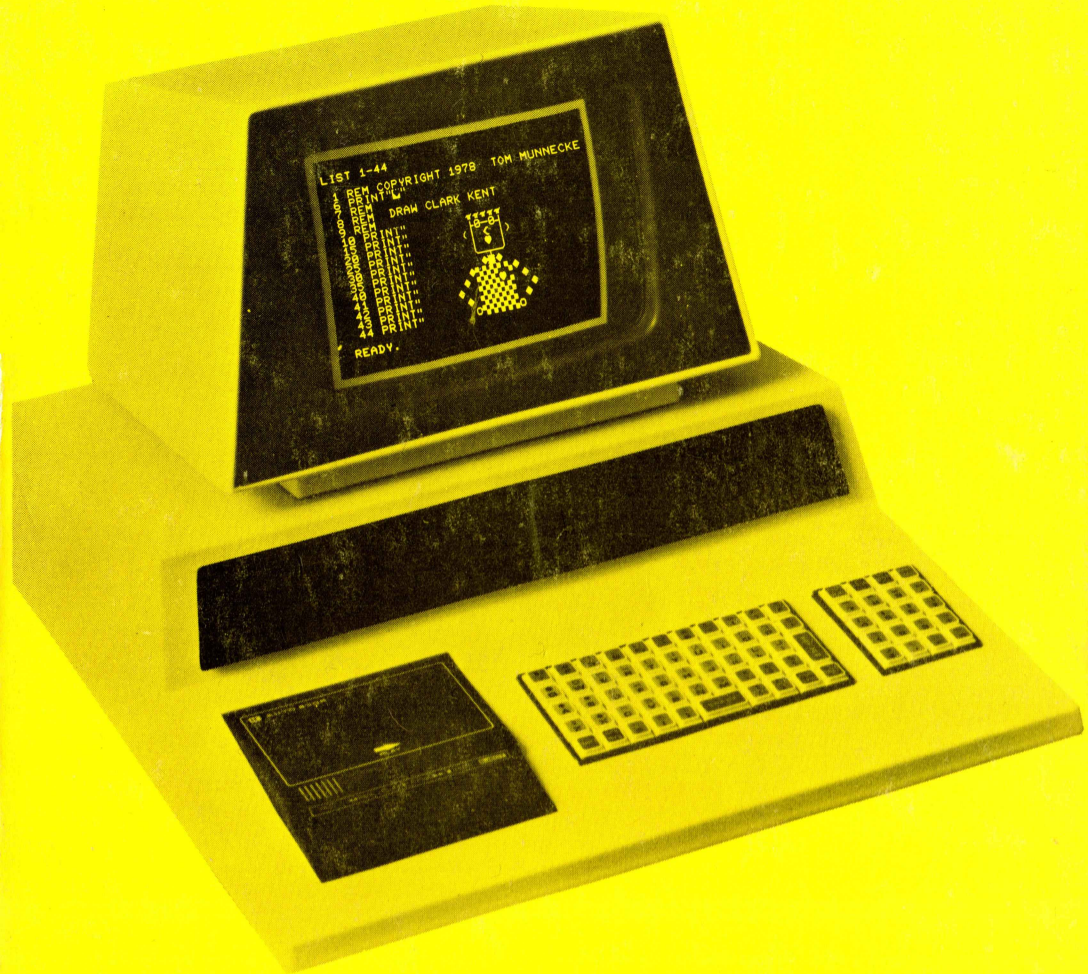
C. LORENZ



Programmier Handbuch

ISBN 3-921682-49-5

Es kann keine Gewähr dafür übernommen werden, daß die in diesem Buche verwendeten Angaben, Schaltungen, Warenbezeichnungen und Warenzeichen, sowie Programmlistings frei von Schutzrechten Dritter sind. Alle Angaben werden für die Amateurzwecke mitgeteilt. Alle Daten und Vergleichsangaben sind als unverbindliche Hinweise zu verstehen. Sie geben auch keinen Aufschluß über eventuelle Verfügbarkeit oder Liefermöglichkeit. In jedem Falle sind die Unterlagen der Hersteller zur Information heranzuziehen.

Nachdruck und öffentliche Wiedergabe, besonders die Übersetzung in andere Sprachen ist verboten. Programmlistings dürfen weiterhin nicht in irgendeiner Form vervielfältigt, verbreitet oder auf Datenträgern gespeichert werden. Auch das Kopieren für private Zwecke ist verboten. Alle Programmlistings sind Copyright der Fa. Ing. W. Hofacker GmbH.(Copyright C 1979). Verboten ist weiterhin die öffentliche Vorführung und Benutzung dieser Programme in Seminaren und Ausstellungen, Irrtum, sowie alle Rechte vorbehalten.

Copyright by Ing. W. HOFACKER GMBH, Postfach 75437, D-8000 MÜNCHEN 75

1. Auflage 1979

Gedruckt in der Bundesrepublik Deutschland – Printed in West Germany – Imprime' en RFA

PROGRAMMIER HANDBUCH FÜR PET *

* PET ist Warenzeichen der Fa. Commodore

Dieses Buch ist eine Produktion des Ing. W. Hofacker GmbH Verlages. Die Produktion erfolgte unabhängig von Commodore Business Machines und ihren weltweiten Niederlassungen.

Quellenverzeichnis und Literaturnachweis

1. Rockwell Datenbücher und Datenblätter, R6500 Microcomputer Systems Hardware Manual
2. R6500 Microcomputer System Programming Manual
3. Synertec Datenbuch
4. PET Communication with the Outside World
5. An Introduction to your new PET Personal Electronic Transactor von Commodore
6. Die Applikationsberichte von Commodore
7. Informationsbroschüre von Hewlett Packard Interface Bus
8. COMPUVOX und COMPUTONE User Manuals von Microsignal, Sacramento
9. Instruction Manual ADAK-1 PET Thinc. Inc., Futteron, Calif.
10. Programmierfiebel von MOS-Technology Inc.
11. First Book of KIM
12. Div. Manuals von Personal Software
13. Programmbeschreibungen von Creative Software
14. Programmbeschreibungen von Microtronix
15. Programmbeschreibungen von Computers One
16. The PAPER, PET Newsletter, Audubon
17. PET User Notes, Montgommeryville
18. PET User' s Group Newsletter, Berkeley

Anschriften:

Commodore Business Machines Inc.
901 California Avenue
PALO ALTO, California 94304
Tel.: 001 (415) 326-4000

in Deutschland:
Frankfurter Str. 171-175
6078 Neu-Isenburg

in der Schweiz:
Bahnhofstr. 29-31, 2. Stock
Postfach 666
CH-5001 Aarau

Rockwell International GmbH
Microelectronic Devices
Fraunhoferstr. 11
D-8030 München – Martinsried

* PET ist Warenzeichen der Fa. Commodore

Vorwort

Dieses Handbuch will dem Programmierer dort weiterhelfen, wo die mitgelieferten Broschüren aufhören. Programmiertechniken, Programmiertricks und Programmierexperimente sollen Sie bei eigenen Versuchen mit Ideen versorgen. Alle Zusammenhänge werden genau beschrieben und genau erklärt, warum es so funktioniert. Sie können dieses Buch als Handbuch zum PET oder auch zu jedem anderen 6502 Microcomputersystem verwenden. Im Anhang finden Sie eine Zusammenstellung der wichtigsten Datenblätter, sowie wichtige Tabellen, die Sie beim Programmieren von 6502 Systemen immer gebrauchen können. Eine Sammlung mit interessanten Programmierbeispielen soll Ihnen als Anregung und Ideengrundlage dienen.

Wir wünschen Ihnen beim Programmieren viel Erfolg!

München
Frühjahr 1979

C. Lorenz

Inhaltsverzeichnis

Programmierung in Maschinensprache

1. Grundsätzliches
Das Programmieren in Maschinensprache kann gerade beim PET sehr interessant sein. Als Vorteile lassen sich hier ohne lange nachzudenken die folgenden Punkte nennen:

1. Speicherplatzeinsparungen bis zu 50 %
2. Wesentlich schnellere Ausführung

Als Nachteile wären zu nennen, da es bekanntlich kaum etwas gibt, was nur Vorteile hat:

1. Etwas schwierigere Programmierung
2. Wenig gute Unterlagen

Nun, dem letzteren wollen wir heute etwas abhelfen. Nach dem Studium dieses Artikels sollen Sie in der Lage sein, eigene, kleine Programme in Maschinensprache zu schreiben und auch selbst austesten und abarbeiten können. Wir legen wieder den PET als Computer zugrunde. Grundsätzlich gilt jedoch die Ausführung auch für jeden anderen Maschinentyp oder Prozessor.

Wir wollen zuerst mit der einfachen Programmierung (zu Fuß) anfangen. Später wollen wir dann einige Handassemblierungen durchführen. Am Schluß sollen dann noch zwei Assemblerarten besprochen werden. Auch ein kleines Assemblerbeispiel wollen wir durchgehen. Mit den wichtigsten Grundlagen des Microcomputeraufbaues sollten Sie allerdings schon etwas vertraut sein.

Es werden unter den Benutzern von Heimcomputern neben anderen Gruppen zwei besonders wichtige Personenkreise vorhanden sein, die mit einem Personal-Computer arbeiten möchten.

1. Gruppe:
Diejenigen, die aus der EDV kommen und bisher nur BASIC, FORTRAN, COBOL etc. programmiert haben. Sie möchten etwas über Maschinenprogrammierung erfahren.

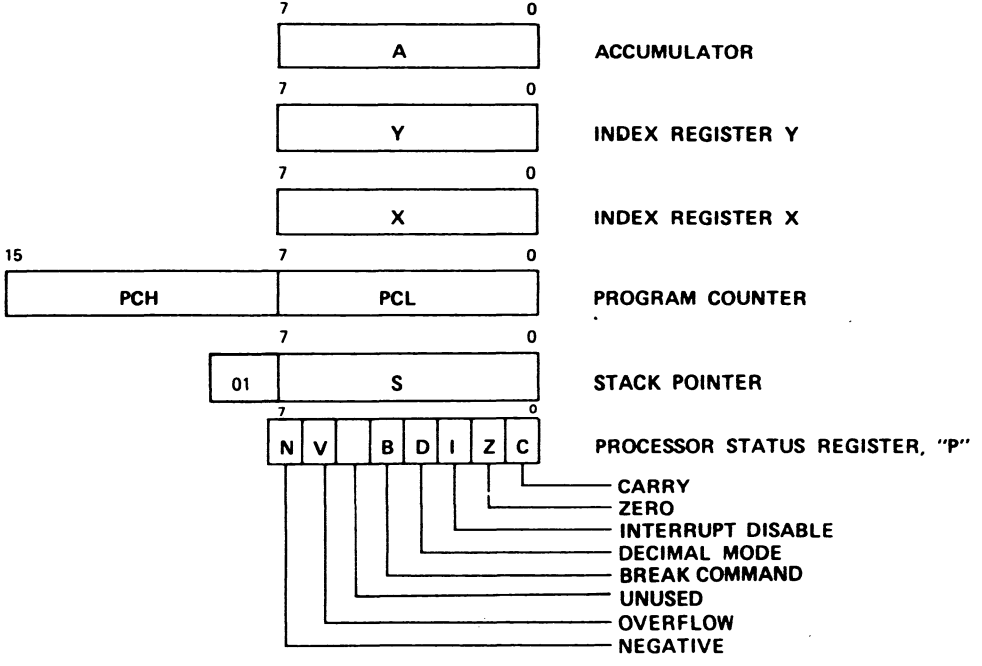
2. Gruppe
Der Anfänger, der noch kein BASIC programmiert hat und vielleicht Kenntnisse bereits von den Hexadezimal-Eingabe-Systemen (KIM, Z-80 KIT, SYM etc.) mitbringt. Jetzt aber in BASIC und Maschinensprache programmieren möchte.

Als Zusatzliteratur können wir Ihnen hier das Buch Nr. 109 (6502 Microcomputer, Aufbau und Programmierung v. Peter Heuer aus dem Hofacker Verlag, München) empfehlen. Hier wird auf über 250 Seiten nur über die Programmierung im Maschinencode berichtet.

R6500 Microprozessorbefehlssatz (alphabetische Reihenfolge)

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right One Bit (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

Programmiermodel für R6500



Adressierungsarten mit zugehörigen Ausführungszeiten in Taktzyklen

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC		2	3	4		4	4*	4*			6	5*	
AND		2	3	4		4	4*	4*			6	5*	
ASL	2		5	6		6	7						
BCC										2**			
BCS										2**			
BEQ										2**			
BIT			3			4							
BMI										2**			
BNE										2**			
BPL										2**			
BRK													
BVC										2**			
BVS										2**			
CLC									2				
CLD									2				
CLI									2				
CLV									2				
CMP		2	3	4		4	4*	4*			6	5*	
CPX		2	3			4							
CPY		2	3			4							
DEC			5	6		6	7						
DEX									2				
DEY									2				
EOR		2	3	4		4	4*	4*			6	5*	
INC			5	6		6	7						
INX									2				
INY									2				
JMP						3							5

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
JSR						6							
LDA		2	3	4		4	4*	4*			6	5*	
LDX		2	3		4	4		4*					
LDY		2	3	4		4	4*						
LSR	2		5	6		6	7						
NOP									2				
ORA		2	3	4		4	4*	4*			6	5*	
PHA									3				
PHP									3				
PLA									4				
PLP									4				
ROL	2		5	6		6	7						
ROR	2		5	6		6	7						
RTI									6				
RTS									6				
SBC		2	3	4		4	4*	4*			6	5*	
SEC									2				
SED									2				
SEI									2				
STA			3	4		4	5	5			6	6	
STX			3		4	4							
STY			3	4		4							
TAX									2				
TAY									2				
TSX									2				
TXA									2				
TXS									2				
TYA									2				

* Add one cycle if indexing across page boundary
** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

Befehle

nach

Hexadezimalzahlen

geordnet

ØØ - BRK
Ø1 - ORA - (Indirect,X)
Ø2 - Future Expansion
Ø3 - Future Expansion
Ø4 - Future Expansion
Ø5 - ORA - Zero Page
Ø6 - ASL - Zero Page
Ø7 - Future Expansion
Ø8 - PHP
Ø9 - ORA - Immediate
ØA - ASL - Accumulator
ØB - Future Expansion
ØC - Future Expansion
ØD - ORA - Absolute
ØE - ASL - Absolute
ØF - Future Expansion
1Ø - BPL
11 - ORA - (Indirect),Y
12 - Future Expansion
13 - Future Expansion
14 - Future Expansion
15 - ORA - Zero Page,X
16 - ASL - Zero Page,X
17 - Future Expansion
18 - CLC
19 - ORA - Absolute,Y
1A - Future Expansion
1B - Future Expansion
1C - Future Expansion
1D - ORA - Absolute,X
1E - ASL - Absolute,X
1F - Future Expansion

2Ø - JSR
21 - AND - (Indirect,X)
22 - Future Expansion
23 - Future Expansion
24 - BIT - Zero Page
25 - AND - Zero Page
26 - ROL - Zero Page
27 - Future Expansion
28 - PLP
29 - AND - Immediate
2A - ROL - Accumulator
2B - Future Expansion
2C - BIT - Absolute
2D - AND - Absolute
2E - ROL - Absolute
2F - Future Expansion
3Ø - BMI
31 - AND - (Indirect),Y
32 - Future Expansion
33 - Future Expansion
34 - Future Expansion
35 - AND - Zero Page,X
36 - ROL - Zero Page,X
37 - Future Expansion
38 - SEC
39 - AND - Absolute,Y
3A - Future Expansion
3B - Future Expansion
3C - Future Expansion
3D - AND - Absolute,X
3E - ROL - Absolute,X
3F - Future Expansion

40 - RTI
41 - EOR - (Indirect,X)
42 - Future Expansion
43 - Future Expansion
44 - Future Expansion
45 - EOR - Zero Page
46 - LSR - Zero Page
47 - Future Expansion
48 - PHA
49 - EOR - Immediate
4A - LSR - Accumulator
4B - Future Expansion
4C - JMP - Absolute
4D - EOR - Absolute
4E - LSR - Absolute
4F - Future Expansion
50 - BVC
51 - EOR - (Indirect),Y
52 - Future Expansion
53 - Future Expansion
54 - Future Expansion
55 - EOR - Zero Page,X
56 - LSR - Zero Page,X
57 - Future Expansion
58 - CLI
59 - EOR - Absolute,Y
5A - Future Expansion
5B - Future Expansion
5C - Future Expansion
5D - EOR - Absolute,X
5E - LSR - Absolute,X
5F - Future Expansion

60 - RTS
61 - ADC - (Indirect,X)
62 - Future Expansion
63 - Future Expansion
64 - Future Expansion
65 - ADC - Zero Page
66 - ROR - Zero Page
67 - Future Expansion
68 - PLA
69 - ADC - Immediate
6A - ROR - Accumulator
6B - Future Expansion
6C - JMP - Indirect
6D - ADC - Absolute
6E - ROR - Absolute
6F - Future Expansion
70 - BVS
71 - ADC - (Indirect),Y
72 - Future Expansion
73 - Future Expansion
74 - Future Expansion
75 - ADC - Zero Page,X
76 - ROR - Zero Page,X
77 - Future Expansion
78 - SEI
79 - ADC - Absolute,Y
7A - Future Expansion
7B - Future Expansion
7C - Future Expansion
7D - ADC - Absolute,X
7E - ROR - Absolute,X
7F - Future Expansion

8Ø - Future Expansion
81 - STA - (Indirect,X)
82 - Future Expansion
83 - Future Expansion
84 - STY - Zero Page
85 - STA - Zero Page
86 - STX - Zero Page
87 - Future Expansion
88 - DEY
89 - Future Expansion
8A - TXA
8B - Future Expansion
8C - STY - Absolute
8D - STA - Absolute
8E - STX - Absolute
8F - Future Expansion
9Ø - BCC
91 - STA - (Indirect),Y
92 - Future Expansion
93 - Future Expansion
94 - STY - Zero Page,X
95 - STA - Zero Page,X
96 - STX - Zero Page,Y
97 - Future Expansion
98 - TYA
99 - STA - Absolute,Y
9A - TXS
9B - Future Expansion
9C - Future Expansion
9D - STA - Absolute,X
9E - Future Expansion
9F - Future Expansion

AØ - LDY - Immediate
A1 - LDA - (Indirect,X)
A2 - LDX - Immediate
A3 - Future Expansion
A4 - LDY - Zero Page
A5 - LDA - Zero Page
A6 - LDX - Zero Page
A7 - Future Expansion
A8 - TAY
A9 - LDA - Immediate
AA - TAX
AB - Future Expansion
AC - LDY - Absolute
AD - LDA - Absolute
AE - LDX - Absolute
AF - Future Expansion
BØ - BCS
B1 - LDA - (Indirect),Y
B2 - Future Expansion
B3 - Future Expansion
B4 - LDY - Zero Page,X
B5 - LDA - Zero Page,X
B6 - LDX - Zero Page,Y
B7 - Future Expansion
B8 - CLV
B9 - LDA - Absolute,Y
BA - TSX
BB - Future Expansion
BC - LDY - Absolute,X
BD - LDA - Absolute,X
BE - LDX - Absolute,Y
BF - Future Expansion

CØ - CPY - Immediate
C1 - CMP - (Indirect,X)
C2 - Future Expansion
C3 - Future Expansion
C4 - CPY - Zero Page
C5 - CMP - Zero Page
C6 - DEC - Zero Page
C7 - Future Expansion
C8 - INY
C9 - CMP - Immediate
CA - DEX
CB - Future Expansion
CC - CPY - Absolute
CD - CMP - Absolute
CE - DEC - Absolute
CF - Future Expansion
DØ - BNE
D1 - CMP - (Indirect),Y
D2 - Future Expansion
D3 - Future Expansion
D4 - Future Expansion
D5 - CMP - Zero Page,X
D6 - DEC - Zero Page,X
D7 - Future Expansion
D8 - CLD
D9 - CMP - Absolute,Y
DA - Future Expansion
DB - Future Expansion
DC - Future Expansion
DD - CMP - Absolute,X
DE - DEC - Absolute,X
DF - Future Expansion

EØ - CPX - Immediate
E1 - SBC - (Indirect,X)
E2 - Future Expansion
E3 - Future Expansion
E4 - CPX - Zero Page
E5 - SBC - Zero Page
E6 - INC - Zero Page
E7 - Future Expansion
E8 - INX
E9 - SBC - Immediate
EA - NOP
EB - Future Expansion
EC - CPX - Absolute
ED - SBC - Absolute
EE - INC - Absolute
EF - Future Expansion
FØ - BEQ
F1 - SBC - (Indirect),Y
F2 - Future Expansion
F3 - Future Expansion
F4 - Future Expansion
F5 - SBC - Zero Page,X
F6 - INC - Zero Page,X
F7 - Future Expansion
F8 - SED
F9 - SBC - Absolute,Y
FA - Future Expansion
FB - Future Expansion
FC - Future Expansion
FD - SBC - Absolute,X
FE - INC - Absolute,X
FF - Future Expansion

Programmieren in Maschinensprache
Hat man keinen Monitor für die Programmierung in Maschinensprache zur Hand, kann man die Maschinensprachenbefehle im (Hex-Code) über den Befehl POKE X, Y in den gewünschten Speicherbereich geben. Nachteil hierbei ist, daß der POKE-Befehl beim PET mit einer dezimalen Adresse und dem einzugebenden Wert hinter dem Komma in dezimal arbeitet.

Wir wollen also gleich mit einem Beispiel beginnen. Hierbei lernt man am besten die Zusammenhänge kennen.

Wir wollen ein Programm in Maschinensprache eingeben, welches eine Reihe von Speicherzellen zu Null macht.
Wir wählen für unsere Experimente jetzt vorerst den Zwischenspeicher für den zweiten Cassettenrecorder. Er liegt an den Adressen 033A bis 03F9 (dezimal 826 bis 1017)

033A	A900	START	LDA 0
033C	A209		LDX 09
033E	9D5003	LOOP	STA $ 350, X
0340	CA		DEX
0341	10FA		BPL
0343	60		RTS

Speicherauszug:
A9 00 A2 09 9D 50 03 CA 10 FA 60

Übersetzt in Dezimal:

169, 0, 162, 9, 157, 80, 3, 202, 16, 250, 96

11 Speicherzellen ab 826 entsprechen 826 – 836.

Die Zellen 0350 bis 0359 sollen mit Nullen angefüllt werden.
0350 Hex = 848
0359 Hex = 856

Nachfolgend sehen Sie jetzt eine Reihe von kleinen Programmen, die Ihnen:

1. Das Maschinenprogramm über DATA Statements in den gewünschten Speicherbereich poked (RUN 9000)

2. Mit RUN 9100 können Sie kontrollieren, ob es auch wirklich eingeschrieben wurde.
3. Mit RUN 9200 können Sie dann die Ergebnisse kontrollieren.

Das Maschinenprogramm im 2. Cassettenpuffer kann nun durch SYS (826) aufgerufen werden. 826 ist der Dezimalwert der hexadezimalen Anfangsadresse 033A.

Gesamtes Versuchsprogramm

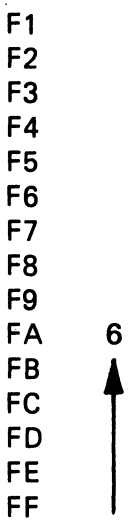
```
READY.

 10 SYS(826)
 15 PRINT PEEK(848)
 20 END
 9000 FOR A=826 TO 836
 9020 READ B
 9055 POKE A,B
 9057 NEXT A
 9060 DATA 169,0,162,9,157,80,3,202,16,250,96
 9070 END
 9100 FOR A =826 TO 837
 9110 PRINT PEEK(A)
 9120 NEXT A
 9130 END
 9200 FOR A=847 TO 866
 9210 PRINT PEEK(A)
 9220 NEXT A
 9230 END
 9500 POKE 1,58
 9510 POKE2,3
 9520 X=USR(Y)
 9530 END
READY.
```

Das oben aufgeführte Maschinenprogramm lädt uns jetzt die Werte 0 in die Speicherplätze 0350 Hex bis 0359 Hex im 2. Cassettenpuffer. Unser Programm liegt am Anfang des 2. Cassettenpuffers bei Adresse Hex. 033A. Dort in diesem Bereich von 033A bis 03F9 (191 Bytes) sind unsere Programme vor der Überschreibung durch BASIC geschützt. Wir sollten diesen für unsere kleinen Maschinenprogramme verwenden, sofern wir den zweiten Cassettenrecorder nicht benötigen. Wie man den anderen Teil des Speichers auch sicher verwenden kann, wollen wir Ihnen später zeigen.

Was macht unser Maschinenprogramm genau:
Wir haben hier zuerst einmal den Akkumulator mit 00 geladen. LDA0. Dann haben wir das X-Register auf 09 gesetzt. Jetzt laden wir die Nullen aus dem Akkumulator in die Speicherzelle 0350 Hex. Der Befehl STA $ 350,X bietet die Möglichkeit der indizierten Adressierung. Der Wert X (Wert 09 im X-Register) wird vom Computer verwendet, um eine effektive Adresse zu berechnen. Der Inhalt des X-Registers wird deshalb zur Adresse 350 dazugezählt. So beginnt also der Akku zuerst seine ersten beiden Nullen in Zelle 350+9 = 359 abzulegen. Der DEX-Befehl zählt das X-Register um eins herunter und wir erhalten so die nächste effektive Adresse 359. So wird nun das X-Register bis auf 00 heruntergezählt. (Dies geschieht über eine Schleife durch BPL. Hat das Register den Wert 00 überschritten, beginnt wieder FF. Diesen Wert sieht der 6502 Computer als negativ an und springt über den BPL-Test dann nach Zeile 0343 Hex.

Der BPL-Befehl ist ein bedingter Testbefehl, der so lange eine Verzweigung bestehen läßt, solange ein positiver Wert dort vorhanden ist, wo wir durch BPL hinspringen.

Der Wert (Opcode) nach BPL ergibt sich aus der Entfernung. Er errechnet sich in unserem Beispiel wie folgt:



Von 0341 10FA bis 9D5003 sind es insgesamt 6 Adressen. Von FF nach unten 6 abgezählt, ergibt den Wert FA. Er wird als Opcode hinter den BPL-Befehl gesetzt.

Der Befehl RTS (Return from Subroutine), Rückkehr aus dem Unterprogramm bringt uns wieder ins BASIC-Programm zurück. Wenn Sie einen BRK-Befehl noch vor den RTS-Befehl setzen, arbeitet zwar Ihr Programm, aber BASIC bringt die Fehlermeldung: „? ILLEGAL QUANTITY ERROR IN 10".

Wenn Sie diese kleine Programmierübung bis jetzt durchgeführt haben, sollten Sie noch einige kleine Übungen durchführen.

1. Ändern Sie das Maschinenprogramm so, daß in die Zellen 0350 bis 0359 die Zahl 44 eingeschrieben wird.
2. Ändern Sie das Programm so, daß der Wert 1 in Zellen geschrieben wird, die zum eigentlichen 8K Arbeitsspeicherbereich des PET gehören. (0400 – 1FFF)

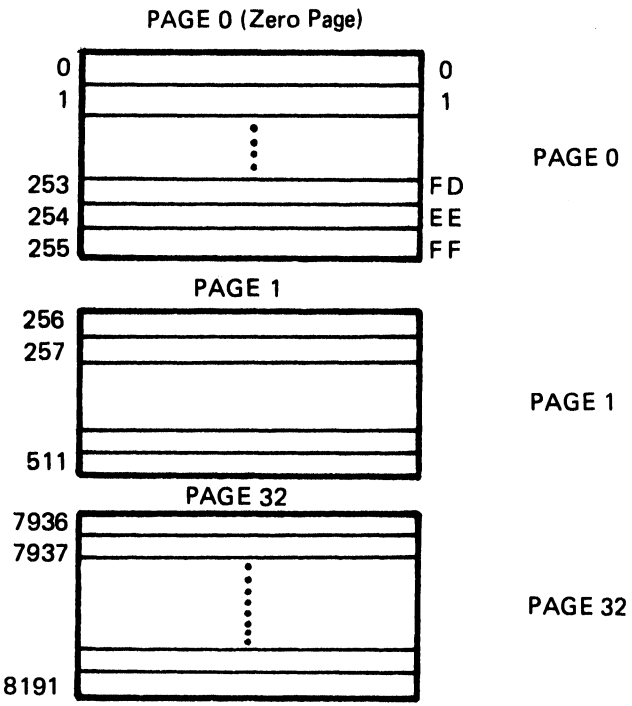
Die Übung 2 führt uns zu unserem nächsten Beispiel. Wie wir wissen, müssen wir unser Maschinensprachenprogramm, wenn wir es nicht an einer geschützten Stelle (z.B. 2. Cassettenpuffer) abspeichern, vor der Überschreibung durch das BASIC schützen. Die Speicherzelle 135 dezimal enthält im PET eine Adresse, die den verfügbaren Speicherraum nach oben begrenzt. Sehen Sie einmal durch ?PEEK (135) nach. Sie finden die Zahl 32. Dies bedeutet, daß der Speicherbereich Ihres PETs bei PAGE 32 endet.

Der Hauptspeicher (RAM) des PET ist in PAGES aufgeteilt, die sich wie folgt aufbauen.

32 Pages mit je 256 Worten Speicherplatz ergeben die 8000 Worte RAM (32 x 256 = 8192 = 8K)

Wenn sich der PET nach dem Einschalten meldet, zeigt er einen geringeren Wert. Dies liegt daran, daß ein kleiner Teil des RAM zur Speicherung von Zwischenwerten etc. reserviert wurde.

PAGE-Aufbau im PET



Wenn wir jetzt unser kleines Beispiel in den Hauptarbeitsbereich schreiben wollen, werden wir uns fragen, wo legen wir es hin? Jetzt müssen wir uns zuerst entscheiden, in welche PAGE. Wie weiß ich jetzt, wo welche Adresse in welcher PAGE liegt?
Ich habe z.B. die Dezimaladresse mit 7950 gewählt (7950=1F0EHex) In welcher PAGE liegt diese Adresse. Wir teilen zuerst einmal 7950 durch 256.

```
7950 : 256 = 31 Rest 14
7936
-------
  14
```

Die Zahl 31 gibt nun die PAGE an, wobei der Übertrag nur die Zeile 14 in der PAGE 31 angibt. PAGE 31 ist in Wirklichkeit die 32. PAGE, da wir bereits mit 0 zu zählen begannen.

Da wir unser kleines Programm jetzt in die PAGE 31 legen wollen, müssen wir diese PAGE vor der Überschreibung durch BASIC schützen. Dies geschieht über den Befehl:

```
POKE 135,31
```

Auf diese Weise haben wir nun ab Adresse 7936 Dez. = 1F00 Hex. den Speicher bis zu 1FFF zur Verfügung (256 Byte).
Versuchen Sie jetzt, in Übung zwei unser Maschinenprogramm ab Adresse 7936 abzuspeichern und über BASIC SYS (7936) aufzurufen und abzuarbeiten.

Beachten Sie bitte beim weiterarbeiten, daß die Bereichsgrenze so lange verbleibt, bis ein anderer Wert eingepoked wird oder das Gerät abgeschaltet wird.
Ähnlich verhält es sich mit dem Maschinenprogramm im zweiten Cassettenpuffer oder im geschützten Speicherbereich. Sie können neue BASIC-Programme einlesen, ohne sie zu zerstören. Es bleibt dort, bis Sie die Maschine ausschalten oder das Programm verändern.

Die USR (X) Funktion

Die USR(X)-Funktion aus dem Commodore BASIC arbeitet ähnlich wie der SYS(X) Befehl. Sie bringt die Ausführung des Computers an die Adresse, die in den beiden Speicherzellen 1 und 2 liegt. Aus diesem Grunde muß von dem USR(X)-Befehl die Anfangsadresse des Maschinenprogrammes in 1 und 2 eingepoked werden.

Beispiel:

```
50 POKE 1, höherwertiges Byte
60 POKE 2, niederwertiges Byte
```

Die Adresse 7950 würde z.B. wie folgt durch POKE eingegeben:

```
50 POKE 1,14
60 POKE 2,31
```

Wobei 14 das dezimale Äquivalent des oberen Adressbytes und 31 das dezimale Aquivalent des unteren Adressbytes ist. Gleichzeitig sind die beiden Zahlen auch Werte für Speicherzelle 14 in PAGE 31 (dezimal). Das Argument X in der Funktion USR(X) soll vom BASIC in das Maschinensprachenprogramm transportiert werden können.

Die USR-Funktion

Die USR-Funktion hängt mit dem direkten Zugriff zum Speicher und dem Prozessor zusammen.
Ähnliche Funktionen im PET sind: POKE, PEEK und SYS. USR ist im Grunde genommen eine Kombination aus diesen drei Funktionen. Auf jeden Fall gibt es nichts, was mit USR gemacht werden kann und mit PEEK, POKE oder SYS nicht simuliert werden könnte. Vom Sprachbereich her gesehen ist der USR-Befehl eine ähnliche Funktion wie INT oder SQR. USR hat ausschließlich numerische Bedeutung und kann in Berechnungen, PRINT-Befehlen und Zuordnungsanweisungen etc. verwendet werden. Der einzige Unterschied zwischen USR und den obengenannten Befehlen ist der, daß der Ausdruck USR(X) nicht unbedingt eine Funktion des Argumentes X sein muß.

Wenn die Funktion USR in einem BASIC-Programm ausgeführt wird, geschehen zwei Dinge:

1. Das Argument in den nachfolgenden Klammern USR(X) wird bewertet und im softwaresimulierten Akkumulator des PETs abgespeichert. Dieser befindet sich in den Speicherzellen $B1 — B5 (fünf Byte binäre Darstellung)

2. Der BASIC-Interpreter macht einen Sprung ins Unterprogramm (JSR) an den Speicherplatz, welcher durch den USR-Vektor bestimmt wird. Die Speicherplätze $01 (low) und $02 (high) enthalten diese Adresse (niederwertiger Teil der Adresse in $01 und das obere Adressbyte in $02)

Da in Zelle 00 ein unbedingter Sprungbefehl C4 steht, welcher bereits durch ein RESET-Signal bei der Initialisierung hier platziert wurde, wird der Unterprogrammaufruf von dieser Stelle aus erfolgen.

Nachdem dies geschehen ist, übernimmt das Unterprogramm in Maschinensprache die komplette Kontrolle über den PET. Obowhl noch weiterhin von allen möglichen Quellen hier Interrupts erscheinen, steht der PET voll und ganz dem Unterprogramm zur Verfügung. Wenn irgendetwas falsch läuft, kann der PET nur durch Ein-/Ausschalten wieder zum „Laufen" gebracht werden. Aus diesem Grunde wird demjenigen empfohlen, der sehr viel in Maschinensprache programmiert, sich einen RESET-Schalter nachträglich einzubauen.

Wenn das Unterprogramm wieder zurück an BASIC gegeben wird, ($ CED2 und $C6EE für SYS), welches durch den Befehl RTS geschehen kann, kann der Betrieb in BASIC weitergehen, als ob nichts geschehen wäre. Der Wert aus USR(X) ist der Wert, der auch dann im 5 Byte Akkumulator zurückbleibt, es sei denn, er würde vom Unterprogramm selbst verändert.

Unser Beispielprogramm zur Einschreibung der Nullen in den Speicher kann auch durch den Befehl USR ausgeführt werden. Das Programm sieht dann wie folgt aus:

```
9500 POKE 1,58
9510 POKE 2,3
9529 X = USR (Y)
9530 END
```

Sie können nachprüfen, ob es funktioniert, wenn Sie das gesamte Versuchsprogramm einlesen, RUN 9000 geben und dann RUN 9500 eingeben. Anschließend können Sie mit RUN 9200 kontrollieren, ob die Speicherinhalte verändert wurden.

Wir haben jetzt mit unserem PET in Maschinensprache programmiert und alles ohne ein zusätzliches Hilfsmittel erledigt. Im nachfolgenden Teil wollen wir einen Monitor zur Erleichtung dieser Arbeiten verwenden.

USR (Funktion) Beispiel

0000	4C	3A	03
033A	20	A7	D0
33D	A5	B3	
33F	A6	B4	
341	85	B4	
343	86	B3	
345	A2	00	
347	A1	B3	
49	A8		
34A	8A		
034B	4C	78	D2

X = USR(I)
$-32768 \leq I \leq 32767$
$0 \leq X \leq 255$

Bringt den Inhalt des Bytes, dessen Adresse in I spezifiziert ist.

```
10000   DATA 32, 167, 208, 165, 179, 166, 180, 133
10100   DATA 180, 134, 179, 162, 0, 161, 179, 168
10200   DATA 138, 76, 120, 210
10300   FOR I = 826 TO 845
10400   READ N : POKE I,N
10500   NEXT
10600   POKE 1,58
10700   POKE 2,3
```

Befehlsliste

in

alphabetischer

Reihenfolge

nach

mnemonischen

Befehlen

geordnet

The following notation applies to this summary:

A	Accumulator
X, Y	Index Registers
M	Memory
P	Processor Status Register
S	Stack Pointer
✓	Change
—	No Change
+	Add
∧	Logical AND
-	Subtract
∀	Logical Exclusive Or
↑	Transfer from Stack
↓	Transfer to Stack
→	Transfer to
←	Transfer to
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	Operand
#	Immediate Addressing Mode

Note: Shown in parentheses at the top of each table a reference number (Ref: XX) which directs the user to the particular Section in the R6500 Microcomputer Family Programming Manual in which the instruction is defined and discussed.

ADC

Add memory to accumulator with carry

ADC

Operation: A + M + C → A, C

N Z C I D V
✓ ✓ ✓ — — ✓

(Ref: 2.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	ADC # Oper	69	2	2
Zero Page	ADC Oper	65	2	3
Zero Page, X	ADC Oper, X	75	2	4
Absolute	ADC Oper	6D	3	4
Absolute, X	ADC Oper, X	7D	3	4*
Absolute, Y	ADC Oper, Y	79	3	4*
(Indirect, X)	ADC (Oper, X)	61	2	6
(Indirect), Y	ADC (Oper), Y	71	2	5*

* Add 1 if page boundary is crossed.

AND

"AND" memory with accumulator

AND

Logical AND to the accumulator

Operation: A ∧ M → A

N Z C I D V
✓ ✓ — — — —

(Ref: 2.2.4.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	AND # Oper	29	2	2
Zero Page	AND Oper	25	2	3
Zero Page, X	AND Oper, X	35	2	4
Absolute	AND Oper	2D	3	4
Absolute, X	AND Oper, X	3D	3	4*
Absolute, Y	AND Oper, Y	39	3	4*
(Indirect, X)	AND (Oper, X)	21	2	6
(Indirect), Y	AND (Oper), Y	31	2	5*

* Add 1 if page boundary is crossed.

ASL

ASL *Shift Left One Bit (Memory or Accumulator)*

ASL

Operation: C ← [7|6|5|4|3|2|1|Ø] ←Ø

N	Z	C	I	D	V
✓	✓	✓	–	–	–

(Ref: 10.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ASL A	ØA	1	2
Zero Page	ASL Oper	Ø6	2	5
Zero Page, X	ASL Oper, X	16	2	6
Absolute	ASL Oper	ØE	3	6
Absolute, X	ASL Oper, X	1E	3	7

BCC

BCC *Branch on Carry Clear*

BCC

Operation: Branch on C = Ø

N	Z	C	I	D	V
–	–	–	–	–	–

(Ref: 4.1.2.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BCC Oper	9Ø	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BCS

BCS *Branch on carry set*

BCS

Operation: Branch on C = 1

N Z C I D V
- - - - - -

(Ref: 4.1.2.4)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BCS Oper	B0	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to next page.

BEQ

BEQ *Branch on result zero*

BEQ

Operation: Branch on Z = 1

N Z C I D V
- - - - - -

(Ref: 4.1.2.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BEQ Oper	F0	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to next page.

BIT

BIT *Test bits in memory with accumulator* **BIT**

Operation: $A \wedge M$, $M_7 \rightarrow N$, $M_6 \rightarrow V$

Bit 6 and 7 are transferred to the status register. N Z C I D V

If the result of $A \wedge M$ is zero then Z = 1, otherwise M_7 ✓ – – – M_6

Z = Ø

(Ref: 4.2.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	BIT Oper	24	2	3
Absolute	BIT Oper	2C	3	4

BMI

BMI *Branch on result minus* **BMI**

Operation: Branch on N = 1 N Z C I D V

– – – – – –

(Ref: 4.1.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BMI Oper	3Ø	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BNE

BNE *Branch on result not zero*

BNE

Operation: Branch on Z = 0

N ⇕ C I D V

– – – – – –

(Ref: 4.1.2.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BNE Oper	DØ	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BPL

BPL *Branch on result plus*

BPL

Operation: Branch on N = Ø

N ⇕ C I D V

– – – – – –

(Ref: 4.1.2.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BPL Oper	1Ø	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BRK

BRK *Force Break*

Operation: Forced Interrupt PC + 2 ↓ P ↓

N Z C I D V
– – – 1 – –

(Ref: 9.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	BRK	ØØ	1	7

1. A BRK command cannot be masked by setting I.

BVC

BVC *Branch on overflow clear*

Operation: Branch on V = 0

N Z C I D V
– – – – – –

(Ref: 4.1.2.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BVC Oper	5Ø	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

BVS

BVS *Branch on overflow set*

BVS

Operation: Branch on V = 1

N Z C I D V
- - - - - -

(Ref: 4.1.2.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BVS Oper	7Ø	2	2*

* Add 1 if branch occurs to same page.

* Add 2 if branch occurs to different page.

CLC

CLC *Clear carry flag*

CLC

Operation: Ø → C

N Z C I D V
- - Ø - - -

(Ref: 3.0.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLC	18	1	2

CLD

CLD *Clear decimal mode*

CLD

Operation: Ø → D

N Z C I D V
– – – – Ø –

(Ref: 3.3.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLD	D8	1	2

CLI

CLI *Clear interrupt disable bit*

CLI

Operation: Ø → I

N Z C I D V
– – – Ø – –

(Ref: 3.2.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLI	58	1	2

CLV

CLV *Clear overflow flag*

CLV

Operation: Ø → V

N Z C I D V

– – – – – Ø

(Ref: 3.6.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLV	B8	1	2

CMP

CMP *Compare memory and accumulator*

CMP

Operation: A - M

N Z C I D V

✓ ✓ ✓ – – –

(Ref: 4.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CMP #Oper	C9	2	2
Zero Page	CMP Oper	C5	2	3
Zero Page, X	CMP Oper, X	D5	2	4
Absolute	CMP Oper	CD	3	4
Absolute, X	CMP Oper, X	DD	3	4*
Absolute, Y	CMP Oper, Y	D9	3	4*
(Indirect, X)	CMP (Oper, X)	C1	2	6
(Indirect), Y	CMP (Oper), Y	D1	2	5*

* Add 1 if page boundary is crossed.

CPX

CPX *Compare Memory and Index X*

CPX

Operation: X - M

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 7.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPX #Oper	EØ	2	2
Zero Page	CPX Oper	E4	2	3
Absolute	CPX Oper	EC	3	4

CPY

CPY *Compare memory and index Y*

CPY

Operation: Y - M

N Z C I D V
✓ ✓ ✓ – – –

(Ref: 7.9)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPY #Oper	CØ	2	2
Zero Page	CPY Oper	C4	2	3
Absolute	CPY Oper	CC	3	4

DEC

DEC *Decrement memory by one*

DEC

Operation: M - 1 → M

N Z C I D V

✓ ✓ – – – –

(Ref: 10.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	DEC Oper	C6	2	5
Zero Page, X	DEC Oper, X	D6	2	6
Absolute	DEC Oper	CE	3	6
Absolute, X	DEC Oper, X	DE	3	7

DEX

DEX *Decrement index X by one*

DEX

Operation: X - 1 → X

N Z C I D V

✓ ✓ – – – –

(Ref: 7.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	DEX	CA	1	2

DEY

DEY *Decrement index Y by one*

Operation: Y - 1 → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	DEY	88	1	2

EOR

EOR *"Exclusive-Or" memory with accumulator*

Operation: A ⊻ M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.2.4.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	EOR #Oper	49	2	2
Zero Page	EOR Oper	45	2	3
Zero Page, X	EOR Oper, X	55	2	4
Absolute	EOR Oper	4D	3	4
Absolute, X	EOR Oper, X	5D	3	4*
Absolute, Y	EOR Oper, Y	59	3	4*
(Indirect, X)	EOR (Oper, X)	41	2	6
(Indirect),Y	EOR (Oper), Y	51	2	5*

* Add 1 if page boundary is crossed.

INC

INC *Increment memory by one*

INC

Operation: M + 1 → M

N Z C I D V
✓ ✓ – – – –

(Ref: 10.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	INC Oper	E6	2	5
Zero Page, X	INC Oper, X	F6	2	6
Absolute	INC Oper	EE	3	6
Absolute, X	INC Oper, X	FE	3	7

INX

INX *Increment Index X by one*

INX

Operation: X + 1 → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.4)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	INX	E8	1	2

INY

INY *Increment Index Y by one*

INY

Operation: Y + 1 → Y

N Z C I D V

✓ ✓ – – – –

(Ref: 7.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	INY	C8	1	2

JMP

JMP *Jump to new location*

JMP

Operation: (PC + 1) → PCL

(PC + 2) → PCH

N Z C I D V

– – – – – –

(Ref: 4.0.2)
(Ref: 9.8.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Absolute	JMP Oper	4C	3	3
Indirect	JMP (Oper)	6C	3	5

JSR

JSR *Jump to new location saving return address*

JSR

Operation: PC + 2 ↓, (PC + 1) → PCL
(PC + 2) → PCH

N Z C I D V
– – – – – –

(Ref: 8.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Absolute	JSR Oper	20	3	6

LDA

LDA *Load accumulator with memory*

LDA

Operation: M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.1.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDA #Oper	A9	2	2
Zero Page	LDA Oper	A5	2	3
Zero Page, X	LDA Oper, X	B5	2	4
Absolute	LDA Oper	AD	3	4
Absolute, X	LDA Oper, X	BD	3	4*
Absolute, Y	LDA Oper, Y	B9	3	4*
(Indirect, X)	LDA (Oper, X)	A1	2	6
(Indirect), Y	LDA (Oper), Y	B1	2	5*

* Add 1 if page boundary is crossed.

LDX

LDX *Load index X with memory*

LDX

Operation: M → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.0)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDX # Oper	A2	2	2
Zero Page	LDX Oper	A6	2	3
Zero Page, Y	LDX Oper, Y	B6	2	4
Absolute	LDX Oper	AE	3	4
Absolute, Y	LDX Oper, Y	BE	3	4*

* Add 1 when page boundary is crossed.

LDY

LDY *Load index Y with memory*

LDY

Operation: M → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDY #Oper	AØ	2	2
Zero Page	LDY Oper	A4	2	3
Zero Page, X	LDY Oper, X	B4	2	4
Absolute	LDY Oper	AC	3	4
Absolute, X	LDY Oper, X	BC	3	4*

* Add 1 when page boundary is crossed.

LSR

LSR *Shift right one bit (memory or accumulator)*

LSR

Operation: Ø → 7 6 5 4 3 2 1 0 → C

N	Z	C	I	D	V
Ø	✓	✓	–	–	–

(Ref: 10.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	LSR A	4A	1	2
Zero Page	LSR Oper	46	2	5
Zero Page, X	LSR Oper, X	56	2	6
Absolute	LSR Oper	4E	3	6
Absolute, X	LSR Oper, X	5E	3	7

NOP

NOP *No operation*

NOP

Operation: No Operation (2 cycles)

N	Z	C	I	D	V
–	–	–	–	–	–

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	NOP	EA	1	2

ORA

ORA *"OR" memory with accumulator*

ORA

Operation: A V M → A

N Z C I D V
✓ ✓ – – – –

(Ref: 2.2.4.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	ORA #Oper	Ø9	2	2
Zero Page	ORA Oper	Ø5	2	3
Zero Page, X	ORA Oper, X	15	2	4
Absolute	ORA Oper	ØD	3	4
Absolute, X	ORA Oper, X	1D	3	4*
Absolute, Y	ORA Oper, Y	19	3	4*
(Indirect, X)	ORA (Oper, X)	Ø1	2	6
(Indirect), Y	ORA (Oper), Y	11	2	5*

* Add 1 on page crossing

PHA

PHA *Push accumulator on stack*

PHA

Operation: A ↓

N Z C I D V
– – – – – –

(Ref: 8.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PHA	48	1	3

PHP

PHP *Push processor status on stack*

PHP

Operation: P↓

N Z C I D V
– – – – – –

(Ref: 8.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PHP	Ø8	1	3

PLA

PLA *Pull accumulator from stack*

PLA

Operation: A ↑

N Z C I D V
✓ ✓ – – – –

(Ref: 8.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PLA	68	1	4

PLP

PLP *Pull processor status from stack*

PLP

Operation: P ↑

N Z C I D V

From Stack

(Ref: 8.12)

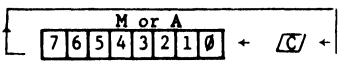
Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PLP	28	1	4

ROL

ROL *Rotate one bit left (memory or accumulator)*

ROL

Operation:



N Z C I D V

✓ ✓ ✓ – – –

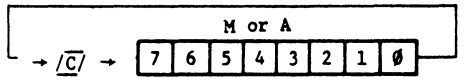
(Ref: 10.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ROL A	2A	1	2
Zero Page	ROL Oper	26	2	5
Zero Page, X	ROL Oper, X	36	2	6
Absolute	ROL Oper	2E	3	6
Absolute, X	ROL Oper, X	3E	3	7

ROR

ROR *Rotate one bit right (memory or accumulator)* **ROR**

Operation:



(Ref: 10.4)

N	Z	C	I	D	V
✓	✓	✓	-	-	-

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ROR A	6A	1	2
Zero Page	ROR Oper	66	2	5
Zero Page,X	ROR Oper,X	76	2	6
Absolute	ROR Oper	6E	3	6
Absolute,X	ROR Oper,X	7E	3	7

RTI

RTI *Return from interrupt* **RTI**

Operation: P↑ PC↑

(Ref: 9.6)

N	Z	C	I	D	V
From Stack					

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	RTI	4Ø	1	6

RTS

RTS *Return from subroutine* **RTS**

Operation: PC↑, PC + 1→ PC

(Ref: 8.2)

N	Z	C	I	D	V
-	-	-	-	-	-

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	RTS	6Ø	1	6

SBC

SBC *Subtract memory from accumulator with borrow*

SBC

Operation: A − M − $\overline{C}$ → A

N Z C I D V

Note: $\overline{C}$ = Borrow (Ref: 2.2.2)

✓ ✓ ✓ — — ✓

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	SBC #Oper	E9	2	2
Zero Page	SBC Oper	E5	2	3
Zero Page, X	SBC Oper, X	F5	2	4
Absolute	SBC Oper	ED	3	4
Absolute, X	SBC Oper, X	FD	3	4*
Absolute, Y	SBC Oper, Y	F9	3	4*
(Indirect, X)	SBC (Oper, X)	E1	2	6
(Indirect), Y	SBC (Oper), Y	F1	2	5*

* Add 1 when page boundary is crossed.

