen zu lösen.

RPB

ISBN 3-7723-1731-6