SEC

SEC *Set carry flag*

SEC

Operation: 1 → C

N Z C I D V

(Ref: 3.0.1)

— — 1 — — —

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SEC	38	1	2

SED

SED *Set decimal mode*

SED

Operation: 1 → D

N Z C I D V

– – – – 1 –

(Ref: 3.3.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SED	F8	1	2

SEI

SEI *Set interrupt disable status*

SEI

Operation: 1 → I

N Z C I D V

– – – 1 – –

(Ref: 3.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SEI	78	1	2

STA

STA *Store accumulator in memory*

STA

Operation: A → M

N Z C I D V

(Ref: 2.1.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STA Oper	85	2	3
Zero Page, X	STA Oper, X	95	2	4
Absolute	STA Oper	8D	3	4
Absolute, X	STA Oper, X	9D	3	5
Absolute, Y	STA Oper, Y	99	3	5
(Indirect, X)	STA (Oper, X)	81	2	6
(Indirect), Y	STA (Oper), Y	91	2	6

STX

STX *Store index X in memory*

STX

Operation: X → M

N Z C I D V

(Ref: 7.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STX Oper	86	2	3
Zero Page, Y	STX Oper, Y	96	2	4
Absolute	STX Oper	8E	3	4

STY

STY Store index Y in memory

Operation: Y → M

N Z C I D V
– – – – – –

(Ref: 7.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STY Oper	84	2	3
Zero Page, X	STY Oper, X	94	2	4
Absolute	STY Oper	8C	3	4

TAX

TAX *Transfer accumulator to index X*

Operation: A → X

N Z C I D V
✓ ✓ – – – –

(Ref: 7.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TAX	AA	1	2

TAY

TAY *Transfer accumulator to index Y*

TAY

Operation: A → Y

N Z C I D V
✓ ✓ – – – –

(Ref: 7.13)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TAY	A8	1	2

TYA

TYA *Transfer index Y to accumulator*

TYA

Operation: Y → A

N Z C I D V
✓ ✓ – – – –

(Ref: 7.14)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TYA	98	1	2

TSX

TSX *Transfer stack pointer to index X*

TSX

Operation: S → X

N Z C I D V

✓ ✓ – – – –

(Ref: 8.9)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TSX	BA	1	2

TXA

TXA *Transfer index X to accumulator*

TXA

Operation: X → A

N Z C I D V

✓ ✓ – – – –

(Ref: 7.12)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TXA	8A	1	2

TXS

TXS *Transfer index X to stack pointer*

TXS

Operation: X → S

N Z C I D V

– – – – – –

(Ref: 8.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TXS	9A	1	2

Programm zur Umwandlung von Zahlen

Da wir beim Programmieren in Maschinensprache ständig Zahlen einer bestimmten Basis in Zahlen einer anderen Basis umrechnen müssen, brauchen wir ein Umrechnungsprogramm. Das nachfolgend gezeigte Listing gibt Ihnen ein komplettes Programm zur Umrechnung von Hex- in Dezimalzahlen und umgekehrt, sowie Umrechnung in Oktal- und Binärzahlen.

```
READY.

 1 POKE 59468,14
 2 DIM B(20)
 5 PRINT"3"
 6 PRINTTAB(10)"BASIS UMWANDLUNG"
 8 PRINT""
 12 PRINT"    ICH WANDLE ZAHLEN IN DEZIMAL,OKTAL,"
 13 PRINT"OKTAL-SPLIT, HEX, UND BINAER."
 20 PRINT"EINGABE BASIS,ZAHL,UND GE-  WUENSCHTE UMRECHNUNG"
 25 PRINT:PRINT
 30 PRINT"Decimal='D'  Octal='O'  Octal-Split='S' "
 31 PRINT"Hexidecimal='H'   Binary='B'"
 32 PRINT"ICH KANN NUR 7 STELLEN VER-  ARBEITEN"
 33 GOTO 100
 35 PRINT "EINE ANDERE ZAHL(J,N)?";
 36 GET A$:IF A$="" THEN GOTO 36
 37 IF A$="J" THEN GOTO 120
 38 PRINT"3":PRINT"BASIS UMRECHNGN":PRINT"SERVUS BOY!!":PRINT""
 39 POKE 59468,12: END
 40 PRINT "VON WELCHER BASIS (D,O,S,H,B)?";
 42 GET Z$:IF Z$="" THEN GOTO 42
 45 PRINTZ$
 50 IF Z$="H" THEN 1400
 55 PRINT""
 60 INPUT "UMZUWANDELNDE ZAHL";A
 65 PRINT""
 70 PRINT "ZU BASIS.... (D,O,S,H,B) ?  ";
 74 GET Y$:IF Y$="" THEN GOTO 74
 76 PRINT Y$
 80 PRINT""
 90 IF Z$="S" THEN 1260
 92 IF Z$="D" THEN 490
```

```
94 IF Z$="O" THEN 580
96 IF Z$="B" THEN 640
98 GOTO 640
100 PRINT"(DRUECKE EINE TASTE)";
110 GET R$:IF R$="" THEN GOTO 110
120 PRINT"3"
130 PRINTTAB(10)"TNT BASE CONVERSION"
140 PRINT"";
150 GOTO 40
490 IF Y$="O" THEN 540
495 IF Y$="H" THEN 1600
500 IF Y$="S" THEN 1135
510 Z=2
520 Y$="BINARY ="
530 GOSUB 800
535 GOTO 35
540 Z=8
545 PRINT
550 Y$="OCTAL ="
560 GOSUB 800
565 GOTO 35
580 Z=8
590 IF Y$="D" THEN 620
595 IFY$="S"THEN GOSUB920:A=G:GOTO1135
596 IFY$="H"THEN GOSUB 920:A=G:GOTO1600
600 GOSUB 920
610 A=G: GOTO 510
620 GOSUB 920
630 GOTO 35
640 Z=2
670 IF Y$="D" THEN 700
675 IFY$="S"THEN GOSUB920:A=G:GOTO1135
676 IFY$="H"THEN GOSUB920:A=G:GOTO 1600
680 GOSUB 920
690 A=G: GOTO 540
700 GOSUB 920
710 GOTO35
800 N=0
810 N=N+1
830 B=INT(A/Z)
840 C=A-(B*Z)
850 B(N)=INT(C+.5)
860 A=B
```

```
870 IF B>0 THEN 810
872 IF Y$="H" THEN 912
875 IF Y$="S" THEN 912
880 PRINT Y$;
885 GOTO 900
890 N=N-1
895 IF N=0 THEN 911
900 PRINT B(N);
910 GOTO 890
911 PRINT
912 RETURN
920 C=1:E=(-1)
925 G=0
930 B=A/C
935 E=E+1
940 IF B<Z THEN 970
950 C=C*10
960 GOTO 930
970 D=INT(A/C)
980 H=(Z^E)*D
1000 D=D*C
1010 A=A-D
1020 G=G+H
1025 C=C/10
1040 E=E-1
1050 IF E=>0 THEN 970
1055 IF Z$="S" THEN 1070
1060 PRINT"DECIMAL =";G
1065 PRINT
1070 RETURN
1135 PRINT"OCTAL SPLIT =";
1140 B(3)=0: B(2)=0: B(1)=0
1150 B=INT(A/256)
1160 C1=A-(B*256)
1165 X=0
1170 A=B
1180 Z=8
1190 GOSUB 800
1200 N=3
1210 PRINTB(N);
1220 N=N-1
1230 IF N>0 THEN 1210
1240 X=X+1
```

```
1245 IF X>1 THEN PRINT: GOTO 35
1250 A=C1:B(3)=0:B(2)=0:B(1)=0:GOTO1180
1260 REM OS
1265 PRINT
1270 C=1000: Z=8
1280 B=INT(A/C)
1290 C1=A-(B*C)
1294 A=B
1300 GOSUB 920
1310 G1=G*256
1320 A=C1
1330 GOSUB 920
1340 G=G+G1
1350 PRINT"DECIMAL =";G
1355 A=G
1360 IF Y$="B" THEN PRINT:GOTO 510
1362 IF Y$="H" THEN 1600
1364 IF Y$="O" THEN 540
1366 GOTO 35
1400 PRINT"":INPUT "  UMZURECHNENDE ZAHL   ";X$:PRINT:PRINT
1401 PRINT"WELCHE BASIS (D,O,S,H,B) ? ";
1402 GET Y$:IF Y$="" THEN GOTO 1402
1403 PRINT Y$:PRINT
1405 HL=0
1406 G=0
1407 HJ=LEN(X$)
1410 FOR I=1 TO HJ
1415 XX$=MID$(X$,I,1)
1420 H=ASC(XX$)
1425 HL=HJ-I
1430 IF H=<57 THEN 1445
1435 B(I)=(H-55)*(16^HL)
1440 GOTO 1450
1445 B(I)=(H-48)*(16^HL)
1450 NEXT I
1455 FOR I=1 TO HJ
1460 G=G+B(I)
1465 NEXT I
1468 PRINT:PRINT:PRINT
1470 PRINT"DECIMAL =";G
1475 PRINT
1480 A=INT(G+.5)
1485 IF Y$="O" THEN 540
```

```
 1490 IF Y$="B" THEN 510
 1495 IF Y$="S" THEN 1135
 1500 GOTO 35
 1600 Z=16
 1601 PRINT:PRINT"HEXDECIMAL = ";
 1610 GOSUB 800
 1611 GOTO 1630
 1615 N=N-1
 1625 IF N=0 THEN 1660
 1630 IF B(N)=>10 THEN 1650
 1635 A1=B(N)+48
 1636 PRINT CHR$(A1);
 1640 GOTO 1615
 1650 A1=B(N)+55
 1655 GOTO 1636
 1660 PRINT
 1661 GOTO 35
 1670 END
READY.
```

Ein leistungsfähiger Monitor für den PET

HEX–MONITOR

Dieser leistungsfähige Monitor wurde in BASIC geschrieben und erlaubt den direkten Zugriff in die Speicherzellen des PET. Weiterhin können Programme auch in Maschinensprache gestartet und abgearbeitet werden.

Nach dem Einlesen des Monitors über den Cassettenrecorder Ihres PETs meldet sich der Monitor mit einer Aufforderung zur Eingabe eines Befehls. Zu diesem Zeitpunkt können Sie folgende Kommandos eingeben:

M = Ändern einer Adresse

D = Dumplistingausgabe vom Speicher auf den Bildschirm (Hexadezimal+ASCII)

L = Auslesen vom Speicher auf den Bildschirm (nur in ASCII)

S = Umwandeln von Dezimalzahlen in Hexadezimalzahlen und umgekehrt.

Z = Um ein Programm aus Data-Statements in BASIC direkt in den PET-Speicher zu lesen

G = Starten eines Programmes

P = PEEK in eine Speicherzelle

Beschreibung der einzelnen Kommandos:

M–Kommando:
Das M-Kommando wird dazu verwendet, um die Speicherinhalte des PETs anzusehen. Dies geschieht Byte für Byte. Nachdem Sie M eingegeben haben, wartet der Computer auf eine Adresse. Es muß nun die Adresse in Hexadezimal eingegeben werden, die zu der gewünschten Speicherzelle gehört. Der Computer wird dann die Adresse mit dem entsprechenden Inhalt auf dem Bildschirm anzeigen. Jetzt kann der Inhalt dieser Speicherzelle geändert werden oder Sie können auch dem Computer sagen, daß er zu einer anderen Adresse gehen soll.

Im Speicheränderungsmode kann wie folgt gearbeitet werden:

1. –NN sagt dem Computer, daß er NN Speicherplätze zurückgeht. Wobei NN eine zweistellige Hexadezimalzahl ist.

2. + NN sagt dem Computer, daß er NN Speicherplätze vorwärtsgeht, wobei NN wieder eine zweistellige Hexadezimalzahl sein muß.

3. Das Zeichen für die Division (/) führt diesen Mode (Betriebszustand) wieder zurück in den Monitor.

4. = NN ermöglicht das Einfügen eines Hexadezimalwertes NN in einen bestimmten Speicherplatz.

Alle anderen Befehle werden ignoriert und der Computer geht an die nächste Speicherzelle.

Mit den vorgenannten 4 Kommandos können Sie im Speicher Ihres PETs herumspringen und die gewünschten Zellen mit einem bestimmten Inhalt laden.

Der Befehl D ermöglicht das Anzeigen der Speicherinhalte in Hexadezimal und ASCII. Die erste Frage, die hier vom Computer kommt, ist, wo soll er anfangen. Sie brauchen jetzt nur die gewünschte Anfangsadresse in Hexadezimal einzugeben. Die nächste Frage des Computers ist die gewünschte Endadresse, bis zu der ausge-

listet werden soll. Auch hier geben Sie die gewünschte Endadresse in Hexadezimal ein. Auf diese Weise kann sehr elegant in ganze Speicherblöcke eingesehen werden.

Das Auslisten kann durch Drücken einer beliebigen Taste gestoppt werden. Gleichzeitig kehren Sie zum Monitorprogramm zurück.

Der Befehl L ist ein Kommando ähnlich wie der D-Befehl. Es werden jedoch nur ASCII-Daten gelistet. Auch hier werden Anfangs- und Endadresse in Hexadezimal eingegeben. Zum Monitor kann durch Drücken einer beliebigen Taste zurückgekehrt werden.

Der Befehl S kann zur Hexadezimal–Dezimal–Umrechnung und umgekehrt verwendet werden. Nach der Eingabe von S muß dem Computer die umzurechnende Dezimalzahl für eine Dezimal/Hexadezimal-Umrechnung eingegeben werden.

Für die Hexadezimal/Dezimal-Umrechnung wird ein $ -Zeichen mit anschließender Hexadezimalzahl eingegeben. Wenn Sie z. B. eine Dezimalzahl in eine Hexadezimalzahl umwandeln wollen, geben Sie $ 277 ein.

Der Computer antwortet dann automatisch mit der Hexadezimalzahl. Wenn Sie aber von Hexadezimal in Dezimal umrechnen wollen, geben Sie die gewünschte Hexadezimalzahl mit einem vorausgehenden Dollarzeichen ein (z. B. $ 3C40H). Der Computer antwortet dann umgehend mit der zugehörigen Dezimalzahl. Wenn Sie die Zahl 0 eingeben, verlassen Sie diesen Mode und gelangen wieder in den Monitor.

Der Befehl Z erlaubt Ihnen ganze Blöcke in Datenstatements im Speicher abzulegen. Nach Eingabe des Befehls Z fragt der Computer wieder nach einer Adresse. Dies ist die Anfangsadresse im Speicher, bei der die Eingabe beginnt. Anschließend fragt der Computer nach dem Namen des Programmes, welches in diesen Speicherblock geladen werden soll. Der Computer sucht dann die Datastatements nach diesem Namen ab. Das Format der Programme, die in Datastatements abgelegt werden sollen, ist wie folgt.

1. Der Programmname muß als erster Ausdruck in einem Datastatement enthalten sein.
2. Zu Beginn des Ausdruckes muß ein Asterisk gesetzt werden.

Um z. B. ein Programm unter dem Namen TEM zu speichern, muß folgendes eingegeben werden:

1. Zeilennummer
2. Leerzeichen
3. DATA
4. Leerzeichen
5. Asterisk
6. TEM
7. Komma
8. Programmtext
9. DATA * TEM, 4C, 3F, 9C, 3F

Die Liste wird abgeschlossen, indem ein DATA-Statement gegeben wird, welches das Wort END enthält. Dies ist notwendig, damit beim Einlesen durch den Programmlader kein END of DATA ERROR gegeben wird.

Wenn das Programm geladen wird, zeigt der Computer die hexadezimalen Bytes wie sie geladen und ausgeführt werden an. Am Ende sagt Ihnen der Computer, wieviele Bytes geladen wurden und nennt Ihnen die Anfangsadresse. Der Befehl Z sollte mit großer Sorgfalt verwendet werden, da evtl. wichtige Daten überschrieben werden könnten.

Der Befehl GO wird dazu benutzt, ein Programm an einer bestimmten Stelle im Speicher zu starten. In der Monitor-Originalversion ist der Speicherplatz mit der Adresse 2000 Hex festgelegt. Er kann jedoch für den Programmtest geändert werden.

Der Befehl P dient dazu, um für eine bestimmte Zeitdauer in eine Speicherzelle einzusehen. Auf diese Weise kann man Veränderungen in einer Speicherzelle beobachten. Nach Eingabe von P fragt Sie der Computer, welches die erste Adresse ist, deren Inhalt hexadezimal ausgegeben werden soll. Die nächste Frage ist, wie lange oder wie groß die Anzahl der Bytes sein soll, die auf dem Bildschirm gezeigt werden soll.
Zum Beispiel können Sie P mit Startadresse 200 und Länge 3 eingeben, und Sie sehen, wie die PET-Echtzeituhr den Speicher verändert.

Auch ist es recht brauchbar, wenn man sich den parallelen Eingangsport ansieht und die Adresse E 841 mit einer Länge von 1 eingibt. Dieser Mode kann durch Drücken einer beliebigen Taste wieder verlassen werden.

Wir wollen nun einmal unser kleines Programm mit Hilfe des Hex-Monitors eingeben und starten.

Das Hexmonitorprogramm wollen wir jetzt zum Einlesen unseres kleinen Programmes einlesen. Wir schreiben ab Zeile 3040 den Namen unseres Programmes NUL mit anschließenden Hexadezimalwerten ein.

```
3040 DATA NUL,"*NUL", A9, 00, A2, 09, 9D, 50, 03, CA,10,FA,60
3041 DATA END
```

In Zeile 3041 geben wir den Ausdruck END ein. Dieser sorgt dafür, daß nur die vorher eingegebenen 11 Bytes in Hexadezimal eingelesen werden. Der nachfolgende Inhalt aus dem Beispiel DLY kann ruhig überschrieben werden. Das Programm DLY wird als Beispielprogramm mit dem Monitor mitgeliefert. Die Zeilen 3040 bis 3999 stehen für Ihre Eingaben voll zur Verfügung. Als Programmnamen dürfen nur Werte mit drei Buchstaben verwendet werden.

```
READY.

 12 DIMH$(16):FORI=0TO15
 20 READH$(I):NEXTI
 22 DATA0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
 25 PRINT:PRINT"ENTER COMMAND:  ";
 30 GETC$:IFC$=""GOTO30
 35 PRINTC$
 40 IFC$="M"GOTO200
 50 IFC$="D"THENS=8:GOTO400
 55 IFC$="L"THENS=32:GOTO400
 60 IFC$="S"GOTO600
 62 IFC$="Z"GOTO700
 65 IFC$="G"GOTO2000
 67 IFC$="P"GOTO100
 68 IFC$="R"GOTO6000
```

```
69 IFC$="T"GOTO8000
70 PRINT"WHAT? ":GOTO25
100 INPUT"START ADDRESS,LENGTH";X$,V:GOSUB800:PRINT1
110 FORI=0TOV-1:B=PEEK(X+I):PRINTH$(B/16);H$(BAND15);:NEXT
117 FORI=1TOV:PRINT"==";:NEXT
120 GETC$:IFC$=""GOTO110
130 GOTO25
200 INPUT"ADDRESS";X$
209 GOSUB800:A1=X-1:GOTO260
210 IFA1<00RA1>2^16GOTO200
220 X=PEEK(A1):GOSUB1000
230 INPUTX$
231 Y$=LEFT$(X$,1):IFLEN(X$)>1THENX$=RIGHT$(X$,LEN(X$)-1):
    GOSUB800
232 IFY$="-"THENA1=A1-X-1:GOTO260
233 IFY$="+"THENA1=A1+X-1:GOTO260
234 IFY$="*"THENX$=X$:GOSUB800:A1=X-1:GOTO239
236 IFY$="/"GOTO25
237 IFY$<>"="GOTO260
239 IFX<0GOTO260
240 IFX>255THENGOTO25
250 POKEA1,X:IF PEEK(A1)<>XTHENPRINT"?":GOTO25
260 A1=A1+1:X=A1:GOSUB1004:GOTO220
400 INPUT"DUMP FROM";X$:GOSUB800
410 X1=X:IFX<0GOTO25
420 INPUT"DUMP TO";X$:GOSUB800
430 E=X:IFX<0GOTO25
440 FORI=X1TOESTEPS
450 PRINT:X=I:GOSUB1004
455 IFS>9GOTO500
460 FORJ=0TOS-1
470 X=PEEK(I+J):GOSUB1000:NEXT
490 PRINT"  ";
500 FORJ=0TOS-1
510 A=PEEK(I+J):IFA<32ORA>95THEN PRINT".";:GOTO550
520 PRINTCHR$(A);
550 NEXTJ
555 GETC$:IFC$>""THEN25
560 NEXTI
570 GOTO25
600 PRINT:INPUT"DECIMAL OR $HEX ";X$
601 IFLEFT$(X$,1)="$"THENX$=RIGHT$(X$,LEN(X$)-1):GOSUB800:
    PRINTX:GOTO600
```

```
605 X=VAL(X$):IFXTHENGOSUB1004:GOTO600
608 GOTO25
700 INPUT"ADDRESS";X$:B$=X$:GOSUB800:A1=X
710 IFX<00RX>2^16GOTO25
715 RESTORE:K=0
720 INPUT"PROGRAM";P$
730 READX$:IFX$="END"THENPRINT"NOT FOUND":GOTO715
740 IFX$<>P$GOTO730
745 READX$:IFLEFT$(X$,1)="*"THENPRINT:PRINTA1+K;X$:GOTO745
747 GOSUB800:IFX<00RX>255GOTO760
750 POKEA1+K,X:IF PEEK(A1+K)<>XTHENPRINT"ROM..?":GOTO700
755 PRINTX$" ";:K=K+1:GOTO745
760 PRINT:PRINTK;"BYTES LOADED AT",B$
770 A%=A1/256:POKE1,A1-A%*256:POKE2,A%
780 GOTO25
800 X=0:L=LEN(X$):FORA=0TOL-1
810 C$=MID$(X$,A+1,1)
820 FORD=0TO15:IFC$=H$(D)THENGOTO850
825 NEXTD
830 X=-1
840 RETURN
850 X=X*16+D
860 NEXTA
870 RETURN
912 DIMH$(16):FORI=0TO15
920 READH$(I):NEXTI
922 DATA0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
925 RETURN
1000 A=1:GOTO1009
1004 A=256:GOTO1009
1006 A=65536
1009 Y=X
1020 Z=Y/A:B%=Z-256*INT(Z/256)
1025 PRINTH$(B%/16);H$(B%AND15);
1040 IFA<9THENPRINT" ";:RETURN
1050 A=INT(A/256):GOTO1020
2000 X=USR(1):X=PEEK(825):GOSUB1000:PRINT"("CHR$(X)")",:
     GOTO2000
3040 DATA DLY,"*DLY",A9,00,8D,3C,03,A9,EA
3045 DATA 38,E9,01,B0,03,CE,3C,03,AC
3050 DATA 3C,03,10,F3,60
3060 DATA "*DE2",A9,00,8D,3C,03,A9,EA
3065 DATA 4A,4E,3C,03,90,E5,09,80,B0,E2
```

```
 3200 DATA"*TX",78,20,40,03,2D,39,03,8D,41,E8
 3205 DATA20,40,03,A2,08,29,FE,4E,39,03
 3208 DATA69,00,8D,41,E8,20,40,03,CA
 3210 DATAD0,F0,09,01,8D,41,E8,58,60
 3301 DATA "*RX",A9,00,8D,41,E8,78,A2,08,A9,01,2C,41,E8
 3305 DATA 30,FB,20,40,03
 3320 DATA20,55,03,AD,41,E8,29,80
 3325 DATA4E,39,03,0D,39,03,8D,39,03
 3330 DATA20,40,03,CA,D0,EC,2E,39,03,4E,39,03,58,A9,02,8D,41,
      E8,60
 3999 DATA END
 4000 P=59457
 4002 GET N:IFN<>0THENC=N*3
 4004 POKEP,0
 4005 FORI=1TOC:NEXT
 4006 POKEP,255:GOTO4002
 4008 GOTO4002
 6000 POKE59459,15:SYS(908):PRINTCHR$(PEEK(825));:GOTO6000
 7000 T=TI:N=10:FORI=1TON:SYS(855):NEXT
 7005 PRINT(TI-T)*16.666667/N-3.283334"MILLISECONDS":END
 8000 POKE59459,15:GETC$:IFC$=""GOTO8000
 8005 PRINTC$;:POKE825,ASC(C$):SYS(870):GOTO8000
READY.
```

Monitor NUL mit geändertem Inhalt

Wenn das Hex-Programm in DATA-Statements eingegeben worden ist, starten wir wieder mit RUN den Monitor. Z bringt uns in den Eingabemodus für Hex-Programme. Wir geben die Anfangsadresse mit 033A Hex ein und den Namen aus dem DATA-Statement NUL. Jetzt wird das Programm in den Speicher eingelesen. Nachdem das Programm eingelesen wurde, können wir wieder den Monitor starten und den Befehl G geben.
Jetzt wird das Programm ausgeführt. Die Anfangsadresse steht in Zeile 2000 mit 825. Sie kann bei Bedarf entsprechend geändert werden. Bei einer Anfangsadresse 033A sollte sie auf 826 abgeändert werden.

Nachdem der Befehl G gedrückt wurde, muß nach ca. 1 — 2 s das Programm über RUN/STOP angehalten werden.

Der Monitor wird jetzt neu gestartet und mit dem Befehl D – 0350 bis 0359 können wir im Speicher nachsehen, ob unser Programm gelaufen ist.

Jetzt können Sie Ihre Kenntnisse und die Praxis noch durch einige Versuche ausbauen.
Der Hex-Monitor wird Ihnen dann bei allen späteren Maschinenspracharbeiten gute Dienste leisten, da Sie direkt in Hexadezimal programmieren können.

Supermonitor für den PET

Commodore Terminal Interface Monitor mit Disassembler

TIM ist ein Terminal Interface Monitor Programm für die 65XX Microcomputerprodukte der Fa. MOS-Technology. Es eignet sich besonders für die Entwicklung von Programmen in Maschinensprache und ermöglicht die Ankopplung dieser Programme an BASIC.
Um die Leistungsfähigkeit dieses Monitorprogrammes noch zu erhöhen, haben wir es mit einem Disassembler ausgerüstet.
Spätere PET-Computer werden den TIM im ROM haben. Alle anderen Versionen müssen den TIM-Monitor über Cassette in den Arbeitsspeicher laden. In jedem Falle wird jedoch das Programm von BASIC in den Monitor mit Hilfe des SYS-Befehls übertragen. Die Ausführung des Programmes erfolgt dann vom TIM.
Um die richtige Startadresse für Ihren TIM herauszufinden, beachten Sie bitte die Speicheraufteilung im mitgelieferten PET-Bedienungshandbuch.

Befehle, die Sie über das PET-Keyboard eingeben, starten im Monitor sofort die Programmausführung.

1. Anzeigen oder Ändern von Speicherplätzen oder Registern
2. Breakpoints setzen
3. Laden und Abspeichern von binären Quellprogrammen (LOAD und SAVE)

Bei der Änderung von Speicherzelleninhalten führt der TIM-Monitor eine automatische Lese-Nachschreibe-Operation durch, damit gesichert ist, daß man in eine vorhandene Speicherzelle geschrieben hat.

Der TIM-Monitor enthält auch einige Unterprogramme, welche durch das Anwenderprogramm aufgerufen werden können.

Diese Routinen sind:

1, Lesen und Schreiben von Zeichen auf dem Bildschirm
2. Hexadezimaleingabe eines Bytes und Eingabe einer CRLF-Reihenfolge

M = Display Memory (Speicherinhaltsanzeigen)
R = Display Registers (Registerinhaltsanzeigen)
G = GO (Beginn mit der Programmausführung)
X = Übergang zu BASIC
S = SAVE
D xxxx Disassemble, beginnend an Adresse xxxx

L = LOAD (Laden eines Programmes)
Z. Zt. sind zwei verschiedene Monitorversionen erhältlich:
1. SUPERMON
2. SUPERMON TOM

Beide Versionen arbeiten ähnlich und unterscheiden sich vom Befehlssatz her nur in der Handhabung des Disassemble-Befehls.

Disassemblieren bei der Version Supermon:
G 0800 (Return) ⟨ (Space) 0000, 0010 Return
Rückkehr mit X in den Monitor, von dort aus wieder mit X zurück in BASIC
Disassemblieren beim Supermon TOM geschieht durch Eingabe von D 0000 Return. Jetzt wird von Adresse 0000 Hex. an eine Bildschirmfüllung voll disassembliert.

Beispiele:

```
: M  C 000, C010
: C000  1DC7  48  C6  35  CC  FF  C7
': C008  C5  CA  DF  CA  70  CF  23  CB
.: C010  9C  C8  9C  C7  74  C7  IF  C8
```

Im Befehl zur Anzeige eines Speicherinhaltes müssen Anfangsadresse und Endadresse genau

angegeben werden. (4 Digit Hexadezimalzahlen)

Um einen Speicherinhalt zu ändern, fahren Sie mit dem Cursor auf dem Bildschirm an die gew. Zahl, ändern diese und drücken die RETURN-Taste.

```
.R
.:    PC  SR  AC  XR  YR  SP
  C6ED  00  80  00  20  F5
```

Bei jedem Eintritt in den TIM-Monitor oder beim Verlassen des Monitors werden die Register zwischengespeichert und aufbewahrt. Sie können genau so, wie die Speicherplatte im obigen Beispiel mit dem Cursor geändert werden.

```
. G C38B
```

Der GO-Befehl zum Starten eines Programmes kann wahlweise mit oder ohne Anfangsadresse geschehen. Wenn nichts angegeben wird, wird der Programmzählerstand, der mit dem R-Befehl angesehen werden kann, als Anfangsadresse verwendet werden.

```
.X
READY
```
Bringt den PET in das BASIC-Programm

```
.Lϕ1, MONITOR
PRESS PLAY ON TAPE #1
OK
FOUND MONITOR
LOADING
```

Beim LOAD-Befehl darf nichts weggelassen werden! Die Nummer und der File-Name muß unbedingt angegeben werden.
Das Betriebssystem antwortet wie in BASIC mit „PRESS PLAY ON TAPE #1".

Die Speicheradresse wird so geladen, wie Sie in der File-Adresse angegeben wurde. Sie mußte vorher natürlich im SAVE-Befehl eingegeben worden sein.
Unterprogramme in Maschinensprache können auch vom BASIC her geladen werden. Es muß jedoch darauf geachtet werden, daß keine BASIC-Variablen verwendet werden, da der entsprechende Pointer dann hinter das zuletzt geladene Byte gesetzt würde.

```
.S 01, MONITOR, 0400, 076D
PRESS PLAY + RECORD ON TAPE #1
```

Endadresse Commodore TIM = 076D Hex.
Endadresse Supermon TOM = 0A2E Hex.

Alle Daten, incl. Anfangs- und Endadresse müssen angegeben werden.
Um einen Befehl wieder rückgängig zu machen, geben Sie einfach RETURN ein oder drücken die Stop-Taste.

Interrupt und Breakpoint Betrieb

BRK ist ein Software-Unterbrechungs-Befehl, welcher die CPU veranlaßt, die Befehlsausführung sofort zu stoppen. Programmzähler und P-Register werden im STACK (Stapelspeicher) zwischengespeichert (gesichert). Nun springt das Programm durch einen Vektor an die Speicherstellen $ 021B und $ 021C. TIM initialisiert diesen Vektor, um ihn auf sich selbst zu richten.

Wenn der Anwender diesen Vektor nicht ändert, bleibt das Programm im Monitor, auch wenn ein BRK-Befehl gegeben wurde. Er druckt dann B* aus und zeigt damit an, daß die Unterbrechung durch einen Breakpoint gekommen ist und nicht durch einen CALL. (CALL = C*....)

Auch werden die Register wie nach dem R-Befehl angezeigt. Der Monitor wartet jetzt auf weitere Befehle. Beachten Sie, daß nach einem BRK-Befehl, dessen Vektor auf den TIM-Monitor gerichtet ist, der Programmzähler des Anwenders jetzt auf die Speicherzelle zeigt, die unmittelbar nach dem BRK-Befehl kommt.

In jedem Fall sollte der Anwender des BRK-Befehls folgendes beachten:

> BRK arbeitet als ein Zwei Byte-Befehl, wobei der Programmzähler zwei Byte hinter dem BRK-Befehl stehen bleibt. (RTI = Return from Interrupt)

Der IRQ (Interrupt Request) ist im PET normalerweise auf ein ISR gerichtet, welches die Uhr steuert und das Tastenfeld jede 1/60 Sekunde abfragt. Wenn der Vektor geändert wird, und die Maschinensprachenunterroutine diesen Vektor nicht zwischenspeichert, muß ein Power-on RESET gegeben werden.

NMI ist im PET nicht verwendet. Dieser Anschluß ist ständig hoch gehalten.
RESET Vektoren auf einen Kalt-Start des BASIC löschen den gesamten Speicher. TIM muß dann über einen SYS-Befehl wieder geladen werden.

TIM-Monitor-Aufrufe und spezielle Speicherplätze

JSR	WRT	$ FFD2	Schreiben eines Zeichens
JSR	RDT	$ FFCF	Eingabe eines Zeichens
JSR	GET	$ FFE4	Holen eines Zeichens
JSR	CRLF	$ 04F2	Eingabe CR (Carriage Return)
JSR	SPACE	$ 063B	Eingabe 2 Leertasten
JSR	WROB	$ 0613	Eingabe 2 Byte
JSR	RDOB	$ 0660	Lesen eines Bytes
JSR	HEXIT	$ 0687	ASCII / Hex in A

$ 04 – $ 22 Zero PAGE
$ 400 – $ 76C absolute RAM

$ 23 – $ 5A sind „ Zero page locations" im BASIC Eingangs-Puffer. Sie können verwendet werden, wenn BASIC diesen Speicherbereich nicht benützt. Der Pufferspeicher für die zweite Cassette $ 33A - $ 3FF ist ein recht praktischer Speicherplatz für Informationen. Natürlich nur solange man nicht die zweite Cassette benützt. Andere Speicherplätze kann man nur mit großer Vorsicht benutzen, da man nie weiß, wo die PET-Software noch etwas ablegt.

Testen des MONITORS

Schalten Sie Ihren PET ein und bringen Sie ihn in den gewohnten Befehlsmode für BASIC. Legen Sie die Cassette ein und drücken Sie Shift/ RUN. Laden Sie das Programm wie gewohnt. Nach dem Einlesen sehen sie folgendes auf dem Bildschirm:

```
C*   PC  SR  AC  XR  4R  SP
. C6ED  29  00  88  89  FE
```

Es können kleine Veränderungen vorkommen, aber im großen und ganzen sieht die Anzeige wie oben aus.

Die Anzeige der Registerinhalte ist, wie schon erwähnt, die erste Anzeige beim Eintreten in den Monitor.
Sie besteht aus C* Registerinhalte: Programmzähler, Prozessorstatus, Akkumulator, X-Index, Y-Index und Stack-Pointer.

Beachten Sie, daß alle TIM-Eingaben und Ausgaben in Hexadezimal erfolgen.
Nachdem der TIM-Monitor diese Registeranzeige vorgenommen hat, ist er in der Lage, Befehle entgegenzunehmen. TIM gibt den „READY-STATUS" durch die Antwort „ o " auf dem Bildschirm wieder.

Die CPU-Register können auch mit dem R-Befehl angesehen werden. Der Monitor sollte dann wie oben antworten, jedoch ohne das Asterisk-Zeichen hinter dem C.

Angezeigte Werte können mit dem Cursor geändert werden. Beachten Sie, da- Zwischenräume zwischen Daten und Adressen sein müssen. (4 Digit für Hex Adressen, 2 Digits für den Byte-Inhalt).

Speicherinhalte können mit dem M-Befehl angezeigt und dann geändert werden.
Wenn Sie .M 0100, 0107 eingeben, sehen Sie z. B. folgendes:

```
              0  1  2  3  4  5  6  7
.: 0100      10 35 00 30 30 30 30 30
```

Nun können Sie wieder mit dem Cursor ändern und anschließend neu anzeigen.

```
*      = S 33A
CRLF = S 4F2
WRT  = S FFD2
```

```
33A  20  F2 04  CHSET  ISR CRLF
33D  A2  20             LDX LS20
33F  8A          LOOP   TXA
340  20  D2 FF         ISR WRT
343  E8                 INX
344  E0  60             CPX LS60
346  D0  F7             BNE LOOP
348  00                 BRK
349  4C  3A 03          JMP CHSET
```

Benutzen Sie den M-Befehl, um das folgende Testprogramm mit dem Namen CHSET einzugeben. Es druckt den ASCII;Zeichen Vorrat auf den Bildschirm. Dann können Sie wieder mit dem Cursor entsprechende Änderungen durchführen.

Geben Sie M 033A 034B ein und ändern Sie mit dem Cursor entsprechend die Zelleninhalte.

```
.M   033A  034B

.:   033A 20 F2 04 A2 20 8A 20 D2
.:   0342 FF E8 E0 60 D0 F7 00 4C
.:   034A 3A 03
```

Das CHSET Programm wurde so assembliert, daß es den Speicherbereich für den 2. Cassetten-Puffer benutzt.

.G 033A Eingabe führt nun das Programm aus

Der gesamte ASCII Zeichenvorrat sollte nun auf dem Bildschirm erscheinen.
Achten Sie darauf, daß die Adresse jetzt im Programmzähler steht und das Programm jetzt durch GO ohne nachfolgende Adresse gestartet werden kann.

Nun wollen wir versuchen, dieses kleine Programm mit BASIC zu verbinden. Diese Technik ist sehr interessant und erlaubt das Einbauen von Maschinenprogrammen in die BASIC Programme.
Zuerst löschen wir den BRK-Befehl in Speicherzelle $ 348 durch ein RTS (Return Subroutine $ 60).

Wechseln Sie den USR Funktions-Vektor in Zelle 1 und 2 so, daß er auf das Unterprogramm, beginnend an Speicherzelle $ 33A gerichtet ist.

```
.: 0000 4C 3A 03
```

(höherwertiges Byte zuerst)
Verlassen Sie nun dem Monitor durch Eingabe X und geben Sie in den BASIC-Mode

```
.X
READY
```

ein.
Prüfen Sie nun nach, ob die Ankopplung des Programmes auch erfolgt ist und geben Sie folgendes ein:

A = USR (0) → RETURN und es erscheint ASCII

SYS (3* 256 + 3 * 16 + 10) → RETURN und es erscheint der Monitor

U-Bootjagd für PET	49.--
Einarmiger Bandit (Spielautomat) für PET	29.--
Black Jack (17 und 4) für PET,	49.--
JOYSTICK mit Programm für PET	149.--
Biorythmus für PET	49.--
Krieg der Sterne (Star Wars) für PET,	29.--
Krieg der Sterne mit Abschießen für PET	29.--
Las Vegas für PET	29.--
Haushalts-Utilityprogramme I für PET	29.--
Haushalts-Utilityprogramme II für PET	29.--
Finanzprogramm I und II (Haush.Fin.) f. PET	138.--
Finanzprogramm III, Immob.Hausk. f. PET	69.--
Partyprogramm (Party Time) für PET	49.--
Lernprogramme (Educator I) für PET	29.--
Lernprogramme (Educator II) für PET	39.--
Musikprogramme für PET	49.--
Hex-Monitor für den PET	19,80
Super-Monitor für den PET	19,80
Musikplatine mit Software f. PET Bausatz	178,--
Schachprogramm für PET	79,--
Linear Joystick mit Softw. Bausatz	199,--
Programmiertricks für PET	39,--
BASIC-Kurs für PET	99,--
Graphik und Bewegung f. PET	29,--
BASIC Software Volume I	99.--
BASIC Software Volume II	99.--
BASIC Software Volume III	149.--
BASIC Software Volume IV	39.--
BASIC Software Volume V	39.--
Your home computer	24.80
My Computer Likes Me (BASIC-Einf.)	9.80
Computer Games (PCC-Games)	9.80
Beginners Guide to Microprocessors	29,80
BASIC-Cookbook	24,80

Telefonische Bestellungen:
08024/7331

Einzelpreis DM

6502 Assembler mit dem PET

Beim Erstellen größerer Programme ist die Umsetzung der Mnemonic-Befehle von Hand in die OPCODES und die Berechnung der Sprungadressen etc. recht langwierig. Aus diesem Grunde ist es jetzt von Vorteil, einen Assembler zu verwenden. Der Assembler ermöglicht es Ihnen, daß Sie Ihr Programm in Mnemonics (Kurzbezeichnungen) schreiben können. Der Assembler übersetzt Ihnen dann dieses Programm in den Maschinencode.

Was macht eigentlich ein Assembler?
Auf jeden Fall ist es einfacher, in Hexadezimal zu programmieren, als Schalter zu bewegen (binäre Schaltereingabe).
Aber jeder, der es einmal versucht hat, größere Programme im Hex-Code zu schreiben, wird einen Assembler gar sehnsüchtig begrüßen.

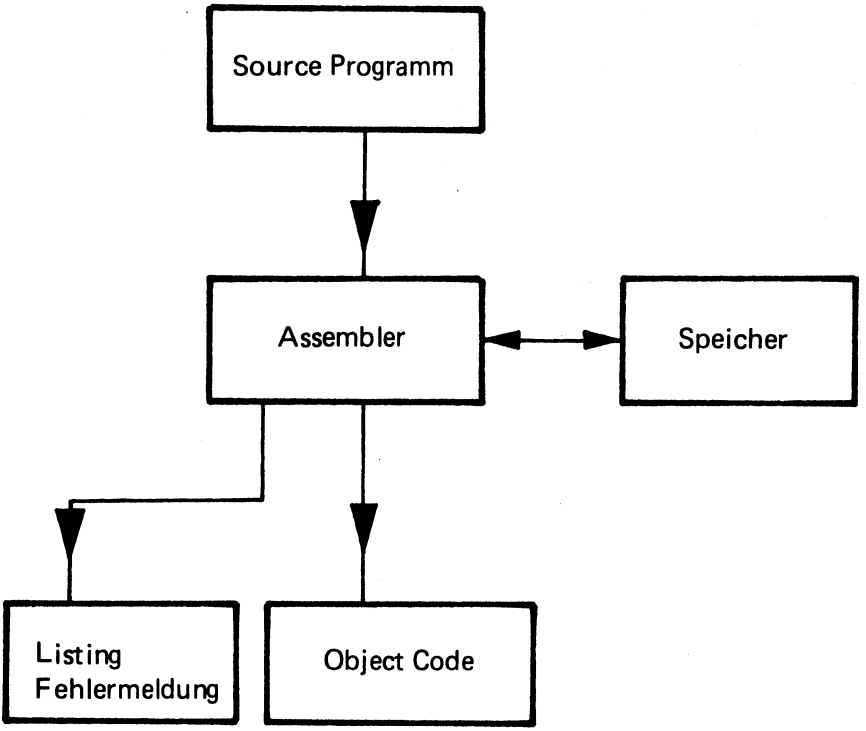
Ein Assembler ist ein Computerprogramm, welches ein Programm in Assemblersprache in den zugehörigen Maschinencode übersetzt. (Binär oder Hex.)

Wenn der Computer, der den Code später verarbeiten soll, einen eigenen Assembler beinhaltet und verarbeitet, nennt man den Assembler „resident". Wenn nicht, spricht man von einem Cross-Assembler. Die Assemblersprache ist eine Symbolsprache mit einfachen Ausdrücken, die auf die Funktion in gewissem Sinne hinweist . (Englisch)
JUMP = JMP = Springe für Sprungbefehl etc.

Diese Symbolsprache ist für den Programmierer in jedem Falle bequemer, als das Hantieren mit Hexadezimalzahlen. Im weitesten Sinne arbeiten alle Assembler ähnlich. Auf diese Weise profitieren Sie in jedem Falle mehrfach, wenn Sie die Assemblierung für einen Prozessortyp erlernen.

Auf jeden Fall sollten Sie sich vorher mit der internen Struktur (Register, Befehlsliste, Adressierungsarten etc.) etwas vertraut machen. Im Gegensatz zur Programmierung in BASIC, FORTRAN, PL/1, ALGOL etc.. wo Sie fast keine computerinternen Kenntnisse benötigen.

Im Grunde genommen ist die Assembler-Programmierung auch eine Programmierung auf Maschinenebene. Der Assembler macht nur die Programmierungsart sehr bequem.



Wie das Bild zeigt, nennt man die Eingabe in einen Assembler Source-Programm oder Quellprogramm. Sie können aus einem Speicherbereich oder von einem Cassettenspeicher gelesen werden.. Das Ausgangsprodukt des Assemblers (Objectcode) kann im Hauptspeicher abgelegt werden.

Grundsätzlich gibt es bei der Assemblerprogrammierung vier verschiedene Programmfelder.

1. Name der Speicherzelle (Label Feld)
2. Befehl oder OPCODE-Feld
3. Operand oder Adress-Feld
4. Ein Feld für Erläuterungen und Kommentierung

Quellprogramm (Beispiel)

	Platz oder Label	Mnemonics oder OPCODE	Adresse Operanden	Bemerkungen
1	TO		$033A	
2	FROM		$043A	
3			$0100	
4	MOV	LDX	# $FF	
5	LOOP	LDA	FROM X	
6		STA	TO X	
7		DEX		
8		BNE	LOOP	
9		LDA	FROM X	
10		STA	TO X	
11		BRK		
12		END		

Erklärungen:
Ein $-Zeichen vor einer Zahl bedeutet Hexadezimal
Ein # bedeutet immediate (unmittelbare Adressierung)

Wie Sie sehen, sind die Befehle im Quellprogramm numeriert. So hat man eine recht gute Übersicht. Das Quellprogramm kann vom Assembler aus einen bestimmten Speicherbereich oder auch von einer Cassette oder Diskette gelesen werden. Der Assembler liest aus dem Source-Code einen Befehl nach dem anderen. Wenn der Befehl gelesen wird, bereitet der Assembler schon die Übersetzung in den OPCODE vor.

Die meist verbreitetsten Assembler sind Zweipassassembler. Dies bedeutet, daß der Assembler zwei komplette Durchläufe über das gesamte Quellenprogramm ausführen muß.

Bestimmte Regeln bezüglich dieser Felder können von Assembler zu Assembler verschieden sein.

Hinweise und Mnemonic Befehle

Die Mnemonic-Befehle für einen bestimmten Prozessor werden vom Hersteller des Prozessors festgelegt. In unserem Falle hier beim 6502 ist es MOS-Technology (Rockwell + Synertex als Zweitlieferanten).

Meist sind Mnemonics leicht verständlich und lassen sich auch leicht behalten. Neben den Mnemonics enthalten die Assemblersprachen auch Direktiven.

Wegen der wichtigen Bedeutung muß der Anfänger sich unbedingt mit den Direktiven beschäftigen.

Assembler Direktive werden auch oft Pseudo Ops genannt, da sie sich nicht wie Mnemonics auf einen bestimmten Maschinencode beziehen. Unter anderem legen die Direktiven beim Assembler fest, wo der OPCODE im Speicher abgelegt werden soll und wo das Assemblerprogramm endet. Ein Assembler kann jedoch noch wesentlich mehr Direktive haben, je nachdem, wie „hoch" er entwickelt ist. Man sollte sich sehr intensiv mit ihnen beschäftigen, da die richtige Anwendung die Programmierung wesentlich erleichtert.

Wie arbeitet ein Assembler?

Während des ersten Durchlaufes werden die LABELS und das OP-CODE-Feld durchgearbeitet. Als Ergebnis liefert der erste Durchgang eine TABELLE mit allen symbolischen Adressen und den kompletten Adressangaben. Bei der Programmierung zu Fuß muß man sich alle Adressen selbst ausrechnen, was oft sehr zeitraubend ist.

Der zweite Durchgang bringt dann den OPCODE an die richtige Adresse.
Der Assembler kennt alle Mnemonics und interpretiert alle Direktiven.
Meistens wird auch während des zweiten Durchlaufes ein vollständiges Assembler-Listing erstellt. Der Objektcode kann in einem bestimmten Teil des Hauptspeichers abgelegt, oder auch gleich auf einen Cassettenspeicher etc. gelesen werden.

In unserem Falle wollen wir jetzt speziell auf den Assembler von Personal Software für den PET eingehen. (Vertretung für Central Europa: Ing. W. Hofacker GmbH)
Dieser Assembler wird auf Cassette mit einem sehr ausführlichen englischen Manual geliefert. Er ist in BASIC geschrieben.
Das gesamte Paket enthält:

1.	Manual	
2.	ASSEM2	Ein Zwei-Pass-Assembler (Mit Musterprogramm)
3.	EDIT	Texteditor
4.	ASSEM 1	Ein-Pass-Assembler
5.	EXEC	Ausführungsmonitor für Maschinenprogramme. Ein Disassembler kann die Maschinensprache-Programme wieder zurück in die Mnemonic-Form bringen und in hexadezimale Zeichen umsetzen. in DATA-Statements kann dann das Maschinenprogramm, wie in BASIC, geladen werden.

Zweipassassembler

Wir wollen nun mit Hilfe des Assemblers auf Cassette einmal unser kleines Beispielprogramm (auf der Assembler-Cassette) assemblieren. Dazu laden wir das erste Programm auf der Cassette in den Speicher.

Für den ersten Kontakt mit dem Assembler ist ein Musterprogramm im ASEM 2 eingebaut. Dieses können Sie sich erst einmal zur Information ansehen.
Laden Sie die Assembler-Cassette und geben Sie RUN.
Auf dem Bildschirm erscheint:

SOURCE FILE?

Geben Sie jetzt ein Anführungszeichen mit anschließendem RETURN ein. Diese Anweisung führt dazu, daß der Assembler ein Quellprogramm aus DATA-Statements liest.

Am Anfang des ASEM 2 befindet sich ein Beispielprogramm. Der Quell- und Maschinencode wird zweimal aufgelistet, zuerst im 1. Durchgang, dann im 2. Durchgang. Sie werden erkennen, daß die assemblierten Adressen für die symbolischen Labels erst im zweiten Durchgang ihre richtigen Werte erhalten. Wenn die Assemblierung fertig ist, erscheint:

```
EXECUTE?
```

Geben Sie jetzt Y (Return) ein. Am oberen Bildschirm sollte nun HI THERE erscheinen.

```
BREAK
READY
```

Nun geben Sie LIST ein und sehen sich das Programm einmal an.

Der Zweipassassembler ist in der Lage, ein Quellprogramm von einer Datencassette zu lesen. Da es sich um einen Zweipassassembler handelt, muß die Datei 2 x hintereinander eingelesen werden. Es gibt jetzt hier zwei Möglichkeiten. Entweder Sie laden das Quellprogramm an den Anfang einer Cassette und laden es zweimal hintereinander, oder Sie laden es zweimal auf der Cassette und laden unmittelbar hintereinander.

Wenn Sie den Zweipassassembler verwenden wollen, um von einem Datenfile (Datei) zu lesen, die zweimal hintereinander geladen ist, können Sie wie folgt vorgehen:

```
SOURCEFILE? NULL
RECORDED TWICE ? Y
```

Der Assembler läuft jetzt durch beide Zyklen und dann fragt er uns, ob wir das Programm starten wollen.

Wenn wir ein Programm assemblieren wollen, welches nur einmal aufgezeichnet wurde, gehen wir wie folgt vor:

```
SOURCE FILE ? NULL
RECORDED TWICE ? N
```

In diesem Falle stoppt der Recorder nach dem ersten Durchlauf. Wir spulen dann zurück und drücken irgendeine Taste. Nun wird der zweite Teil eingelesen.
Je nach Aufgabenstellung sollten Sie zwischen dem Einpass- und dem Zweipassassembler wählen.

Der Text Editor
Bevor wir ein Programm assemblieren können, müssen wir es erst einmal eingeben, evtl. korregieren und auf Cassette als File (Datei) abspeichern.

Wir wollen uns jetzt etwas mit dem Texteditor beschäftigen. Wir laden das Programm mit dem Namen EDITOR und nach RUN finden wir am oberen Bildrand die 6 Möglichkeiten.

LIST = Auslisten	(L)
INPUT = Eingabe	(I)
DELET = Zerstören	(D)
SEARCH = Suchen	(S)
READ = Lesen	(R)
WRITE = Schreiben	(W)

Angenommen, wir wollen ein Programm eingeben, so drücken wir I. Jetzt erscheint auf dem Bildschirm:

```
INPUT BEFORE LINE?
```

Wenn wir mit Zeile 1 anfangen wollen, geben wir 1 ein. Es erscheint ein Fragezeichen, Zeilennummer, Komma und Anführungszeichen.

```
?1,"
```

Sie fahren jetzt mit dem Cursor hinter das Anführungszeichen und geben ein kleines Assembler-Programm ein.

```
? 1, " LDX # 0          "
? 2, " LABEL TXA        "
? 3, " STA 32767,X      "
? 4, " RTS              "
? 5, " ←                "
```

Es muß immer zwischen den Anführungszeichen in einer Reihe eingegeben werden. Wenn die Eingabe beendet werden soll, wird der kleine Pfeil nach links (←)eingegeben. Jetzt haben wir die Wahl, alles zu speichern und sollten daher irgendeine Taste (nur nicht X) drücken.

Wir erhalten dann die Meldung über den freien Speicherbereich. Dann wartet der Computer auf eine neue Anweisung. Mit SHIFT CLEAR löschen Sie jetzt das Geschriebene vom Bildschirm. Die Befehle am oberen Bildschirmrand bleiben erhalten. Jetzt können Sie auch eine Änderung vornehmen, wenn Sie z. B. (I) eingeben. Es erscheint:

```
INPUT BEFORE LINE ? 0
```

Sie geben 0 ein und der Computer meldet sich mit einem Fragezeichen. Der Computer weiß jetzt, daß Sie eine einzelne Zeile eingeben wollen.

Diese Art der Eingabe verwendet man zweckmäßigerweise zur Korrektur.
Sie geben dann Ihre Änderung z.B. wie folgt ein: Listen Zeile 1,

```
? 1," LDX # 0"
```

Gehen Sie mit dem Cursor über die Null und überschreiben Sie diese z.B. durch eine 1. Dann geben Sie RETURN und die Änderung ist erfolgt.
Sie können aber auch mit dem Cursor in eine neue Zeile gehen und die zu ändernde Zeile noch einmal neu schreiben.

Nachdem die Änderung erfolgt ist, geben Sie wieder I ein und

```
INPUT BEFORE LINE ? 4

? 4, " INX          "
? 5, " BNE LABEL    "
? 6, ←              "
```

Drücken Sie irgendeine Taste (nicht X). Nun wieder Shift/Clear und L mit Return. Nun wird das gesamte Programm gelistet.

```
? 1, " LDX # 1          "
? 2, " LABEL TXA        "
? 3, " STA 32767,X      "
? 4, " INX              "
? 5, " BNE LABEL        "
? 6, " ←                "
```

Um es auf Cassette zu speichern, geben Sie jetzt W ein.
Auf dem Bildschirm erscheint:

```
WRITE FROM, TO ? 1,6
FILENAME ? NUL
```

(unbedingt richtig eingeben oder einfach ein Komma mit RETURN)

Nun wird das Programm auf die Cassette geschrieben. Anschließend erscheint das Listing noch einmal mit dem Namen. Wenn Sie den Versuch machen wollen, es noch einmal einzulesen, können Sie dies mit dem Befehl R tun.

Nach Drücken der Taste R und Return meldet sich der Computer mit:

```
LOAD BEFORE LINE ? 1
FILENAME ? NUL
```

Nun wird das Programm noch einmal geladen. Es wird jetzt an die Adressen 7 bis 12 geladen. Sie können dies sich durch List bestätigen lassen.

Wir haben jetzt festgestellt, wie man ein Programm editiert und auf eine separate Datencassette (File) laden kann.

Assemblieren mit dem Zweipassassembler
Nun laden wir den Assembler ASEM 2 wieder in den PET und geben RUN ein. Jetzt erscheint auf dem Bildschirm wieder:

```
SOURCE FILE ? NUL
RECORDED TWICE ? N
```

Wir geben NUL und Nein und laden die File-Cassette. Nach dem ersten Durchgang sollte

REWIND TAPE, PRESS ANY KEY

ausgedruckt werden. Wir spielen die Cassette zurück, drücken eine beliebige Taste und Laden die File Cassette noch einmal.

Am Ende steht das fertig assemblierte Programm auf Ihrem Bildschirm. Wenn in Ihrem Quellprogramm keine besondere Aussage bezüglich Anfangsadresse gemacht wird, wird das assemblierte Programm im Zwischenspeicher des 2. Cassetteninterfaces ab Adresse 033A (826) abgelegt. Dort befindet sich auch noch unser kleines Assemblerbeispiel.

Eine neue Anfangsadresse kann im Quellprogramm z.B. durch *= $2000 festgesetzt werden. Achten Sie jedoch darauf, daß der Speicher des PET bereits mit dem Assembler belegt ist. Aus diesem Grunde ist es für denjenigen, der größere Programme als 198 Byte assemblieren möchte, sehr wichtig, daß eine Speichererweiterung angeschafft wird. Mit der 8K PET 2001 Ausführung sollten deshalb in erster Linie nur Programme im 2. Cassettenspeicher assembliert werden.

Der EXECUTER /Disassembler
Solange das assemblierte Programm noch im 2. Cassettenpuffer vorhanden ist, kann mit dem Programm „EXEC" das Maschinenprogramm wieder in eine hexadezimale Form umgewandelt und in DATA-Statements übernommen werden. Es wird dann als Teil des Executer-Programms wie ein normales BASIC-Programm auf Cassette abgespeichert.

Laden Sie deshalb jetzt das EXEC-Programm.
Auf dem Bildschirm erscheint:

EXECUTE, DISASSEMBLE, BINARY, HEXBASIC

Sie können jetzt irgend eines dieser Optionen wählen, z.B. (D).

DSMBL ADDR? 033A
INSRUCS ? 6

Jetzt werden 6 Befehle ab 033A Hex Adresse disassembliert.

Zur Ausführung des Programmes geben Sie (E) ein.

```
EXECUTE ADDR ? 033A
PARAMETER ? 0
```

Um das Programm in Hexadezimal zu verwandeln und in DATA-Statements zu verwandeln, drücken Sie jetzt erst einmal (H)

```
START ADDR ?  033A
END ADDR ?    0344
DATA STMT NO? 1100
```

Nun erscheinen die DATA-Statements auf dem Bildschirm. Wenn alle vorhanden sind, erscheint READY.

Der Cursor geht in die obere Bildschirmecke und läuft hinunter zum RUN 100 Befehl. Nun kann ein neuer Befehl eingegeben werden. Die DATA-Statements sind nun Bestandteil des BASIC-EXECUTER Programmes.

Nun können Sie die gekoppelten Programme auf einer neuen Cassette mit SAVE "NUL" abspeichern.
(Weitere Informationen finden Sie im Original englischen Manual, welches mit dem Assembler geliefert wird. Best.-Nr. P25, DM 96,--
(Erhältlich beim Fachhandel oder direkt beim Verlag)

PET-Files (Dateien)

Eine der besonderen Fähigkeiten des PET ist die Möglichkeit der Speicherung von Dateien auf Cassette, sowie das Abrufen von Daten von der Cassette.

Wenn Sie Daten auf eine Cassette speichern wollen, sollten Sie wie folgt vorgehen:

1. OPEN eingeben mit nachfolgender Datei, Numerierung und Filename
2. PRINT # mit Dateiennummer
3. Schließen der Datei durch CLOSE

Der OPEN-Befehl bereitet eine Datei vor. Er gilt beim Lesen oder Schreiben von Dateien. Man kann bis zu vier Parameter als Unterscheidungsmerkmale eingeben. Eine logische Dateiennummer zwischen 1 und 255. Sie kann eine Konstante oder auch eine Variable sein. Weiterhin eine Bezeichnung für die externe Einheit, auf die ausgegeben werden soll.
1 = 1. Cassette
2 = externe Cassette

Der dritte Parameter legt die Richtung fest. Eingabe oder Ausgabe. Er kann drei verschiedene Werte annehmen, 0,1 oder 2. 0 ergibt eine Eingabe von Cassette (READ). 1 bedeutet Ausgabe (WRITE). Wenn die Datei durch CLOSE abgeschlossen wird, wird automatisch eine End of File (EOF) Markierung auf das Band geschrieben. Dies geschieht erst nach dem letzten Datenblock.

Eine zwei bedeutet auch Schreiben (WRITE), jedoch wird nach dem letzten Datenblock ein (EOF) und nach (EOF) noch eine End of Tape (EOT)-Markierung auf das Band geschrieben.

Als letzter Parameter kann eine FILE hinzugefügt werden.

Ein Statement könnte also wie folgt aussehen:

```
10 OPEN 35, 1, 1," FILE1"
```

Nun wollen wir gleich einige Versuche machen.

```
10 A = 1
20 OPEN A
RUN
PRESS PLAY ON TAPE 1
```

A darf zwischen 1 und 255 liegen. Null und Minuswerte wie z.B. –1 sind nicht erlaubt. 256 bringt einen Illegal Quantity Error. Eine String-Variable ist auch nicht erlaubt.

Konkretes Beispiel:
Wir wollen jetzt einmal Informationen auf das Band schreiben und anschließend lesen.

```
READY.

 10 OPEN 1,1,2,"PROBE"
 20 FOR J=1 TO 5
 30 PRINT# 1,J
 40 NEXT J
 50 CLOSE 1
 60 PRINT"BITTE ZURUECKSPULEN UND EINE BELIEBIGE"
 70 PRINT"TASTE DRUECKEN !"
 80 GETA$
 90 IF A$="" THEN 80
100 OPEN 1,1,0,"PROBE"
110 FOR J=1 TO 5
120 INPUT#1,X
130 PRINT X
140 NEXT J
150 PRINT"FERTIG"
160 CLOSE 1
READY.
```

```
READY.

 10 OPEN 1,1,2,"PROBE 2"
 20 FOR J=1 TO 5
 30 PRINT#1,CHR$(160+J)
 40 NEXT J
 50 CLOSE 1
 60 PRINT"BITTE ZURUECKSPULEN UND IRGEND EINE     TASTE"
 70 PRINT"DRUECKEN !"
 80 GETA$:IF A$=""THEN 80
 100 OPEN 1,1,0,"PROBE"
 110 FOR J=1 TO 5
 120 INPUT#1,X$
 130 PRINT X$
 140 NEXT J
 150 PRINT"FERTIG"
 160 CLOSE 1
READY.
```

Wir haben jetzt die Variable X (Zeile 110 und 120) ausgedruckt. Mit jedem PRINT kann nur eine Variable ausgegeben werden. Nun wollen wir einmal Zeichen (Strings) ausgeben.
Ändern Sie unser erstes Programm bitte wie folgt ab:

```
30 PRINT #1, CHR$ (160 + J)
120 INPUT #1, X$
130 PRINT X$
```

Das geänderte Programm ließt uns jetzt Zeichen auf die Cassette. (Strings) Es können auch Buchstaben geschrieben werden, je nach der Zahl in Klammern in Zeile 30.

Die Zahlenwerte können Sie der Tabelle im Abschnitt über PET-Graphik entnehmen . Versuchen Sie jetzt einmal selbst Zahlen und Strings zu verwenden. Jetzt wollen wir einmal zwei Spalten ausgeben. Das Programm wird wie folgt geändert.

```
30  PRINT #1, CHR$ (160+J);", "; CHR$ (210 + J)
120   INPUT #1, X$, Y$
130   PRINT X$, Y$
```

Genau wie Strings, können wir jetzt Variable eingeben.

Bis jetzt haben wir immer nur fünf oder zehn Zeichen oder Zahlen in unsere Datei eingeschrieben. Wenn jedoch eine große Menge von Daten ansteht, müssen mehrere Datenblocks auf die Cassette geschrieben werden. In einer alten PET-Version wurden diese einzelnen Blöcke zu eng aneinander geschrieben, so daß beim Lesen der Files Fehler entstanden.

Nun wollen wir uns ein kleines Unterprogramm ansehen, welches dafür sorgt, daß wir keine Probleme bei der Eingabe bekommen.

```
6500 IF Z < = PEEK (625) GOTO 6540
6510 POKE 59411,53
6520 Z = 1 TO 60:NEXT Z
6530 POKE 59411,61
6540 Z = PEEK (625): RETURN
```

Diese Unterroutine sollte nach jedem PRINT # Befehl aufgerufen werden. Die Routine gilt nur für den eingebauten Cassettenrecorder, wenn Sie den 2. Cassettenrecorder verwenden wollen, wird POKE 59411,53 zu POKE 59456,207 und POKE 59411,61 wird zu POKE 59456,223.

Erweitert sieht ein solches Hilfsprogramm wie folgt aus:

```
 63000 REM SCHREIBEN AUF CASSETTE 1
 63010 REM A6$ ENTHAELT DIE ZEILE IN
 63020 REM DIE GESCHRIEBEN WERDEN SOLL
 63030 T6=TI:IF LEN(A6$)<190 THEN 63070
 63040 PRINT#1,LEFT$(A6$,189);
 63050 IF TI-T6>120THEN GOSUB 63100
 63060 T6=TI:PRINT#1,RIGHT$(A6$,LEN(A6$)-189)
 63065 GOTO 63080
 63070 PRINT#1,A6$
 63080 IF TI-T6>120 THEN GOSUB 63100
 63090 RETURN
 63100 REM EINSCHALTEN DES CASSETTENRECORDERMOTORES FUER 0.1
       SEKUNDE
 63110 POKE59411,53:T6=TI
 63120 IF TI-T6<6 THEN 63120
 63130 POKE 59411,61:RETURN
 63150 PRINT"DRUECKE EINE TASTE WENN FERIG!"
 63160 REM ABFRAGEN DER TASTE
 63170 GET A6$:IF A6$="" THEN 63170
 63180 RETURN
READY.
```

Der PRINT # Befehl druckt nach Einlesen der Daten-Cassette alles genau das aus, was eingegeben wurde, ohne Rücksicht auf evtl. Zeilenrücklauf etc.

Beispiel:

```
10 OPEN 1, 1, 1
20 X$ = " ABCDEFGHJK"
30 FOR I = 1 TO 10
40 PRINT X$;
50 PRINT # 1, X$;
60 NEXT I
70 CLOSE 1
```

Sehen Sie sich dieses Programm einmal an. Bei mehr als 40 Zeichen wird kein RETURN eingesetzt. Die Daten gehen einfach in die nächste Zeile über.

Die Größe des Eingabepuffers
Versuchen Sie einmal folgendes Programm:

```
10 INPUT X$
20 PRINT X$
RUN
```

Jetzt geben Sie ca. 50 – 100 Zeichen ein und drücken RETURN. Sie werden sehen, daß der PET nur 80 Zeichen in seinen Eingangspuffer aufnehmen kann. Wenn mehr eingegeben werden, gehen die ersten 80 Zeichen verloren.

```
READY.

 10 PRINT"LADEN AUF EINE DATENCASSETTE"
 20 GOSUB 63150
 30 OPEN 1,1,2,"DATAA"
 40 FOR X=1 TO 10
 50 FOR Y=1 TO 5
 60 A6$=STR$(X*Y)
 70 GOSUB 63000
 80 NEXT Y
 90 PRINT"AUFNAME";X;"BEENDET"
 100 NEXT X
```

```
110 CLOSE 1
120 PRINT"SCHREIBEN BEENDET"
130 STOP
150 NEXT Y
190 PRINT"SCHREIBEN BEENDET"
63000 REM SCHREIBEN AUF CASSETTE 1
63010 REM A6$ ENTHAELT DIE ZEILE IN
63020 REM DIE GESCHRIEBEN WERDEN SOLL
63030 T6=TI:IF LEN(A6$)<190 THEN 63070
63040 PRINT#1,LEFT$(A6$,189);
63050 IF TI-T6>120THEN GOSUB 63100
63060 T6=TI:PRINT#1,RIGHT$(A6$,LEN(A6$)-189)
63065 GOTO 63080
63070 PRINT#1,A6$
63080 IF TI-T6>120 THEN GOSUB 63100
63090 RETURN
63100 REM EINSCHALTEN DES CASSETTENRECORDERMOTORES FUER 0.1
      SEKUNDE
63110 POKE59411,53:T6=TI
63120 IF TI-T6<6 THEN 63120
63130 POKE 59411,61:RETURN
63150 PRINT"DRUECKE EINE TASTE WENN FERIG!"
63160 REM ABFRAGEN DER TASTE
63170 GET A6$:IF A6$="" THEN 63170
63180 RETURN
READY.
```

Laden von Daten von der Cassette in den Speicher

Das Laden von einer Datencassette (File) in den Speicher erfolgt durch den INPUT#Befehl.

Nach INPUT# erfolgt die logische File-Numerierung und eine Variable.

Beispiel: INPUT # 1,A

Die logische File-Nummer muß mit der im vorherherigen OPEN-Befehl festgelegten File-Nummer übereinstimmen. Die dritte Zahl im OPEN-Statement muß jedoch eine 0 sein, damit von der Cassette gelesen werden kann, z.B. OPEN 1,1,0,"DATA".

Der INPUT-Befehl liest von einer Cassette auf die gleiche Weise, wie über die Tastatur eingeschrieben wird. Bei der Zahleneingabe müssen die einzelnen Daten durch Komma getrennt werden. Mit dem Befehl INPUT # können auch Zeichenketten gelesen werden. Komma und einige andere Sonderzeichen werden dann auch auch als Strings angesehen, es sei denn, man setzt sie in Anführungszeichen.

Ein Programm zum Abspeichern und Einlesen eines DATA-Files

Das nachfolgende Programm stellt eine gute Grundlage für das Arbeiten mit Dateien dar. Das Programm enthält ein Menue und hat vier Betriebsarten.

1. Speichern auf Cassette
2. Laden von Cassette
3. Laden von Cassette mit Anzeige des Cassetteninhaltes
4. Rückkehr und Stop

Das gezeigte Listing soll als Demonstrationsmodel dienen, Sie können es dann für Ihren speziellen Anwendungsfall abändern.

```
READY.

 9 GOTO 900
 10 PRINT"3LADEN AUF EINE DATENCASSETTE"
 2GOSUB 63150
 30 OPEN 1,1,2,"DATAA"
 40 FOR X=1 TO 10
 50 FOR Y=1 TO 5
 60 A$=STR$(X*Y)
 70 GOSUB 63000
 80 NEXT Y
 90 PRINT"AUFNAME";X;"BEENDET"
 100 NEXT X
 110 CLOSE 1
 120 PRINT"SCHREIBEN BEENDET"
 130 STOP
 150 NEXT Y
 190 PRINT"SCHREIBEN BEENDET"
 200 END
 210 PRINT"3LADEN VON DATEN VON DER CASSETTE"
```

```
220 GOSUB 63150
230 OPEN 1,1,0,"DATAA"
240 FOR X=1 TO 10
250 FOR Y=1 TO 5
260 INPUT#1,Z
270 IF Z<>X*Y THEN 330
280 NEXT Y
290 PRINT"AUFNAHME";X;"IST IN ORDNUNG"
300 NEXT X
310 PRINT"ALLES IN ORDNUNG UND FERTIG"
320 STOP
330 PRINT"FEHLER IN";X;Y
340 STOP
350 GOTO 280
400 END
500 REM ANZEIGE DER DATEN VON CASSETTE
510 PRINT"3LOAD VON CASS UND ANZEIGE
520 GOSUB 63150
530 OPEN 1,1,0,"DATAA"
540 FOR X= 1 TO 3
550 B$="":C$=""
560 GET#1,A$
570 IF A$=CHR$(13) THEN 630.
580 B$=B$+"  "+A$
590 Z$=STR$(ASC(A$))
600 IF LEN(Z$)<3THENZ$="  "+Z$:GOTO600
610 C$=C$+Z$
620 GOTO 560
630 PRINT B$
640 PRINT C$
650 NEXT X
660 STOP
670 END
900 PRINT"3WAEHLE EINE MOEGLICHKEIT"
9q0 FOR M=1 TO 1000:NEXT M
1000 PRINT"31 SPEICHERN AUF CASSETTE"
1010 PRINT"2 LADEN VON CASSETE"
1020 PRINT"3 LADEN UND ANZEIGEN VON CASSETE"
1030 PRINT"4 STOP UND RUECKKEHR"
1040 GET D$
1050 IF D$="" THEN 1040
1060 D=VAL(D$)
1070 ON D GOSUB 10,210,500,1600
```

```
1080 GOTO 900
1600 PRINT"3AUF WIEDERSEHEN!"
1601 PRINT"IHR HOFACKER VERLAG MUENCHEN
6END
63000 REM SCHREIBEN AUF CASSETTE 1
63010 REM A6$ ENTHAELT DIE ZEILE IN
63020 REM DIE GESCHRIEBEN WERDEN SOLL
63030 T6=TI:IF LEN(A6$)<190 THEN 63070
63040 PRINT#1,LEFT$(A6$,189);
63050 IF TI-T6>120THEN GOSUB 63100
63060 T6=TI:PRINT#1,RIGHT$(A6$,LEN(A6$)-189)
63065 GOTO 63080
63070 PRINT#1,A6$
63080 IF TI-T6>120 THEN GOSUB 63100
63090 RETURN
63100 REM EINSCHALTEN DES CASSETTENRECORDERMOTORES FUER 0.1
      SEKUNDE
 63110 POKE59411,53:T6=TI
 63120 IF TI-T6<6 THEN 63120
 63130 POKE 59411,61:RETURN
 63150 PRINT"DRUECKE EINE TASTE WENN FERIG!"
 63160 REM ABFRAGEN DER TASTE
 63170 GET A6$:IF A6$="" THEN 63170
 63180 RETURN
READY.
```

Der erste Teil des Programmes lädt Daten auf die Cassette. (Zeile 30 bis 200) Der zweite Teil lädt Daten von der Cassette in den Computer und kontrolliert, ob die Daten richtig gelesen wurden. (Zeile 210 bis 400). Ab Zeile 500 finden Sie ein Programm zur Anzeige der von Cassette geladenen Daten.

Der Befehl GET# und ST in Verbindung mit Dateien

Am Schluß dieses Kapitels wollen wir noch ganz kurz die Befehle GET und ST besprechen.

Der GET#Befehl liest ein Zeichen von einer Datencassette auf die gleiche Weise, wie er sich ein Zeichen von der Tastatur herholt. GET kann in Verbindung mit numerischen Variablen und Zeichenketten (Strings) verwendet werden. In Verbindung mit Variablen ist der GET #Befehl nur mit Vorsicht anzuwenden, da nur die Zeichen 0–9,

+ und – und der Punkt verwendet werden dürfen. Es wird daher empfohlen, möglichst nur in Verbindung mit Strings zu arbeiten.

Ein einfaches Programm zur Darstellung des Inhaltes einer Datencassette:

```
10 OPEN 1,1,0,"PROBE"
20 GET #1,A$
30 PRINT A$;
40 GOTO 20
```

Verwenden Sie nun einmal dieses kleine Programm, um die Daten zu lesen, die Sie mit Listing 1 und RUN 10 auf einer separaten Cassette erzeugen können.
Es liest Ihnen die Zahlen 1 bis 5 und eine zusätzliche 1 aus und druckt dies auf dem Bildschirm aus.
Die Cassette läuft weiter. Die zusätzliche Eins stammt aus einer Endmarkierung. Sie kann zusammen mit dem ST-Befehl zum Anhalten der Cassette verwendet werden.
Dies könnte dann wie folgt aussehen:

```
10 OPEN 1,1,0"PROBE"
20 GET #1, A$
30 IF ST > 0 THEN 60
40 PRINT A$;
50 GOTO 20
60 CLOSE 1
```

ST ist eine besondere BASIC-Variable, die nach jeder Ein-/Ausgabe Operation vom Computer festgelegt wird. Die einzelnen Bits dieses Statuswortes ST werden entsprechend den Bedingungen bei einer Ein-/Ausgabeoperation gesetzt. Mit Hilfe der UND-Funktion kann dieses Statuswort vom BASIC her abgefragt werden. Der Code sieht wie folgt aus:

4 = Kurzer Block von Daten
8 = Langer Block von Daten
16 = Fehlermeldung
32 = Checksumme ist fehlerhaft
64 = Ende eines Files
128= Ende des Bandes

Werden mehrere Bits gesetzt, so addieren sich die Stellenwerte.
Z.B., wenn eine Aufprüfung ST = 28 ergibt, so sind 4 + 8 + 16 gesetzt.
Will man z.B. das Ende einer Datei anzeigen, kann man

```
70 IF (ST) AND 64 THEN PRINT " DATEIEN– ENDE"
```

eingeben.

Ein Abschließen der Datei könnte auch über

```
80 IF (ST) AND64 THEN 90
90 CLOSE 1
```

erfolgen.

Aufzeichnung eines Programmes in ASCII auf Cassette

Oft ist es von Vorteil, wenn man ein Programm als Datenfile lesen kann. Da mit dem INPUT-Befehl nicht alle Zeichen direkt übernommen werden, läßt sich durch das Abspeichern in ASCII Zeichen ein brauchbares Ergebnis erzielen.

Mit

```
OPEN 1,1,2"NAME"
```


```
CMD1,1
LIST
CLOSE 1
```

läßt sich ein Programm im ASCII auf Cassette speichern. Es kann dann als Datenfile gelesen werden. Der Computer kann damit arbeiten und evtl. Änderungen durchführen und wieder auf Cassette abspeichern. Eine Übernahme in den Arbeitsspeicher könnte über den Bildspeicher und den Cursor geschehen. (Siehe Programmiertricks)

Messen von Zeiten (256 Mikrosekunden)

Der eingebaute PIA6522 im PET bietet Ihnen die Möglichkeit, mit einem eingebauten Zeitgeber genaue Zeitmessungen durchzuführen. Der Zeitgeber wird durch verschiedene Register gesteuert, die vom Programm her angesteuert werden können. Der Zeitgeber zählt von 255 auf Null herunter und fängt dann wieder bei 255 von vorne an.

In einfachen Applikationen ist es möglich, den Intervall-Zeitgeber im 6522 vom BASIC-Programm aus über PEEK und POKE zu erreichen. PEEK (A) bringt den Inhalt einer Adresse A auf den Bildschirm. A kann hierbei eine Adresse eines Speicherplatzes oder eines Eingangsregisters sein.

POKE A,B bringt den Wert B in die Adresse A, die auch wiederum eine Speicherzelle oder ein Register sein kann. Das nachfolgende Programm greift über PEEK und POKE in den Intervalltimer und mißt die Zeit zwischen zwei Befehlen in Zeile 100 und Zeile 200.

```
1 DT = 0: TH = 59465:TA = 246
100 POKE TH,0
200 DT = TA = PEEK (TH)
```

Ein Ergebnis von DT = 35 besagt, daß 8,96 Millisekunden (35 Einheiten x 256 Microsekunden) zwischen den Statements in Zeile 100 und 200 vergangen sind. Der Maximalwert für DT ist 246.

Die Methode eignet sich bestens zur Messung von Zeitintervallen kleiner 63 Millisekunden.

Messung von Einheiten im Bereich von einer Microsekunde
Der Intervalltimer hat die Registeradresse 59464. Dieser zählt jede Microsekunde. Wenn er bei Null angelangt ist, veranlaßt er den langsameren 256 Microsekunden-Zeitgeber dazu, jetzt um Eins weiter herunterzuzählen.

Messen verschiedener Einheiten

Das erste Betriebssystem des PET verwendet die folgenden Speicherplätze, um TI und TI$ zu berechnen.

PEEK (514)	zählt von 1 bis 255 in ca. 4 Sekunden
PEEK (513)	zählt von 0 bis 255 in 18 Minuten
PEEK (512)	zählt von 0 bis 80 in 24 Stunden

Für Ihre Zeitmessungen in Programmen oder für externe Steuerung von Zeitabläufen können Sie die obigen Adressen in Ihrem Programm verwenden.

BASIC-Befehls Vergleichsliste

SPC(X) In vielen Fällen kann dieser Befehl durch TAB(X) ersetzt werden.
Er läßt sich auch durch folgende Routine ersetzen:

```
10 FOR I = 1 TO X
20 PRINT "      "
30 NEXT I
```

: # # # .# # Dies ist eine Zeile aus einem PRINT USING-Befehl. PRINT USING läßt sich nicht direkt ersetzen. Am besten, man zerstört diese Zeile und setzt anstelle von PRINT USING einfach PRINT.

```
MAT READ A FOR I = 1 TOX
           FOR J = 1 TO Y
           READ A (I,J)      A (X,Y)
           NEXT J
           NEXT I
```

```
MAT INPUT AFOR I = 1 TOX
           FOR J = 1 TOY
           INPUT  A (I,J)    A(X,Y)
           NEXT J
           NEXT I
```

```
MAT A=ZER  FOR I = 1 TO X
           FOR J = 1 TO Y
           A (I,J) = 0       A (X,Y)
           NEXT J
           NEXT I
```

```
MAT A=CON  FOR I = 1 TO X
           FOR J 1 TO Y
           A (I,J) = 1
           NEXT J
           NEXT I
```

A(X,Y)

MAT READ A,B, C

Wobei A,B und C als A(X,Y), B(R,S), C (V,W)

```
FOR I = 1 TO X
FOR J = 1 TO Y
READ A (I,J)
NEXT J
NEXT I
FOR I = 1 TO R
FOR J = 1 TO S
READ B (I,J)
NEXT J
NEXT I
FOR I = 1 TO V
FOR J = 1 TO W
READ C(I,J)
NEXT J
NEXT I
```

RANDOMIZE: Diesen Befehl können Sie aus den Programmen streichen, wenn Sie die RND(X) Funktion vom PET entsprechend richtig festlegen. Achten Sie bitte auf den Wert X.

Ein-/Ausgabe-Programmierung mit PET

Ein-/Ausgabeprogrammierung beim PET

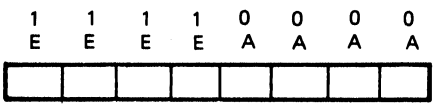
Der PET besitzt, wie die meisten Anwender wissen, einen IEEE488 Standardbus zum Anschluß externer Geräte. Für provisorische und schnelle Ein-/Ausgabeprogrammierung eignet sich jedoch der parallel Ein-/Ausgabeport besser. Er läßt sich wesentlich einfacher handhaben als der IEEE488 Bus.

Der Ein-/Ausgabeport wird durch das Datenrichtungsregister gesteuert. Es kann jede einzelne Leitung wahlweise als Eingang oder als Ausgang durch das Programm festgelegt werden. Dies geschieht durch den Befehl:

POKE 59459,0 Alle Ports werden Eingänge
POKE 59459,255 Alle Ports werden Ausgänge

Will ich nur vier der acht I/O-Leitungen zu Eingängen programmieren und die restlichen vier sollen Ausgänge bleiben, so gebe ich ein:

POKE 59459,240



Jetzt sind die Ports PA0–PA3 Eingänge und die Ports PA4–PA7 Ausgänge. 240 ist das dezimale Äquivalent der Binärzahl 11 11 000. So können Sie jede beliebige Kombination zwischen 0 und 255 poken und bestimmte Ports zu Ein- oder Ausgängen programmieren.

Damit wäre erst einmal die Richtung des Datenflusses festgelegt. Übrigens, diese Konfiguration bleibt so lange erhalten, bis Sie geändert wird oder der PET abgeschaltet wird.

Nach dem Einschalten des PETs schaltet er automatisch alle I/O-Leitungen auf Eingang. Sie können dies leicht nachprüfen, indem Sie

```
10 ?PEEK(59459)
RUN
```

eingeben.

Nachdem wir jetzt die Richtung des Datenflusses festlegen können, wollen wir versuchen, Daten auf die Ports zu geben. Hierzu gibt es, im PET eine besondere Speicherzelle mit der dezimalen Adresse 59457 = (Hex. = E841).

Alles was Sie nach POKE 59459,255 in diese Zelle hineinschreiben, erscheint am Ausgangsport PA0 – PA7. Alles was Sie nach POKE 59459,0 aus dieser Speicherzelle herauslesen, wurde von außen an den Port angelegt. (5V–TTL oder C MOS Pegel). Im offenen Zustand finden Sie eine log. 1, bei nach Masse verbundenen Ports wird eine log. 0 erkannt.

So, jetzt wollen wir zum besseren Verständnis etwas mit diesen Ausgangsleitungen „spielen".

```
READY.

 10 REM MINIDEMO I/O
 20 POKE59459,255
 30 FOR I=0 TO 255
 40 POKE59457,I
 60 FOR J=0 TO 255
 65 FOR Y=1TO1000:NEXTY
 70 PRINT"DER INHALT DES EIN/AUSGABE-"
 75 PRINT"REGISTERS IST:";PEEK(59457)+1
 80 NEXTI:NEXTJ
 90 NEXT N
 95 GOTO 30
READY.
```

Dieses Programm setzt die Ausgangsports A0–A7 nacheinander auf die verschiedenen möglichen Werte. Nach dem Durchlaufen der Schleife bleibt das Ausgaberegister auf 255 = 11111111 stehen.
Sie können jetzt dieses kleine Programm so erweitern, daß Sie jeden gewünschten Ausgangswert oder bestimmte Folgen auf dem Port erzeugen können. Wir wollen jetzt einmal versuchen, ein Lauflicht zu programmieren. Hierzu verwenden wir das obenstehende Programm als Grundlage. Im obigen Programm wird in Zeile 30 durch die FORNEXT-Schleife jede ganze Zahl zwischen 0 und 255 erzeugt. D. h., auf dem Port erscheinen innerhalb eines Durchlaufes alle möglichen Kombinationen.

Beim Lauflicht soll aber immer nur ein Ausgang aktiviert werden.
Wir erreichen dies, indem wir nur die Zahlen 2, 4, 8, 16, 32, 64 und 128 poken. Auf diese Weise entsteht zwischen den Ausgängen eine lauflichtartige Weiterbewegung.

Die Grundgeschwindigkeit können Sie durch Ändern der Verzögerungsschleife in Zeile 65 erreichen.

```
READY.

 4 REM COPYRIGHT C ING.W.HOFACKER GMBH
 5 REM MINIDEMO I/O NR 2
 7 REM LAUFLICHT
 8 REM IN ZEILE 65 BESTIMMT DIE
 9 REM ZAHL 100  DIE GESCHWINDIGKEIT
 10 REM MINIDEMO I/O
 20 POKE59459,255
 23 FOR I=1TO 7
 24 X=2↑I
 26 PRINT X
 30 POKE59457,X
 60 FOR J=0 TO 255
 65 FOR Y=1TO100 :NEXTY
 70 PRINT"DER INHALT DES EIN/AUSGABE-"
 75 PRINT"REGISTERS IST:";PEEK(59457)
 80 NEXT I
 95 GOTO 23
READY.
```

Die gewünschten Werte für POKE können auch aus Datastatements gelesen werden. Hierzu soll das nachfolgende Beispiel dienen. DATA 300 wird eingegeben, damit kein „OUT OF DATA ERROR" erscheint.
Zur Kontrolle druckt das Programm den Status der Ports noch einmal aus. Die Geschwindigkeit läßt sich durch Zeile 25 leicht ändern.

```
READY.

 10 REM MINIDEMOI/O NR.3
 15 READ NUM
 20 IF NUM>255 THEN RESTORE:GOTO 15
 30 POKE59457,NUM
 40 GOTO 15
 100 DATA 1,2,4,8,16,32,64,128
 110 END
READY.
```

```
READY.

 10 REM MINIDEMOI/O NR.3
 15 READ NUM
 20 IF NUM>255 THEN RESTORE:GOTO 15
 25 FOR I=1TO100:NEXT
 30 POKE59457,NUM
 35 PRINT PEEK(59457)
 40 GOTO 15
 100 DATA 1,2,4,8,16,32,64,128
 110 DATA 300
READY.
```

Zum Schluß:

Jetzt haben Sie einige Grundlagen der Ein-/Ausgabeprogrammierung kennengelernt. Sie können jetzt selbst die verschiedensten Konfigurationen und Steuerungen entwerfen und ausführen. Sie brauchen dazu keinen Lötkolben.

Alles kann über die Tastatur geschehen. Wenn Sie z. B. die Geschwindigkeit des Lauflichtes ändern wollen, brauchen Sie keinen Kondensator oder Widerstand zu ändern. Sie schreiben einfach in Zeile 25 des Programms „Minidemo3" anstelle der 100 eine 200 und schon ist die Frequenz geändert.

Als wertvolles Hilfsmittel zur Kontrolle der Zustände am Port könnten Sie auch das Experimentierboard und den Logiktester aus dem Artikel „Programmexperimente" in Heft 1/79 verwenden. Wir wünschen Ihnen viel Spaß!

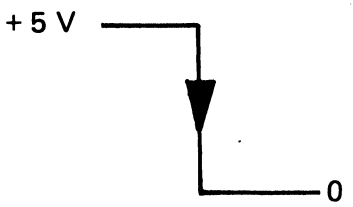
Zum Thema: Ein-/ Ausgabeprogrammierung

Wir haben bis jetzt nur die 8 Port Ausgänge am PET User Port betrachtet. Neben diesen beiden 8 Leitungen, die wahlweise als Eingänge oder als Ausgänge geschaltet werden können, haben wir dort am User Port noch die Klemmen CA1 und CB2. Diese Anschlüsse sind sog. „Handshake Lines" und dienen zum Quittungsaustausch mit externen Geräten.

CA1 Anschluß B auf der Steckerleiste (CA1) ist der Anschluß für ankommende Quittungssignale

CB1 Anschluß M auf der Steckerleiste (CB2) ist der Anschluß für ausgehende Quittungssignale

Im Ausgangszustand verhält sich CA1 so, daß auf eine negative Flanke getriggert wird, d. h. ein Impulsübergang von log. „1" auf log. „0" wird erkannt und registriert.

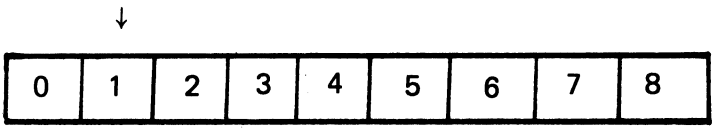


Wenn Ihr externes System eine positive Flanke erzeugt, also ein Impulsübergang von log. „1" nach log. „0" stattfindet, so können Sie auch diesen Impuls erkennen, wenn Sie das Quittungsverhalten des CA1 über BASIC ändern.
POKE 59468, PEEK (59468) OR1. Dadurch wird das Kontrollbit im VIA-Baustein 6522 (Siehe auch im Datenblattanhang) von 0 auf 1 geändert. Die Änderung geschieht im Peripheral Control Register, deren Bit 0 (CA1-Control) das Interrupt Bit enthält. Wenn dieses Bit eine log. „0" ist, so wird eine negative Flanke erkannt, ist es log „1", wird eine positive Flanke erkannt. Entsprechend wird ein Unterbrechungsflag gesetzt.

Wenn an Pin B ein positiver Impuls erscheint und das Unterbrechungsflag gesetzt wird, so bedeutet dies, daß ein externes Gerät oder Schaltung bereit ist, Daten auf die 8 Eingangsports zu geben.

Jetzt wird im Interrupt-Flag-Register im 6522 ein Bit gesetzt.

Im Interrupt Flag - Register wird Bit 1 im Zusammenhang mit dem CA1 Pin verändert.



Das Interrupt-Flag-Register hat die dezimale Adresse 59469. Vom BASIC her können Sie auf dieses Bit durch den Befehl: WAIT 59469,2 aufprüfen. Dieser Befehl stoppt die Programmausführung in BASIC solange, bis der Inhalt des Registers 59469 mit dem Wert 2 logisch UND verknüpft einen Wert von 1 ergibt.
Die logische UND-Verknüpfung führt jedoch nur zu einer log. „1" am Ausgang, wenn alle Eingänge log. „1" sind, d. h. der Inhalt des Flag-Registers (59469) wird mit dem Wert 2 = binär 0000 0010 logisch verknüpft und auf den Wert 1 geprüft. Im Normalfalle befindet sich im Bit 2 des Kontrollregisters eine 0. Mit WAIT 59469,2 ergibt die logische UND-Verknüpfung eine Null. D. h. BASIC arbeitet nicht. Erst wenn eine positive Flanke erscheint, und das Bit 2 des Kontrollregisters zu 1 macht, ergibt die logische UND-Verknüpfung eine 1 und der BASIC-Interpreter beginnt wieder zu arbeiten. Seien Sie deshalb vorsichtig, wenn Sie WAIT 59469,2 eingeben, da das Programm dann BASIC verläßt und auf einen Impuls an CA1 wartet. Sie können dann nur wieder ins BASIC zurück, wenn Sie abschalten oder einen Impuls + 5 V an CA1 bringen.

Nach Ausführung des WAIT-Befehles können die Daten an den Ports PA0 bis PA7 eingelesen werden.

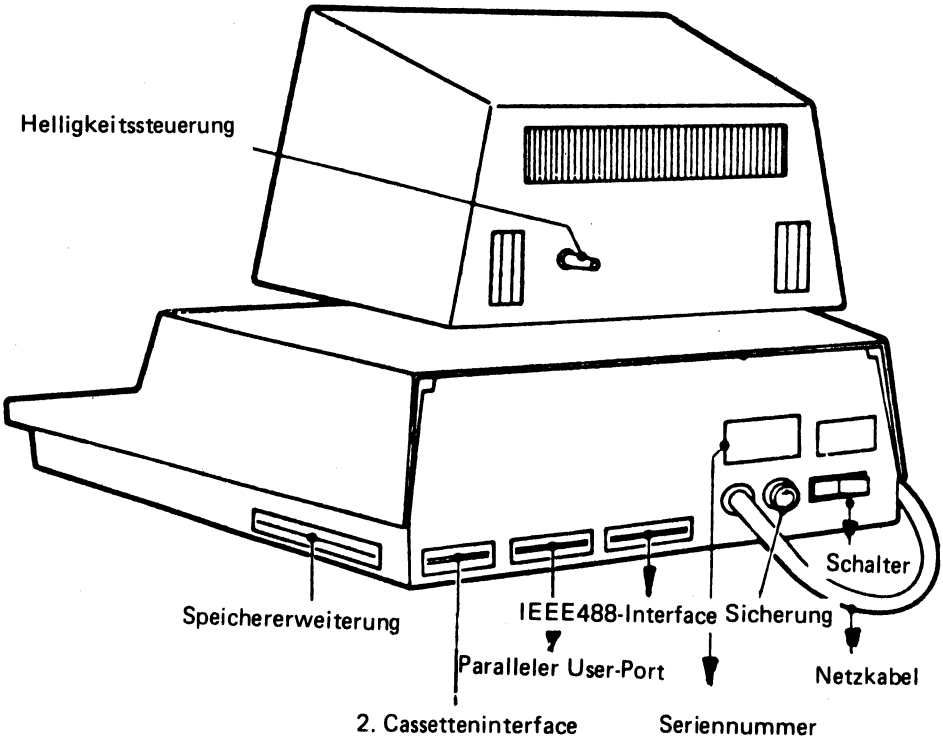
```
N = PEEK (59457)
```

59457 ist hier die Adresse für das Ein-/Ausgabe-Register A mit Quittungsbetrieb (mit Handshaking). Der Inhalt des Registers wird nun gelesen und als Variable N mit einem dezimalen Wert zwischen 0 und 255 versehen. (Je nachdem, was von außen angelegt wurde.) Wie schon erwähnt, wird das CA1-Flag durch Lesen wieder zurückgesetzt und kann sofort wieder zum Erkennen nachfolgender Daten verwendet werden. Jetzt haben Sie die Funktion des CA1 Einganges etwas näher kennengelernt. Wir empfehlen Ihnen, einige Experimente damit durchzuführen. Sie können jetzt z.B. ganze ASCII-Zeichen in eine String-Variable einlesen.

```
100  A$ = "      "
120  FOR J = 1 TO 72
130  WAIT 59469,2
140  N = PEEK (59457) AND 127
150  IF N = 13 THEN 180
160  A$ = A$ + CHR$ (N)
170  NEXT J
180  PRINT A$
```

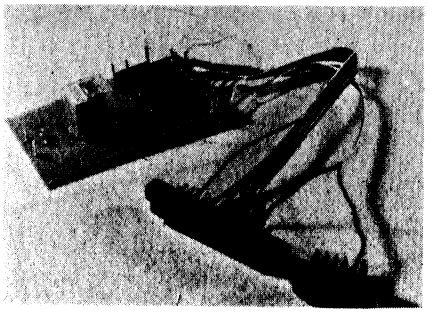
Mit diesem Programm können bis zu 72 ASCII-Zeichen eingelesen werden. Die Eingabe ist auf 7 Bit begrenzt. Werden andere Zeichen eingegeben, kann das Programm durch ein Wandler-Programm in BASIC abgeändert werden.

Die Anschlüsse des PET



Programmiertechnik für PET

Ein praktisches Ausgabeinterface für den PET



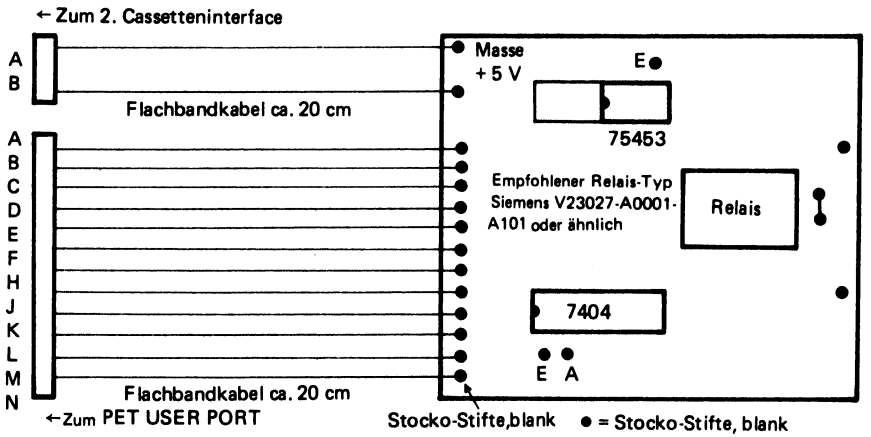
Für verschiedene Anwendungen wie Steuerungen, Musikerzeugung etc. benötigt man beim PET immer wieder einen Stecker mit Anschlußkabel. Hier hat es sich bewährt, wenn man eine kleine, steckbare Platine anfertigt, welche es ermöglicht, daß man die einzelnen Ein- oder Ausgänge leicht mit Steckern erreichen kann.

Wir haben für diesen Zweck eine kleine Zusatzschaltung aufgebaut, die wir Ihnen heute gerne vorstellen möchten.

Für die softwaremäßige Behandlung der Zusatzschaltung gibt es ein kleines BASIC-Programm. Mit diesem Programm können Sie folgende Steuerfunktionen programmieren

1. Ein-/Aus Wechsel in bestimmten Abständen (auf die Sekunde genau)
2. Ein/Aus Wechsel zu einer bestimmten Zeit
3. Stellen der internen Uhr
4. Einschalten über die Tastatur
5. Ausschalten über die Tastatur
6. STOP

Schaltung der Platine

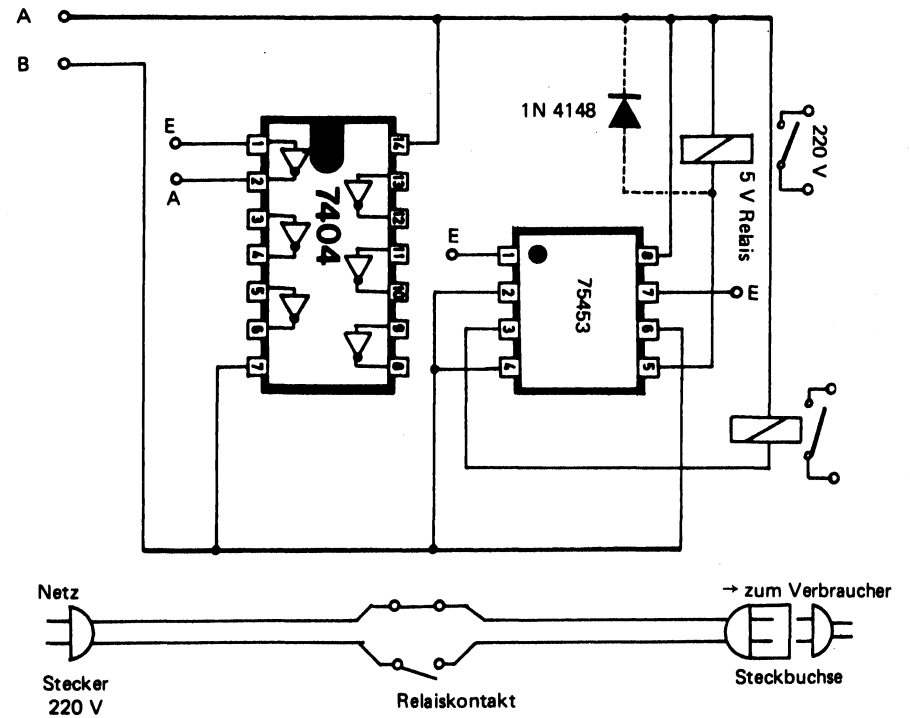


Stecker für USER PORT CINCH 251-12-90-160
A = Masse
B = CA1
C–L = PA0 bis PA7
M = CB2
N = Masse

Überall wo wir Stocko-Stifte gesetzt haben, können wir später mit den zugehörigen Steckern die gewünschten Verbindungen herstellen.

Am Anschluß für den zweiten Cassettenrecorder (von hinten gesehen, links neben dem USER PORT-Anschluß) lassen sich über einen kleinen Stecker (CINCH 250-06-90-170) +5V und Masse abnehmen. Wir brauchen also keine externe Stromversorgung.

vom 2. Cassettenrecorder-Anschluß



Schaltung für das Ausgabeinterface

Von den blanken Steckerstiften (A-N) lassen sich nun Verbindungen zum Eingang des Relaistreibers 75453 oder zum Eingang des Inverter-Treibers 7404 herstellen. Das Relais ist mit einer der beiden Treiberstufen im 75453 fest verbunden. Für ein weiteres Relais ist noch ein Treiberelement im Baustein 75453 enthalten.

Als Verbraucher haben wir bei unseren Versuchen kleine 220 V Wechselstromverbraucher, wie Stereoanlage, Radiogerät etc. verwendet.

Programm zur Steuerung des Ausgabeinterfaces

```
READY.

 10 POKE59459,255
 20 GOTO3000
 100 REM EINSCHALTEN
 110 POKE59457,255
 120 RETURN
 200 REM AUSSCHALTEN
 210 POKE59457,0
 220 RETURN
 300 REM WECHSELN
 310 IF PEEK(59457)=0THEN POKE59457,255:PRINT"EIN ";:GOSUB400:PRINT:RETURN
 320 IF PEEK(59457)=255THEN POKE59457,0
 325 PRINT"AUS ";
 330 GOSUB400:PRINT:RETURN
 400 PRINTLEFT$(TI$,2)":"MID$(TI$,3,2)":"RIGHT$(TI$,2)
 410 RETURN
 1000 REM EIN/AUS ALLE N SEKUNDEN
 1010 INPUT"ZEITDAUER ZWISCHEN WECHSELN";N
 1015 GOSUB300
 1020 N=N*60
 1030 T=TI+N
 1035 GETC$:IFC$<>""THEN RETURN
 1040 IFTI<TTHEN1035
 1050 GOSUB300
 1060 GOTO1030
 2000 REM WECHSELN BEI DER UHR
 2005 GOSUB400
 2010 PRINT"ZU WELCHER ZEIT WILLST DU SCHALTEN"
 2020 PRINT"DRUECKE 000000 WENN FERTIG"
 2030 FORMX=1 TO 10
 2040 PRINT"ZEIT ";MX
 2050 INPUT T$
 2060 IF LEN(T$)<>6THENPRINT"DIE ZEIT MUESSEN 6 ZAHLEN SEIN":GOTO2040
 2070 T(MX)=VAL(T$)
 2080 IFT(MX)=0THEN2100
 2090 NEXTMX
 2100 FORJ=1TOMX:PRINTT(J):NEXT
 2110 I=1
 2115 PRINT
 2130 IFT(I)<VAL(TI$)THEN2200
 2140 PRINT"1========";:GOSUB400
 2150 GOTO2130
 2200 GOSUB300
 2210 I=I+1
 2220 IFT(I)=0THENRETURN
 2230 GOTO2115
 2800 PRINT"WIEVIEL UHR IST ES?"
 2810 INPUT TIME$:RETURN
 2999 PRINT"AUF WIEDERSEHN!!!!":END
 3000 PRINT"3WAEHLE EINE MOEGLICHKEIT
 3010 PRINT"1. WECHSELN IM ZEITRAUM
 3020 PRINT"2. WECHSELN ZU BESTIMTEN ZEITEN
 3030 PRINT"3. STELLEN DER UHR
 3040 PRINT"4. EINSCHALTEN
 3050 PRINT"5. AUSSCHALTEN
 3060 PRINT"6. STOP
 3100 GETC$
 3110 IFC$=""THEN3100
 3200 C=VAL(C$)
 3210 ON C GOSUB 1000,2000,2800,100,200,2999
 3220 GOTO3000
READY.
```

Programmiertricks für PET

Programmiertricks für den PET

Dieser Artikel soll Ihnen einige einfache Programmiertricks zeigen.
Die einzelnen Demo-Programme befinden sich hintereinander auf der Cassette. Lassen Sie die „Play-Taste" gedrückt und geben Sie SHIFT RUN zum Laden des nächsten Programmes ein!
Die Anführungszeichen können Sie beim SAVE-Vorgang am Ende des Namens weglassen. Dasgleiche gilt für PRINT-Statements, bei denen am Schluß keine Leerzeichen mehr kommen. Listen Sie dieses Programm einmal aus und Sie werden dies leicht erkennen können.

Mehrfaches Speichern von Programmen hintereinander auf einer Cassette.

Nehmen wir an, Sie wollen ein Programm mit dem Namen WPP dreimal hintereinander auf einer Cassette abspeichern.
Was müssen Sie tun?
Sie geben ein SAVE " WPP und drücken die RETURN-Taste. Nun erscheint auf dem Bildschirm ' Press Play and Rec. on Tape # 1'. Jetzt drücken Sie PLAY und REC. Daraufhin erscheint auf dem Bildschirm WRITING WPP. Drücken Sie jetzt sofort die STOP-Taste, damit der Cassettenrecorder anhält. Mit dem Cursor gehen wir jetzt an den Anfang der Zeile SAVE"WPP. Mit Hilfe der Tasten SHIFT und INST/DEL schieben wir jetzt den Ausdruck SAVE"WPP so weit nach rechts, bis die Anweisung FOR I = 1 TO 3 noch dazwischen paßt. Hinter den Ausdruck SAVE"WPP wird dann ein Anführungszeichen und ein Doppelpunkt gesetzt. Dann schließen wir die Schleife mit NEXT I.

Das ganze sieht dann wie folgt aus:

```
FORI=1to3:SAVE"WPP":NEXTI
```

Jetzt brauchen Sie nur noch die RETURN-Taste drücken und das Programm wird 3 mal hintereinander geladen.
Achtung: Es dürfen keine Leerräume entstehen, sonst geht der PET in seine Cursor-Betriebsart. Sie kommen hier durch einfaches Drücken der RETURN-Taste wieder heraus. Gehen Sie dann wieder an die Stelle und ändern Sie wie gewohnt weiter.

Oft wünscht man sich eine Wiederholfunktion wie bei einer Schreibmaschine. Sie können dies durch folgendes Programm realisieren.

```
 4 REM DIE GEWUENSCHTE TASTE OEFTER
 DRUECKEN BIS DAUERFUNKTION EINTRITT
 5 REM WIEDERHOLFUNKTION
 10 GET A$:IF A$=" " GOTO 10
 20 P=PEEK(515):D=6
 30 PRINT A$;
 40 T=TI+D
 50 IF PEEK(515)=255 GOTO10
 60 IF TI<TGOTO50
 70 D=6:GOTO 30
READY.
```

Kleinbuchstaben

POKE 59468,14
In dieser Betriebsart werden die Tasten mit SHIFT zu kleinen Buchstaben. In den normalen Betrieb kehren sie wieder zurück mit POKE 59468,12. Da diese Betriebsart genau umgekehrt der unserer normalen Schreibmaschine ist, können Sie dies durch folgendes Programm wieder umkehren. Für große Buchstaben muß jetzt die SHIFT-Taste gedrückt werden. Normalschrift und Negativschrift erreichen Sie einfach durch Drücken der RVS/OFF-Taste.

```
 10 REM KLEINBUCHSTABEN
 20 REM POKE59468,14
 30 REM IN DIESER BTRIEBSART WERDEN DIE
 40 REM TASTEN MIT SHIFT ZU KLEINEN
 50 REM BUCHSTABEN
 60 REM IN DEN NORMALEN BETRIEB KOMMEN
 70 REM SIE WIEDER MIT POKE59468,12
 80 REM ZURUECK.
 90 REM DA DIESE BETRIEBSART UMGEKEHRT
 100 REM DER UNSERER NORMALEN SCHREIB-
 110 REM MASCHINE IST,KOENNEN SIE DIES
 120 REM DURCH FOLGENDES PROGRAMM WIEDER        UMKEHREN
 295 REM FUER GROSSE BUCHSTABEN MUSS JETZT DIE SHIFT TASTE GEDRUECKT WERDEN
 297 REM NORMALSCHRIFT UND NEGATIVSCHRIFT GESCHIEHT EINFACH DURCH RVS/OFF TASTE
 300 POKE59468,14:C=32
 310 C=198-C:T=TI+15:PRINT CHR$(C);"=";
 320 GET Z$:IF Z$<>""GOTO 350
 330 IF TI<TGOTO320
 340 GOTO 310
 350 Z=ASC(Z$):IFZ>64ANDZ<91THENZ$=CHR$(Z+128)
 360 IF Z>192ANDZ<219THENZ$=CHR$(Z-128)
 370 PRINT" =";Z$;:GOTO 320
READY.
```

Programm „FIND"

Dieses Programm druckt Ihnen jede gewünschte Funktion, Zahl oder Ausdruck aus und gibt Ihnen die Zeile an, wo das gesuchte Wort steht. Beispiel: In Zeile 1 geben Sie 1 FOR ein und starten mit RUN 9000.

```
 9000 A=1025:X=PEEK(1029):FOR J=1 TO 1E3:FORK=A+4 TO A+83
 9001 P=PEEK(K):IF P=XTHEN GOSUB 9005
 9002 IF P<>OTHEN NEXT K
 9003 A=256*PEEK(A+1)+PEEK(A):IF A>OTHEN NEXT J
 9004 STOP
 9005 FOR L=1 TO 80:Y=PEEK(1029+L):IF Y=0THENPRINT256*PEEK(A+3)+PEEK(A+2);:RETURK
 9009 REM JEDEN BEFEHL AUS UND GIBT DIR
 9010 REM DIE ZEILE AN,WO DER GESUCHTE
 9011 REM BEFEHL IST.BEISPIEL:IN ZEILE1
 9012 REM GIBST DU 1 FOR EIN UND START-
 9013 REM EST MIT RUN 9000
READY.
```

```
10 REM DAS VORANGEGANGENE PROGRAMM WURDEDURCH EINEN TRICK AUS ZWEI EINZELNEN
20 REM PROGRAMMEN ZUSAMMENGESETZT
30 REM BEI DIESEM VORGANG WIRD DER BILD-SPEICHER DES PET ALS ZWISCHENSPEICHER
40 REM BENUTZT. WIE WIR WISSEN,WIRD DAS
50 REM ALTE PROGRAMM IM RAM DES PET
60 REM GELOESCHT,WENN EIN NEUES PROGRAMMEINGELADEN WIRD.SICHER HABEN WIR AUCH
70 REM SCHON BEOBACHTET,DASS WAEHREND
80 REM DES LADEVORGANGES RESTLICHE PROGRAMMTEILE AUF DEM BILDSVCHIRM ERHALTEN
90 REM BLEIBEN.DIES MACHEN WIR UNS JETZTZUNUTZE.DER ANZUKOPPELNDE PROGRAMMTEIL
100 REM KANN DESHALB IMMER NUR SO GROSS
110 REM WIE UNSERE BILDSCHIRMFLAECHE
120 REM SEIN.ABER IMMERHIN WIR BRAUCHEN
130 REM SIE NICHT WIEDER NEU ZU SCHREIBEN.DIESER TRICK EIGNET SICH GUT BEIM
140 REM ANKOPPELN VON UNTERPROGRAMMEN
```

```
 150 REM WIR WOLLEN NUN DIESES PROGRAMM
 160 REM AN DAS PROGRAMM 'FIND' ANHAENGENWIR FINDEN DAS PROGRAMM FIND ALS
 170 REM NAECHSTES PROGRAMM AUF DIESER
 180 REM CASSETTE.
 190 REM WIR LADEN ALSO ZUNAECHST DAS PROGRAMM'FIND' IN UNSEREN PET
 200 REM NUN LISTEN WIR ES UND SCHIEBEN ES FAST BIS GANZ AN DEN OBEREN BILD
 210 REM SCHIRMRAND
 220 REM JETZT SAEUBERN WIR MIT DEM CURSOR UND DER SPACE TASTE ALLE UNNOETIGEN
 230 REM SCHRIFTTEILE VOM BILDSCHIRM WEG.UND GEHEN MIT DEM CURSOR DIREKT UNTER
 240 REM DIE LETZTE PROGRAMMZEILE
 250 REM WIR GEBEN JETZT EINFACH LOAD EINUND LEGEN DIE CASSETTE MIT DIESEM
 260 REM PROGRAMM EIN.WIR LADEN JETZT DIESES PROGRAMM EINFACH IN DEN SPEICHER
 270 REM AUF DEN BILDSCHIRM BLEIBT DAS
 280 REM PROGRAMM 'FIND'STEHEN
 290 REM IST DER LADEVORGANG ABGESCHLOSSEN,DRUECKEN WIR DIE 'HOME'TASTE
 300 REM JA NICHT 'SHIFT'-'HOME'
 310 REM DER CURSOR IST JETZT OBEN IN DERECKE.WIR DRUECKEN JETZT SO LANGE DIE
 320 REM RETURN TASTE,BIS DER CURSOR
 330 REM UNTEN AM BILDSCHIRMRAND
 340 REM WIEDER ANGEKOMMEN IST.
 350 REM JETZT IST DAS PROGRAMM VOM BILDSCHIRM AUCH IN DEN SPEICHER UEBERNOMMEN
 360 REM GEBEN SIE JETZT LIST-ES MUESSTEN BEIDE PROGRAMME IM SPEICHER SEIN
READY.
```

Wir wollen nun dieses Programm an das Programm „FIND" anhängen.

Wir laden also zunächst das Programm „FIND" in unseren PET. Nun listen wir es und schieben es fast bis ganz an den oberen Bildschirmrand. Jetzt säubern wir mit dem Cursor und der SPACE-Taste alle unnötigen Schriftteile vom Bildschirm weg und gehen mit dem Cursor direkt unter die letzte Programmzeile. Jetzt geben wir LOAD ein und legen die Cassette mit diesem Programm ein. Dieses Programm wird nun in den Speicher geladen. Auf dem Bildschirm bleibt das Programm „FIND" stehen. Ist der Ladevorgang abgeschlossen, drücken wir die HOME-Taste. Achtung: Nicht SHIFT-HOME.
Der Cursor ist jetzt oben in der Ecke. Wir drükken jetzt so lange die RETURN-Taste, bis der Cursor unten am Bildschirmrand wieder angekommen ist.

Jetzt ist das Programm vom Bildschirm auch in den Speicher übernommen. Geben Sie jetzt LIST ein, es müßten beide Programme im Speicher sein.

Nachfolgend sehen Sie das über den Bildschirm angekoppelte Programm in seiner endgültigen Form.

```
10 REM DAS VORANGEGANGENE PROGRAMM WURDEDURCH EINEN TRICK AUS ZWEI EINZELNEN
20 REM PROGRAMMEN ZUSAMMENGESETZT
30 REM BEI DIESEM VORGANG WIRD DER BILD-SPEICHER DES PET ALS ZWISCHENSPEICHER
40 REM BENUTZT. WIE WIR WISSEN,WIRD DAS
50 REM ALTE PROGRAMM IM RAM DES PET
60 REM GELOESCHT,WENN EIN NEUES PROGRAMMEINGELADEN WIRD.SICHER HABEN WIR AUCH
70 REM SCHON BEOBACHTET,DASS WAEHREND
80 REM DES LADEVORGANGES RESTLICHE PROGRAMMTEILE AUF DEM BILDSVCHIRM ERHALTEN
90 REM BLEIBEN.DIES MACHEN WIR UNS JETZTZUNUTZE.DER ANZUKOPPELNDE PROGRAMMTEIL
100 REM KANN DESHALB IMMER NUR SO GROSS
110 REM WIE UNSERE BILDSCHIRMFLAECHE
120 REM SEIN.ABER IMMERHIN WIR BRAUCHEN
130 REM SIE NICHT WIEDER NEU ZU SCHREIBEN.DIESER TRICK EIGNET SICH GUT BEIM
140 REM ANKOPPELN VON UNTERPROGRAMMEN
150 REM WIR WOLLEN NUN DIESES PROGRAMM
160 REM AN DAS PROGRAMM 'FIND' ANHAENGENWIR FINDEN DAS PROGRAMM FIND ALS
170 REM NAECHSTES PROGRAMM AUF DIESER
180 REM CASSETTE.
190 REM WIR LADEN ALSO ZUNAECHST DAS PROGRAMM'FIND' IN UNSEREN PET
```

```
 200 REM NUN LISTEN WIR ES UND SCHIEBEN ES FAST BIS GANZ AN DEN OBEREN BILD
 210 REM SCHIRMRAND
 220 REM JETZT SAEUBERN WIR MIT DEM CURSOR UND DER SPACE TASTE ALLE UNNOETIGEN
 230 REM SCHRIFTTEILE VOM BILDSCHIRM WEG.UND GEHEN MIT DEM CURSOR DIREKT UNTER
 240 REM DIE LETZTE PROGRAMMZEILE
 250 REM WIR GEBEN JETZT EINFACH LOAD EINUND LEGEN DIE CASSETTE MIT DIESEM
 260 REM PROGRAMM EIN.WIR LADEN JETZT DIESES PROGRAMM EINFACH IN DEN SPEICHER
 270 REM AUF DEM BILDSCHIRM BLEIBT DAS
 280 REM PROGRAMM 'FIND'STEHEN
 290 REM IST DER LADEVORGANG ABGESCHLOSSEN,DRUECKEN WIR DIE 'HOME'TASTE
 300 REM JA NICHT 'SHIFT'-'HOME'
 310 REM DER CURSOR IST JETZT OBEN IN DERECKE.WIR DRUECKEN JETZT SO LANGE DIE
 320 REM RETURN TASTE,BIS DER CURSOR
 330 REM UNTEN AM BILDSCHIRMRAND WIEDER
 340 REM WIEDER ANGEKOMMEN IST.
 350 REM JETZT IST DAS PROGRAMM VOM BILDSCHIRM AUCH IN DEN SPEICHER UEBERNOMMEN
 360 REM GEBEN SIE JETZT LIST-ES MUESSTEN BEIDE PROGRAMME IM SPEICHER SEIN
 9000 A=1025:X=PEEK(1029):FOR J=1 TO 1E3:FORK=A+4 TO A+83
 9001 P=PEEK(K):IF P=XTHEN GOSUB 9005
 9002 IF P<>OTHEN NEXT K
 9003 A=256*PEEK(A+1)+PEEK(A):IF A>OTHEN NEXT J
 9004 STOP
 9005 FOR L=1 TO 80:Y=PEEK(1029+L):IF Y=0THENPRINT256*PEEK(A+3)+PEEK(A+2);:RETURK
 9009 REM JEDEN BEFEHL AUS UND GIBT DIR
 9010 REM DIE ZEILE AN,WO DER GESUCHTE
 9011 REM BEFEHL IST.BEISPIEL:IN ZEILE1
 9012 REM GIBST DU 1 FOR EIN UND START-
 9013 REM EST MIT RUN 9000
READY.
```

Das vorangegangene Programm wurde durch einen kleinen Trick aus zwei einzelnen Programmen zusammengesetzt. Bei diesem Vorgang wird der Bild-Speicher des PETs als Zwischenspeicher benutzt. Wie wir wissen, wird das alte Programm im RAM des PETs gelöscht, wenn ein neues Programm eingeladen wird. Sicher haben wir auch schon beobachtet, daß während des Ladevorganges restliche Programmteile auf dem Bildschirm erhalten bleiben. Dies machen wir uns jetzt zunutze. Der anzukoppelnde Programmteil darf deshalb immer nur so groß wie unsere Bildschirmfläche sein. Aber immerhin, wir brauchen diesen Programmteil nicht wieder neu zu schreiben. Dieser Trick eignet sich gut zum Ankoppeln von Unterprogrammen.

```
 10 REMAUSDRUCKEN DES NAMENS EINES PROGRAMMES,WELCHES SICH GERADE IM PET BEFINDET
 20 FOR I=639TO660:PRINTCHR$(PEEK(I));
 30 NEXT
 40 END
READY.


READY.

 10 REM LADEN EINES PROGRAMMES MIT DEM
 20 REM MOMMENTANEN NAMEN
 30 N$=" "
 40 FORI=639TO 660:N$=N$+CHR$(PEEK(I))
 50 NEXT
 60 SAVEN$
READY.
```

Achtung: Cassettenrecorder muß nach Einlesen dieses Programms angehalten werden!!!

Programmier-experimente für PET

Programmierexperimente mit dem PET

Allgemeines

Der PET bietet mit seinen, von außen leicht zugänglichen Ein-/Ausgabeports ideale Möglichkeiten für Programmierexperimente. Dieser Vorteil ist unter anderem dem recht interessanten Interfacebaustein 6522 zu verdanken, der sich im PET befindet. Auf diesen Baustein wollen wir jetzt kurz eingehen.

Der 6522, ein universeller Interface-Adapter-Baustein (VIA), ist eine Weiterentwicklung des 6520 Peripheral Interface Adapters. Der 6520 ist voll hardwarekompatibel zum bekannten 6820 von Motorola.

Der 6522 besitzt die gleichen Eigenschaften wie der 6520, hat jedoch zusätzlich zwei leistungsfähige Zeitgeber, ein internes Schieberegister (seriell/parallel und parallel/seriell) sowie zwischengespeicherte Eingänge. Die Steuerung peripherer Geräte kann durch zwei bidirektionale Ports erfolgen. (PA0-PA7 und PB0-PB7) Die Ports PA0-PA7 sind beim PET von außen über eine Steckerleiste leicht zugänglich. Als Stecker können verwendet werden:

1. CINCH 251-12-90-160 für User Port

Die Ausgangsports PB0 -PB7 werden im PET für den IEEE488 Anschluß sowie zur Behandlung der anderen Ein-/Ausgabeeinheiten (wie Video, Cassette etc.) verwendet.
Jeder dieser Ports kann vom Programm her als Eingabe oder Ausgabekanal geschaltet werden. Einzelne Ein-/Ausgabeleitungen können weiterhin durch die Zeitgeber gesteuert werden, da man vom Programm her sehr leicht Rechteckwellen erzeugen kann. Man kann aber auch auf die gleiche Weise von außen kommende Impulse zählen und ins Programm übernehmen.

Die Kontrolle dieser Funktionen geschieht durch interne Register des 6522 Bausteins.

Wir können jetzt schon erkennen, welche Möglichkeiten es hier gibt, sich bei Programmexperimenten auszutoben. Natürlich sind die I/O-Ports des 6522 CMOS und TTL-compatibel. D. h., Sie können direkt TTL-Gattereingänge oder CMOS-Gattereingänge ansteuern.

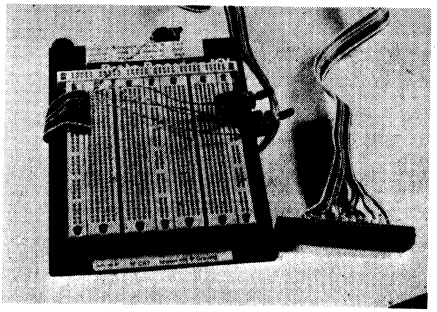
Die Richtung des Datenflusses (Eingabe oder Ausgabe von Daten) kann durch ein spezielles Datenrichtungsregister gesteuert werden. Jeder beliebige Anschluß kann wahlweise als Eingang oder Ausgang festgelegt werden.

Am parallelen User-Port Ihres PET finden Sie dann an der unteren Klemmleiste noch die Anschlußbezeichnungen CA1 und CB2. Dieses sind auch direkte Verbindungen zum 6522.

CA1 ist eine Handshake Leitung (Quittungsleitung) für eingehende Signale. Sie kann auch als Flankendetektor verwendet werden. CB2 arbeitet ähnlich wie CA1, kann aber zusätzlich noch als Eingang oder Ausgang des Schieberegisters in 6522 programmiert werden. Sie werden über das periphere Kontrollregister (PCR) gesteuert. Wie man mit diesem Ausgabeport recht interessante Experimente durchführen kann, wollen wir im folgenden Artikel zeigen. Es soll nur eine Anregung und ein Anstoß für eigene weitere Experimente sein.

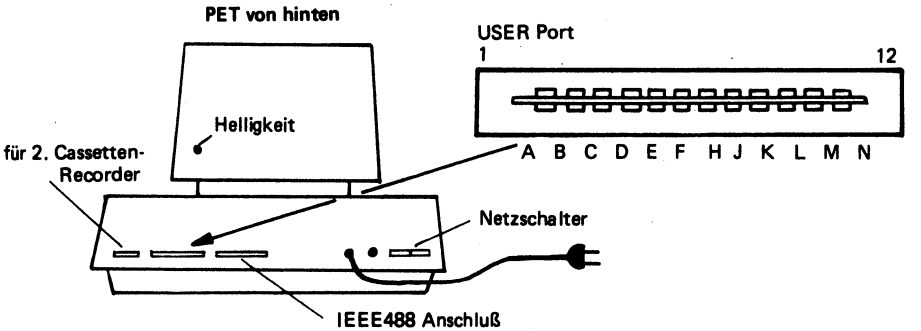
Anschluß einer Universal-Experimentierplatine

Für den externen Aufbau von kleinen Experimenten eignet sich ein Experimentierboard (s. Bild) mit einem Anschlußkabel recht gut. Über ein Flachbandkabel schließt man das Board an den User-Port an.



Wie Sie sicher auf dem Bild erkennen können, haben wir uns die Anschlußbezeichnungen mit Farbe des Verbindungsdrahtes oben am Board vermerkt. So wissen wir auf den ersten Blick, mit welchem Anschluß wir es zu tun haben. Wir können jetzt Schalter anbringen und von außen Informationen an den PET geben. In umgekehrter Richtung können wir jedes gewünschte 8 Bit Muster vom PET her über den Port nach außen geben.

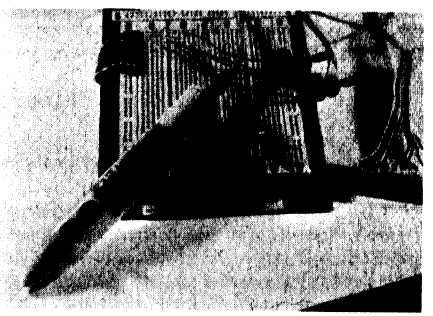
Hierzu haben wir ein kleines Programm geschrieben, welches Ihnen die Arbeit mit dem Experimentierboard erleichtert.



Der User Port ist von außen zugänglich. Die Kontakte sind auf der Leiterbahn oben und unten angebracht. Für Verbindungen sollte man auf jeden Fall einen Stecker aufstecken. (CINCH 251-12-90-160 Typ 50 24EE-30). Dort kann man dann das Flachbandkabel anlöten und mit dem Universal Experimentierboard verbinden.

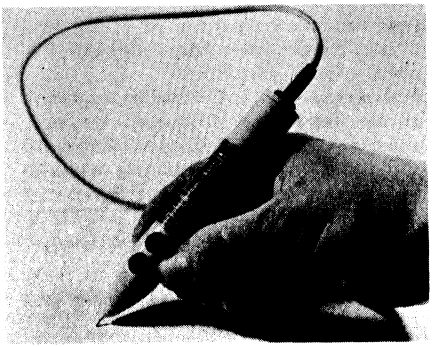
Achtung ganz wichtig! Jegliches Aufstecken und Abziehen von Steckern an sämtlichen externen Anschlüssen des PETs darf nur bei abgeschaltetem Gerät erfolgen!!

Programmbeschreibung:
Das kleine Hilfsprogramm kann den Port abfragen oder Informationen hinausgeben. Nach dem Einlesen und Eingabe von RUN wird gefragt, ob wir die acht als Eingänge geschalteten I/O-Leitungen abfragen wollen, oder ob wir über alle 8 Leitungen Informationen hinausgeben möchten. Entscheiden Sie sich für das Erste, zeigt Ihnen der Computer die offenen Eingänge



(log. 1) mit 255 = 1 1 1 1 1 1 1 1. Jetzt können Sie beim Experimentieren die Eingänge verändern und haben die Information auf dem Bildschirm. Im zweiten Falle können Sie eine gewünschte Information (8 Bit binär) an die Ports ausgeben. Die Eingabe erfolgt dezimal. Hiermit lassen sich dann recht praktische Steuerschaltungen programmieren.

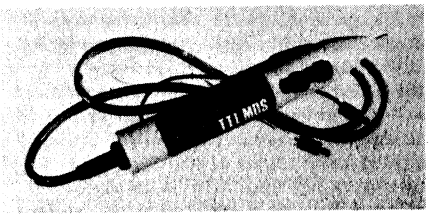
Ein weiteres, wichtiges Hilfsmittel, welches Sie bei solchen Experimenten gut gebrauchen können, ist ein Logiktester (Bild). Mit diesem Tester können Sie leicht die einzelnen Logik-



pegel in der Experimentierschaltung und am Port nachkontrollieren. Er kann durch den Fachhandel oder direkt beim Verlag für nur DM 24,80 als Bausatz und für DM 39,80 als Fertiggerät bezogen werden.
Wir wünschen Ihnen beim Experimentieren viel Erfolg.

```
5 COPYRIGHT BY ING W. HOFACKER GMBH CMD5
10 REM THIS PROGRAM DISPLAYS THE IO PORT
20 REM IN BOTH DECIMAL AND BINARY
37 PRINT"3"
39 S$(0)="0":S$(2)="1"
40 GOTO 200
41 POKE59459,0
60 PRINT"DECIMAL     ";PEEK(59457)
62 P=PEEK(59457)
63 GOTO90
65 FORI=1TO8
70 PRINTPAND1;:P=P/2
80 NEXTI
90 PRINT"BINARY      "ABS((PAND128)=128);
91 PRINTABS((PAND64)=64);
92 PRINTABS((PAND32)=32);
93 PRINTABS((PAND16)=16);
94 PRINTABS((PAND8)=8);
95 PRINTABS((PAND4)=4);
96 PRINTABS((PAND2)=2);
97 PRINTABS((PAND1)=1);
110 PRINT"INHALT DES REGISTERS FUER
        DATENRICHTUNG";PEEK(59459)
111 FOR X=1TO 10
115 X=PEEK(59459)
116 NEXT
120 GOTO60
200 PRINT"ABFRAGEN='1',AUSGABE='2'";
210 INPUT N
220 IF N=1 THEN 41
230 IF N=2 THEN 300
300 REM AUSGABE VON INFORMATIONEN
305 POKE59459,255
310 PRINT"WELCHE AUSGANGSBITS";K
330 INPUT K
340 IF K>255THEN RESTORE:GOTO300
350 POKE59457,K
370 L=PEEK(59457)
380 GOTO 60
READY.
```

Dieses Listing wurde über einen Drucker ohne Graphiksymbole ausgedruckt. Wir bitten Sie deshalb, bei der Eingabe in Ihren PET darauf zu achten.



Im obigen Bild sehen Sie noch einmal den praktischen Logiktester, der Ihnen bei Ihren Experimenten viele nützliche Dienste erweisen kann.

Aufbau und Arbeitsweise

Zuerst verbinden Sie das Board (AP-Products oder CSC) mit einem Flachkabel und dem CINCH-Stecker 251-12-90-160/50-24 EE-30 für den USER PORT.

Dann können Schalter und die Aufkleber mit den einzelnen Bezeichnungen angebracht werden. PET ausschalten und Stecker so auf den USER PORT aufsetzen, daß die Anschlußreihe mit der Zahlenbezeichnung 1-12 nach oben zeigt. Nun den PET einschalten und Cassette einlesen. Sie können jetzt zwischen den Betriebsarten „Abfragen" oder „Ausgabe" wählen. Im ersten Falle können Sie den PORT abfragen, welche Signale von außen anstehen. (Offen = log. 1 = High). Kurzschluß des Portausganges nach Masse = log. 0 = Low). In der Betriebsart „Ausgabe" können Sie Signale über die PORTs per Programm ausgeben. Es wird eine externe 5V-Stromversorgung empfohlen. Auf dem Experimentierboard können Sie jetzt Ihre eigenen Schaltungen leicht aufbauen.

Analog/Digital und Digital Analog-Wandler für PET

Beschreibung: Die ADAK-Platine kann ohne weitere Zusätze in den User Port des PET gesteckt werden. Mit der Platine wird eine Cassette mit Betriebssoftware, Spielen und wichtigen Utilityprogrammen sowie eine ausführliche Anleitung geliefert. Sensoren, Lautsprecher, Verstärker etc. können sofort angeschlossen werden. (Joystick, Musik)

Ganz besonders interessant sind die Maschinenunterprogramme, die zu einer schnelleren Behandlung der Ports auch in Ihre engeren BASIC-Programme eingefügt werden können.
Eine ausführliche Anleitung zur Programmierung in Maschinensprache sowie ein speziell hierfür entwickeltes Programm gehören zur Grundausstattung.

Software:

1. Programme in Maschinensprache zum Eingeben, Einlesen und Verarbeiten von 6502 Anweisungen aus dem 6502 Standardbefehlssatz.
2. Analog/Digital-Umwandlung. Successive Approximation. Bis zu acht Sensoren können angeschlossen werden (Potentiometer etc.).
3. Fourier Synthese. Erzeugen von Wellenformen zur Kombination bis zu 11 harmonischen Obertönen in jeder beliebigen Proportion.
4. Erzeugung von Musik. Einführungsprogramme in die Technik zur Erzeugung von zweistimmigen Liedern. (Gesteuerte Wellenformstruktur) 8 Oktaven gleichförmig temperierte Tonleiter mit Halbtönen. Tempo-Kontrolle und Möglichkeit zur Phasenverschiebung.
5. Einige Beispiele zur Erleichterung der Handhabung sind beigefügt (Spiele, Lieder etc.)

Bestell-Information

ADAK-1 A — Universales Analg Interface mit Stekkern und Anleitungen. **DM 299,--**

ADAK-1 PET — Analoge Interface-Karte für 8K PET 2001. Mit Steckern, Anleitung und Software Cassette **DM 399,--**

Spezifikationen: Betriebsspannung + 5 V / 20 mA (ohne Lautsprecher). Kann beim PET der Versorgung des 2. Cassettenrecorders entnommen werden.

Signalleitungen:	8 Bit Datenport, eine Kontrolleitung für den Multiplexer, eine Eingangsleitung für den Komparator. TTL oder CMOS-kompatibel.
Analoge Eingänge:	0 bis +5V, 1 MΩ Impedanz 8 Eingangsleitungen.
DAC-Linearität:	± 1/2 LSB
Tiefpassfilter:	5 Pole, 7 KHz Butterworth
Lautsprecher-impedanz	4, 8 oder 16 Ω
A/D-Umwandlg.:	Successive Approximation, 4 Microsekunden Einstellzeit.
Anschlüsse:	12 pin und 15 pin

Bedienungsanleitung ADAK-1-PET

Technische Beschreibung:
ADAK-1 wird durch ein 8 Bit-Wort vom User Port Ihres Computers gesteuert. Dieses 8 Bit Wort steuert den Ausgang eines monolithischen Digital/Analog Umsetzers. Eine Null am Eingang bringt am Ausgang des D/A-Umsetzers den Spannungswert 0V. Eine 255_{10} oder FF_{16} bringt den Ausgang auf den Wert der Referenzspannung. (nominal 2,4V) Dazwischenliegende Werte erzeugen entsprechende Spannungswerte dazwischen.

Der D/A-Ausgang wird jetzt noch einmal verstärkt und geglättet (durch drei Teile eines 5 poligen 7KHz Butterworth Filters). Der Filter-Ausgang (Pin 1 des LM324) wird normalerweise mit dem Eingang eines HIFI-Verstärkers verbunden. Die Ausgangswerte sind typisch 0 bis + 4 V.

Für manche Anwendungen kann es von Vorteil sein, wenn man den Audio-Verstärker aus einem Teil des LM324 zusammen mit zwei Transistoren mitverwendet. Man schließt dann nur noch einen Lautsprecher mit 4, 8 oder 16Ω an die Anschlüsse 3 und 4 des Analogsteckers. Das ist alles, was Sie zur Anwendung als D/A-Wandler z. B. bei der Musikerzeugung zu tun haben.
ADAK-1 kann jedoch auch Ihren Computer dazu veranlassen, daß er bis zu acht analoge Eingänge digitalisieren kann.

Die Analog/Digital-Wandlung ist etwas komplexer als die D/A-Wandlung. Pin 12 des Digital-Steckers muß an einen Ein-Bit-Ausgangs-Port des Computers angeschlossen werden. Wenn dieser Kontrollanschluß auf log. 0 liegt, werden die Signale der vier niederwertigen Bits am 8Bit Computer-Ausgangsport in einen Zwischenspeicher (CD4042) übernommen. Wenn die Kontrolleitung dann auf log. 1 geht, hält der CD 4042 die übernommene Information. Für einen einwandfreien Betrieb muß gewährleistet sein:

1. Pin 12 muß auf log. „0" gehalten werden
2. Ausgabe des richtigen Wortes über den 8Bit Ausgabeport
3. Pin 12 muß auf log. „1" gebracht werden

Der CD4051 (8 Kanal Multiplexer) wird von den restlichen niederwertigen Bits aus dem CD4042 Speicher gesteuert. Wenn Sie den analogen Eingang Nr. 5 digitalisieren wollen, (Analogstecker Pin 5) muß das Wort $05 auf den Ausgabeport gegeben werden. Gleichzeitig muß der Zwischenspeicher CD4042 aktiviert werden. Dies geschieht durch den Digital-Stecker Pin 12. Er muß auf log. „0" liegen.

Natürlich werden die Werte $\$13_{10}$ oder $\$0D_{16}$ auf Ihrem Ausgangsport auch den Analogeingang Nr. 5 aktivieren.

Bit 3 wird dazu verwendet, den Oszillator (CD4011) zu aktivieren. Wenn Bit 3 auf log. 1 liegt, während der Zwischenspeicher gesperrt ist, wird der Komparator-Ausgang (Digital-Stecker Pin 3) auf log. 1 liegen. Nur wenn das analoge Eingangssignal größer als das analoge Äquivalent des digitalen Wortes ist. Es wird log. „0" sein, wenn der analoge Eingang log. „0" ist. Wenn das Kontrollbit 3 log. „0" ist, und der analoge Eingang kleiner ist, als sein digitales Äquivalent, wird vom Comparator eine 1 MHz Rechtecksspannung erzeugt. Dies ist recht praktisch für interruptgesteuerte Software-Programme.

Um die Platine als Analog/Digital-Konverter zu betreiben, muß ein spezielles Programm zur Behandlung dieser successiven Approximation verwendet werden.

Dies geschieht ähnlich wie bei einem Zahlenratespiel. In dem der Computer rät, ob sich der Wert in der oberen oder unteren Hälfte des Gesamtbereiches befindet. (80_{16}). Wenn der Komparator-Ausgang (Digitale Stecker Pin 3) fest-

stellt, daß dieser Wert zu niedrig ist, wird die nächste Schätzung auf 3/4 des vollen Bereiches gelegt. (CO_{16}). Im anderen Falle wird es auf 1/4 des Gesamtbereiches (40_{16}) gebracht.

Durch Aufteilen der Differenz wird der analoge Eingang in 8 Bit eingeteilt. Wir werden auf diese Technik später noch eingehen.

ADAK-1 benötigt einen 8 Bit Ausgangs-Port für die Analog/Digital-Umsetzung. Die unteren 4 Bit werden auch in Verbindung mit einer Steuerleitung zur Auswahl einer aus 8 Datenleitungen verwendet. Außerdem legen sie fest, ob der Komparatorausgang ein einfacher Gleichstrompegel oder eine Impulsfolge ist. Der Komparatorausgang führt dann in den Computer zur A/D-Umwandlung.

Analog Anschluss (Analog-Stecker)

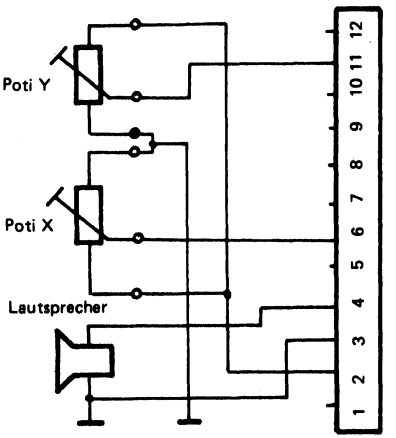
Der 12polige Analog-Stecker wird wie folgt angeschlossen:

1. Anschluß eines Joysticks

Dieser Stecker hat seine Lötfahnen an der unteren Seite. Rechts mit 1 beginnend und nach links bis 12 ansteigend. Auf keinen Fall Signale größer + 5V oder negative Spannungen hier anlegen. Wenn Joysticks angeschlossen werden, bitte darauf achten, daß weder +5V noch Masse an die Abgreifer angeschlossen wird. Bei Lautsprecheranschluß an Pin 3 und Pin 4 erwarten Sie bitte keine Lautstärken.

12 pin Analogstecker Anschlußliste

Pin #	Funktion	gewöhnlich angeschlossen an:
1	DAC Output	HiFi Verstärker
2	+ 5 V	Spannungsversorgung f. Joystick
3	Masse	Analog Signal Return
4	Audio Verstärker Ausgang	Lautsprecher
5	Analog Eingang # 5	0 bis + 5 V Analog Signal
6	Analog Eingang # 1	Abgreifer des Joysticks X
7	Analog Eingang # 7	0 bis + 5 V Analog Signal
8	Analog Eingang # 3	0 bis + 5 V Analog Signal
9	Analog Eingang # 4	0 bis + 5 V Analog Signal
10	Analog Eingang # 2	0 bis + 5 V Analog Signal
11	Analog Eingang # 0	Abgreifer des Joysticks Y
12	Analog Eingang # 6	0 bis + 5 V Analog Signal



Typische Anschlüsse zum Analog-Stecker

Digitale Stecker-Anschlüsse

Die richtigen Anschlüsse erfolgen automatisch, wenn die Karte auf den User-Port aufgesteckt wird. Die +5V Versorgungsspannung wird dem Anschluß für das 2. Cassetteninterface entnommen. Nachdem die Platine eingesteckt wurde, werden die 8 digitalen Eingangsleitungen mit dem DATA-Register (Speicherzelle 59471) verbunden. Das Datenrichtungsregister muß vorher mit 59559,255 gepoked werden. (POKE 59559,255). Die Kontrolle des Zwischenspeichers erfolgt über CB2, während der Komparator-Ausgang mit CA1 verbunden wird. Die Ausgangsspannung an Pin 1 des Analog-Steckeranschlusses ist nun proportional zu dem Digitalwert, der in die Speicherzelle 59471 gepoked wird. 0—255 entspricht 0 bis + 4V.

Software Behandlung der Platine

Es wird in jedem Falle vorausgesetzt, daß man mit BASIC schon gearbeitet hat. Da die Programmierung in Maschinensprache relativ kompliziert ist, wollen wir uns jetzt ein wenig mit dieser Technik befassen. Sie sollten auf jeden Fall den Befehlssatz des 6502 Prozessors kennen und die entsprechenden Handbücher besitzen. (Hofacker Verlag, Best.Nr. 80/42 und 80/43). Die mitgelieferten Programme können auf einfache Weise für den persönlichen Gebrauch abgeändert und verwendet werden. Die Maschinenprogramme sind in BASIC DATA-Statements enthalten und machen Ihnen so die Arbeit etwas leichter.

A-Seite der mitgelieferten Programmcassette

Das erste Programm ist ein Hinweisprogramm und zeigt Ihnen alle Programme auf der ersten Seite. Weiterhin lädt es ein Maschinenunterprogramm in die Speicherzellen 826 bis 1022. Wenn Sie später Ihre eigenen Programme schreiben wollen, sollten Sie diese Routine mitverwenden. Wenn diese Routine einmal geladen ist, können alle weiteren Programme eingelesen werden. Sie bleibt so lange im Speicher, bis der PET abgeschaltet wird.

Spiele

Alle drei Spiele arbeiten mit Joysticks an den Analog-Steckereingängen 0 und 1 (Pin 11 und Pin 6). Das digitale Äquivalent der Eingangsspannung wird in den Speicherzellen 920 und 921 abgelegt. Das Analog/Digital-Umwandlungsprogramm befindet sich in den Speicherzellen 826 bis 913. Über den Befehl SYS (826) digitalisiert dieses Maschinensprachenprogramm Analogwerte, die an Eingang 0—7 anliegen. Der Digitalwert wird dann in den Speicherzellen 920 bis 927 abgelegt.

Wenn die Joysticks nicht richtig arbeiten, kann ein Diagnostik-Programm mit RUN 500 gestartet werden. Dieses Programm zeigt auf dem Bildschirm das digitale Äquivalent der Eingänge 0—7 an. Die Eingänge 0 und 1 sollten bei Bewegung des Joysticks die Werte 0 bis 255 einnehmen. Wenn nicht, trennen Sie die Joystick-Verbindungen zwischen dem Analog-Stecker Pin 11 (Eingang 0) und dem Analog-Stecker Pin 2 (+5V).

Stellen Sie jetzt eine Verbindung her. Der Eingang 0 sollte nun 255 und der Eingang 1 0 anzeigen. Wenn der Pin 11 (Eingang 0) auf Masse (Pin 3) gelegt wird, muß 0 angezeigt werden. Wenn nicht eine dieser Möglichkeiten eintritt, laden Sie bitte das Programm mit dem Namen „MACHINE" und vergleichen Sie die Speicherinhalte 826 bis 913 mit dem Listing. Wenn Sie Fehler finden, so ändern Sie diese mit dem Programm „WRITE" entsprechend ab.

Richtig angeschlossene Joysticks bringen an den Zuleitungen zu den Eingängen 0 und 1 einen Spannungswert zwischen 0 und +5V. (Bei Bewegung des Joysticks) Die Spiele sind so programmiert, daß der zu bewegende Punkt auf dem Bildschirm sich im Ausgangszustand in der linken unteren Ecke des Bildschirmes befindet. X(Eingang 1) und Y (Eingang 0) sind dann 0.

Wenn alles richtig angeschlossen ist, sollten Sie auch Töne hören. (Space Flight und Chase) Wenn der PET einmal „abstürzt", liegt meist irgendwo ein Fehler im Maschinensprachenprogramm. Über das Programm MACHINE können dann leicht die Speicherinhalte getestet werden.

A/D-Programm	=	Zelle 826 bis 913
Tonroutine	=	Zelle 930 bis 979

Es könnte nun möglich sein, daß Sie im Eingangsprogramm „ADAK-1-SIDE A" einige DATA-Statements ändern müssen. Dies kann wie folgt geschehen. Spielen Sie die Cassette zurück und geben LOAD „ADAK-1-SIDE A" ein. Dann RUN 5000. Anschließend laden Sie das Programm entsprechend der Anleitung „Laden von Maschinenprogrammen".

Das Analog/Digital-Umwandlungsprogramm arbeitet nur über die Eingänge 0, 1 und 2. Dies liegt an der $03 Grenze in Speicherzelle 827, die eine Erhöhung der Umsetzgeschwindigkeit ermöglicht. Wenn Sie mehrere der Eingänge benutzen wollen, geben Sie POKE 827,8 ein, um alle 8 Eingänge zu aktivieren.

Das Unterprogramm zur Erzeugung der Toneffekte liegt in den Speicherzellen 930 bis 979. Es erzeugt einfache Rechtecksignale. Die Amplitudencharakteristik ist steil ansteigend und logarithmisch abfallend. Die Tonhöhe wird durch den Wert in Speicherzelle 980 beeinflußt. Ein Verdoppeln dieses Wertes bringt eine Halbierung der Frequenz. (Senkt die Tonhöhe um eine Oktave).

Die Dauer der Note wird durch den Wert in Speicherzelle 981 bestimmt. Eine Verdoppelung dieses Wertes bringt eine Verdoppelung der Tondauer. Die Tondauer wird zusätzlich auch durch die Tonhöhe verändert. Töne mit niedrigeren Frequenzen erhalten eine längere Tondauer. Man sollte deshalb hier einige eigene Versuche und Experimente selbst anstellen. Wenn Sie immer die gleiche Tondauer haben wollen, sollten Sie den Wert in Speicherzelle 981 etwas erhöhen, wenn Sie die Tonhöhe in Speicherzelle 980 senken wollen.

Das Programm „MACHINE"

Dieses Programm besteht aus einem Disassembler und einem „WRITE„Programm. Wenn Sie die Option „READ„ wählen, zeigt der Disassembler den Speicherinhalt in Standard 6502 Mnemonics. Wenn Sie dieses Programm mit der Speicherzelle 826 testen, sollte es das A/D-Umsetzer-Programm anzeigen. Wenn ein kleines Sternchen ausgedruckt wird, heißt das, daß der Disassembler den Befehl nicht verstanden hat. Dies kommt daher, daß der Befehlssatz des 6502 ständig erweitert wurde. Es muß nicht bedeuten, daß es ein unerlaubter Befehl ist.

Der Teil „WRITE" aus diesem Programm wandelt die Standard 6502 Hex Eingaben in Dezimalwerte und gibt sie über POKE in den Speicher. Wenn Sie Maschinenspracheprogramme und BASIC-Programme mischen möchten, kann es vorkommen, daß ein BASIC-Programm das Maschinenprogramm überschreibt und zerstört. Dies kommt daher, daß die BASIC-Befehle ab Speicherzelle 1024 aufwärts abgelegt werden. Die Ablage von Strings jedoch erfolgt vom Speicherende an. Alles, was dazwischen liegt, wird dann zerstört. Bei den ersten Programmen haben wir deshalb den Speicherbereich des 2. Cassetteninterfaces verwendet. In anderen Fällen kann man jedoch auch den Speicherbereich des PET eingrenzen. Der PET hat ein Memory Limit-Register, welches den Raum für das BASIC-Programm festlegt. Das Register hat die Adresse 135. Sehen wir durch PEEK (135) in diese Zelle, finden wir die Zahl 32. Beim 8K PET entspricht dies 23 Teilblöcken (Pages) des Speichers, a' 250 K Byte. Unser Musik-Programm auf der B-Seite der Cassette benötigt etwa 1K Speicher. Aus diesem Grunde wird in die Speicherzelle 135 die Zahl 25 gepoked. Dadurch werden die letzten 7 Speicherblöcke vom BASIC geschützt.

Bevor man ein eingegebenes Maschinenprogramm startet, sollte man es genauestens prüfen und mit dem READ-Programm testen. Alle Sprünge und Verzweigungen müssen überprüft werden. Ein Fehler führt dazu, daß sich der PET irgendwo aufhängt und eine Rückkehr nur über den Ausschalter möglich ist. Das mühsam eingegebene Programm geht verloren.

Das DEBUG-Programm

Das DEBUG-Programm setzt an die Stelle, an der Sie einen Breakpoint haben möchten einen RTS-Befehl (Return from Subroutine). Nachdem der Anfang und der Breakpoint des Programmes festgelegt wurde, wird das Programm in den Speicherzellen 990 bis 1022 geladen, die vorher gespeicherten Werte in den ACU geladen, das X und Y-Register werden geladen und die Flags gesetzt.

Dann wird das Programm in dem gewählten Bereich abgearbeitet. (Anfang bis Breakpoint). Registerinhalte und Flags werden abgespeichert und anschließend kehrt das Programm ins BASIC zurück. Die Inhalte der Register und Flags werden angezeigt. Anfang und Breakpoint müssen den 6502-Befehlen entsprechen. Ansonsten bricht das Programm zusammen. Ein einfaches READ-WRITE-Programm ist noch vorhanden. Es ermöglicht Ihnen eine einfache Änderung während der Korrekturphase (DEBUG).

Weiterhin finden Sie eine „File-Handling"-Routine, die es Ihnen ermöglicht, Maschinenprogramme zu laden und zu speichern. Es wird in jedem Falle empfohlen, eingegebene Programme vor dem ersten RUN zu speichern.

Abspeichern auf Cassette
Das letzte Programm auf Seite A der Cassette ist ein Spezialprogramm zum Abspeichern von Maschinenspracheprogrammen. Der SAVE-Teil verwendet die Anfangs- und Endadresse des Programmes in DATA-Statements. Dann wird der Bildschirm gefüllt. Jetzt muß die RETURN-Taste für jedes DATA-Statement gedrückt werden. Dadurch wird es in den Speicher übernommen. Das letzte DATA-Statement bringt sie wieder zum DATA-Generations-Programm zurück. Wenn alle Speicherinhalte in DATA-Statements verwandelt sind, kann das Programm wie ein normales BASIC-Programm mit SAVE abgespeichert werden.

Wenn das Baud dann später in den PET geladen wird, werden die DATA-Statements in den gewünschten Speicherbereich gepoked. Wenn das Programm in einen anderen Speicherbereich geladen werden soll, brauchen nur die DATA-Statements für Anfangs- und Endadresse vor dem Laden geändert zu werden. (Erste Datenzeile)

Wenn Sie das Programm in einen ungeschützten Speicherbereich des PET laden, sollten Sie am Anfang des Programmes den POKE 135,X-Befehl geben. Wobei X kleiner als 32 sein muß. (Je nach Programmgröße)

Die Seite 2 der Cassette enthält als erstes das Musikprogramm in Maschinensprache (Speicherzellen 7680–8125). Es ist in verschiedene Unterprogramme unterteilt, um den verschiedenen, speziellen Aufgaben gerecht zu werden.

Lieder:
Dieses Programm lädt vier Demonstrationsprogramme in den Speicher. (1. Teil des Liederbereiches 6912–7679). Die Lied-Platzierungsinformation (6900–6911) wird auch geladen. Dieses Programm berechnet und lädt auch die Tonhöhentabelle (8128–8188). Diese Tabelle ist so aufgebaut, daß immer zwei Byte-Werte einer von 12 Noten zugeordnet werden.
(12 Noten der gleichmäßig temperierten Tonleiter mit ihren Viertelnoten für die höchste Oktave)

Music
Der erste Teil des Musik-Programmes erzeugt eine Hälfte der Sinuswellen und lädt sie in die Speicherzellen 6665 bis 6783. Die Sinuswellen-Tabelle wird dann vom Programm „HARMONIC" dazu benutzt, um eine komplexe Wellenform in den Speicherzellen 6400 bis 6655 zu erzeugen.
Das Programm „HARMONIC" beginnt damit, daß die 11 Verhältnisregister gelöscht werden (6816 bis 6826).

Jedesmal, wenn vom Programmierer ein neues Verhältnis der harmonischen Oberwellen gewählt wird, werden alle vorher gewählten Oberwellen in jede einzelne, vorher gewählte Harmonische unterteilt, auf 255 normalisiert und im zugehörigen Register für die harmonischen Verhältnisse abgespeichert. Das Maschinenprogramm HARM (7936–8032) berechnet, nachdem die zugehörigen Werte mit den Werten der Sinuswellen-Tabelle multipliziert worden sind, alle harmonischen Verhältnisse und speichert sie in der Wellenformtabelle. Diese Multiplikation für jedes harmonische Verhältnis wird 256 X wiederholt.

Die Wellenformtabelle enthält dann das gesamte Mittel aller 11 harmonischen Wellenformen mit ihrer Dominanz. Die Dominanz entspricht den Verhältnissen, die vom Programmierer gewählt werden können.

Wenn der Programmierer nur die erste Harmonische wählt, wird die Wellenform-Tabelle nur eine Sinuswelle enthalten. Wenn er nur die dritte Harmonische wählt, wird sie drei Sinuswellen beinhalten. Wenn nur die ungeraden Harmonischen mit dem relativen Verhältnis von 1000/N gewählt werden, wird die Wellenformtabelle eine Rechteckwelle etc. enthalten.

Wenn die Wellenformtabelle gefüllt ist, wird das Unterprogramm „PLAY" (7678) aufgerufen, um die Töne abzuspielen.

Vorher werden noch die zugehörigen Register für Tonhöhe und Tondauer mit Konstanten geladen. Dieser gesamte Ablauf wird dann für jeden der 11 harmonischen Obertöne wiederholt, bis die F-Taste gedrückt wird.

Das Programm COMPOSE bringt die vom Programmierer ausgewählten Noten in eine Form, die das Programm PLAY verstehen und abspielen kann. Wenn das Lied „A" als zu kompensierendes Lied ausgewählt wird, wird COMPOSE die Noten ab Zeile 6912 in den Speicher laden. Die Noten werden in drei Byte Paketen geladen:

1. Tondauer (Beat Duration)
2. Note A (Tonhöhe)
3. Note B (Tonhöhe)

Die ausgewählte „Beat Duration" im Bereich von 1/4 bis 60 wird mit 4 multipliziert und im ersten Speicherplatz abgelegt.
Da das Maschinenspracheprogramm eine Null als Programmende erkennt, wird durch Drücken der Taste F eine Null an diesen Speicherplatz gebracht. Die Note A kann einfach ein „C" oder auch ein „3C#+" sein. Das BASIC-Programm wird die Note A in eine Zahl umwandeln, die vom Maschinenspracheprogramm erkannt wird und in die richtige Tonhöhe verwandelt wird.

Das Maschinensprachenprogramm DECODE dekodiert die Noten, indem die 3 niederwertigen BITs als Oktave interpretiert werden. Diese Oktaven-Bezeichnung wird später dazu verwendet, die Tonhöhe in Schritten von 2 herunterzuteilen. Die Tonhöhe wird von den fünf höherwertigen BITs bestimmt. Sie wird aus der Tonhöhentabelle (8128–8188) hergeleitet. Damit die Tonhöhe richtig gewählt wird, wird die ausgewählte Note mit 8 multipliziert und zu einer Oktave hinzugezählt. Sie wird dann in der zweiten Speicherzelle abgelegt. Die Note B wird auf gleiche Weise erzeugt und in Speicherzelle 3 abgelegt. Diese Gruppe von drei Speicherplätzen gehören zur ersten Note. Eine Null wird dann im folgenden Speicherplatz abgelegt und die PLAY-Routine wird aufgerufen. Die letzte Note wird dann mit der ausgewählten Tonhöhe und Dauer abgespielt. Achtung: Die „Beat-Duration" ist ein relativer Wert, die absolute Dauer kann während des Abspielens durch Ändern des Tempos gewählt werden.

Dieser Lied-Komponier-Vorgang wird solange fortgesetzt, bis die Taste „F" = Finish gedrückt wird. Eine Null wird dann in die nächste Zelle für die „Beat Duration" gegeben und das COMPOSE-Programm wird verlassen.

Wenn Sie jetzt wissen wollen, wie weit Sie gekommen sind, drücken Sie die P-Taste (PLAY). Eine Null wird im nächsten Register abgelegt, das Lied wird abgespielt und am Ende können Sie wieder zum Compose zurückkehren, wo Sie die Komposition unterbrochen haben.

Wenn das Lied fertig ist, wird der Platz des folgenden freien Speicherraumes in das nächste Lied Index-Register geladen. So muß, wenn die letzte Note (Null) des Liedes B in Zelle 7580 liegt, die Speicherzelle 7580+3 für das Lied C im C-Register abgespeichert werden. Von Zelle 7000 an finden Sie ein Diagnoseprogramm.

Dieses Programm ermöglicht die codierten Anfangsadressen für jedes Lied und ermöglicht das Auslisten des Liederspeichers in Dreierreihen.

Sie können auch den PET dazu bringen, daß er Zufallsnoten selbst generiert, indem Sie in den Liederbereich Zufallszahlen eingeben. Vergessen Sie bitte nicht, am Ende eines Liedes eine Null einzugeben, denn sonst spielt der PET den ganzen Speicherinhalt ab.

Der Teil „PLAY" des Music-Programmes funktioniert sehr einfach. Das gewählte Tempo wird in die Speicherzelle 6865 geladen und die Lader werden in der gewünschten Reihenfolge abgespielt. Das Maschinenprogramm PLAY erledigt die ganze Arbeit. BASIC dient Ihnen nur noch dazu, die Maschinenroutine an den Anfang des nächsten Programmes zu bringen.

Wenn Sie weiter in die Maschinenprogrammierung einsteigen wollen, verwenden Sie das Programm „MACHINE".

Die Programme „HARM" und „MULT" arbeiten zusammen und errechnen die Wellenform-Tabelle. Sie arbeiten mit Fourier-Synthese-Techniken in Verbindung mit einer Sinustabelle. Sie können das BASIC-Programm so ändern, daß nach jedem Lied die Wellenformtabelle neu berechnet wird. Das DIRECTOR-Programm wird zusammen mit dem Programm DECODE dazu verwendet, den 3Byte-Code zu entschlüsseln und in die entsprechende Tonhöhe und Tondauer zu verwandeln.

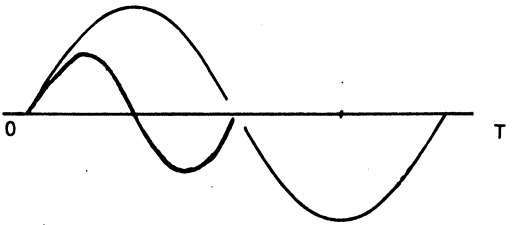
Die Tonhöhenwerte werden durch Oktaventeilung der zwei Byte-Informationen in der PIT-Tabelle ermittelt. Die PLAY-Routine nimmt diese Werte als Werte für Ihre Schrittgröße in der Wellenformtabelle. Wenn die Note B eine Oktave höher liegt als Note A, so wird sie in doppelt langen Schritten entlang der Wellentabelle schreiten. Dabei macht sie einen kompletten Zyklus in der halben Zeit und erzeugt eine Frequenz mit doppelter Tonhöhe. Die AUX-Routine sucht nun nur noch das Ende jeder Note und enthält eine Reihe von Leerzeiten.

Diese freie Zeit kann zur Ausführung sinnvoller

Befehle genutzt werden. (Vibrato, Tremolo, Phasenverschiebungen etc.)
Beispiel: Interessante Töne mit Phasenverschiebung können dadurch erzeugt werden, daß die doppelten Schrittwerte (6793–6796) so geändert werden, daß sie sich leicht von dem echten, doppelten Wert unterscheiden (6789–6792)

Theorie der Computer-Music
Alle Töne sind das Ergebnis einer bestimmten Mischung aus verschiedenen Tonfrequenzschwingungen. Töne wie z. B. das Geräusch eines Flugzeuges, des Windes oder des Lautes „sch" sind Mischungen von vielen Frequenzen in einem bestimmten Frequenzband. Diese Mischung wird auch mit „Geräusch" bezeichnet. Musiktöne sind grundsätzlich genau kontrollierte Mischungen von Sinuswellen. Die Frequenzen der einzelnen Sinuswellen verhalten sich zueinander im Verhältnis ganzzahliger Teiler.

Die Grundfrequenz ist immer die niedrigste Frequenz. Die zweite Harmonische oder 1. Oberschwingung hat die doppelte Frequenz.



$f = \frac{C}{T}$ Grundschwingung

$f2 = \frac{2C}{T}$ 1. Oberschwingung, 2. Harmonische

$f3 = \frac{3C}{T}$ 2. Oberschwingung, 3. Harmonische

Die 2. Oberschwingung hat die dreifache Frequenz. (3. Harmonische) u. s. w. Das relative Verhältnis dieser harmonischen Frequenzen ist der bestimmende Faktor für die unterschiedliche Toncharakteristik der einzelnen Instrumente.

In Wirklichkeit ist echte Musik wesentlich komplexer. Jede Note hat nicht nur eine bestimmte Tonhöhe und eine eigene harmonische Zusammensetzung. Sie verändert auch Ihre harmonische Zusammensetzung und die relative Lautstärkencharakteristik während eine Note gespielt wird. (Laut und schnell ansteigende Töne, langsam abklingende Töne).

Mit einem Digital/Analog-Wandler am Microcomputer läßt sich die harmonische Zusammensetzung eines Tones recht präzise einstellen. Jedoch sind die heute verfügbaren Microprocessoren meist noch zu langsam, um die harmonischen Verhältnisse während des Tones zu verändern. Sogar die größten Computer sind heute noch nicht in der Lage, richtige Musik in Echtzeit zu simulieren. Die ADAK-1-Platine ermöglicht es Ihnen, mit dem PET die harmonische Struktur eines Tones oder Anstiegs- und Abfallcharakteristik eines Tones zu steuern. (Aber nicht beides gleichzeitig) Auf jeden Fall werden Sie feststellen, daß es wesentlich mehr Spaß macht, Musik mit einem A/D-Wandler zu machen als nur einfache Rechteckwellenmusik zu erzeugen.

Die ADAK-1-Platine ermöglicht Ihnen eine Kontrolle der harmonischen Struktur eines Tones, genau wie dies die großen Computer für Musiksynthese tun. Der einzige Unterschied liegt darin, daß eine 256 x 8Bit lange Wellenform-Tabelle benutzt wird.

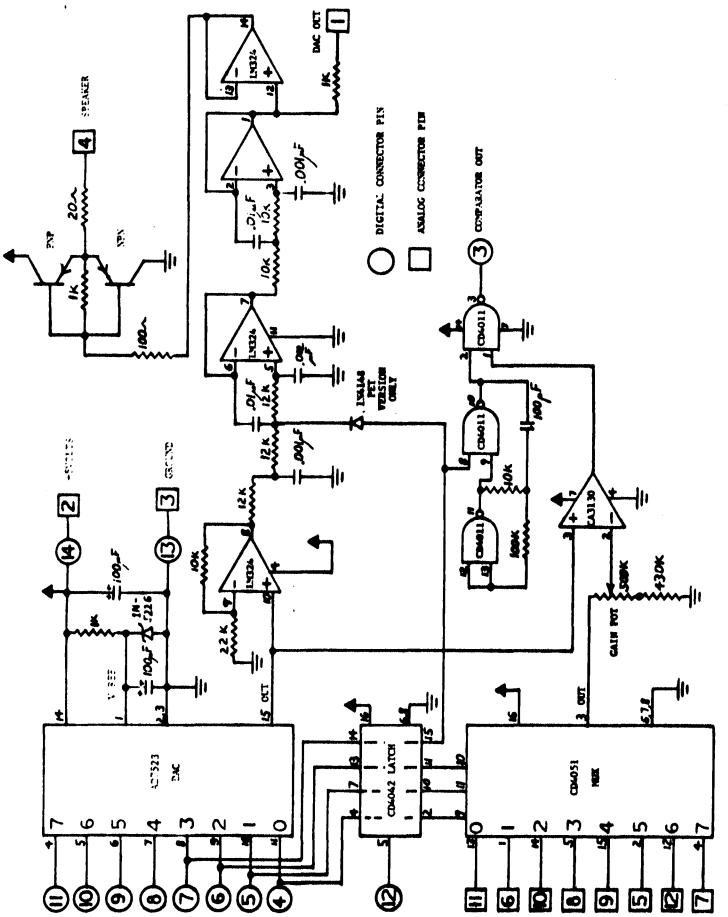
Große Computer verwenden meist eine 8K x 32 Bit lange Wellenform-Tabelle.

Auf jeden Fall muß jede Anstrengung unternommen werden, um so präzise wie möglich zu sein. Zu den zugehörigen Softwareprogrammen wird die Tonhöhe in 16 Bit Arithmetik errechnet. Die unteren 8 Bit werden als Übertrag gespeichert. Dies führt zu einer wesentlich genaueren Berechnung.

Die Wellenform-Tabelle wird durch Multiplikation der relativen Verhältnisse jeder Harmonischen mit der Position in der Sinustabelle errechnet. Nehmen wir an, wir wollen eine Wellenform-Tabelle für 50 % Grundfrequenz und 25 % 1. Oberwelle und 25 % 2. Oberwelle erstellen. Wenn wir dann später eine Note mit 1000 Hz spielen wollen, wird die resultierende Wellenform ein Gemisch von

50 %	1000 Hz Sinuston
25 %	2000 Hz Sinuston
25 %	3000 Hz Sinuston

Das Abspielen der Wellenform-Tabelle geschieht dadurch, daß wir einen Pointer (Zeiger) entlang der Tabelle führen und die Werte in den Ausgabeport bringen. Der D/A-Wandler bringt dann die entsprechende Ausgangsspannung. Das Endergebnis ist dann eine Wellenform, die eine ähnliche Form wie die Wellenform-Tabelle hat. Die Frequenz ist proportional den Schritten, die vom Pointer ausgeführt werden.



ADAK-1 SCHALTBILD

PET MEMORY MAP

Location	Mnemonic	Function
826-913..	SAR	A-D conversion Program
920-927	ANSWER	Digitized inputs 0-7
930-979	SOUND	Sound generating program
980	PITCH	Pitch for SOUND
981	DELAY	Duration for SOUND
990-1022	DEBUG	Break-point routine
1023	DATA	Data pointer
6400-8188	MUSIC	Music Area
6400-6655	WTAB	Wavetable
6656-6783	STAB	Sinewave table
6784	LONG	Note duration
6785-6786	MUL,TIP	Two numbers to be multiplied
6787-6788	LPRA,LPRB	Note A&B remainders
6789-6792	HPIB,LPIB,HPIA,LPIA	High and low pitch increment bytes
6793-6796	HPDB,LPDB,HPDA,LPDA	Double increments
6797	NOW	Harmonic table pointer
6798-6808	HPT	Pointers for 11 harmonics
6814-6815	TEM-P,TEM-M	Temporary addition & subtraction
6816-6826	HPRO	Proportions of 11 harmonics
6864	OCT	Temporary octave storage
6865-6866	T1,T2	Permanent & temporary tempo
6867	NOTE	Note pointer
6868-6869	TEMX,TEMY	Temporary X&Y storage
6870	PTEM	Temporary pitch storage
6871	WAIT	Delay register
6900-6910	SPT	Locator for songs 0-L
6912-7679	SONG	Reserved for 12 songs
7680-7803	DIRECTOR	Controls the notes and duration
7804-7851	PLAY	Moves along the wavetable
7852-7935	AUX	Determines end of note
7936-8032	HARM	Computes wavetable
8048-8065	MULT	8 bit multiplier routine
8080-8125	DECODE	Decodes pitch & octaves
8128-8188	PIT TABLE	Pitch table

SAR LISTING

Register	Machine Code	Function
826	A9 FF	LDAIM 255
828	8D 43 E8	STA 59459
831	A9 00	LDAIM 0
833	8D 4B E8	STA 59467
836	A2 03	LDXIM 3
838	AD 4C E8	LDA 59468
841	8D 97 03	STA 919
844	CA	DEX
845	10 07	BPL 7(TO 854)
847	AD 97 03	LDA 919
850	8D 4C E8	STA 59468
853	60	RTS
854	A9 C1	LDAIM 193
856	8D 4C E8	STA 59468
859	8E 41 E8	STX 59457
862	A9 E1	LDAIM 225
864	8D 4C E8	STA 59468
867	A9 80	LDAIM 128
869	8D 96 03	STA 918
872	8D 41 E8	STA 59457
875	A0 08	LDYIM 8
877	88	DEY
878	D0 FD	BNE 253(TO 877)
880	AC 41 E8	LDY 59457
883	A9 02	LDAIM 2
885	2D 4D E8	AND 59469
888	F0 05	BEQ 5(TO 895)
890	98	TYA
891	4D 96 03	EOR 918
894	A8	TAY
895	4E 96 03	LSR 918
898	F0 07	BEQ 7(TO 907)
900	98	TYA
901	4D 96 03	EOR 918
904	4C 68 03	JMP 872
907	98	TYA
908	9D 98 03	STAX 920
911	4C 4C 03	JMP 844

SOUND LISTING

Register	Machine Code	Function
930	A9 C9	LDAIM 201
932	8D 4C E8	STA 59468
935	8D 41 E8	STA 59457
938	AD 97 03	LDA 919
941	8D 4C E8	STA 59468
944	A9 FF	LDAIM 255
946	8D A1 03	STA 929
949	AC D5 03	LDY 981
952	8D 41 E8	STA 59457
955	AE D4 03	LDX 980
958	EA	NOP
959	EA	NOP
960	EA	NOP
961	EA	NOP
962	CA	DEX
963	D0 F9	BNE 249(TO 958)
965	AD A1 03	LDA 929
968	4D 41 E8	EOR 59457
971	88	DEY
972	D0 EA	BNE 234(TO 952)
974	4E A1 03	LSR 929
977	D0 E2	BNE 226(TO 949)
979	60	RTS

DEBUG LISTING

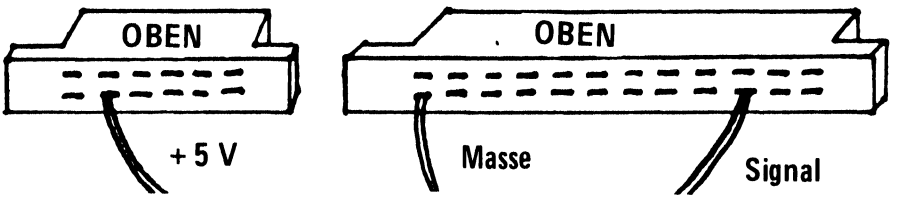
Register	Machine Code	Function
990	AD DC 03	LDA 988
993	48	PHA
994	28	PLP
995	AD D9 03	LDA 985
998	AE DA 03	LDX 986
1001	AC DB 03	LDY 987
1004	20 3A 03	JSR 826
1007	8D D9 03	STA 985
1010	8E DA 03	STX 986
1013	8C DB 03	STY 987
1016	08	PHP
1017	68	PLA
1018	8D DC 03	STA 988
1021	58	CLI
1022	60	RTS

Spracheingabe für PET

Das nachfolgend beschriebene Spracheingabegerät gehört zu den preiswertesten Systemen dieser Art. Die Funktion basiert aber darauf, daß eher festgestellt wird, ob gesprochen wird, als was gesprochen wird. (Unterschied zur Spracherkennung). Diese Art macht es möglich, zu geringen Kosten eine einfache Spracheingabe zu fertigen. In vielen Fällen kann jedoch diese Spracheingabe eine Spracherkennungsschaltung ersetzen.

Die Spracheingabeschaltung entnimmt ihre Betriebsspannung dem 2. Cassetteninterface-Anschluß. Sie benötigt ca. 5 mA. Der Ausgang des Gerätes wird mit Pin 7 (PA7) des User-Ports verbunden (Signal) und die Masse an Pin A gelegt.

Für den Anschluß benötigen Sie zwei verschiedene Stecker (1. einen Stecker mit 24 Kontakten CINCH 251-12-90-160 und einen Stecker mit 12 Kontakten CINCH 250-06-90-170. Der Anschluß sollte wie folgt gemacht werden.



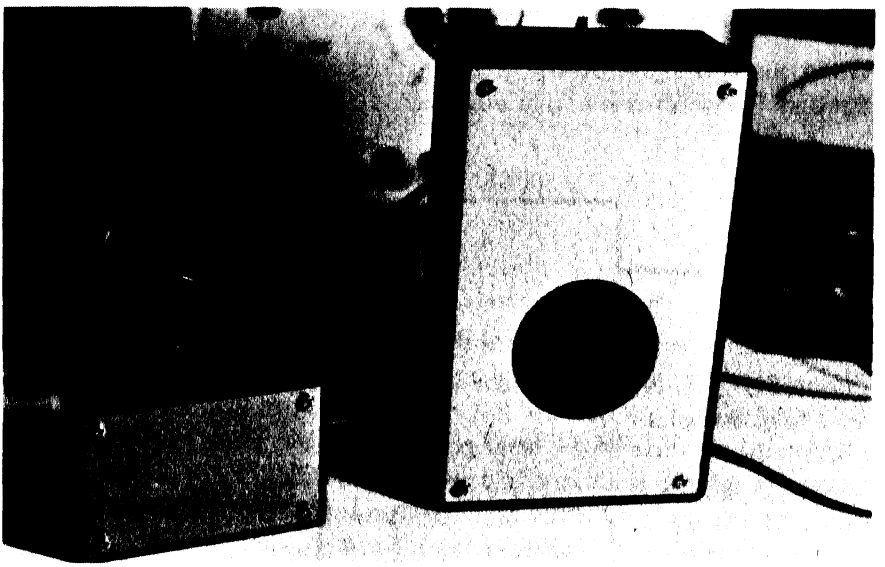
Bitte achten Sie darauf, daß nur in angeschaltetem Zustand ein Stekker aufgesteckt oder abgezogen werden darf.
Wenn Sie das Eingabegerät angesteckt haben, können Sie mit folgendem Programm die Signale der Spracheingabe auf dem Bildschirm sichtbar machen.

Die kleinen Buchstaben im nun folgenden Listing bedeuten folgendes:

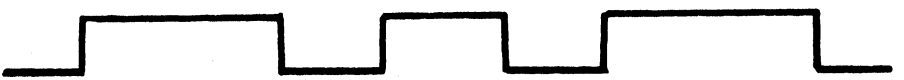
ch = CLEAR HOME
r = SHIFT R
e = SHIFT B

```
100 POKE 59459,0
110 IF PEEK(59471)<128 GOTO 110
120 PRINT "ch"
130 CTR=1
140 IF PEEK(59471)>128 GOTO 190
150 CTR=CTR+1
160 IF CTR>30 GOTO 110
170 PRINT "r";
180 GOTO 140
190 CTR=0
200 PRINT "e";
210 GOTO 140
```

Laden Sie Listing 1 und geben RUN. Dann sprechen Sie bitte in das Eingabemikrophon. (Im Bild links)



Sie werden folgende Wellenformen auf dem Bildschirm sehen:

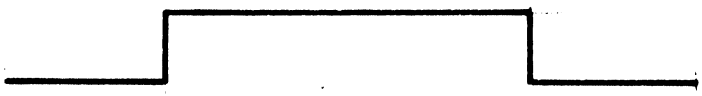


Das Signal auf dem Bildschirm schwankt zwischen logisch „1" und logisch„0". Wenn gesprochen wird, ist es log. „1". Wenn Sie ständig sprechen, füllt sich der Bildschirm mit Signalen. Wenn Sie für 2 Sekunden schweigen, stoppt das Programm und die Signale bleiben auf dem Bildschirm erhalten. Sie können diese 2 Sekunden verändern, indem die Zahl 30 in Zeile 160 auf 40 erhöhen. Sie haben dann die Möglichkeit, längere Pausen einzulegen als 2 Sekunden. Jeder Ton, nach dem Stoppen des Programmes, startet einen neuen Zyklus.

Arbeitsweise der Spracheingabe

Sie arbeitet auf einem ganz einfachen, jedoch sehr leistungsfähigen Prinzip. In einer patentierten Schaltung werden Schallwellen empfangen, gefiltert und so aufbereitet, daß am Ausgang eine log. „1" entsteht, wenn ein Ton vorhanden ist und eine log. „0" erzeugt wird, wenn kein Ton vorhanden ist.

Wenn Sie z.B. „JA" (sehr langsam aussprechen) sagen, könnte dies auf dem Bildschirm wie folgt aussehen.



Wenn Sie „NEIN" sagen, (sehr kurz und abgehackt) dann könnte dies wie folgt aussehen:

Man kann die Spracheingabeschaltung auch zur Spracherkennung einsetzen. Hier erkennt der Computer dann nur, wie es ausgesprochen wurde. Wenn Sätze erkannt werden sollen, so ist die Information entscheidend, wieviele Worte in diesem Satz gesprochen wurden.

In diesem Falle arbeitet das System wie ein „Wortzähler". Im Betrieb, bei dem ein Wort erkannt werden soll, kommt es auf die Länge des Wortes an.

Im Betrieb als Wortzähler muß folgendes geschehen:

1. Feststellen, wenn ein Wort beginnt
2. Kurze Störimpulse müssen unterdrückt werden
4. Festlegung, wenn der Satz zu Ende ist.

Dies kann mit dem folgenden Unterprogramm errechnet werden:

```
2000 NN=0
2005 PTR=0
2010 IF PEEK(59471)<128 GOTO 2040
2020 NN=NN+1
2030 GOTO 2010
2040 IF NN<6 GOTO 2140
2050 CTR=0
2060 CTR=CTR+1
2070 IF PEEK(59471)>127 GOTO 2100
2080 IF CTR>30 GOTO 2130
2090 GOTO 2060
2100 IF CTR<7 GOTO 2050
2110 PTR=PTR+1
2120 GOTO 2050
2130 PTR=PTR+1
2140 RETURN
```

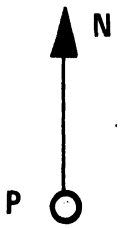
Das Unterprogramm nimmt an, daß ein Satz dann zu Ende ist, wenn die Ruheperiode 2 Sekunden überschritten hat.
Die maximale Ruhezeit zwischen zwei Worten muß also unter 2 Sekunden liegen. Hier gibt es unzählig viele Anwendungsmöglichkeiten. Das Programm VOICETRAP ist im Lieferumfang des Spracheingabegerätes enthalten). Es demonstriert eine einfache aber sehr interessante Anwendung, wobei die Anzahl der Worte pro Satz gezählt wird.

PTR gibt die Anzahl der Worte an. Je nachdem, ob PTR = 1,2,3 oder 4 ist, wird eine Maus in vier verschiedene Richtungen gelenkt. Wenn PTR gleich Null oder größer als vier ist, wird dies nicht erkannt. Auf jeden Fall ist es wichtig, daß jedes Wort klar und deutlich ausgesprochen wird. Wenn zwei Worte ineinanderlaufen, wird dies als ein Wort erkannt. Wenn Sie zu weit auseinander sind, kann die Pause so

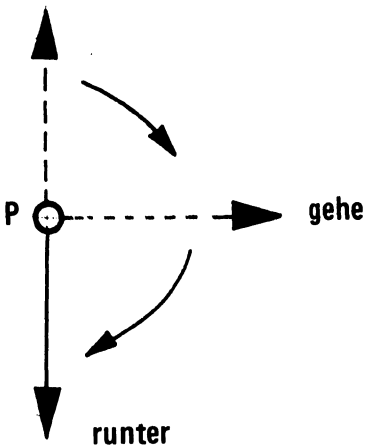
groß werden, daß zwei Worte in verschiedenen Sätzen erkannt werden. Auch die Lautstärke sollte bei der Spracheingabe genügend hoch sein. Die Empfindlichkeit des Eingabegerätes wurde etwas reduziert, damit Störgeräusche genügend unterdrückt werden.

Deshalb sollte das Mikrophon ca. 3 – 5 cm vom Mund entfernt sein. Eine andere Methode zur Kontrolle eines Computers über die Sprache wird im Programm VOICEMAZE demonstriert. Nehmen wir einmal an, Sie wollen den Cursor auf dem Bildschirm über Spracheingabe steuern. Es soll festgelegt werden, daß ein imaginärer Zeiger immer dann in seiner Richtung verändert wird, wenn ein Wort gesprochen wird. Die Endrichtung, die der Cursor am Ende des Satzes einnimmt, ist dann die Richtung, in die sich der Cursor bewegen soll.

Nehmen wir an, daß wir mit der imaginären Zeigerstellung wie folgt beginnen:



Nun sagen wir „GEHE RUNTER". Dies sind zwei Worte und bringen unseren Zeiger um zwei Einheiten in einer Drehung nach rechts um den Punkt P:



In der Endrichtung zeigt dann der Pfeil nach unten. Würden wir eingeben:"GEHE NUN ENDLICH RUNTER MEIN FREUND", so erhalten wir das gleiche Ergebnis. Nur läuft unser Zeiger einmal ganz herum. Es ist sehr wichtig, daß alle Worte klar und abgegrenzt ausgesprochen werden.

Nun sind Sie sicher in der Lage, ein Programm zu schreiben, welches den Cursor mit Hilfe Ihrer Stimme auf dem Bildschirm bewegen kann.

Der imaginäre Zeiger kann durch einen Zähler bis vier ersetzt werden, welcher immer in der Reihenfolge 1, 2, 3, 4, 1, 2, 3,4, 1, 2.... durchzählt.

Spracheingabe mit Wortlängenerkennung

Die verschiedenen Worte in unserem Sprachgebrauch haben unterschiedliche Längen. (Tondauer) Ein Beispiel für die Unterschiede geben Ihnen die Worte „JA" und „RATIONALISIERUNG". Dieser Unterschied kann von einem Computer (Ihrem PET) durch ein einfaches Programm erkannt werden. Natürlich können auch mehr als 2 Worte unterschieden werden, sicherer arbeitet jedoch die Schaltung mit nur zwei Worten.

Mit der Unterscheidung zwischen einem langen und einem kurzen Wort kann man schon recht praktische Schaltungen realisieren. Es können z.B. in der Produktion an Bändern Mikrophone angebracht werden, und das Band kann durch Zuruf eines bestimmten Wortes gestoppt oder eingeschaltet werden. Die Hände bleiben dabei frei. Auch Ratespiele mit dem Computer können sehr leicht entworfen werden.

Nachfolgend soll ein Programm dargestellt werden, welches Ihnen bei der Auswahl der richtigen Worte helfen soll. Das Programm gibt die Länge des Wortes auf dem Bildschirm aus. Sprechen Sie die beiden Worte, von denen Sie glauben, daß der Längenunterschied recht groß ist, ins Mikrophon und prüfen Sie das Ergebnis. Versuchen Sie es mit vielen Worten und nehmen Sie die Worte mit der größten Differenz.

```
300 POKE 59459,0
310 IF PEEK(59471)<128 THEN 310
320 CTR=0
330 CTR=CTR+1
340 IF PEEK (59471)>128 THEN 330
350 PRINT " WORD LENGTH = "; CTR
360 GOTO 310
```

Wenn Sie die geeigneten Worte gefunden haben, sprechen Sie diese mehrmals hintereinander und vergleichen die Dauer pro Wort. Auch bei gleichen Worten kann es Streuungen geben.

Wenn einmal zwei Worte zwar verschiedener Bedeutung, doch die gleiche Länge haben, versuchen Sie es mit anderen Worten.

Auch die Aussprache kann helfen. In die Länge ziehen des Wortes oder abgehackt und kurz aussprechen.

Das nachfolgende Programm liefert gute Ergebnisse mit einem langen Wort „JA" und einem kurz abgehackten Wort „NEIN",

"JA" lang ausdehnen
"NEIN" ganz kurz aussprechen.

```
300 POKE 59459,0
310 IF PEEK(59471)<128 THEN 310
320 CTR=0
330 CTR=CTR+1
340 IF PEEK(59471)>128 THEN 330
350 IF CTR<7 THEN 310
360 IF CTR<13 THEN 390
370 PRINT "YES"
380 GOTO 310
390 PRINT  "NO"
400 GOTO 310
```

Zeile 350 dient zur Störimpulsunterdrückung. (Impulse und Geräusche, die kürzer als 7 Einheiten sind)

„NEIN" wird mit Längeneinheiten zwischen 7 und 12 erkannt
„JA" wird mit Längeneinheiten zwischen 13 und länger erkannt (Siehe Statement 360). Alles, was länger als 13 Einheiten ist, wird als JA erkannt.

Diese Konstanten wurden experimentell ermittelt. Sie können sie jedoch nach Ihrer eigenen Stimme justieren. Es ist möglich, die Zähltechnik und die Wortlängentechnik zu mischen. Sollten Sie einmal selbst etwas experimentieren, so würden wir uns über Ihre Mitteilung freuen.

Einen kompletten Hardwaresatz mit Spracheingabe und Tonausgabe sowie reichlicher Software können Sie unter der Bestell-Nr. P45 für DM 249,-- incl. Mwst. beziehen.

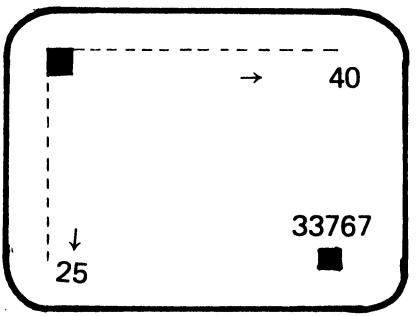
POKE und Bildschirm

Der POKE-Befehl bringt einen Wert in eine bestimmte Speicherzelle.
Die Befehlsform sieht wie folgt aus:

POKE A,B
Wobei A = die Adresse in dezimal
B = der Wert in dezimal

Beide Werte A und B können auch Variable sein.
Wir können den POKE-Befehl jetzt dazu verwenden, Zeichen an einer bestimmten Stelle auf dem Bildschirm zu erzeugen.
Der Bildschirm kann also wie ein Speicher betrachtet werden.

Der erste Punkt oben links
ist die Adresse: 32768



Beispiel:
POKE 32768,1
POKE 32768,49

„49" = Zahl die 1 entspricht.
(s. Tabelle)

B darf nur zwischen 0 und 255 liegen → Siehe Tabelle

Will man ein Zeichen an einer anderen Stelle darstellen, muß zu 32768 immer 1 dazugezählt werden. Mit 40 kommen Sie in die nächste Zeile.

Unterprogramm: Zeichen durch C gegeben
L = Zeilennummer (1 – 25)
P = Position (1 – 40)

```
1000 POKE 32768 + (40*(L-1))+(P-1),C
1010 RETURN
```

```
100  FOR L = 1 TO 25
110   FOR P = 1 TO 40
120     C = C + 1 : IF C > 255 THEN C = 1
130     GOSUB 1000
140    NEXT P
150  NEXT L
160  END
1000 POKE 32768 + ( 40 * (L – 1 ) + (P–1), C
1010 RETURN
```

```
100  FOR I = 0 TO 999
110  C = C + 1 : IF C > 255 THEN C = 1
120  POKE 32768 + I,C
130  NEXT I
140  END
```

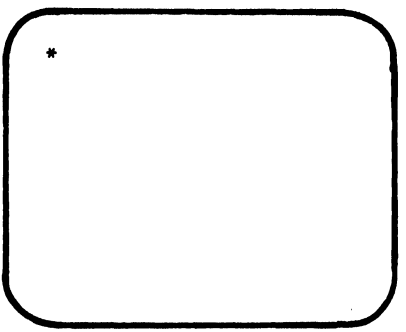
PEEKING in den Bildschirm

PEEK (L)	L = Platz in dezimal im Speicher
C = PEEK (L)	Nimmt den Inhalt von L und macht es zur Variablen C. Es wird immer der Code entsprechend der Tabelle gezeigt.

Home * Return

PRINT PEEK (32768)
42
42 = * Asterik-Sternchen

POKE 32808,42



Auf diese Weise können Sie die Codewörter für jedes Zeichen herausfinden.

PET-Zeichencodierung

Zeichen können auf dem PET Bildschirm auf zwei verschiedene Arten dargestellt werden.

1. Durch Poken des zugehörigen Zeichens in den Bildspeicher. Bildspeicheradresse von 32768 bis 33767 Dez.
1. Ausdrucken durch PRINT

Neben diesen Eigenschaften wird der Speicherbereich durch zwei verschiedene Charaktergeneratoren interpretiert.

1. Im Standardbereich muß Zelle 59468 12 dezimal oder XXXX 110X enthalten. (59468,12)

2. Im Betrieb für Kleinbuchstaben wird in die Zelle 59468,14 eingepoked.

In der nachfolgenden Tabelle zeigen die OFF und RVS-Werte, welche Zahl in den Bildspeicher durch POKE eingegeben werden muß, um das zugehörige Zeichen darzustellen.

OFF RVS	CHR$
	105 233 169
	122 250 186
	94 222 222
	95 223 223

	OFF	RVS	CHR$
@	64	192	192
	0	128	64
A	65	193	193
	1	129	65
B	66	194	194
	2	130	66
C	67	195	195
	3	131	67
D	68	196	196
	4	132	68
E	69	197	197
	5	133	69
F	70	198	198
	6	134	70
G	71	199	199
	7	135	71
H	72	200	200
	8	136	72
I	73	201	201
	9	137	73
J	74	202	202
	10	138	74
K	75	203	203
	11	139	75
L	76	204	204
	12	140	76
M	77	205	205
	13	141	77
N	78	206	206
	14	142	78
O	79	207	207
	15	143	79

	OFF	RVS	CHR$
P	80	208	208
	16	144	80
Q	81	209	209
	17	145	81
R	82	210	210
	18	146	82
S	83	211	211
	19	147	83
T	84	212	212
	20	148	84
U	85	213	213
	21	149	85
V	86	214	214
	22	150	86
W	87	215	215
	23	151	87
X	88	216	216
	24	152	88
Y	89	217	217
	25	153	89
Z	90	218	218
	26	154	90
[	91	219	219
	27	155	91
\	92	220	220
	28	156	92
]	93	221	221
	29	157	93
↑ π	94	222	222
	30	158	94
←	95	223	223
	31	159	95

RETURN		141
		13

CLR SCREEN HOME		147
		19
OFF RVS ON		146
		18

	OFF	RVS	CHR$
SP	96	224	160
	32	160	32
/	97	225	161
	33	161	33
//	98	226	162
	34	162	34
#	99	227	163
	35	163	35
$	100	228	164
	36	164	36
%	101	229	165
	37	165	37
&	102	230	166
	38	166	38
'	103	231	167
	39	167	39
(	104	232	168
	40	168	40
)	105	233	169
	41	169	41
*	106	234	170
	42	170	42
+	107	235	171
	43	171	43
.	108	236	172
	44	172	44
-	109	237	173
	45	173	45
•	110	238	174
	46	174	46
/	111	239	175
	47	175	47

	OFF	RVS	CHR$
Ø	112	240	176
	48	176	48
1	113	241	177
	49	177	49
2	114	242	178
	50	178	50
3	115	243	179
	51	179	51
4	116	244	180
	52	180	52
5	117	245	181
	53	181	53
6	118	246	182
	54	182	54
7	119	247	183
	55	183	55
8	120	248	184
	56	184	56
9	121	249	185
	57	185	57
:	122	250	186
	58	186	58
;	123	251	187
	59	187	59
<	124	252	188
	60	188	60
=	125	253	189
	61	189	61
>	126	254	190
	62	190	62
?	127	255	191
	63	191	63

CURSOR ↑↓	145	
	17	
CURSOR ←→	157	
	29	

INST DEL	148	
	20	
RUN STOP	131	
	3	

Beispiel: Die Zahl 83 in Zelle 32768 ergibt ein Herz. Wird die Zahl 211 eingepoked, wird das Herz negativ dargestellt. Dem Buchstaben S entspricht als Wert die Zahl 19. Negativ wird S durch die Zahl 147.

Beispiel:
```
10 POKE 32768,83
RUN
```

Das Ausdrucken durch PRINT geschieht über den PRINT CHR $-Befehl. Hierzu finden Sie den Wert ganz rechts in der Spalte.

POKE 32768,83 und PRINT CHR$ (211) bewirken das gleiche. Mit beiden Befehlen kann vom Programm her eine Kontrolle über die einzelnen Zeichen erfolgen.

Das nachfolgende Programm druckt Ihnen den Vorrat der graphischen Zeichen aus.

```
READY.

 1 REM DARSTELLUNG ASCII FUER GRAFIK
 10 PRINT"ANZEIGE ASCII FUER GRAFIK"
 11 PRINT
 12 PRINT"DRUECKE'RUN/STOP' HALTEN -":PRINT
 20 FOR J=1TO20
 25 I=J+160
 30 W$=CHR$(I):X$=CHR$(I+25)
 35 Y$=CHR$(I+50):Z$=CHR$(I+75)
 40 PRINT W$;" =";  I; TAB(10); X$;" =";  I+25;
 45 PRINT TAB(20); Y$; " =";  I+50; TAB(30); Z$; " =";  I+75
 50 PRINT
 55 IFJ<>10THEN65
 60 GOSUB 100
 65 NEXTJ
 70 GOTO999
 100 REM ZEITSCHLEIFE
 110 FOR K=1 TO 5000
 120 NEXT K
 130 RETURN
 999 END
READY.
```

	Zeichen auf dem Bildschirm	ASCII No.	Zeichen auf Tastatur
→ OFF / RVS		146	OFF
→ OFF / RVS		18	RVS
→ RUN / STOP		3	STOP
→ CLEAR / HOME		147	CRL
→ CLEAR / HOME		19	HOME
↑ CURSOR / → ↓ CURSOR		145	HOCH
→ ↑ CURSOR / ↓ CURSOR		17	RUNTER
← CURSOR / → → CURSOR		157	LINKS
→ ←CURSOR / → CURSOR		29	RECHTS
INST / → DEL		148	INST

Bildschirm des PET

Wenn Sie die Graphikeigenschaften des PET voll nutzen wollen, wird Ihnen die nachfolgende Skizze wertvolle Dienste leisten. Sie können das gewünschte Bild zuerst in seiner wirklichen Form eintragen und dann den entsprechenden Character-Code (Zeichencode) einsetzen. Die Adresse läßt sich dann ganz leicht ermitteln. Heben Sie sich das Original auf und fertigen Sie sich Fotokopien an, in die Sie Ihre eigenen Eintragungen machen können.

Links finden Sie die Dezimalwerte der Adressen am linken Bildschirmrand. Wenn Sie die Zahlen in der horizontalen Reihe dazuaddieren, finden Sie den Wert für den gewünschten Bildpunkt.

Beispiel: Die Mitte des Bildschirmes ist Adresse

33248 + 10 + 9 = 33267

Ein Herz in der Mitte wäre dann POKE 33267,83.

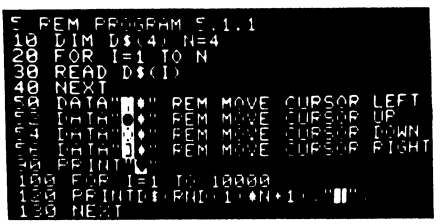
Adresse		0	1	2	3	4	5	6	7	8	9	+10 0	1	2	3	4	5	6	7	8	9	+20 0	1	2	3	4	5	6	7	8	9	+30 0	1	2	3	4	5	6	7	8	9
3 2 7 6 8																																									
3 2 8 0 8																																									
3 2 8 4 8																																									
3 2 8 8 8																																									
2 3 9 2 8																																									
3 2 9 6 8																																									
3 3 0 0 8																																									
3 3 0 4 8																																									
3 3 0 8 8																																									
3 3 1 2 8																																									
3 3 1 6 8																																									
3 3 2 0 8																																									
3 3 2 4 8																																									
3 3 2 8 8																																									
3 3 3 2 8																																									
3 3 3 6 8																																									
3 3 4 0 8																																									
3 3 4 4 8																																									
3 3 4 8 8																																									
3 3 5 2 8																																									
3 3 5 6 8																																									
3 3 6 0 8																																									
3 3 6 4 8																																									
3 3 6 8 8																																									
3 3 7 2 8																																									

Copyright 1979 by Ing. W. Hofacker GmbH, Tegernseerstr. 18, 8150 Holzkirchen

Programmierung von Bewegungsabläufen

Programmieren von Bewegungsabläufen auf dem PET

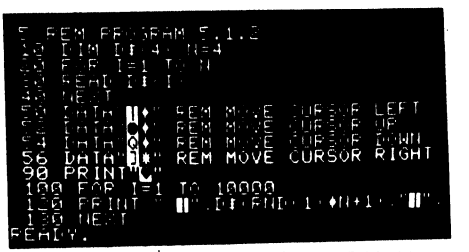
Eines der faszinierendsten Dinge, die man mit dem PET-Computer programmieren kann, sind sich bewegende Bilder auf dem Video-Display. Bei einem Heimkinoprojektor werden die einzelnen Bilder in einer schnellen Folge zu einem bewegten Bild. Auf dem PET-Bildschirm werden die Zeichen in einer schnellen Folge ausgewechselt. Als Beispiel betrachten wir uns das folgende Programm. Es wird einen laufenden Streifen auf dem Bildschirm zeichnen.



Zeile 10 legt ein Array von vier möglichen Zeichenreihen (character strings) und zwar für auf, ab, links und rechts.
Zeile 20 ist eine FOR NEXT-Schleife, welche das Array mit Data-Statements aus den Zeilen 50 – 56 füllt. Beachten Sie bitte, daß Zeile 50 das Graphiksymbol „ Cursor nach links", Zeile 52 das Graphiksymbol „ Cursor nach oben" usw. enthält. Bedenken Sie, daß alle Graphiksymbole inclusive „ DELETE„ gedruckt werden können. Zeile 100 ist das Ende der Schleife und erzeugt die wandernde Bewegung. Sie ist so gesetzt, daß die Schleife 10.000 mal durchlaufen wird. Sie kann natürlich mit der BREAK-Taste unterbrochen werden.

Zeile 120 ist das eigentliche Herz des Programmes. Der Ausdruck, welcher dem PRINT-Statement folgt, wählt über einen Zufallsgenerator eine der Richtungen willkürlich aus und druckt dies aus. Dieses bringt den Cursor an eine neue Stelle auf dem Bildschirm. Nun druckt es ein Asterisk aus und bildet einen neuen Rahmen. Beachten Sie, daß das PRINT-Statement mit einem Semikolon endet, damit die Cursor-Position nicht gestört wird.
Zeile 130 beendet die Schleife, welche in Zeile 100 begann.

Programm für einen laufenden Cursor



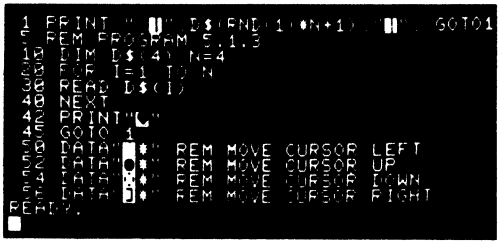
Hier scheint das Asterisk auf dem Bildschirm umherzuschießen. Bei Dunkelheit können Sie auch einen nachfolgenden Streifen sehen. Je nachdem, bis der Phosphor am Bildschirm seine Nachleuchtkraft verloren hat.

Eine schnellere Programm-Version

Nun haben wir bereits eine Reihe von schönen Programmen kennengelernt. Jetzt wollen wir sie etwas optimieren, indem wir die Anzahl der Anweisungen innerhalb der Schleife reduzieren.

Wir bringen den „Backspace"-Befehl zum Array D$ und löschen die gesamte Zeile 120. Um die FOR-Schleife zu beseitigen, geben wir einfach GOTO ein. Die Schleife wird nun zur ersten Zeile des Programmes gebracht und es entsteht keine Wartezeit, um die Zeilennummer zu finden. Das Programm muß ab jetzt jedoch mit RUN 10 gestartet werden.

Programm 5.1.3, optimierte Version von 5.1.2



Eine andere einfache, aber sehr effektvolle Bewegungstechnik besteht darin, den Bildschirm um eine Zeile weiterspringen zu lassen. Dies geschieht dadurch, daß Carriage Returns mit nachfolgenden PRINT-Anweisungen ohne Ausdruck eingegeben werden.

```
05 REM Progr. 5.1.4
10 FOR I = 1 TO25
20 PRINT
30 NEXT I
40 REM: Dieses Programm läßt das Bild
        am oberen Ende um eine Zei-
        le weiterspringen.
```

Wenn dies zu schnell geschieht, können Sie die Geschwindigkeit durch eine Verzögerungsschleife herabsetzen.

```
05 REM Progr. 5.1.5
10 FOR I = 1 TO25
20 PRINT
25 FOR X = 1 TO 250: NEXT X
30 NEXT I
40 REM: Dieses Programm läßt das Bild
        am oberen Ende um eine Zei-
        le weiterspringen.
```

Dieser Effekt kann dazu benutzt werden, ein Objekt über den oberen Rand des Bildschirms hinauslaufen zu lassen.

Auch kann ein sich bewegender Hintergrund hinter einem stehenden Bild erzeugt werden. Dies verschafft den Eindruck eines sich abwärts bewegenden Bildes. Ähnlich wie bei einem Tiefseetaucher oder Fallschirmspringer.

Objekte können so programmiert werden, daß deren ursprüngliche Lage am unteren Ende des Bildschirmes ist und durch eine Bewegungs-Schleife soviele Vorschübe erzeugt werden, bis sie den oberen Bildrand erreicht haben.

Bewegungskontrolle mit POKE und PEEK-Befehlen

Der PET-Bildschirm besteht aus einem Block von 1000 Zeichen, die über die Speicherzellen 32 768 bis 33768 zu erreichen sind. So kann das Programm mit dem Befehl POKE ein bestimmtes Zeichen an eine gewünschte Stelle auf dem Bildschirm direkt bringen. Dies geschieht ohne Rücksicht auf die Cursorsteuerung, die bei der PRINT-Anweisung gefordert wird.

Da der Bildschirm aus 25 Reihen mit je 40 Zeichen besteht, kann das folgende Programm mit dem POKE-Befehl ein Asterisk (*) an eine gewünschte Stelle auf dem Display bringen.

Die Formel, wie man die Platzierung aus Reihe und Spalte in die Bildschirm-Adresse umrechnen kann, sieht wie folgt aus:

Bildschirmadresse = 32768 + (Reihe *40)+ Spaltenzahl

```
05 REM Progra. 5.1.6
10 REM POKE * IN REIHE, SPALTE
20 FOR I = 1 TO 1000
30 INPUT „ EINGABE REIHE, SPAL-
   TE"; R,C
40 POKE 32768 + ( R - 1) * 40 + (C-1),42
50 NEXT I
```

Anmerkung: Die Zahl 42 in Zeile 40 des Programmes ist die Zahl für *. Es können beliebig andere Zeichen eingesetzt werden. Siehe Tabelle im PET-Manual.

Die Reihen und Spalten beginnen mit den Zahlen 0 und nicht mit 1, so daß wir 1 abziehen müssen. Ähnlich können wir mit dem PEEK-Befehl nachsehen, ob eine Bildschirmadresse schon belegt ist oder nicht.

```
05 REM Progr. 5.1.7
10 REM PEEK AT SCREEN
20 FOR I = 1 TO 1000
30 INPUT „EINGABE REIHE, SPALTE"; R, C
40 PRINT„ DIES IST ZEICHEN";
50 PRINT PEEK (32768 + (R-1) * 40 + (C -1))
60 NEXT I
```

Beachten Sie, daß der PEEK-Befehl eine Zahl liefert. Ist die Bildschirmadresse frei, erhalten wir eine „32".

A = 65
B = 66

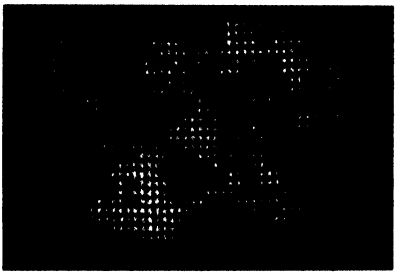
usw. Siehe Liste in Ihrem PET-Handbuch

Weiterhin ist zu beachten, daß PEEK und POKE-Befehle immer nur zu einem festen Bildschirmspeicherplatz führen. Sie können keine Bewegung über den Bildschirm erzeugen. Bei Benutzung des POKE-Befehls muß darauf geachtet werden, daß die POKE-Adresse genau stimmt. Wenn eine Adresse außerhalb des Bildschirmbereiches „ gepoked" wird, kann dies zu nichtvorauszusehenden Ergebnissen führen. Das Schwierige daran ist, daß Sie den Fehler nicht gleich bemerken und dann später überrascht sind, wo der Fehler herkommt.

Sie haben jetzt einige Grundlagen der Programmierung von Bewegungsabläufen auf dem PET kennengelernt. Es bleibt nun Ihrer Phantasie überlassen, neue eigene Programme mit Hilfe dieser kleinen Programme selbst zu entwickeln. Sollten Sie originelle Programme entwickelt haben, schreiben Sie uns, wir werden diese dann in einem zukünftigen Beitrag veröffentlichen.

Nun überarbeiten wir unser Programm für das laufende Sternchen. Anstelle des PRINT-Befehls verwenden wir nun den POKE-Befehl.

```
05 REM 5.1.7
10 REM WANDERING ASTERISK
15 REM USING POKE
20 DIM V (4) : P = 32768:N = 4 : R = 500
22 DATA + 40, –40, + 1, – 1
30 FOR I = 1 TO N
40 READ V (I)
50 NEXT I
60 R0 = R : REM SAVE OLD POSITION
70 R = P + V ( RND ( I ) * N + 1 )
80 IF R ⟨0 THEN R = R + 40
90 IF R ⟩ 1000 THEN R = R – 40
100 POKE P + R, 42 : REM NEW ASTERISK
105 POKE P + R0, 32 : REM BLANK OUT OLD ASTERISK
110 GOTO 60
```



Kurzbeschreibung des Programms

Zeile 20 erzeugt ein Array mit dem Namen V, welches die Ausgangszustände der jeweiligen Asterisk-Position enthält. P ist der Anfang des Bildschirms und R ist die Ausgangsposition des Asterisk. Zeile 22 enthält die DATA-Statements um den Asterisk in die verschiedenen Richtungen zu bewegen. Beachten Sie, daß – 40 eine Aufwärtsbewegung erzeugt (eine Reihe aufwärts). Die Zahl + 40 entspricht einer Abwärtsbewegung. Bewegung nach links ist – 1 und Bewegung nach rechts ist + 1.
Die Zeilen 30 bis 50 bilden eine Schleife, welche das Array V mit den Kontrollbefehlen lädt. In Zeile 70 wird die neue Lage des Asterisk durch eine Zufallsfunktion ausgewählt. Eine Zahl zwischen 1 und 4 wird erzeugt, welche dann zum Subskript von V wird und zu R hinzugezählt wird. Die Zeile 100 poked die neue Asterisk-Position. Die Zeilen 80 und 90 sorgen dafür, daß der Asterisk nicht außerhalb des Bildschirmes gerät.

Graphik mit dem PET

Graphik mit dem PET

Das Zeichnen von Bildern auf dem PET-Bildschirm ist sehr einfach und bringt dem Programmierer auch recht viel Freude. Man braucht dazu noch nicht einmal Programmierkenntnisse. Gemachte Fehler können auf einfachste Weise korregiert werden und das vollständige Bild kann wie ein Programm gespeichert werden. Es gibt zwar einige Einschränkungen, die vom begrenzten Zeichenvorrat herrühren, aber kompliziertere Bilder lassen sich dann durch eine Kombination von Graphik auf dem Schrim und vom Programm gezeichneten Symbolen erstellen.
(Siehe Clark Kent-Programm auf Cassette).

Der gesamte Graphik-Symbol und Zeichensatz besteht aus folgenden Zeichenfamilien:

P24 Graphik und Bewegung DM 29,--

1. Graphik

Gerade Linien	5.2.1
Vertikale Linien	5.2.2
Horizontale Linien	5.2.3
Ecken u. Schnittpunkte	5.2.4
Runde Ecken	5.2.5
Helle u. dunkle Schatten, Kartensymbole	5.2.6

Bewegung:

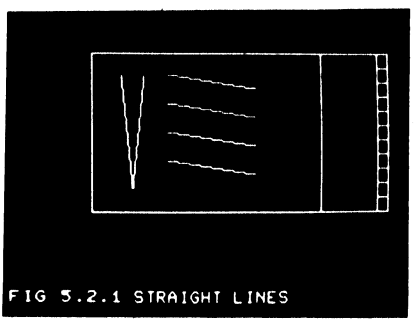
5.1.1
5.1.2
5.1.3
5.1.4
5.1.5
5.1.6
5.1.7.1
5.1.7
SUPERMANN
MONDLANDSCHAFT

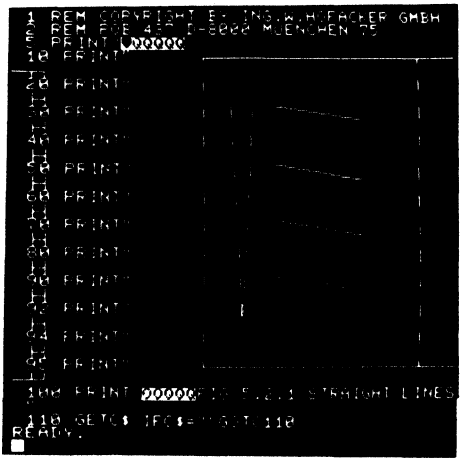
1. Vertikale Linien
2. Horizontale Linien
3. Ecken (Corners)
4. Kreuzungspunkte
5. Runde Ecken
6. Hell- und dunkel-schattierte Zeichen
7. Spielkartensymbole

Beachten Sie auch, daß jedes dieser Symbole (Revers) negativ dargestellt werden kann. Dies erhöht natürlich den gesamten Zeichenvorrat wesentlich. Auf diese Weise bewegt sich ein Symbol, welches in der untersten Zeichenhälfte ein Kästchen darstellt, beim Drücken der REVERSE-TASTE nach oben.

Taste drücken

Um ein einfaches Bild zu zeichenen, braucht man nur den Bildschirm zu löschen und das Bild mit Hilfe des Cursors zu zeichnen. Wenn das Bild fertig ist, wird es durch Schreiben von Zeilennummern und PRINT-Befehlen in den Speicher übernommen. Die Zeilennummern und PRINT—Befehle müssen an den linken Zeilenanfang geschrieben werden.





Programmlisting für Gerade Linien (Straight Lines)
Bild 5.2.1

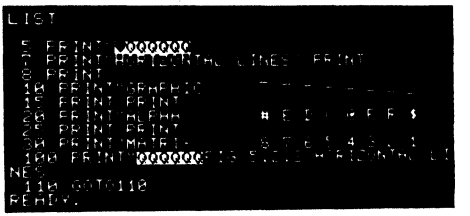


Bild 5.2.3

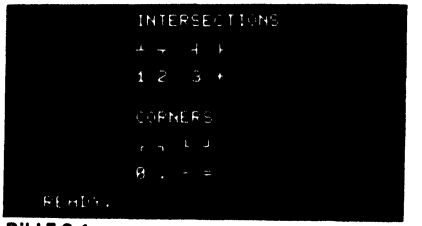
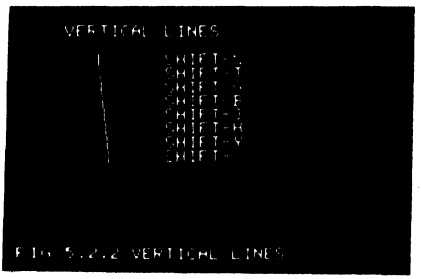


Bild 5.2.4



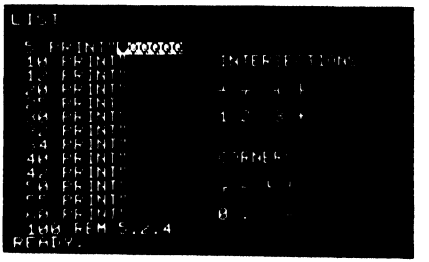


Bild 5.2.4

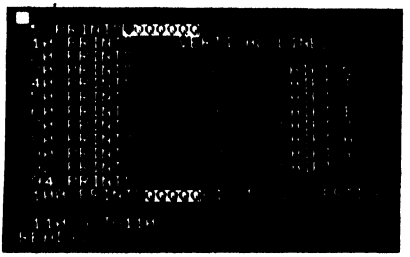


Bild 5.2.2

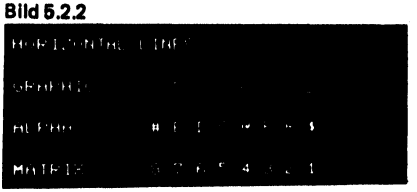


Bild 5.2.3

Runde Ecken, helle und dunkle Schattensymbole sowie die Spielkartensymbole können Sie leicht dem Tastenfeld des PET entnehmen.

Folgende Punkte müssen dabei beachtet werden:

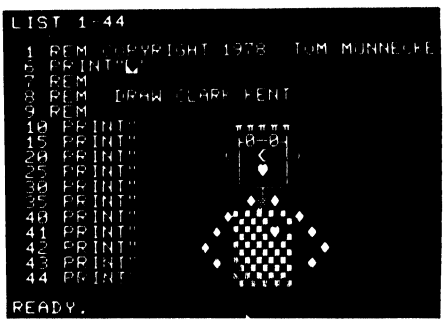
1. Kein Punkt des gezeichneten Bildes darf nach der Fertigstellung näher als 5 Zeichen von der linken Bildhälfte entfernt sein.
2. Die rechte Seite des Bildes kann ruhig bis zum rechten Bildschirmende reichen.
3. Bei der Zeilennummerierung muß darauf geachtet werden, daß alle Zeilenzahlen die gleiche Stellenzahl haben. (z. B. 001, 002, 020 etc.)

4. Wenn das Bild einmal durch Zeilen- und Befehlseingabe zu einem Programm gemacht wurde, kann das Bild auch an eine gewünschte Stelle in einem anderen Programm gebracht werden. Man braucht jetzt nur die Zeilennummern zu ändern.

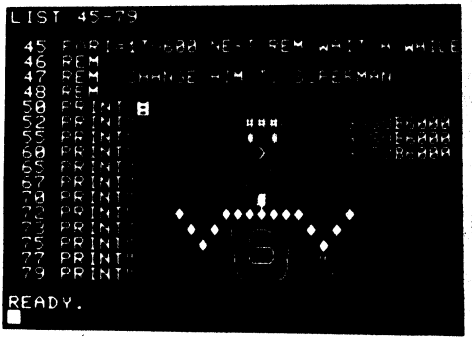
Will man z. B. ein Mondlandeprogramm erstellen, so kann man zuerst die Landschaft auf dem Bildschirm zeichnen und ins Programm übernehmen. Anschließend könnte man dann das Raumschiff auf dem Schirm zeichnen und mit unterschiedlichen Zeilennummern auch ins Programm aufnehmen.

Beispiel für Graphik und Bewegung

Superman Clark Kent
Im Listing Zeile 1 – 44 sehen Sie den ersten Teil des Programmes.
Das erste Bild, welches nach dem Programmstart auf dem Bild erscheint, wurde mit Hilfe der Cursorsteuerung auf den Bildschirm gezeichnet. Nach Beendigung der Zeichnung wurden die PRINT-Befehle am linken Bildrand angefügt. Man fährt mit dem Cursor in die erste Zeile, in der sich Zeichensymbole befinden. Dies geschieht am besten mit der HOME-Taste und der Taste zum Abwärtsbewegen des Cursors. Man gibt z. B. 10 PRINT" (Return) ein und kann sofort die zweite Zeile schreiben. Das zweite Anführungszeichen kann weggelassen werden. Zeile 6 des Programmes löscht den Bildschrim, so daß sich keine Zeichen überlagern können.
Die Zeilen 10 bis 44 enthalten das Anfangsbild unseres „Superman".



Die Programmzeilen 45 – 79 erzeugen einen neuen „SUPERMAN", welcher dem ersten überlagert wird. Dadurch entsteht der Eindruck, daß unser SUPERMAN sich verwandelt.



Die Zeile 45 ist nichts anderes als eine Verzögerungsschleife, die durch 600 Umläufe geht, damit das Bild etwas länger auf dem Bildschirm zu sehen ist.
Zeile 60 bringt den Cursor wieder an die linke obere Ecke des Bildschirms, so daß die Programmzeilen 52 – 79 den neuen Superman zeichnen können. Die GOSUB-Befehle GOSUB 6000 am rechten Ende der Zeilen 52, 55 und 60 sind Unterprogrammaufrufe, die eine Verzögerung des Hauptprogrammes bewirken und so eine Pause zwischen der Veränderung erzeugen.

Dieses Bild wurde auch durch einfaches Ändern des ersten Bildes erzeugt.(Mit Cursor)

Anschließend wurden die Zeilenzahlen verändert und das Bild ins Programm übernommen.

Achten Sie darauf, daß hier die rechten Anführungszeichen gesetzt werden müssen, da auch Leerzeichen mit aufgenommen werden sollen. Diese Leerzeichen überdecken dann das erste Bild. (s. Zeile 77 und 79)

Listing Zeile 100 – 1080

Dieser Teil des Programmes wird dazu benutzt, daß es so aussieht, als würde der Supermann sprechen. Diese Veränderungen werden aus DATA-Statements genommen. Zeile 900 ist eine weitere Verzögerungsschleife, welche den Anfang der Sprachbewegung hinauszögert. Zeile

910 startet den Ablauf, indem sie den Cursor in die linke obere Ecke bringt. Zeile 1000 setzt den DATA-Pointer, so daß ein neuer Teil mit Worten gestartet werden kann.

```
LIST 100-1080

 100 REM
 101 REM  MAKE HIM SAY THE WORDS
 102 REM
 900 FORI=1TO800 NEXT
 910 PRINT"█"
 1000 RESTORE
 1060 READ CT,N$
 1061 IF CT=0 THEN PRINT"█                    "
GOTO1100
 1062 PRINT"SQQQQQQQQQQQQQQQQ ,"          "N$
"
 1064 FORK=1TOCT
 1065 FORH=1TO90 NEXTH
 1066 PRINT"SQQQ" PRINT"           |▬|"
 1067 FORH=1TO40 NEXTH
 1068 PRINT"SQQQ"  PRINT"           |-|"
 1069 FORH=1TO90 NEXTH
 1070 NEXTK
 1080 GOTO1060

READY.
█
```

Zeile 1060 zählt die Bewegungen, die für das Wort erzeugt werden müssen. Auf diese Weise erscheint es dem Betrachter, als würde der Superman eine bestimmte Anzahl von Lippenbewegungen durchführen.

Die Zeilen 1064 bis 1070 ändern das Zeichen, welches den Mund des Superman darstellt. (offen oder geschlossen)
Zeile 1066 ist die offene Mundstellung, während Zeile 1068 den geschlossenen Mund repräsentiert. Die Verzögerungsschleifen dazwischen sorgen für eine angemessene Geschwindigkeit.

```
LIST 1100-1128

 1100 REM PUT A SMILE ON HIS FACE
 1101 REM
 1102 PRINT"SQQQ"  PRINT"          |' '|
 [illegible]
 1112 REM
 1113 REM WINK HIS EYE
 1114 REM
 1115 PRINT"SQ"  PRINT"          |• -|
 1116 REM
 1117 REM BLOW A KISS
 1118 REM
 1119 PRINT"SQQQ"  PRINT"          | ♥ |
 1120 REM
 1121 REM  MAKE HIM FLY OFF THE SCREEN
 1122 REM
 1128 FORI=1TO40 PRINT X=1↑1↑1 NEXT

READY.
```

Listing Zeile 1100 – 1128

Nachdem der Superman seine Worte ausgesprochen hat, wird er sich vom Erdboden lösen und über den oberen Bildschirmrand abfliegen.

Die Zeilen 1100 – 1128 lösen diese Aufgabe. Dies geschieht dadurch, daß die Zeichen in seinem Gesicht verändert werden.
Zeile 1102 bringt ein Lächeln in sein Gesicht. Zeile 1115 schließt das eine Auge.

Die Zeile 1128 enthält eine Schleife, welche den Bildschirm veranlaßt, das Bild nach oben durchzuschieben. Dies läßt den Eindruck entstehen, als ob unser Superman sich über den oberen Bildschirmrand hinaus entfernt.
Die kleine Verzögerung in Zeile 1128 sorgt dafür, daß dies nicht zu schnell abläuft.

Listing Zeile 2000 – 2040

Die Worte, die vom Superman gesprochen werden, sind in DATA-Statements enthalten. (Ab Zeile 2018) Die erste Zahl in jedem Statement repräsentiert die Anzahl der Silben, wobei das zweite Wort den zu sprechenden Text darstellt.

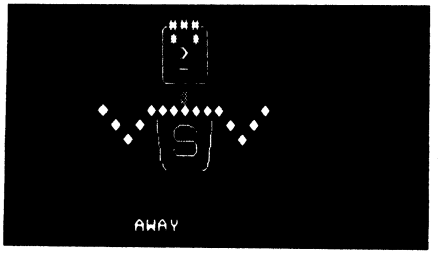
```
LIST 2000-2040

 2000 REM
 2002 REM
 2004 REM THESE ARE THE WORDS HIS SAYS
 2006 REM & THE NUMBER OF LIP MOVEMENTS
 2008 REM HE MAKES PER WORD.  EACH DATA
 2010 REM STATEMENT HAS THE COUNT,THEN
 2012 REM THE WORD TO BE DISPLAYED.
 2014 REM SPEECH ENDS WITH A COUNT OF
 2016 REM ZERO.
 2017 REM
 2018 DATA 1,"UP"
 2019 DATA 1,"UP,UP"
 2020 DATA 1,"AND"
 2030 DATA 2,"AWAY"
 2040 DATA 0 ,0

READY.
```

Listing Zeile 2900 – 3060

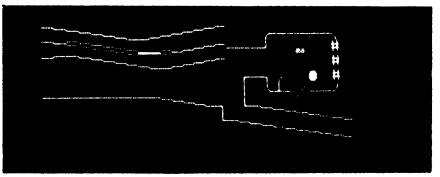
Nachdem der Superman über den oberen Bildschirmrand weggeflogen ist, muß er wieder zurückkommen, da wir ihn gerne fliegen sehen möchten. Um die Bewegungseffekte noch etwas zu verbessern, soll sein Mantel im Wind wehen.

Die Zeilen 2900 – 3060 zeichnen auf einfache Weise die Gestalt des fliegenden Supermannes. (Erstellung erfolgt einfach mit dem Cursor). Zeile 3000 löscht den Bildschirm.
Die Zeilen 4000 – 5000 zeichnen den Mantel und erzeugen den Eindruck, als ob dieser im Winde weht.
Die Zeichen in Zeile 4000 (Y$ und X$) enthalten den sich bewegenden Mantel. Zeile 4005 führt dazu, daß sich der Mantel 10 x bewegt. Zeile 4010 führt die entsprechenden Schritte über den Mantel durch. Zeile 4015 führt eine Rotationsbewegung durch, wobei 4020 und 4022 den Mantel 3 x ausdrucken.

```
LIST 2900-3060

 2900 REM
 2902 REM CLEAR THE SCREEN AND DRAW HIM
 2904 REM FLYING HORIZONTALLY. THE CAPE
 2906 REM WILL COME LATER.
 3000 PRINT"
 3001 PRINT"
 3010 PRINT"
 3020 PRINT"
 3030 PRINT"

 3040 PRINT"
 3050 PRINT"
 3060 PRINT"
READY.
```




```
LIST 4000-5000

 4000 Y$=
 4005 FOR I=1 TO 10
 4010 FOR I=1 TO 16 STEP 2
 4015 X$=RIGHT$(X$,14)+MID$(Y$,I,2)
 4020 PRINT"
 4025 PRINT"
 4030 NEXT I,J
 5000 GOTO 6 REM START AGAIN

READY.
```



Diese Programmierung erzeugt ähnliche Zustände, wie wir es von einer Filmkamera her kennen. Eine Bewegung wird durch viele kleine Veränderungen erzeugt.

Zeile 5000 schließt das Programm ab und der GOTO 6-Befehl sorgt dafür, daß das Programm wieder von vorne beginnt.

Im obigen Bild sehen Sie, wie sich der verkleidete Superman nach oben über den Bildschirmrand entfernt.

Das andere Bild zeigt Ihnen, wie der Superman mit fliegendem Mantel sich durch die Luft bewegt.

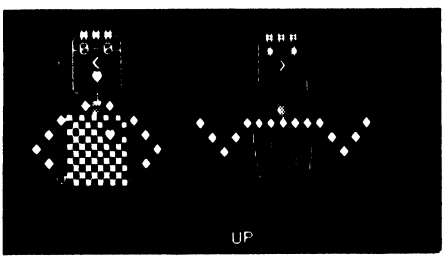
Programmierbeispiel für einen Hintergrund:

Eine Gebirgslandschaft kann auch wieder ganz einfach über den Bildschirm eingegeben werden, die PRINT-Statements davorgesetzt, ergeben ein praktisches Unterprogramm.

```
LIST 6000-6010

 6000 REM
 6002 REM DELAY LOOP
 6004 REM
 6005 FOR I=1 TO 900 NEXT
 6010 RETURN
READY.
```

Listing Zeile 6000 – 6010

Dieses Unterprogramm bringt eine Verzögerung von ca 1 s und kann von verschiedenen Programmteilen her aufgerufen werden. Eine praktische Technik für Verzögerungen.




```
RUN

READY.
```

```
LIST

 10 PRINT"
 20 PRINT"
 30 PRINT"
 40 PRINT"
 50 PRINT"
 60 PRINT"
 70 PRINT"
 80 PRINT"

 90 PRINT"
 95 PRINT"
READY.
```

Bildschirmprogrammierung und Bewegungsabläufe (Kleine Unterprogramme)

Bildschirmspiele mit Bewegung bringen dem Programmierer besonders viel Spaß. Wir wollen jetzt noch einige kleine Tips und Hilfen geben, wie Sie auf einfache Weise sich bewegende Figuren programmieren können. Wie Sie sicher bereits wissen, gibt es hier grundsätzlich zwei Methoden:

1. Erzeugung von Zeichen durch PRINT-Befehle
2. Erzeugung von Zeichen durch POKE

1. Erzeugung durch PRINT-Befehle
In einer Schleife wird ein Zeichen z.B. vom linken auf den rechten Bildschirmrand gedrückt.
Der Befehl PRINT „(Leertaste) (Zeichen) (Cursor links) " sorgt dafür, daß ein Zeichen gedruckt wird und nach rechts verschoben wird. Die zweite Schleife bringt das Zeichen wieder zurück.

Sie können das nachfolgende Programm als Unterprogramm in kleinen Spielen verwenden, bei denen Bälle hin und her laufen sollen.

```
READY.

 10 PRINT"3";
 20 FOR J= 1 TO 39
 30 PRINT" q€";
 40 NEXT
 45 END
 50 FOR J=1 TO 39
 60 PRINT" €€q€";
 70 NEXT
 80 GOTO 20
READY.
```

Die Bewegung des Balles vom oberen nach dem unteren Bildschirmrand und zurück kann über das nachfolgende Programm erfolgen.

```
READY.

 10 PRINT"3";
 20 FOR I= 1 TO 24
 30 PRINT" ←↓q←";
 40 NEXT I
 50 FOR I=1TO 24
 60 PRINT" ←↑q←";
 70 NEXT
 80 GOTO20
READY.
```

In der Diagonalen kann der „Ball" mit folgendem kleinen Unterprogramm bewegt werden.(Von oben links nach rechts unten)

```
READY.

 10 PRINT"3";
 20 FOR I= 1 TO 24
 30 PRINT"←↓q←";
 40 NEXT I
 50 FOR I=1TO 24
 60 PRINT" ←↑q←";
 70 NEXT
 80 GOTO20
 100 PRINT"3"
 110 FOR I=1 TO 24
 120 PRINT" ←↓q←";
 130 NEXT
 140 FOR I=1 TO 24
 150 PRINT" ↑q←";
 160 NEXT
 170 GOTO 110
READY.
```

Starten mit RUN 100

Wie man über den Befehl POKE eine Information in den Bildschirmspeicher geben kann, können Sie im Abschnitt Graphik und Bewegung nachlesen.

Da die Listings auf einem Drucker ohne graphische Symbole hergestellt wurden, hier noch die Bedeutungen:

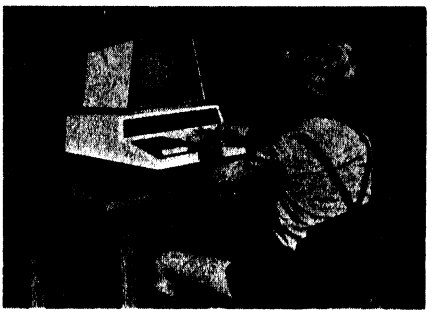
9 = Q mit Shift
← = Cursor links
→ = Cursor rechts
↑ = Cursor aufwärts
↓ = Cursor abwärts
3 = CLEAR/HOME

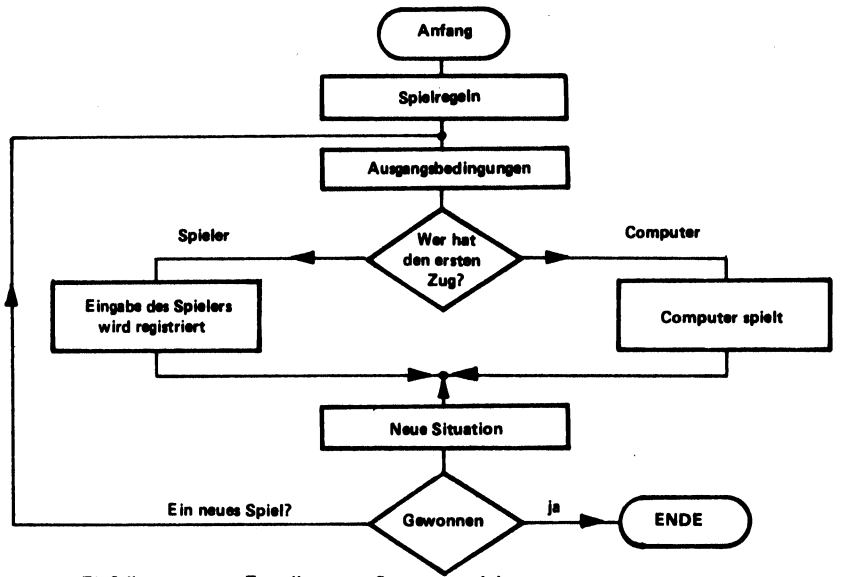
Computerspiele

Computerspiele

Wozu überhaupt Spiele? Spiele führen zu einer besseren Vorstellungskraft und konstruktiven Reaktion auf das Spiel. Der Ausgang von Spielen ist dabei meist unwichtig. Während eines Spiels übernehmen wir Risiken, ertragen eine bestimmte Ungewissheit und spielen meist weiter, ohne das Spiel besser zu kennen. Unsere Fähigkeiten, schöpferisch auf neue und unerwartete Situationen zu reagieren werden ständig gefordert. Wir entwickeln dabei unsere Entscheidungsfähigkeit und die Fähigkeiten, Probleme zu lösen. Für Kinder und Erwachsene bringen Spiele immer Freude.

Warum eigentlich Computerspiele?
Der Computer ist der ideale Spielpartner für Kinder und Erwachsene. Gerade für Kinder dürfte er ein sehr lehrreicher und geduldiger Spielpartner sein. Er steht geduldig in der Ecke und wartet darauf, daß die Kinder ihn hervorholen, wenn es ihnen paßt.





Allgemeines Flußdiagramm zur Erstellung von Computerspielen

NOTIZEN

Der Erwachsene kann mit Computerspielen sich leicht in die Programmiersprache einarbeiten. Viele haben einmal mit einfachen Computerspielen angefangen und verwenden Ihre erworbenen Programmierkenntnisse heute zur Erstellung von Buchhaltungsprogrammen.

Über das Schreiben von Spielprogrammen
Im nachfolgenden Teil wollen wir einmal versuchen, selbst kleine Spielprogramme zu entwerfen und zu programmieren. Dabei sollte man wie folgt vorgehen:

1. Spielregeln aufstellen
2. Festlegung der Anfangsbedingungen
3. Wer spielt (Computer oder Programmierer)
4. Wer macht den ersten Spielzug?
5. Wie sieht die neue Lage jetzt aus?
6. Führt der nächste Spielzug zu einem Gewinn oder Verlust?

Ein Zahlenratespiel
1. Der Computer denkt sich eine Zahl zwischen 1 und 100. Der Programmierer muß versuchen, diese Zahl herauszufinden. Nach jedem Rateversuch sagt der Computer, ob die geratene Zahl zu groß oder zu klein ist.
2. Anfangsbedingungen: Computer „denkt" sich eine Zahl und fragt.
3. Computer generiert die Zahl
4. Der Programmierer antwortet zuerst
5. Neue Situation je nachdem, ob zu groß oder zu klein.
6. Ist die Zahl erraten ? (Ja / Nein)

Beispiel 1
Listing eines sehr einfachen Zahlenratespieles

```
READY.

 100 REM ING W HOFACKER GMBH
 110 REM ZAHLENRATESPIEL
 120 PRINT"SPIELREGELN:
 130 PRINT"ICH DENKE MIR EINE ZAHL
 140 PRINT"ZWISCHEN 1 UN 100 (NUR
 150 PRINT"GANZE ZAHLEN BITTE!) DU
 160 PRINT"SOLLST MEINE GEDACHTE
 170 PRINT"ZAHL RATEN. ICH SAGE DIR
 180 PRINT"DANN OB DU ZU GROSS ODER
 190 PRINT"ODER ZU KLEIN GERATEN HAST
 200 REM DER COMPUTER DENKT SICH JETZT
 210 REM EINE ZAHL
 220 LET X=INT(100*RND(1))+1
 230 PRINT
 240 PRINT
 250 PRINT"ICH HABE EINE ZAHL DU KANNST
 260 PRINT"JETZT RATEN-VIEL GLUECK!!!!!
 300 REM DER PROGRAMMIERER KANN JETZT
 310 REM ANFANGEN ZU RATEN
 320 PRINT
 330 PRINT"WAS GLAUBST DU HABE ICH MIR
 340 PRINT"GEDACHT";
 350 INPUT A
 360 IF A=X THEN 500
 370 IF A>XTHEN400
 380 PRINT"DIESE ZAHL IST ZU KLEIN"
 390 GOTO 300
 400 PRINT"DIESE ZAHL IST ZU GROSS"
 410 GOTO 300
 500 REM DER PROGRAMMIERER HAT DIE ZAHL
 510 REM ERRATEN
 520 PRINT
 530 PRINT"RICHTIG DU HAST DIE ZAHL GE-
 540 PRINT"RATEN-AUF EIN NEUES SPIEL!!
 550 PRINT
 560 GOTO 200
READY.
```

Beispiel 2:
Listing eines Ratespiels, bei dem der Computer Ihre gedachte Zahl erraten kann.

```
READY.

 100 REM ING W HOFACKER GMBH
 110 REM DER COMPUTER ERRAET NUN DIE
 120 REM VON IHNEN GEDACHTE ZAHL
 130 PRINT"BITTE DENKEN SIE SICH EINE
 140 PRINT"GANZE ZAHL ZWISCHEN 1 UND
 150 PRINT"316.SCHREIBEN SIE DIESE ZAHL
 160 PRINT"AUF EIN STUECK PAPIER UND
 170 PRINT"TEILEN SIE DIE ZAHL DURCH 5
 180 PRINT
 190 PRINT
 200 PRINT"GEBEN SIE MIR DEN UEBERTRAG
 210 PRINT"ODER AUCH REST EIN!";
 230 INPUT X5
 240 PRINT
 250 PRINT"TEILEN SIE NUN DIE GEDACHTE
 260 PRINT"ZAHL DURCH 7 UND GEBEN SIE
 270 PRINT"MIR WIEDER DEN REST EIN!";
 280 INPUTX7
 290 PRINT
 300 PRINT"NUN TEILEN SIE GEDACHTE ZAHL
 310 PRINT"DURCH 9 UND GEBEN SIE MIR DEN
 320 PRINT"VERBLEIBENEN REST EIN !";
 330 INPUT X9
 340 PRINT
 350 REM BERECHNUNG DER ZAHL
 360 Y=126*X5+225*X7+280*X9
 370 W=Y-INT(Y/315)*315
 380 PRINT
 390 PRINT"ICH FREUE MICH IHNEN SAGEN
 400 PRINT"ZU KOENNEN ,DASS DIE VON
 410 PRINT"IHNEN GEDACHTE ZAHL";W
 420 PRINT" WAR"
 430 END
READY.
```

Beispiel 3
Casino-Simulation

```
READY.

 10 REM ING W HOFACKER GMBH
 20 PRINT"EIN SPIELER GEHT MIT 1000 DM
 30 PRINT"INS CASINO UND SETZT JEDE
 40 PRINT"MINUTE DM 1.- AUF ROT.JEDER
 50 PRINT"RUN DURCHGANG DIESES PROGRAMMES
 60 PRINT"SIMULIERT EINE GESPIELTE
 70 PRINT"STUNDE UND GIBT IHNEN AM
 80 PRINT"SCHLUSS DAS VERBLEIBENDE
 90 PRINT"KAPITAL"
 91 PRINT"WARTEN SIE BITTE - ES  DAUERT
 92 PRINT"EINE WEILE BIS UNSER SPIELER
 93 PRINT"WIEDER KOMMT !!!!!!!!!!!!!!!
 100 REM ING W HOFACKER GMBH
 110 REM ROULETTE SIMULATION
 120 X=1000
 130 FOR Y=1 TO 60
 140 LET X=X-1
 150 RESTORE
 160 REM SIMULATION DER TELLERDREHUNG
 170 LET W=INT(37*RND(1))
 175 IFW=0 THEN 250
 180 FOR J=1 TO 18
 190 READ A
 200 IFW=A THEN 230
 210 NEXT J
 220 GOTO 250
 230 REM AUS BZW.EINZAHLUNG
 240 LET X=X+2
 250 NEXT Y
 255 PRINT"DER SPIELER HAT NOCH DM";X
 260 PRINT"ER GEHT JETZT WIEDER INS CASINO -"
 261 PRINT"WARTEN WIR BIS ER WIEDER KOMMT!!!!"
 262 PRINT
 263 PRINT
 264 PRINT
 265 GOTO 91
 270 DATA 1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36
 280 END
READY.
```

Biorythmus für PET

```
READY.

 1 REM  COPYRIGHT  1978  TOM  MUNNECKE
 2 PRINT"3";TAB(8)"&&&&&&&&&&&&":PRINTTAB
       (8)"&BIORHYTHMS&"
 3 PRINTTAB(8)"&&&&&&&&&&&&"
 20 M$="DAY "
 100 DIMX$(31)
 105 DIMM$(12)
 110 M$(1)="JAN"
 111 M$(2)="FEB"
 112 M$(3)="MAR"
 113 M$(4)="APR"
 114 M$(5)="MAY"
 115 M$(6)="JUN"
 116 M$(7)="JUL"
 117 M$(8)="AUG"
 118 M$(9)="SEP"
 119 M$(10)="OCT"
 120 M$(11)="NOV"
 121 M$(12)="DEC"
 200 P9=6.283185
 300 P1=23
 310 P2=28
 320 P3=33
 400 D1=P9/P1
 410 D2=P9/P2
 420 D3=P9/P3
 430 DATA31,28,31,30,31,30,31,31,30,31,30
 431 DATA31
 450 INPUT"ENTER NAME";N$
 460 PRINT:PRINT"ENTER 'M' TO PRINT A
    MONTH'S DATA OR    'Y' FOR A YEAR'S";
  461 INPUTX$
  500 N1=0
  510 PRINT"BIRTH MONTH,DAY,YEAR";
```

```
511 INPUTB1,B2,B3
520 IFB3<100THENB3=B3+1900
530 IFB1>2THEN600
540 IFB1=2ANDB2=29THEN600
550 Q1=INT(B3-1900)/4
551 Q2=(B3-1900)/4
552 IFQ1<>Q2THEN600
560 N1=1
600 PRINT"ENTER START MONTH, YEAR";
601 INPUTC1,C3
608 IFC3<100THENC3=C3+1900
610 IFB3>=C3THEN8100
620 FORJ=1TOB1:READX:NEXT
650 N1=N1+X-B2
660 IFB1=12THEN702
670 FORJ=B1+1TO12:READX:N1=N1+X:NEXT
702 REM
705 IFC3-B3<2THEN750
710 FORJ=B3-1899TOC3-1901
720 IFINT(J/4)=J/4THENN1=N1+1
730 N1=N1+365
740 NEXTJ
750 RESTORE
760 IFC1=1THEN810
770 FORJ=1TOC1-1:READX:N1=N1+X:NEXT
810 Q3=INT((C3-1900)/4)
811 Q4=(C3/4)
812 IFQ3<>Q4THEN900
820 IFC1>2THENN1=N1+1
900 I1=N1
910 I2=N2
920 I3=N3
930 READX
955 PRINT""
970 PRINT"3";"BIOCHART `` ";N$
975 PRINT"BIRTHDATE `` ";B2;M$(B1);"";B3
976 PRINT
977 PRINT"C=COGNITIVE"
978 PRINT"P=PHYSICAL"
979 PRINT"S=SENSITIVITY"
985 PRINT""
995 L=0
997 GOSUB2000
```

```
998 D=0
1000 L=L+1
1100 FORZZ=1TO31:X$(ZZ)=" ":NEXT
1130 X$(16)="}"
1200 Y1=INT(15*SIN((L+I1)*D1)+16.5)
1210 Y2=INT(15*SIN((L+I2)*D2)+16.5)
1220 Y3=INT(15*SIN((L+I3)*D3)+16.5)
1250 X$(Y1)="P2"
1260 X$(Y2)="S2"
1270 X$(Y3)="C2"
1280 IFY1=Y2THENX$(Y1)="q"
1290 IFY1=Y3THENX$(Y1)="q"
1300 IFY2=Y3THENX$(Y3)="q"
1350 D=D+1
1360 IFD<X+1THEN1400
1365 S1=S1+1
1370 IFS1=12THEN9999
1375 C1=C1+1
1377 IFC1>12THEN1390
1379 READX
1380 GOSUB2000
1385 GOTO1400
1390 RESTORE
1393 C1=1
1394 C3=C3+1
1395 GOTO1379
1400 PRINTM$(C1);""DTAB(8);
1450 FORJ=1TO31:PRINTX$(J);:NEXT
1500 PRINT
1600 GOTO1000
2000 REM
2002 IFX9=1GOTO9999
2004 IFX$="M"THENX9=1
2010 FORJ=1TO5
2020 PRINT
2030 NEXTJ
2040 PRINT"BIOCHART FOR ";M$(C1);" ";C3
2100 PRINTTAB(5);N$
2170 PRINTTAB(10);"(-)";TAB(34);"(+)"
2180 PRINT
2190 D=1
2200 RETURN
8000 PRINT
```

```
 8010 PRINT"YEAR MUST BE 1900 OR LATER"
 8020 GOTO510
 8100 PRINT
 8110 PRINT"START YEAR MUST BE GREATER"
 8111 PRINT"THAN BIRTH YEAR"
 8120 GOTO600
 9999 END
READY.
```

Wer Interesse hat, kann auch die fertige Cassette mit zwei weiteren Spielen (Hangman und Pferderennen) vom Hofacker-Verlag unter der Bestellnummer P5 für DM 49,-- beziehen.

Da das Listing auf einem Drucker ausgedruckt wurde, der die Graphiksymbole des PETs nicht ausdruckt, muß man bei den PRINT-Befehlen etwas aufpassen.

Die folgenden Zeichen des Druckers entsprechen den angegebenen Graphik-Zeichen des PETs:

3 = CLEAR
& = Graphik-Zeichen Shift &
2 = OFF

REVERSE und Cursor abwärts werden vom Drucker nicht erkannt. Folgende Zeilen müssen noch entsprechend dem Graphik-Mode geändert werden:

Zeile 955 PRINT " (5 x Cursor down)"
Zeile 970 PRINT "(CLEAR)"; „BIOCHART – ";N$
Zeile 975 PRINT „BIRTHDATE–" ; B2;M$ (B1); " ";B3
Zeile 1130 X$ (16) = „(SHIFT B)"
Zeile 985 wie 955
Zeile 1250 X $ (Y1) = „ (Reverse) P (OFF)"
Zeile 1260 X $ (Y2) = „ (Reverse) S (OFF)"
Zeile 1270 X $ (Y3) = „ (Reverse) C (OFF)"

Spielprogramm:

Irrgartenzeichner zeichnet automatisch einen Irrgarten. Länge und Breite können eingegeben werden.

```
READY.

 1 REM TIM HUNTER
 50 PRINT"3 I WILL DRAW A MAZE OF ANY SIZE UP
 55 PRINT"TO WIDTH OF 13 AND LENGTH OF 11.
 100 INPUT"ENTER THE WIDTH AND LENGTH";H,V
 101 IFH>13THENPRINT"TO WIDE":GOTO100
 102 IFV>11THENPRINT"TO LONG":GOTO100
 103 IFH<>1ANDV<>1THEN110
 104 PRINT"MEANINGLESS DIMENSIONS ":GOTO100
 110 DIMW%(H,V):DIMV%(H,V)
 120 PRINT""
 160 Q=0:Z=0:X=INT(RND(1)*H+1)
 165 FORI=1TOH:IFI=XTHEN173
 171 PRINT".--";:GOTO180
 173 PRINT".ST";
 180 NEXTI:PRINT".":C=1:W%(X,1)=C:C=C+1
 200 R=X:S=1:GOTO260
 210 IFR<>HTHEN240
 215 IFS<>VTHEN230
 220 R=1:S=1:GOTO250
 230 R=1:S=S+1:GOTO250
 240 R=R+1
 250 IFW%(R,S)=0THEN210
 260 IFR-1=0THEN530
 265 IFW%(R-1,S)<>0THEN530
 270 IFS-1=0THEN390
 280 IFW%(R,S-1)<>0THEN390
 290 IFR=HTHEN330
 300 IFW%(R+1,S)<>0THEN330
 310 X=INT(RND(1)*3+1)
 320 ONXGOTO790,820,860
 330 IFS<>VTHEN340
 334 IFZ=1THEN370
 338 Q=1:GOTO350
 340 IFW%(R,S+1)<>0THEN370
 350 X=INT(RND(1)*3+1)
 360 ONXGOTO790,820,910
```

```
370 X=INT(RND(1)*2+1)
380 ONXGOTO790,820
390 IFR=HTHEN470
400 IFW%(R+1,S)<>0THEN470
405 IFS<>VTHEN420
410 IFZ=1THEN450
415 Q=1:GOTO430
420 IFW%(R,S+1)<>0THEN450
430 X=INT(RND(1)*3+1)
440 ONXGOTO790,860,910
450 X=INT(RND(1)*2+1)
460 ONXGOTO790,860
470 IFS<>VTHEN490
480 IFZ=1THEN520
485 Q=1:GOTO500
490 IFW%(R,S+1)<>0THEN520
500 X=INT(RND(1)*2+1)
510 ONXGOTO790,910
520 GOTO790
530 IFS-1=0THEN670
540 IFW%(R,S-1)<>0THEN670
545 IFR=HTHEN610
547 IFW%(R+1,S)<>0THEN610
550 IFS<>VTHEN560
552 IFZ=1THEN590
554 Q=1:GOTO570
560 IFW%(R,S+1)<>0THEN590
570 X=INT(RND(1)*3+1)
580 ONXGOTO820,860,910
590 X=INT(RND(1)*2+1)
600 ONXGOTO820,860
610 IFS<>VTHEN630
620 IFZ=1THEN660
625 Q=1:GOTO640
630 IFW%(R,S+1)<>0THEN660
640 X=INT(RND(1)*2+1)
650 ONXGOTO820,910
660 GOTO820
670 IFR=HTHEN740
680 IFW%(R+1,S)<>0THEN740
685 IFS<>VTHEN700
690 IFZ=1THEN730
695 Q=1:GOTO830
```

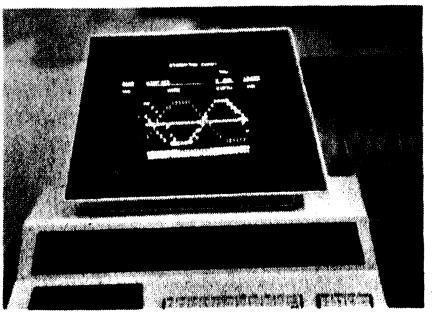
```
700 IFW%(R,S+1)<>0THEN730
710 X=INT(RND(1)*2+1)
720 ONXGOTO860,910
730 GOTO860
740 IFS<>VTHEN760
750 IFZ=1THEN780
755 Q=1:GOTO770
760 IFW%(R,S+1)<>0THEN780
770 GOTO910
780 GOTO1000
790 W%(R-1,S)=C
800 C=C+1:V%(R-1,S)=2:R=R-1
810 IFC=H*V+1THEN1010
815 Q=0:GOTO260
820 W%(R,S-1)=C
830 C=C+1
840 V%(R,S-1)=1:S=S-1:IFC=H*V+1THEN1010
850 Q=0:GOTO260
860 W%(R+1,S)=C
870 C=C+1:IFV%(R,S)=0THEN880
875 V%(R,S)=3:GOTO890
880 V%(R,S)=2
890 R=R+1
900 IFC=H*V+1THEN1010
905 GOTO530
910 IFQ=1THEN960
920 W%(R,S+1)=C:C=C+1:IFV%(R,S)=0THEN940
930 V%(R,S)=3:GOTO950
940 V%(R,S)=1
950 S=S+1:IFC=H*V+1THEN1010
955 GOTO260
960 Z=1
970 IFV%(R,S)=0THEN980
975 V%(R,S)=3:Q=0:GOTO1000
980 V%(R,S)=1:Q=0:R=1:S=1:GOTO250
1000 GOTO210
1010 FORJ=1TOV:IFH=13THENPRINT"1";
1011 PRINT"I";
1012 FORI=1TOH:IFV%(I,J)<2THEN1030
1020 PRINT"    ";:GOTO1040
1030 PRINT"  I";
1040 NEXTI
```

```
 1041 IFH=13THENPRINT"1":GOTO1043
 1042 PRINT
 1043 FORI=1TOH
 1045 IFV%(I,J)=0THEN1060
 1050 IFV%(I,J)=2THEN1060
 1051 PRINT":  ";
 1052 GOTO1070
 1060 PRINT":--";
 1070 NEXTI
 1071 PRINT"."
 1072 NEXTJ
 1075 GETC$:IFC$=""THEN1075
 1080 PRINT"3":CLR:GOTO100
READY.
```

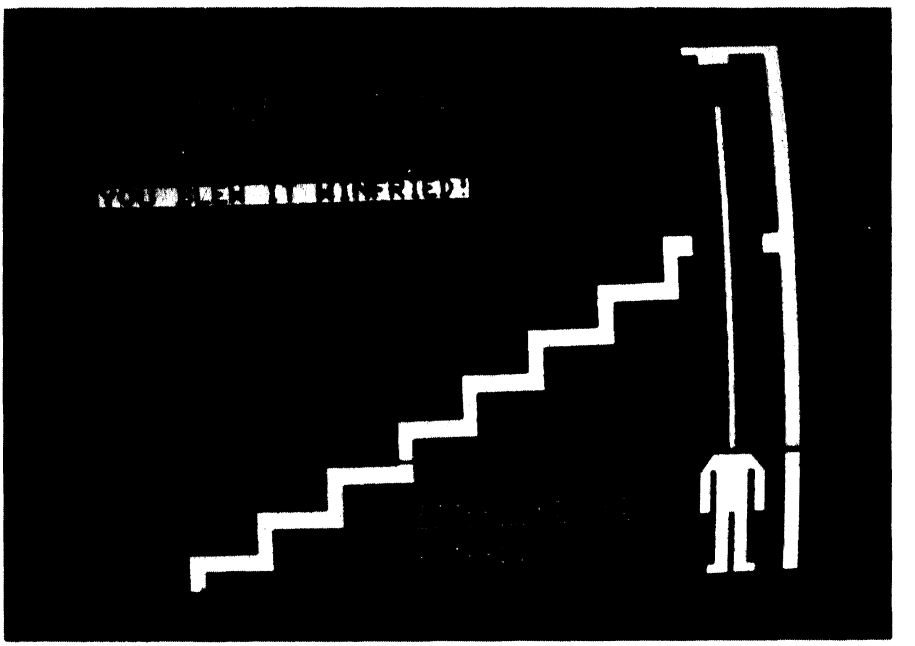
Einige Beispiele von Spielprogrammen

Nachfolgend wollen wir Ihnen einige Fotos zeigen, denen Sie die phantastischen Graphikeigenschaften des PET entnehmen können.

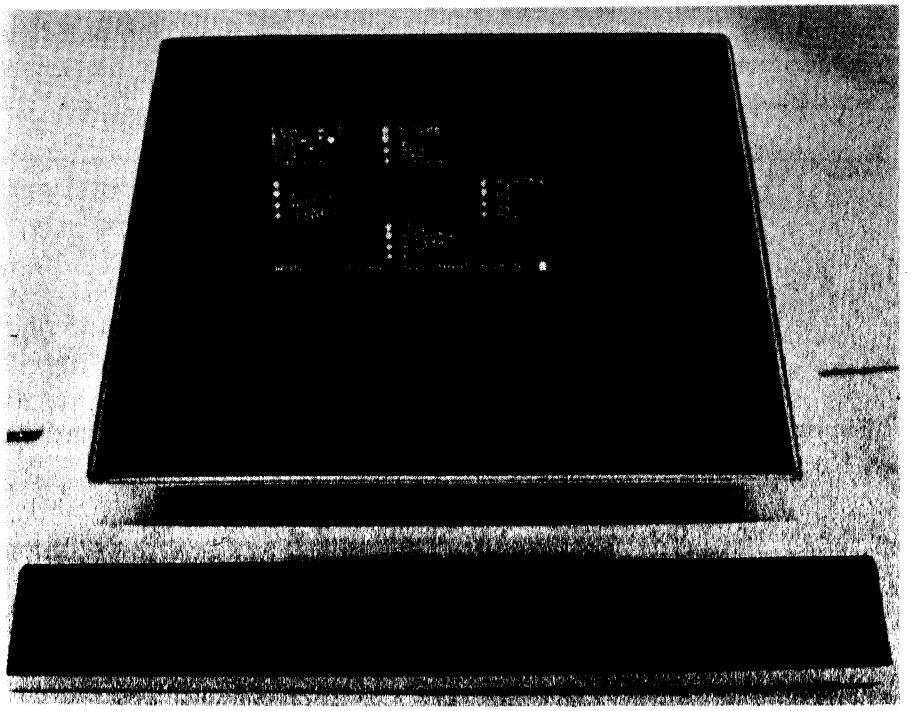
1. Ein Biorythmusprogramm, welches Ihnen den gesamten Ablauf in einem Bildschirminhalt anzeigt.



2. Das Spiel Hangman — Es müssen Worte erraten werden. Wenn Sie das Wort nicht erraten, wird ein Männchen an einen Galgen gehängt.



3. Bridge gegen den PET — Mit diesem Programm können Sie gegen den PET Bridge spielen.



Zeichnen von mathematischen Funktionen auf dem PET 2001

Auf dem PET 2001 können auch mathematische Funktionen dargestellt werden. Allerdings beträgt Auflösung gemäß dem Bildschirmraster in der x-Achse 40 Punkte und in der y-Achse 25 Punkte. Eine Funktion kann somit in ganzen Zahlen zwischen y = 1 bis 24 und x = 1 bis 40 gezeichnet werden. Man muß die Funktion auf diese Zahlen normieren. Z. B. es soll y = sin (x) von 0 bis 2 π gezeichnet werden.

Dabei ist

$$y = \sin(0) = 0$$
$$y = \sin\frac{(\pi)}{2} = 1$$
$$y\ E\ \sin\frac{(3\pi)}{2} = -1$$
$$y = \sin(2\pi) = 0$$

Einmal muß jetzt 2π auf das x-Koordinate und -1 bis $+1$ auf die y-Koordinate normiert werden, d. h.

für x-Achse	$0.\pi$	= 0. Spalte;	$2.\pi$	=	39. Spalte
für y-Achse	-1	= 23.Reihe;	-1	=	0. Reihe

Das ergibt dann folgendes Zeichenprogramm:

I. 100

```
90    PRINT „CLR"
100   FOR S = 0 TO 39
200   Z = 13 – (Int (12*SIN (s*2*π / 39 )))
300   S = S + 1
400   GOSUB 5000
500   NEXT : END
5000   POKE  32727 + 40*Z + S, 42
5100  RETURN
```

Ein Kreis kann mit nachstehender Befehlsreihe auf dem PET gekennzeichnet werden.

II.

```
90    PRINT „CLR"
100   INPUT „CLR Q Q DURCHMESSER X-Achse (12 FUER
      RUND):"; A 1
200   INPUT „Q DURCHMESSER Y-ACHSE ( 9 FUER RUND ):"
      ; A 2
300   PRINT „ Q-LAGE DES MITTELPUNKTES"
400   INPUT „Q- X-Achse (20 FUER MITTE): ";A 3
500   INPUT „ Q Y-Achse (12 FUER MITTE): ";A 4
550   INPUT „ Q Q Q AUSZUDRUCKENDES ZEICHEN:"; A $
580   PRINT „CLR"
600   FOR F = 0 TO 2*π STEP 0,1*π
700   Z + INT ( A 2*COS (F)) + A 4
800   S = INT ( A 1*sin (F)) + A 3
900   GOSUB 5000
1000  NEXT F : END
5000  POKE 32727 + 40*Z + S, ASC (A$)
5100  RETURN
```

Computer-Musik

Computer Musik

Das generieren von Tönen mit Hilfe eines Computers gehört zu den interessantesten Tätigkeiten des Programmierers. Wollte man sich über dieses Gebiet richtig auslassen, würde man weit den Rahmen eines einzigen Buches sprengen. Dieser Artikel soll Sie mit praktischen Grundlagen vertraut machen, mit deren Hilfe Sie dann auch selbst kleine Experimente durchführen können. Programme mit Toneffekten sind noch attraktiver und machen noch mehr Freude.

Um einen möglichst großen Kreis anzusprechen, haben wir den PET als Computersystem verwendet. Sie könnten jedoch die Programme auch leicht für einen anderen Computer auf 6502 Basis (KIM-SYM-ALPHA etc.) umändern. Wichtig ist, daß auch der leistungsfähige Interfacebaustein 6522 vorhanden ist.

Bei all unseren Musikexperimenten in diesem Artikel haben wir es mit Rechteckschwingungen im Tonbereich zu tun. Weiterhin kann immer nur ein Ton zur gleichen Zeit erzeugt werden. Trotzdem lassen sich über den PET USER PORT sehr interessante Musikeffekte generieren. Es besteht beim PET die Möglichkeit, in BASIC oder auch in Maschinensprache Töne zu erzeugen. Will man jedoch Töne in einem großen Frequenzbereich (z. B. 100–500 Hz) erzeugen, muß man auf jeden Fall in Maschinensprache programmieren. Die Ausführungszeit für einen Befehl ist in Maschinensprache wesentlich kürzer (ca. 2–10 μs.)

Zwei verschiedene Möglichkeiten der Musikerzeugung

Wir wollen nur zwei einfache, verschiedene Möglichkeiten zur Musikerzeugung mit dem PET besprechen.

1. Musikerzeugung über den I/O-Port
2. Musikerzeugung über das Schieberegister im 6522 VIA

1. Musikerzeugung über den I/O-Port

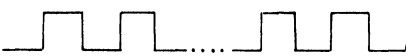
Bevor wir mit unserer Programmierung beginnen, benötigen wir einen „Edge Connector" (Stecker) für den USER Port. Sie können hier einen CINCH-Stecker, Typ 50-24EE-30/251-12-90-160 verwenden. Der Stecker hat die Bezeichnungen 1-12 auf der oberen Reihe und A B C D E F G H J K L M N auf der unteren Reihe. Der Anschluß N unten rechts ist der Masseanschluß. Zwischen diesem Anschluß und einem, der noch festzulegen ist, liegt unser Audiosignal.

Aus welchem anderen Anschluß nun das Signal kommt, hängt davon ab, was wir programmieren.

Erzeugen wir z. B. unsere Töne über den I/O-Port, so können wir als Ausgang einen der Portanschlüsse C–L festlegen.

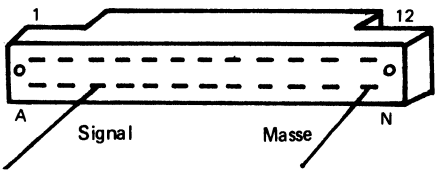
```
READY.

 5 REM MUSIK DEMO 1
 20 POKE59459,1
 30 FOR I=1TO100
 40 POKE59457,I
 50 NEXT I
 60 END
READY.
```

Dieses kleine Programm würde z. B. zwischen Anschluß C und Masse ein Signal erzeugen. POKE 59459,1 schaltet im Datenrichtungsregister den Anschluß PA0 = C als Ausgang. POKE 59457,I gibt den Registerinhalt I aus, welcher durch die FOR NEXT-Schleife ständig erhöht wird.



100 x

Ein Anschluß müßte am Stecker wie folgt aussehen:

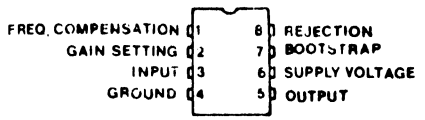


Man kann jetzt hier einen Verstärker (Stereo-Anlage oder Kofferradio) anschließen und die Ausgangstöne hörbar machen.

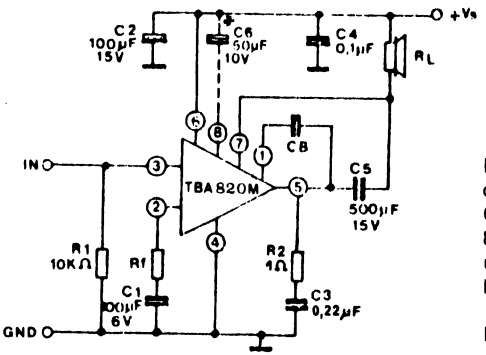
Wer sehr viel mit Computermusik experimentieren möchte, sollte sich ein kleines Plastikgehäuse mit Lautsprecher und eingebautem NF-Verstärker anfertigen. Die Versorgungsspannung kann dann entweder dem 2. Cassetten Interface Port (+ 5 V) oder einer zusätzlichen Batterie entnommen werden.

Eine sehr einfache und praktische Verstärkerschaltung mit dem TBA820M wollen wir Ihnen nachfolgend zeigen. Sie arbeitet an nur einer Betriebsspannung zwischen 4,5 V und 16 V und benötigt ein Minimum an Bauteilen.

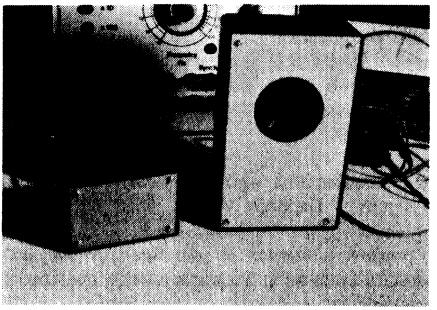
Anschlüsse



Anwendungsbeispiel



CB = 1 nF



Den Verstärker kann man auf eine NF-Buchse führen. Dann läßt sich leicht von jedem gewünschten Ausgang des PETs eine Verbindung zum Verstärker herstellen.

Wir haben nun schon unser erstes kleines Geräuschprogramm ausprobiert. Wie schon erwähnt, sollte man jedoch bei der Erzeugung von Tönen in Maschinensprache programmieren. Da dies jedoch relativ kompliziert ist, wollen wir eine Routine schreiben, die uns ein Maschinenprogramm erzeugt. Diese Routine ist in Wirklichkeit auch ein Programm in BASIC, arbeitet aber wie ein Maschinenunterprogramm.

```
READY.

 1 REM MUSIK DEMO 2
 2 REM ING W HOFACKER GMBH
 5 DATA 120,169,1,141,67,232,162,155
 10 DATA 160,9,169,4,233,1,208,252,136
 20 DATA 208,247,238,65,232,202,208,239
 30 DATA 88,96
 40 POKE59457,I
 50 FOR I=1 TO 27
 60 READ A
 70 POKE(831+I),A
 80 NEXT I
 100 END
READY.
```

Dieses Programm setzt über den POKE-Befehl die Werte aus den DATA-Statements in den 2. Cassettenspeicher des PET. Die Anfangsadresse 831 dezimal liegt in diesem Cassettenspeicher und das Programm ist dort gegen Überschreibungen aus dem BASIC geschützt.

Natürlich darf das 2. Cassettengerät jetzt nicht verwendet werden.
Diese kleine Routine erzeugt beim Aufruf einen TON, dessen Dauer und Frequenz wir in BASIC

festlegen können. Die Tondauer wird durch den Befehl POKE 839,X festgelegt. Die Zahl X kann hierbei zwischen 1 und 255 liegen. Sie entspricht den halben Zykluslängen des TONES, der erzeugt wird.

Die Tonfrequenz wird durch zwei Konstanten festgelegt. Dadurch können über 65000 verschiedene Töne erzeugt werden. Diese beiden Konstanten werden in die Speicherzelle 841 und 843 gebracht. Sie können beide zwischen 1 und 255 liegen.

843 = Tonleiter
841 = Frequenz

Wenn die Konstanten richtig gesetzt sind, kann das Maschinenunterprogramm durch SYS(832) aufgerufen werden.

Beispiel:

```
200 REM MUSIK DEMO 3
205 POKE839,255
210 POKE841,100
215 POKE843,1
220 SYS(832)
240 GOTO200
```

Nun können wir bereits anfangen zu experimentieren. Nach RUN 1 oder einfach RUN wird der Musikgenerator geladen. Nach RUN 200 wird ein Musikgenerator geladen. Nach RUN 200 wird ein TON entsprechend unserem Musik DEMO 3 erzeugt.

Versuchen wir es jetzt einmal mit einer anderen Frequenz, indem wir Zeile 210 in POKE 841,40 ändern.
Achten Sie darauf, daß jetzt auch die Tondauer kürzer geworden ist. Es wurde ja nicht die Konstante für Tondauer mitgeändert. Die Anzahl der Halbwellen ist zwar gleich geblieben, aber jede einzelne Halbwelle ist dafür kürzer geworden. Um interessante Töne zu erzeugen, kann man jetzt die verschiedensten Änderungen vornehmen.

Ändern wir einmal Zeile: 240 GOTO 200
und geben RUN 200 ein.

Komplexe Töne lassen sich nun grundsätzlich auf zwei verschiedene Arten erzeugen.

1. Die Konstanten werden aus einer Tabelle gelesen
2. Die Konstanten ergeben sich aus einer arithmetischen Operation oder einer BASIC-Funktion.

Beispiel zu 1.:
Lesen der Konstanten aus Tabelle :

```
300 REM MUSIK DEMO 4
310 POKE843,1
320 POKE839,150
330 FOR I=1TO9
340 READ B
350 POKE841,B
360 SYS(832)
370 NEXT I
380 DATA 10,20,30,40,50,60,70,80,90
390 END
```

Testen Sie dieses Programm mit RUN 200 und RUN 300. Jetzt können Sie beliebige Melodien erzeugen, indem Sie die gewünschten Konstanten in die DATA-Statements setzen. Sie können dann neben den Frequenzkonstanten auch noch die Konstanten für die Tondauer in DATA-Statements geben und auslesen.

Beispiel zu 2.:

```
400 REM MUSIKDEMO 5
410 POKE843,1
420 POKE839,200
430 FOR J=1 TO 15
440 POKE841,J*10
450 SYS(832)
460 NEXT J
470 END
```

Mit RUN 400 können Sie dieses kleine Beispielprogramm starten. Es liefert eine bestimmte Tonfolge entsprechend der arithmetischen Operation. Sie können dies jetzt ausbauen und weitere eigene Programme schreiben.

Natürlich brauchen Sie den Musikgenerator in den 2. Cassettenspeicher nur einmal zu laden. Er bleibt dort, bis das Gerät ausgeschaltet wird.

Ein weiteres Beispiel zeigt uns einen Ton-Zufallsgenerator. Sie können es mit RUN 500 starten.

```
 500 REM MUSIK DEMO 6
 510 POKE843,1
 520 POKE839,255
 530 I=INT(RND(2)*255)
 540 POKE841,I
 550 SYS(832)
 560 GOTO 530
READY.
```

Jetzt haben wir gesehen, wie man auf einfachste Weise kleine Musikeffekte mit dem PET erzeugen kann. Sie können jetzt diese kleinen Unterroutinen in Ihre Programme einbauen und mit GOTO (Zeilennummer) etc. aufrufen.

Erzeugung von Tönen über das Schieberegister im 6522 VIA

Hier muß der Anschluß des Verstärkers etc. mit der Signalseite an Pin M des USER PORTS angeschlossen werden. (Pin M auf der Steckerleiste entspricht CB2)

In diesem Falle wird das Schieberegister durch POKE 59467,16 in einen freilaufenden (oszillierenden) Zustand versetzt.

Achtung: Wenn dieser Befehl einmal gegeben wurde, ist das Cassetteninterface nicht mehr in Betrieb. Sie können also keine Programme mehr lesen oder schreiben. Wenn Sie also wieder ein Programm laden wollen, müssen Sie POKE 59467,0 eingeben.

Nach POKE 59467,16 geben wir dann
POKE 59466,15 ein. Dies legt den Inhalt des Schieberegisters fest,welcher dann herherumgeschoben wird. Er sollte für ein Rechtecksingal 15, 51 oder 85 sein.

POKE 59464,F Wobei F wie folgt errechnet werden kann:

$$F = \frac{500\,000}{(X + 2) * S}$$

S = 8 bringt 15 ins Schieberegister
S = 4 bringt 51 ins Schieberegister
S = 2 bringt 85 ins Schieberegister
X = ist eine ganze Zahl zwischen 0 und 255

Nachfolgend einige Beispiele:
(Bitte Verstärker an CB2 anschließen)

```
READY.

 10 REM MUSIK DEMO-7
 20 REM ING W HOFACKER GMBH
 30 REM MUSIK UEBER DEN 6522 VIA
 40 POKE59467,16
 50 POKE59466,15
 60 POKE59464,25
 70 POKE59467,0
 80 END
READY.
```

Die einzelnen Programme können Sie mit RUN 3000, RUN 4000 oder RUN 5000 starten.

```
READY.

 3000 REM MUSIK DEMO 8
 3010 REM ING W HOFACKER GMBH
 3020 POKE 59467,16
 3030 POKE 59466,10
 3040 FOR P=1 TO 255
 3050 POKE 59464,P
 3055 NEXT P
 3057 POKE 59467,0
 3060 POKE 59466,0
 3070 END
 4000 REM MUSIK DEMO 9
 4010 REM ING W HOFAKCER GMBH
 4020 POKE 59467,16
 4030 POKE 59466,10
 4040 FOR I=1 TO 2
 4050 FOR X=255 TO 1 STEP -1
 4060 POKE59464,X
 4070 NEXT X:NEXT I
 4080 POKE 59464,0
 5000 REM MUSIK DEMO 10
 5010 REM ING W HOFACKER GMBH
 5020 POKE 59467,16
 5030 POKE 59466,51
 5040 LET X=INT(255*RND(TI)+1)
 5050 POKE59464,X
 5060 FOR I=1 TO 250:NEXT I
 5070 POKE 59464,0
 5080 FOR I=1 TO 250:NEXT I
 5090 GOTO 5040
READY.
```

Programm: Data Musik

Beim nachfolgenden Programm werden die Töne aus Datastatements genommen und abgespielt. Es arbeitet auch über den 6522 VIA.

Die zu spielenden Noten werden ab Zeile 500 in Datastatements eingegeben.

Sie können die gewünschte Note mit nachfolgendem Wert für T eingeben.

Beispiel:
A,2, B,3, C#,2, F,3

C,2 = Viertelnote
F,3 = Viertelnote mit Punkt

Pausen werden durch Eingabe von Z in die Datastatements erreicht. (Z,1 ist eine kurze Pause), Z,7 ist eine lange Pause.

Nachdem ein Lied eingegeben wurde, kann RUN gegeben werden. Der Computer fragt Sie nun nach der Schieberegisterkonfiguration (15, 51 oder 85) für D und nach T2, dem Tempo. Es kann frei gewählt werden.
Die Eingabe erfolgt durch Komma getrennt hintereinander, z. B. 15,20 (Return).

So, nun viel Spaß beim Musik-Generieren.

```
10 REM MUSIK UEBER DAS SCHIEBEREGISTER
20 REM ING W HOFACKER GMBH DØMUENCHEN75
30 PRINT"3MUSIK UEBER 6522 VIA
31 PRINT"ING W HOFACKER GMBH
32 FOR Y=1 TO 2000:NEXT Y
33 PRINT"3BITTE VERSTAERKER
34 PRINT"          AN CB 2 ANSCHLIESSEN!
35 PRINT"NUR BEI AUSGESCHALTETEM GERAET
       ETWAS AM
36 PRINT"USER PORT AUFSTECKEN ODER
       ABNEHMEN!!!!!
37 POKE59467,16
38 INPUT"D(15,51,81),T2(100)";D,T3
39 LET T2=T3
40 POKE59466,D
41 POKE59464,0
42 GOSUB 6800
43 READ R$,T
44 PRINT R$"...."T"     "D"
45 IF R$="XX"THENGOTO 900
46 GOSUB 10000
47 GOSUB 6000
48 GOTO 43
100 D=81:RETURN
101 D=85:RETURN
102 D=10:RETURN
103 D=15:RETURN
104 D=20:RETURN
105 D=30:RETURN
106 D=40:RETURN
107 D=50:RETURN
108 D=5:RETURN
109 D=10:RETURN
499 REM HIER KOMMEN DIE NOTEN HINEIN
500 DATA E1,2,Z,3,E1,2,Z,3,E1,2,D1,2
501 DATA E1,2,G1,2,G1,2,F1,2,D1,2
502 DATA Z,3,D1,2,Z,3,D1,2,A1,2,G1,2
503 DATA E1,2,Z,2,E1,2,Z,2,E1,2,D1,2,E1
       ,2,G1,2,G1,2,F1,2
504 DATA D1,2,E1,2
505 DATA F1,2,A1,2,G1,2,B,2,C1,2,C1,2
506 DATA A,2,Z,3,G,2,Z,3,H,2,Z,3
599 DATA XX,0
900 RESTORE
901 TEST=TEST+1:T2=INT(T2*0.7)
902 ON TEST GOSUB 100,101,102,103,104,
                  105,106,108,109
903 IF TEST>9 THEN GOTO 39
904 GOTO 40:REM ES WIEDERHOLT SICH
908 REM ..RUN 909 LOESCHT DAS
       SCHIEBEREGISTER
909 PRINT PEEK(59467),"59467"
990 POKE59464,221
995 POKE59464,0
999 POKE59467,0:END
6000 POKE59464,R
6005 IF T<1THEN GOTO 6020
6010 ON T GOTO 6100,6200,6300,6400,6600,
              6800,6900
6020 FORT1=1TO2:NEXT T1:RETURN
6100 FOR T1=1 TO 2:NEXT:RETURN
6200 FOR T1=1TO T2*T2:NEXT:RETURN
6300 FOR T1=1TO3*T2:NEXT:RETURN
6400 FOR T1=1 TO 4*T2:NEXT:RETURN
6600 FOR T1=1 TO 6*T2:NEXT:RETURN
6800 FOR T1=1 TO 8*T2:NEXT:RETURN
6900 FOR T1=1 TO12*T2:NEXT:RETURN
6999 REM TONUNTERROUTINEN
10000 REM INTERPRETER
10010 IF R$="BØ"THEN R=251
10020 IF R$="C" THEN R=237
10030 IF R$="C#"THEN R=224
10040 IF R$="D" THEN R=211
10050 IF R$="D#"THEN R=199
10060 IF R$="E" THEN R=188
10070 IF R$="F" THEN R=177
10080 IF R$="F#"THEN R=167
10090 IF R$="G" THEN R=157
10100 IF R$="G#"THEN R=149
10110 IF R$="A" THEN R=140
10120 IF R$="A#"THEN R=132
10130 IF R$="B" THEN R=124
10140 IF R$="C1"THEN R=117
10150 IF R$="C1#"THEN R=111
10160 IF R$="D1" THEN R=104
10170 IF R$="D1#"THEN R=99
```

```
 10180 IF R$="E1" THEN R=93
 10190 IF R$="F1" THEN R=88
 10200 IF R$="F1#"THEN R=83
 10210 IF R$="G1" THEN R=78
 10220 IF R$="G1#"THEN R=73
 10230 IF R$="A1" THEN R=69
 10240 IF R$="Z" THEN R=0
 10999 RETURN
READY.
```

Zum Abschluß unseres ersten kleinen Artikels über Computermusik mit dem PET wollen wir Ihnen noch einige kleine Hinweise geben, wie Sie ein bestehendes Programm mit Toneffekten erweitern können.

Wir haben z. B. ein Computerspiel, bei dem wir gewinnen oder verlieren können. (Black Jack aus P8, Las Vegas Cassette). Je nachdem, wie das Spiel ausgeht, soll ein unterschiedlicher Ton über CB2 erzeugt werden.

Wir setzen also an das Ende des Programmes 2 Unterroutinen mit verschiedenen Toneffekten.

```
z. B. an Adresse 2000 POKE 59467,16
                 2010 POKE 59466,10
                 2020 FOR P=1 TO 255
                 2030 POKE 59464,P
                 2040 NEXT P
                 2050 POKE 59467,0
                 2060 POKE 59466,0
                 2070 RETURN
An Adresse       3000 POKE 59467,16
                 3010 POKE 59466,10
                 3020 FORX=250TO1 STEP-1
                 3030 POKE,X
                 3040 NEXT X
                 3060 POKE 59467,0
                 3060 POKE 59466,0
                 3070 RETURN
```

Nun können wir an den gewünschten Stellen im Programm die GOSUB 2000 oder GOSUB 3000 Befehle einsetzen. Sie können natürlich jetzt die Töne noch variieren und exotische Toneffekte selbst programmieren.

Diese Artikelserie wird fortgesetzt. Schreiben Sie uns Ihre Erfahrungen und senden Sie uns Ihre Programme mit Toneffekten. Wir werden dann eine Audio- und Datencassette mit allen Einsendungen erstellen. Jeder Einsender eines Programmes erhält dann gegen Rückporto und Cassettenpreis von DM 9,80 eine Cassette mit allen Einsendungen. Sie können Ihren Namen und Adresse im Programm angeben und so Kontakte mit anderen PET-Besitzern knüpfen. Einsendeshcluß ist der 30. Februar 1979.

Also dann, gleich ans Programmieren! Viel Spaß ! Ihre ELCOMP-Redaktion

PS. Wir haben für Sie einen kompletten Verstärker mit Anschlußkabeln und Lautsprecher als Bausatz oder Fertiggerät vorbereitet. Dieser Verstärker braucht keine zusätzliche Betriebsspannung. Er wird über das 2. Cassetteninterface versorgt. Schreiben Sie uns, wenn Sie hierfür Interesse haben.

NOTIZEN

IEEE 488

IEEE488 Geräteinterface Bus
Der Interface-Bus des PET
Einführung und Beschreibung

Allgemeine Einführung

Im Jahre 1972 wurden auf einer Tagung des IEC in München ein amerikanischer und ein deutscher Vorschlag eingereicht, die eine Zusammenschaltung eines Meßsystems mittlerer Geschwindigkeit bei vergleichsweise niedrigem Preis für das System Interfacing gestatteten.

Beide Systeme gingen von einem bit-parallelen, byte-seriellen Konzept aus, das als Bus-System ausgelegt einen asynchronen Befehls- und Daten-Austausch zwischen den beteiligten Geräten gestattet. Auf das amerikanische System einigte man sich im September 1974 in Bukarest schließlich. Das amerikanische System basierte auf dem bereits 1965 bei Hewlett-Packard erarbeiteten Interface-System für programmierbare Meßgeräte.

Für den daraus erarbeiteten IEC-Entwurf vom Juni 1975 wurde bezüglich des verwendeten Steckers eine vom amerikanischen Entwurf abweichende Regelung vorgeschlagen.

Die dann relativ lang dauernde Beschlußfassung des IECs hat dazu geführt, daß die USA im April 1975 den amerikanischen Entwurf unter der Bezeichnung IEEE488 als eine Industrienorm dokumentierte. Im Januar 1976 zog das American National Standard Institut (ANSI) nach und publizierte diese Industrienorm als ANSI-Standard MC 1.1. Erst auf einer der letzten Zusammenkünfte des IECs im Frühjahr 77 einigte man sich auf das vom IEC erarbeitete Dokument.

Damit ergab sich die etwas unbefriedigende Situation, daß für ein und den selben Interfacebus zwei verschiedene Stecker verwendet wurden. Weil jedoch bereits nach Abschluß der IEEE488 – Norm viele Hersteller zu dem dort dokumentierten Stecker griffen, liegt der Anteil der Geräte mit dem im IEC-Dokument vorgeschlagenen Stecker heute nur bei ca. 5%.

Auf dem Markt sind jedoch Adapterkabel für diese beiden Steckernormen erhältlich.

Drei Voraussetzungen sind heute erfüllt, um ein Meßdatenerfassungs- und Verarbeitungssystem einfach und kostengünstig auszubauen:

1. Ein international genormter Interfacebus steht zur Verfügung
2. Eine große Anzahl ausgeklügelter Instrumente, teilweise ausgerüstet mit internen Prozessoren (Mikroprozessor)
3. Preisgünstige und einfach zu bedienende Computer. (Tischcomputer, Minicomputer, Rechner-Verbundsystem).

Bereits ungefähr 80 Hersteller liefern über 200 Produkte mit diesem standardisierten Interfacebus. Hewlett-Packard steht mit einer Zahl von über 80 Geräten an der Spitze. Das zahlenmäßige Wachstum dieser Produkte, im Verhältnis der Zeit (Jahre), steigt immer noch überproportional an.

Aufbau des Interfacebusses
Das parallele Bus-Konzept, verglichen mit einem anderen Konzept (z. B. mit dem Sternanschluß-Verfahren, das bisher üblich war) erlaubt den Betrieb eines ganzen Systems an einer „Input/Output"-Karte des Steuergerätes bis zu einer Zahl von 14 anzuschließenden Geräten.

Es handelt sich um ein charakterserielles System. Dadurch wird die Beschränkung auf 16 Leitungen möglich. Alle am Interface Bus angeschalteten Geräte hängen parallel an diesen Leitungen. Die 16 Bus-Leitungen lassen sich in drei Untereinheiten einteilen:

a) Der Datenbus mit 8 Leitungen
b) Der Handshake-Bus mit 3 Transferleitungen
c) Der Kontroll-Bus mit 5 Steuerleitungen.

Die Verbindung zwischen den einzelnen Geräten erfolgt mit einem rein passiven Kabel. Die Bus-Elektronik ist jeweils im Gerät untergebracht. Der Interface-Bus arbeitet mit TTL-Pegel in der „Negativ-True"-Logik (logisch 0 entspricht high Pegel). Die Ausgangstreiber haben ein „Fan-Out" von 30. Damit werden die „Thresh-Hold-Pegel" (Schwellwerte) auch bei voller Gerätezahl erreicht. Das Impedanzverhalten der Verbindungskabel (spezifiziert in der Norm) unter Berücksichtigung der Signalzeiten läßt eine maximale Bus-Länge von

20 m zu, mit der Einschränkung, daß pro Gerät eine Kabellänge von 2m nicht überschritten werden soll. Das bedeutet, daß alle 4 m eine ohmsche Last (Wirklast) vorhanden sein muß. Die Übertragungsgeschwindigkeit auf dem Interfacebus beträgt 250k-Byte pro sec. bis zu einer Buslänge von 20m, wenn alle 2m Einheitslasten angebracht sind und 48 mA-Treiber mit offenem Kollektor verwendet werden. Die maximale Übertragungsgeschwindigkeit von einem Mega-Byte/ sec. wird erreicht bei einer Kabellänge pro Gerät von 0,5 m und wenn 48 mA Tri-State-Treiber verwendet werden.

Die tatsächliche Maximalgeschwindigkeit hängt natürlich sehr stark von dem Zeitverhalten der angeschlossenen Geräte ab. Die höchsten Datenraten werden nur dann erzielt, wenn im Gerät ein Zwischenspeicher für mindestens ein Byte vorhanden ist.

Als Anhaltspunkt für die Übertragungsgeschwindigkeit von der Seite der Kontroller (Stand 1978) kann angenommen werden, daß Tischcomputer mit einer maximalen Übertragungsrate von 50 k-byte pro sec. arbeiten.

Dem Interface-Bus liegt das Konzept zugrunde, kleinere, in sich geschlossene Datenerfassungs- und Verarbeitungssysteme zu betreiben. Für eine Ankopplung an größere Datensysteme (wobei meist auch größere Entfernungen überbrückt werden müssen) eignen sich andere Schnittstellen besser, wie z.B. die V-24-Schnittstelle. Die meisten Tischcomputer besitzen eine solche Anschlußmöglichkeit.

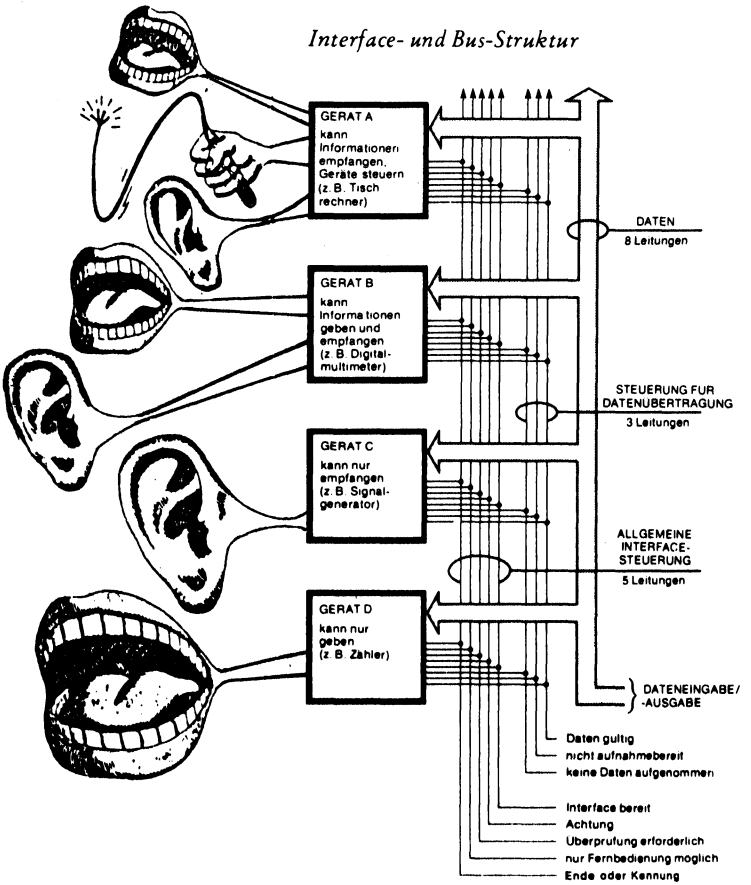
Funktionsweise des Interfacebusses

Der Interface-Bus ist in der Norm definiert:

1. Mechanisch:
Hier gilt der Vorbehalt, daß die IEC 66.22 Norm einen anderen Stekker verwendet als dies in der Norm IEEE 488/ANSI MC 1.1 vorgeschrieben ist. Dieses Problem läßt sich jedoch mit Adapterkabeln umgehen.

2. Elektrisch:
Der Interface Bus arbeitet mit TTL-Pegel in der „Negativ-True"-Logik.

3. Funktionsablauf:
Die Zustandsabläufe, abhängig von den möglichen Gerätefunktionen, sind festgelegt.



4. Nachrichtenübertragung:
Die Codierung interner Nachrichten fällt nicht in den Geltungsbereich der Norm (ausgenommen des Verfahrens der Adressierung mittels des ISO-7bit-Codes). Hier bleibt ein offenes Feld für den Entwickler des einzelnen Herstellers. Damit ist der Implementierung von neuen Technologien innerhalb der Geräte keine Grenze besetzt.

Adressierung

Die Geräte aus dem Interface-Bus können verschiedene Funktionen ausüben (Talker, Listener, Kontroller). Die parallele Zusammenschaltung aller Geräte hat eine bindende Gesetzmäßigkeit zur Folge.

1. Nur ein Kontroller kann auf dem Interface-Bus tätig sein. Nur dieser hat Zugriff zum Kontrollbus (5 Steuerleitungen).
2. Gleichzeitig kann auf dem Interface-Bus nur ein Talker (Sender) tätig sein. Doch können gleichzeitig mehrere Listener (Empfänger) tätig sein.
3. Alle Geräte werden durch den Kontroller mittels der Adressierung in die betreffende Funktion geschaltet. Jedes Gerät der möglichen 15 Geräte hat eine hardwaremäßig einstellbare Adresse.

Für die Adressencodierung wird der ISO-7-bit-Code (ASCII-Code, American Standard Code for Information Interchange) verwendet.

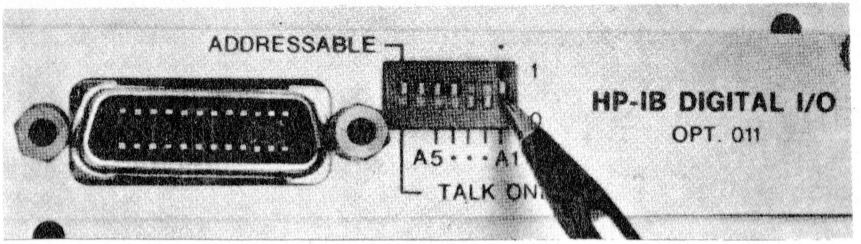
Ein Gerät, welches in der Talker- und Listenerfunktion geschaltet werden kann, hat für jede Funktion eine Adresse. Diese beiden Adressen unterscheiden sich jedoch nur in bit 6 und 7. Ist das bit 7 gesetzt, arbeitet das Gerät als Talker, ist das bit 6 gesetzt, arbeitet das Gerät als Listener.

Bit 1 bis 5 werden am Gerät eingestellt. Weil sich die Bit-Konstellationen eines Gerätes für den Talker- und Listener-Betrieb nur im 6. und 7. bit unterscheidet, gehören immer 2 ASCII-Zeichen unbedingt zusammen. Ist z. B. das Listener-ASCII-Zeichen eines Gerätes die 6, dann ist das Talker-ASCII-Zeichen des Gerätes ein V, bzw. Listener-Adresse 5 gehört zur Talker-Adresse U, usw. Häufig sind bei den Bus-Geräten auch das 6-bit, bzw. bit 7 am Adressenschalter einstellbar. Dieses ist nur notwendig für den Betrieb ohne Kontroller. Z.B. Datenlogger-Betrieb. Dann wird ein Gerät als Talker und ein Gerät als Listener geschaltet. Der Meßzyklus kann dabei von einem Gerät des Interface-Busses mittels des „Handshake Timings" gesteuert werden.

Die bei der Adressierung vom Kontroller gesendeten ASCII-Zeichen werden über die 8 Datenleitungen übertragen. Damit Daten von Adressen unterschieden werden können, wird bei der Adressierung zusätzlich die Kontrolleitung ATN des Kontrollbusses aktiviert.

Wichtig ist, daß bei jeder neuen Adressierung die alten Funktionen gelöscht werden (unbedingt notwendig bei Talkern). Dies erfolgt mit dem Senden einer auf dem Interfacebus nicht bekannten Talker- oder Listener-Adresse.

Dazu verwendet man die Adresse 31 (bit 1 bis 5 gesetzt!). Deshalb darf diese Adresse an keinem Gerät benutzt werden.



Address Switches					Talk Address Character	Listen Address Character
A_5	A_4	A_3	A_2	A_1		
0	0	0	0	0	@	SP
0	0	0	0	1	A	!
0	0	0	1	0	B	"
0	0	0	1	1	C	#
0	0	1	0	0	D	$
0	0	1	0	1	E	%
0	0	1	1	0	F	&
0	0	1	1	1	G	'
0	1	0	0	0	H	(
0	1	0	0	1	I	)
0	1	0	1	0	J	*
0	1	0	1	1	K	+
0	1	1	0	0	L	,
0	1	1	0	1	M	—
0	1	1	1	0	N	.
0	1	1	1	1	O	/
1	0	0	0	0	P	0
1	0	0	0	1	Q	1
1	0	0	1	0	R	2
1	0	0	1	1	S	3
1	0	1	0	0	T	4
1	0	1	0	1	U	5
1	0	1	1	0	V	6
1	0	1	1	1	W	7
1	1	0	0	0	X	8
1	1	0	0	1	Y	9
1	1	0	1	0	Z	:
1	1	0	1	1	[	;
1	1	1	0	0	\	<
1	1	1	0	1	]	=
1	1	1	1	0	^	>

„ Handshake-Timing"
Die Handshakeleitungen (3 Transferleitungen) sollen sicherstellen, daß schnellere Geräte mit schnellen Geräten sowie mit langsamen Geräten kommunizieren können. Jede Datenübertragung von einem Gerät zu anderen Geräten wird nicht eher begonnen, bis alle zugehörigen Geräte ihr Data Accepted gesendet haben.

Wichtig ist es zu wissen, daß nur die Geräte an einem Datenaustausch teilnehmen, die vorher durch die Adressierung dazu bestimmt wurden.

Damit wird es ermöglicht, daß, selbst wenn besonders langsame Geräte wie z.B. Drucker, in ein System integriert sind, zwei sehr schnelle Geräte miteinander auch sehr schnell kommunizieren können.

Auf dieses Handshake-System hat Hewlett-Packard ein Patent. Hewlett-Packard hat sich jedoch gegenüber dem Normenausschuß dazu verpflichtet, dieses Patent jedem der es anwenden möchte, in Lizenz gegen eine einmalige Schutzgebühr von $ 250,-- zu geben. Eine Erklärung seitens des Anwenders über die Verwendung dieser Lizenz ist nicht notwendig.

Bus-Steuerungen
Der Kontrollbus (5 Steuerleitungen) ermöglicht die verschiedenen Kommandos in die Geräte. Die Leitung IFC (Interface Clear) bringt den Interfacebus in einen Beginnzustand. Die Leitung REN (Remote Enable) schaltet die Geräte in den Zustand der Fernbedienung.

Die Leitung SRQ (Service Request) ermöglicht eine Anfrage eines Gerätes an z.B. den Rechner während eines Programmablaufs.

Die Leitung EOI (End or Identify) wird 1. benutzt, um das Ende einer Datenübertragung anzuzeigen und 2. benutzt bei der Identifizierung eines Gerätes, welches z.B. Service Request gesendet hat.
Wenn die SRQ-Leitung von einem Gerät gesetzt ist, hat der Kontroller mittels einer Statusabfrage das Gerät herauszufinden, welches diese Leitung aktiviert hat.

Es gibt zwei mögliche Verfahren. Die Hersteller spezifizieren häufig nur die Möglichkeit der seriellen Abfrage (serielles „poll"-Verfahren). Damit wird jedes Gerät nacheinander vom Kontroller nach dem Statusbyte befragt. Ein bestimmtes oder auch mehrere bits (vom Hersteller vorgegeben) geben dann über den Gerätestatus Auskunft.
Die parallele Abfrage (paralleles „poll"-Verfahren) erlaubt bei bis zu 8 Geräten eine einzige Statusabfrage.

Dann ist jedem Gerät ein bit des 8 bit Bytes zugeordnet (vorher per Programmierung zugewiesen). Die Anzahl der Geräte, die dieses letztgenannte Verfahren erlauben, ist jedoch z. Zt. noch sehr klein.

Wichtig zu wissen ist es, daß ein Setzen der Service Request-Leitung von einigen Rechnern als regelrechter Interrupt gesehen wird, von anderen nur softwaremäßig erfragt werden kann.

Kurze Zusammenfassung der wichtigsten Eigenschaften des Interfacebusses

15 Geräte können zugleich an einem Interfacebus angekoppelt sein (incl. Interfacekarte).

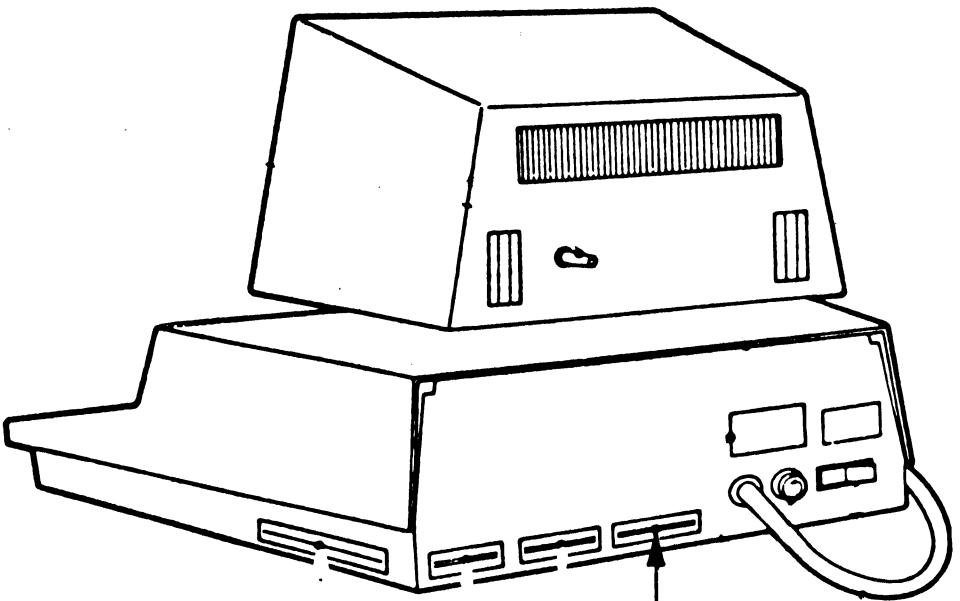
Die gesamte Länge der Interfacekabel darf 20 m nicht überschreiten. Größere Entfernungen können mit zwischengeschalteten, sogenannten Trägerfrequenz- Interface-Einheiten (2-Draht oder 4-Draht-Verbindung, 1000 m) überbrückt werden.

Die maximale Übertragungsgeschwindigkeit ist 1 mega-byte/sec. Diese hohe Geschwindigkeit setzt jedoch besonders kurze Kabellängen voraus (max. 10 m).

Die Leistungsfähigkeit eines Interfacebus Systems wird stark vom eingesetzten Kontroller bestimmt. Die Auswahl des richtigen Kontrollers ist ein systematischer Entscheidungsprozess. Der Preisbereich beginnt bei ca. DM 10.000,-- und ist nach oben durch die heutige Rechnerverbundtechnik eigentlich nicht mehr begrenzt. Als besonders optimal hat sich der Einsatz von Tischcomputern erwiesen. Die heute verfügbaren Tischcomputer arbeiten bereits in einer allgemein üblichen höheren Programmiersprache, wie z.B. BASIC.

Einige kurze Bemerkungen zum IEEE Interface-Port beim PET

Der IEEE488 Interface-Anschluß befindet sich beim PET auf der Rückseite gleich rechts neben dem Netzkabeleingang (s. Bild).

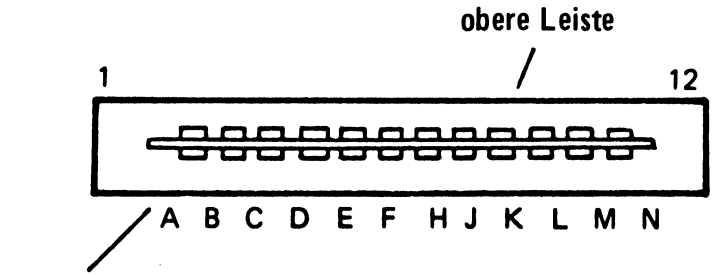


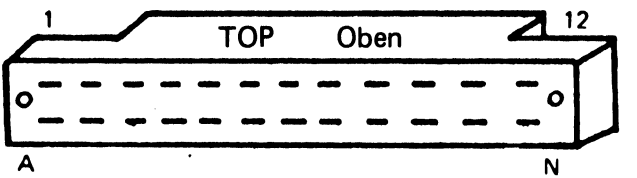
Der IEEE-Ausgang wird in erster Linie vom peripheren Interface-Adapter MPS 6520 angesteuert. Eingänge werden über das A-Register und Daten-Ausgaben über das B-Register initialisiert. Von einigen anderen, internen Bauteilen werden auch noch Leitungen an den IEEE-Bus geführt. 12 der 15 IEEE-Anschlüsse werden über Bus-Treiber an den Ausgang geführt. Die Treiber sind immer aktiviert und haben offenen Kollektor. Sie können Ströme bis zu 48 mA aufnehmen. Als Strom- oder Spannungsquelle haben sie jedoch nur einen bestimmten Innenwiderstand.
Der Befehl PEEK (59424) liest den IEEE-Port aus. Wenn nichts angeschlossen ist, liest er lauter Einsen entsprechend dem Abschlußwiderstand des Ports.

Der Befehl POKE 59426,X schreibt den Wert X in den IEEE-Port.

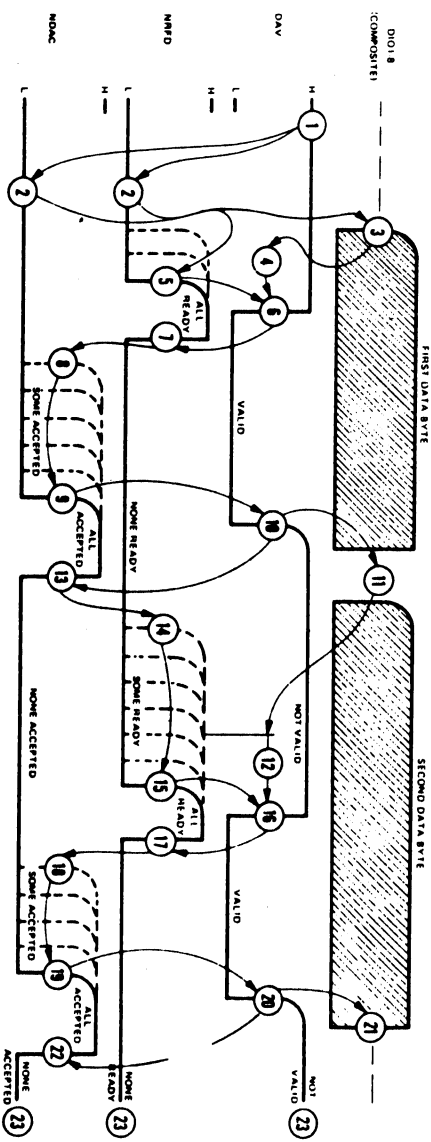
Die Anschlußbelegung des IEEE beim PET

Obere Leiste			Untere Leiste		
PET	IEEE		PET	IEEE	
1	1	D101	A	13	DIO5
2	2	D102	B	14	DIO6
3	3	DIO3	C	15	DIO7
4	4	DIO4	D	16	DIO8
5	5	EIO	E	17	Masse
6	6	DAV	F	18	Masse
7	7	NRFD	H	19	Masse
8	8	NDAC	J	20	Masse
9	9	IFC	K	21	Masse
10	10	SRQ	L	22	Masse
11	11	ATN	M	23	Masse
12	12	Chassi Masse	N	24	Masse





IEEE-Stecker für PET, Type CINCH 251-12-90-160


DIO1-8 (COMPOSITE)
DAV
NRFD
NDAC
H
L
FIRST DATA BYTE
SECOND DATA BYTE
ALL READY
VALID
SOME ACCEPTED
ALL ACCEPTED
NONE READY
NOT VALID
SOME READY
NONE ACCEPTED
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Bedeutungen der Anschlüsse am IEEE488-Bus

Die Anschlüsse D101 – D108 sind die Datenbusanschlüsse. Die Anschlüsse NRFD (Not Ready For Data), DAV (Data Valid) und NDAC (Data Not Accepted) stellen den Transferbus dar. Auf diesem Bus laufen alle Signale, die mit der Ein- und Ausgabe von Daten auf dem Bus zusammenhängen. Sie sorgen dafür, daß anstehende Daten übernommen werden, und daß die Datenübertragung komplett ausgeführt wird, bevor neue Daten wieder angeliefert werden können.

Weiterhin besitzt der IEEE488 einen Management-Kontrollbus, der die an den Bus angeschlossenen Geräte steuert. Er kann drei verschiedene Arten von Geräten behandeln:

1. Talkers – Geräte, die Daten auf den Bus geben
2. Listeners – Geräte, die Daten vom Bus entnehmen
3. Controllers – Kontrollsysteme

In der nachfolgenden Tabelle finden Sie eine Zusammenstellung der zugehörigen Hardwareadressen im PET. Der Management-Bus besteht aus folgenden Anschlüssen: ATN, SRQ, IFC, REN und EIO.

Hex Address	Decimal Address	Bits	IEEE	Mode
E820	59424	0-7	DI01-8	Input
E822	59426	0-7	DI01-8	Output
E821	59425	3	NDAC	Output
E823	59427	3 7	DAV SRQ	Input
E810	59408	6	EOI	Input
E840	59456	0 1 2 6 7	NDAC NRFD ATN NRFD DAV	Input Output Output Input Output

Beschreibung der Transferbusbezeichnungen

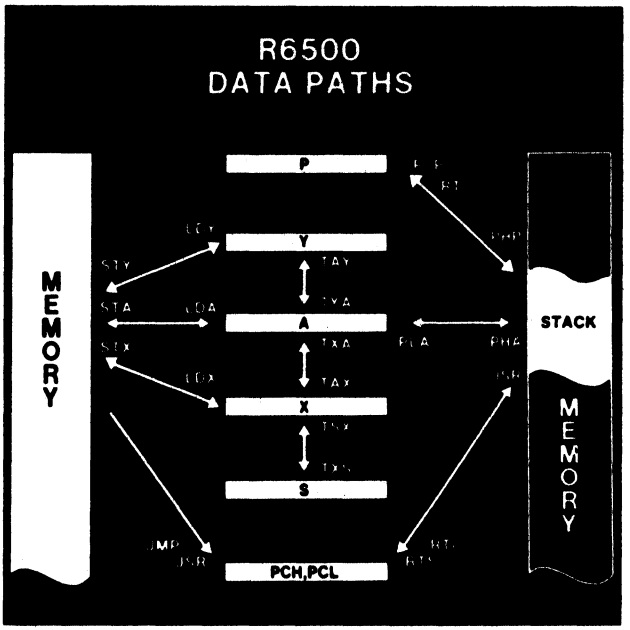
NRFD Not Ready For Data. Wenn dieser Anschluß auf log. „0" liegt, erhalten die Listener nicht das nächste Datenbyte. Wenn alle angeschlossenen Geräte fertig sind, geht der Anschluß NRFD auf log. „1". Dieses sagt dem Talker, daß er das nächste Datenbyte auf den Bus ausgeben kann.

DAV Data Valid. Wenn dieser Anschluß auf log. „0" geht, kann ein Listener ein Datenbyte aus dem Datenbus lesen. Der Talker kann den Anschluß DAV nicht auf log. „0" legen, solange NRFD auf log. „0" liegt. (Zuerst müssen einmal alle Listener fertig sein)

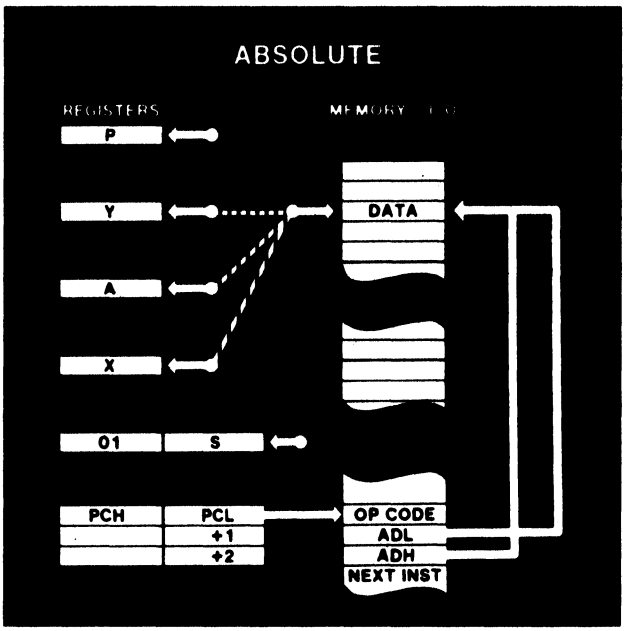
NDAC Data Not Accepted. Dieser Anschluß wird von jedem Listener auf log. „0" gehalten, bis das anstehende Datenbyte gelesen wurde. Wenn NDAC auf log. „1" geht, kann der Talker seine Daten wieder vom Datenbus runternehmen.

R6500

ADRESSIERUNG

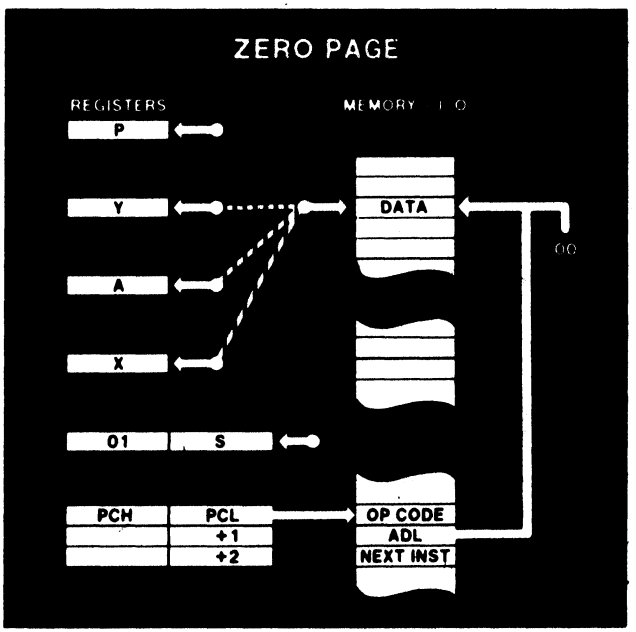


Datenfluß im R6500

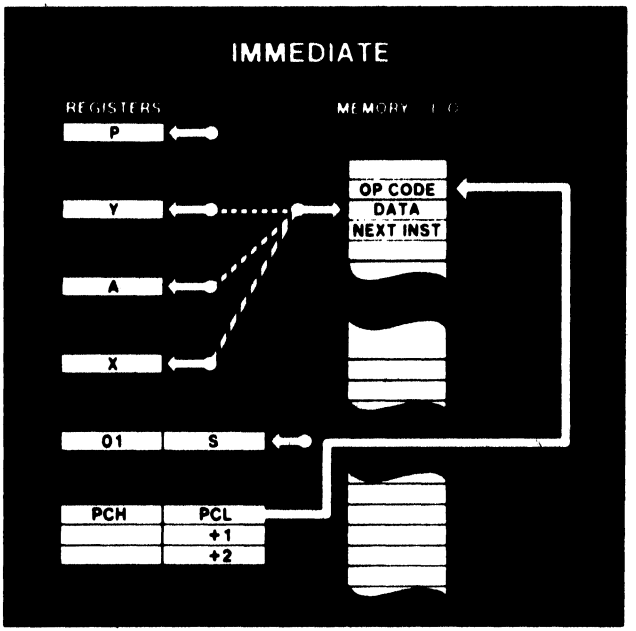


Absolute Adressierung

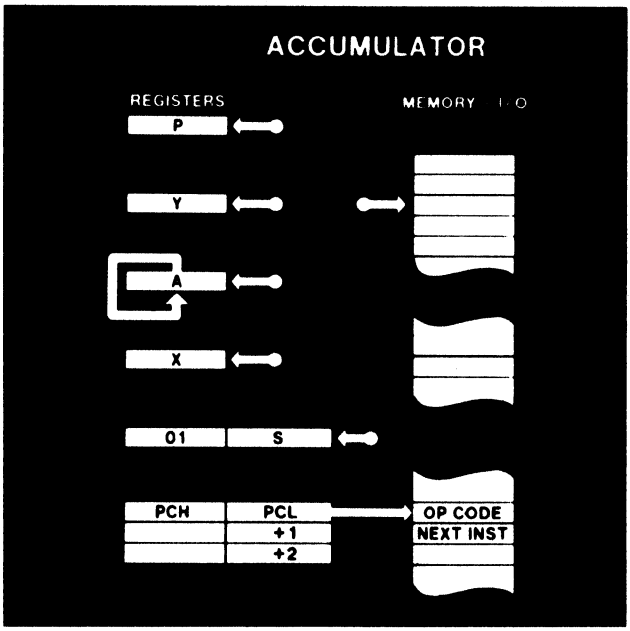
Ein Drei-Byte-Befehl, bestehend aus OPCODE, das zweite Byte enthält das niederwertige Byte der effektiven Adresse ADL. Das dritte Byte enthält das höherwertige Byte der effektiven Adresse (die Adresse, die Daten enthält).
Befehlsimpulse: ADC, DEC, JMP etc.



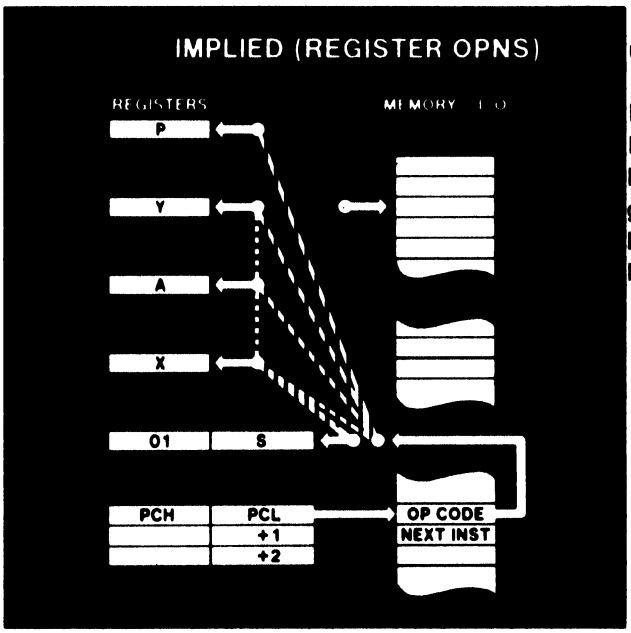
Zero Page Adressierung (Adressierung über die Page 0 = Speicheranfang)
Zweibytebefehl:
1. Byte OPCODE, zweites Byte enthält die effektive Adresse in der Zero Page (00). Es wird einfach angenommen, daß ADH = 00 ist.
Beispiele: CPX, DEC, LDX etc.



Unmittelbare Adressierung
Zwei-Byte-Befehl:
1. Byte enthält den OPCODE. Im zweiten Byte ist ein Wert, der vom Programmierer vorgegeben wird, abgelegt.
Beispiele: ADC, LDA, LDX etc.
Indirekt, indizierte Adressierung.

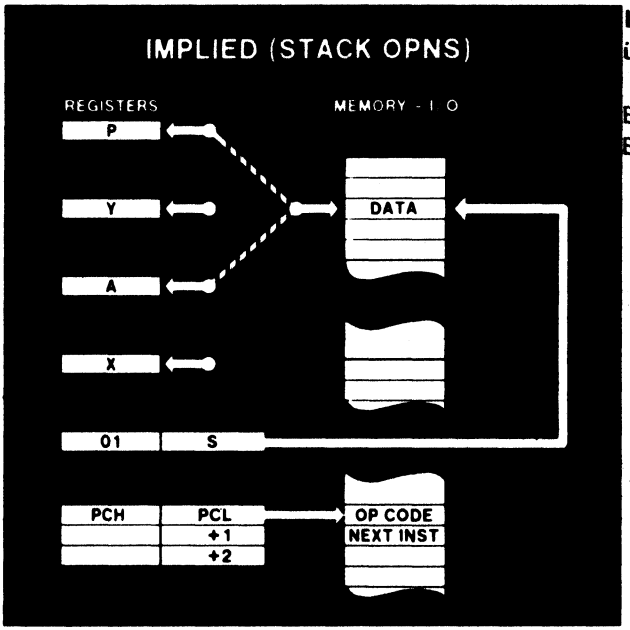


Direkte Manipulation des Akkumulators



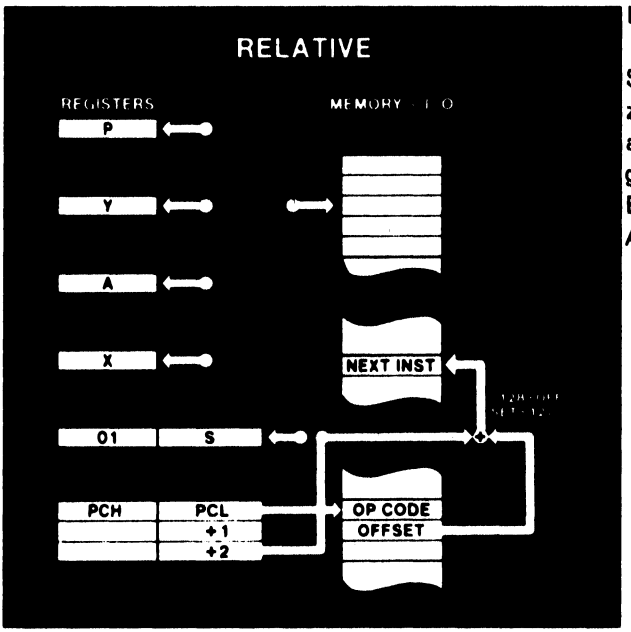
Implizierte Adressierung über Register

Ein Byte-Befehl, innerhalb der CPU werden Bits in den Registern gelöscht oder gesetzt. Beispiele: CLC, INX, NOP, TXA etc.



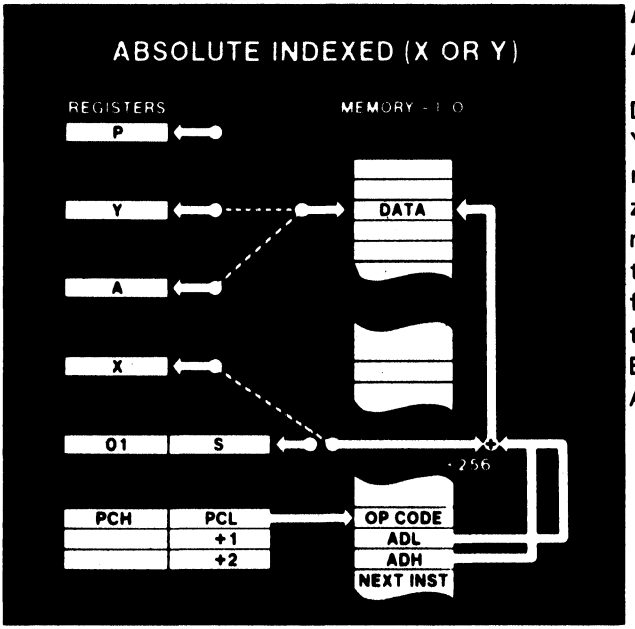
Implizierte Adressierung über Stack

Beispiele: RTS, PLA, BRK



Relative Adressierung

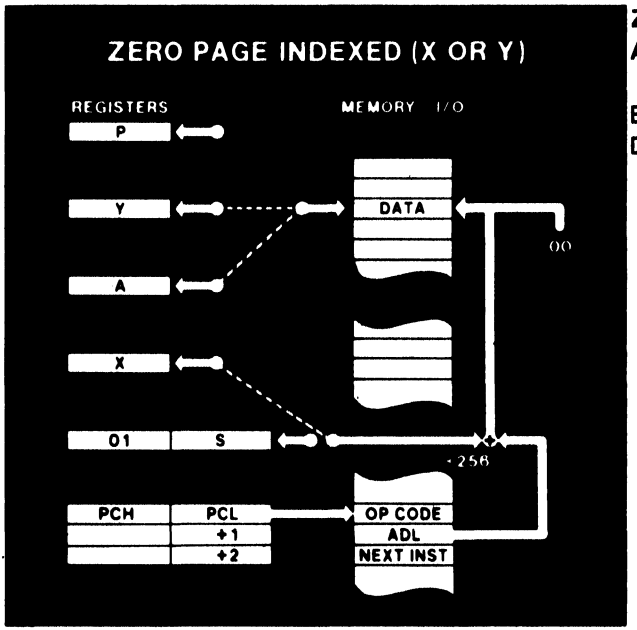
Sie wird meist für Verzweigungsoperationen angewendet. Der Programmzähler wird zur Ermittlung der nächsten Adresse verwendet.



Absolute, indizierte Adressierung

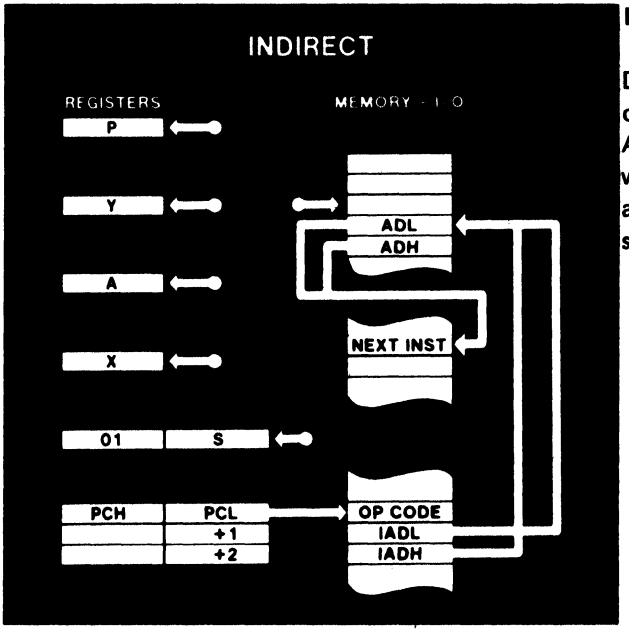
Der Inhalt des X- oder Y-Registers wird zu einer Ausgangsadresse hinzugezählt und so die neue Endadresse ermittelt. Ein Drei-Byte-Befehl. Pagegrenzen beachten.

Beispiel: CPM, ROL, ADC, LDA etc.



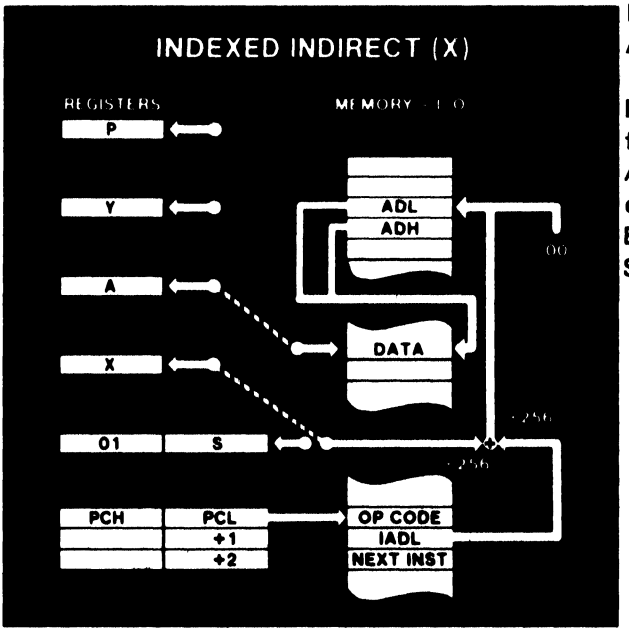
Zero-Page-indizierte Adressierung

Beispiele: AND, CPM, DEC etc.



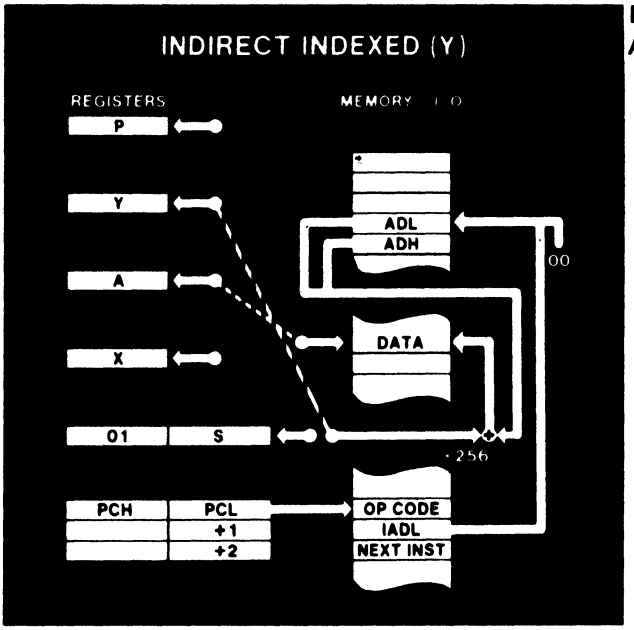
Indirekte Adressierung

Der Befehl besteht aus drei Bytes. Es wird eine Adresse, in der ein gewünschter Befehl steht, aus einer anderen Adresse entnommen.

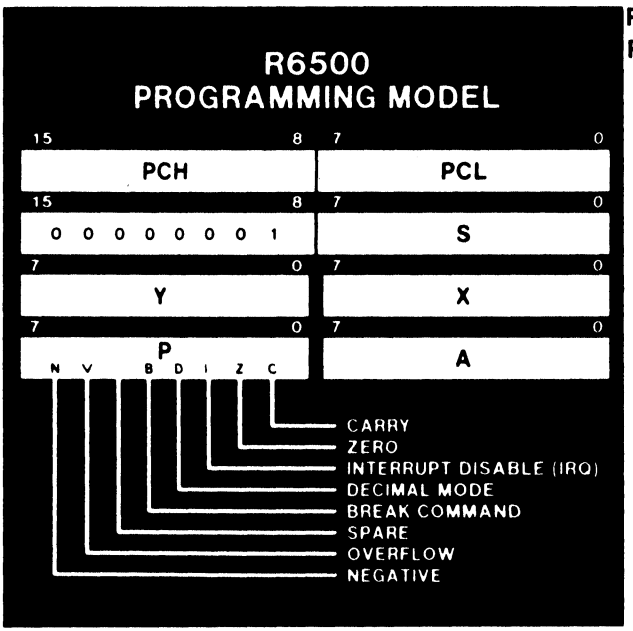


Indizierte, indirekte Adressierung

Der Befehl wird meist für die Abfrage von Ein-/Ausgabegeräten verwendet.
Beispiele: ADC, LDA, STA etc.

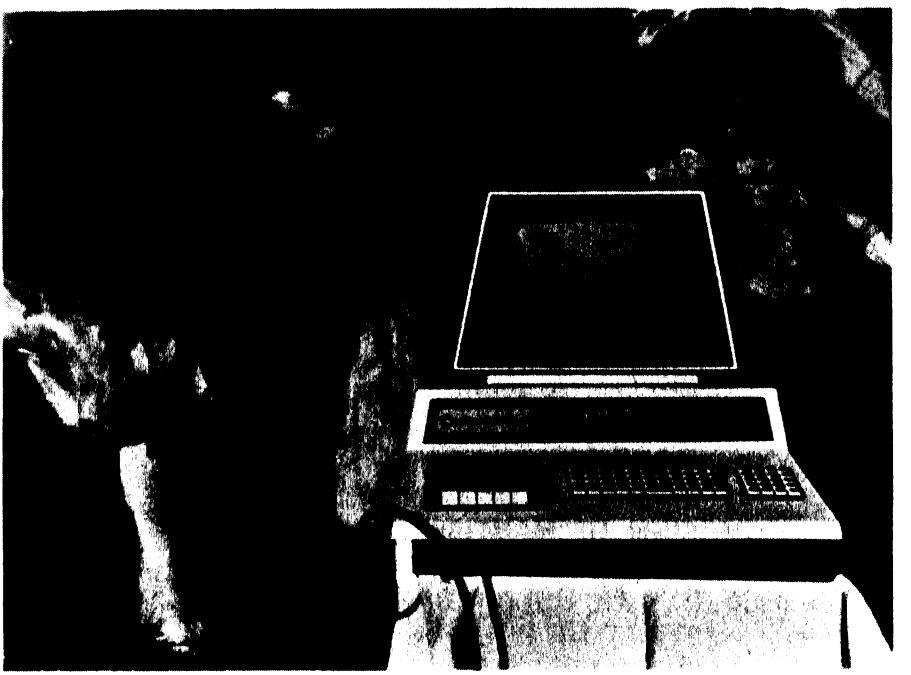


Indirekte, indizierte Adressierung



Programmiermodel des R6500

Speicherbelegung



0000 – 0002 JUMP, USER ADDRESS
0005 CURSOR COLUMN
000A – 005A BASIC INPUT BUFFER
005C BASIC INPUT BUFFER POINTER
005E CURRENT RESULT TYPE (FF)STRING (00)NUMERIC
005F CURRENT RESULT TYPE (80) INTEGER (00) FLOATING POINT
007A – 007B START OF BASIC STATEMENTS
007C – 007D START OF VARIABLE TABLE
007E – 007F END OF VARIABLE TABLE
0080 – 0081 START OF AVAILABLE SPACE
0082 – 0083 BOTTOM OF STRINGS (MOVING DOWN)
0084 – 0085 TOP OF STRINGS (MOVING DOWN)
0086 – 0087 TOP OF MEMORY ALLOCATED FOR BASIC WORKING AREA
0088 – 0089 CURRENT PROGRAM LINE NUMBER
008A – 0088 CURRENT PROGRAM LINE NUMBER SAVED BY END
008C – 008D CURRENT PROGRAM POINTER SAVED BY END
0092 – 0093 DATA STATEMENT POINTER
0094 – 0095 CURRENT VARIABLE SYMBOLS
0096 – 0097 CURRENT VARIABLE STARTING POINT
00AE–00AF POINTER ASSOCIATED WITH BASIC BUFF TRANSFER
0080 EXPONENT + $80
0081 MANTISSA MSB
0082 MANTISSA
0083 MANTISSA (FLOATING POINT ACCU-
0084 MANTISSA LSB MULATOR)
0085 SIGN OF MANTISSA (0 IF ZERO) (+ IF POS.) (– IF NEG)
0088 – 00C0 DYADIC HOLDING AREA
00C2 – START OF ROUTINE FOR FETCHING NEXT BASIC CHARACTER
00C9 – 00CA PROGRAM POINTER
– 00D9 END OF CHARACTER FETCH
00E0 SCREEN POSITION ON LINE
00E1 – 00E2 POSITION OF LINE START

00E3 – 00E4 CURRENT TAPE BUFFER POINTER
00E5 – 00E6 END OF CURRENT PROGRAM
00EA QUOTE MODE (00 IF NOT IN QUOTE)
00EE NUMBER OF CHARACTERS IN FILE NAME
00EF GPIB FILE #
00F0 GPIB COMMAND
00F1 GPIB DEVICE #
00F3 – 00F4 START OF TAPE BUFFER
00F5 CURRENT SCREEN LINE #
00F6 RUNNING CHECKSUM OF BUFFER
00F7 – 00F8 POINTER TO PROGRAM DURING VERIFY,LOAD
00F9 – 00FA FILENAME STARTING POINTER
00FC SERIAL WORD
00FD NUMBER OF BLOCKS REMAINING TO WRITE
00FE SERIAL WORD BUFFER
00FF BASIC
0200 – 0202 CLOCK H.M.S.
0203 MATRIX COORDINATE OF LAST KEY DOWN (255 IF NONE)
0204 SHIFT KEY STATUS (1 IF DOWN)
0205 – 0206 JIFFY CLOCK
0207 CASSETTE 1 ON SWITCH
0208 CASSETTE 2 ON SWITCH
0209 KEYSWITCH PIA
020B LOAD 0, VERIFY 1
020C STATUS
020E REVERSE VIDEO
020F – 0218 KYBD INPUT BUFFER
0219 – 021A HARDWARE INTERRUPT VECTOR
021B – 021C BREAK INTERRUPT VECTOR
0223 KEY IMAGE
0225 CURSOR TIMING

0228 TAPE WRITE
0242 – 024B LOGICAL NUMBERS OF OPEN FILES
024C – 0255 DEVICE NUMBERS OF OPEN FILES
0256 – 025F R/W MODES OF OPEN FILES (COMMAND TABLE)
0262 GPIB TABLE LENGTH
0265 PARITY
0268 POINTER IN FILENAME TRANSFER

026C SERIAL BIT COUNT
0270 TAPE WRITE COUNTDOWN
0273 LEADER COUNTER
0275 0 IF FIRST HALF BYTE MARKER NOT WRITTEN
0276 0 IF SECOND HALF BYTE MARKER NOT WRITTEN
0279 CHECKSUM WORKING WORD
027A – 0339 BUFFER FOR CASSETTE # 1
033A – 03F9 BUFFER FOR CASSETTE # 2
0400 START OF BASIC STATEMENTS
– 1FFF END OF AVAILABLE RAM (8K VERSION)
– 7FFF END OF AVAILABLE RAM EXPANSION
8000 – 8FFF VIDEO RAM
9000 – BFFF AVAILABLE ROM EXPANSION AREA
C000 – E0B0 MICROSOFT „8K" BASIC
E0B5 – E27D SYSTEM SET UP
E294 – E66A VIDEO DRIVER
E66B – E684 INTERRUPT HANDLER
E685 – E758 CLOCK UPDATE, KYBD SCAN (60 HZ INT.)
E75C – E7D4 KYBD ENCODING TABLE
E800 – EFFF PIA'S
F0B6 – F226 GPIB HANDLER
F346 – F82C FIDE CONTROL
F82D – FD15 TAPE CONTROL
FD38 – FFB2 DIAGNOSTICS
FFC0 – FFEC JUMP VECTORS
FFFA–FFFF 6502 INTERRUPT VECTORS (NMI NOT USED IN ORIG VERSIONS)

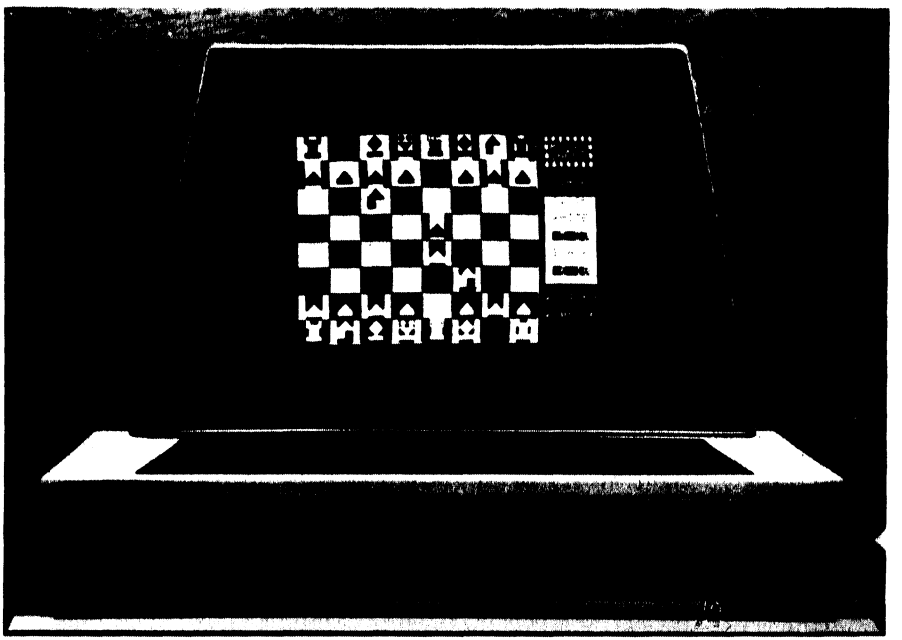
PROGRAMM Beschreibungen

Microchess
Joystick-Programmierung
Dual-Joysticks
Haushaltsfinanzprogramm
Mathematische Praktiken
Gamepac I (Spielepaket)
Household Utility I + II
Musik mit dem PET
Geschäfts- und Buchhaltungs-
programme

MICROCHESS

MICROCHESS

Endlich ist es auch auf dem deutschen Markt zu haben! Das zur Zeit wohl komfortabelste Schachprogramm für Microcomputer: Microchess 1.5. Das in Maschinensprache geschriebene Programm begnügt sich mit 4 K Byte RAM und ist in zwei Ausführungen jeweils auf Cassette lieferbar. Die eine ist für den bekannten PET 2001 von Commodore maßgeschneidert und die andere für den TRS-80 von Radio Shack.
Wir haben uns die Version für den PET-Computer etwas näher betrachtet.
Nachdem das Programm von Cassette geladen ist, meldet sich Microchess auf dem Bildschirm mit Titel und Verfasser. Wird nun eine beliebige Taste gedrückt (als Abschluß jeder Eingabe ist außerdem immer die RETURN-Taste zu drücken), so erscheint auf dem Bildschirm das Schachbrett mit den 32 Figuren in der Grundstellung. Die vom Computer geführten Figuren befinden sich immer am oberen Rand, gleichgültig ob er mit den weißen oder mit den schwarzen Figuren spielen soll.
Standardmäßig nimmt der Computer die weißen Steine und beginnt mit dem ersten Zug, sobald ein „P" eingegeben wurde. Will man selbst mit den weißen Steinen spielen, gibt man anstelle des „P" ein „X" ein und kann dann den ersten Zug (z. B.: E2–E4) eingeben. Wenn man mit den Bezeichnungen E2, H7 u. s. w. nicht ganz vertraut ist, kann man „N" eingeben und es erscheint in jedem der 64 Felder links oben die zugehörige Bezeichnung.
Rechts vom Spielfeld zeigt der Bildschirm sowohl die vom Computer als auch die von seinem Gegenspieler benötigte „Nachdenkzeit" an. Dies



entspricht voll einer Schachuhr, wie sie bei Turnieren verwendet wird. Unter den Zeitangaben wird außerdem die Anzahl der Spielzüge angezeigt.
Beeindruckend ist die Anzahl von acht (!) Schwierigkeitsstufen, mit denen das Programm ausgestattet ist. Die der eigenen Spielstärke entsprechende Schwierigkeitsstufe wird durch Angabe von „ IQ = x" („x" ist zwischen 1 und 8 frei wählbar) eingegeben. Sie kann auch während des Spiels beliebig geändert werden. Vom Schwierigkeitsgrad ist natürlich auch die Dauer der Antworten abhängig. Bei IQ=1 (niedrigste Stufe) antwortet Microchess sehr schnell (nach wenigen Sekunden), bei IQ=8 kann ein Zug 5 Minuten dauern.
Interessant ist, daß sich z. B. nur Stufe 1 vom bekannten Schäferzug überlisten läßt. Diese Stufe ist somit geeignet, auch Anfängern ein Erfolgserlebnis zu verschaffen. Noch ein weiterer beachtlicher Komfort: Microchess erlaubt es seinem Gegenspieler zu jeder Zeit und beliebig oft während des Spiels, die Seiten zu wechseln. Dieser Seitenwechsel wird durch Eingabe eines „X" und eines anschließenden „ P " erreicht.

Durch diese Möglichkeit kann man den Computer letztlich gegen sich selbst spielen lassen, wenn man anstelle des eigenen Zuges immer „ X " u. „ P " eingibt.

Interessant dürfte die Möglichkeit des Seitenwechsels vor allem dann sein, wenn man während des Spiels gegen den Computer von einer schwierigen Situation steht und wissen möchte, welchen Zug der Computer an dieser Stelle in der höchsten Schwierigkeitsstufe wählen würde. Wenn das Spiel z. B. gerade in der Schwierigkeitsstufe 4 läuft, sind dazu folgende Eingaben notwendig:

IQ = 8	(höchste Schwierigkeitsstufe)
X	(Seitenwechsel wird vollzogen)
P	(Der Computer denkt nach und zieht für den Gegner)
IQ = 4	(vorherige Schwierigkeitsstufe wird wieder hergestellt)
X	(Seitenwechsel zurück)
P	(Der Computer denkt nach und zieht wieder für sich selbst)

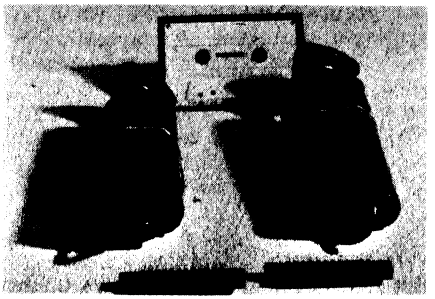
Anschließend kann das Spiel wieder normal fortgeführt werden.

Die Spezialzüge des Schachspiels werden wie folgt eingegeben:
Kurze Rochade: „ O–O" (Buchstabe O)
Lange Rochade: „ O–O–O"
En passant: z. B. „ E5–D6EP"

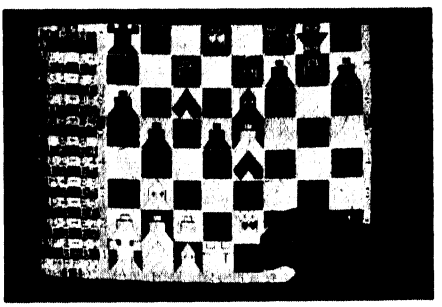
Microchess prüft alle Züge (mit Ausnahme der Rochade und en passant aus Speicherplatzgründen) auf ihre Gültigkeit.
Ungültige und unzulässige Eingaben werden mit einem Fragezeichen beantwortet.

Zusammenfassend kann gesagt werden, daß dieses Schachprogramm aufgrund seiner Vielfalt und seines Komforts eine echte Bereicherung des derzeitigen Softwaremarktes darstellt.

Fritz Schenk



Microchess kann auch mit Joysticks (Steuerknüppeln) geliefert werden. Vorerst ist nur die TRS-80 Version mit zwei Joysticks lieferbar. Sie können mit diesen beiden Steuerknüppeln auch gegeneinander oder gegen den Computer spielen. Die Verschiebung der Figuren geschieht durch Hochheben des Steuerknüppels mit anschließender Bewegung zum gewünschten neuen Feld. In der PET Version wird die Verschiebung durch die Feldangabe über die Tastatur erreicht.

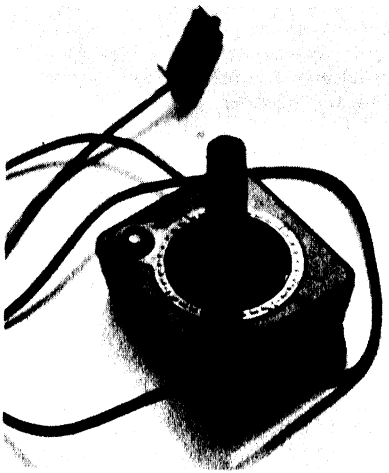


NOTIZEN

Joystick Programmierung

Ein Joystick für den PET

Der PET kann durch den Anschluß eines Joysticks zum interessanten System, zur Simulation von Bewegungsabläufen etc. erweitert werden.



Mit dem Joystick (Steuerknüppel) können Sie dann Objekte, Punkte etc. auf dem Bildschirm von Hand verschieben.

Anschluß des Joysticks:

Drehen Sie den Anschlußstecker so, daß die Seite mit dem Aufkleber „ This side up" nach oben zeigt. Das Kabel zwischen Joystick und Stecker sollte von der rechten Seite kommen, wenn man vor dem PET steht. Nun schieben Sie den Stekker vorsichtig, aber fest auf den parallelen Benutzer Port # 5 des PET.
Sie können auch im PET-Handbuch nachsehen, wo sich dieser befindet. Ist der Joystick angeschlossen, können Sie sofort mit dem Einlesen des ersten Programmes beginnen. Es befindet sich auf einer Datencassette die vom Hofacker-Verlag, Holzkirchen oder durch den Fach und Buchhandel bezogen werden kann. Die Bestellnummer lautet: P 4. Das gesamte Joystick-Paket besteht aus einem digitalen Joystick mit einer Datencassette, welche zwei interessante Programme enthält. Das erste heißt:
„SCRATCHPAD". Um es zu laden, legen Sie die Cassette in den Recorder ein, spielen bis zum Anfang zurück und geben „ LOAD " oder „ LOAD SKETCHPAD" ein. Alle weiteren Anweisungen sind im Programm selbst enthalten.

Beachten Sie, daß aufgrund der internen Codierung des PETs nicht alle Symbole durch einen Knopfdruck zu erreichen sind. Dies kommt daher, daß der ASCII-Code, der zu einem entsprechenden Symbol gehört, nicht der gleiche ist, wie der Code des Character Generators für die Bildschirmanzeige. Im großen und ganzen sind jedoch die meisten Symbole über die Tastatur erreichbar.

Das zweite Programm auf der Joy Stick Cassette heißt MAZE. Um es zu laden, spielen Sie die Cassette zurück und geben „ LOAD MAZE" ein. Alle weiteren Anweisungen finden Sie wieder im Programm selbst. Wenn Ihnen der Cursor im MAZE-Programm zu langsam oder zu schnell erscheint, kann die Geschwindigkeit in Zeile 150 des Programmes geändert werden. Diese Zeile enthält eine FOR–NEXT–Schleife zur Geschwindigkeitsfestlegung. Die Geschwindigkeit kann durch Änderung der oberen Grenze dieser Schleife geändert werden. Wenn Sie z. B. die Geschwindigkeit des Cursors erhöhen wollen, können Sie die Schleife in FOR N = 1 TO 40 : NEXT N ändern.

Eine kleine Einführung in die Programmierung eines Joysticks in BASIC

Obwohl die Programme, welche mit dem Joystick geliefert werden, zur Kontrolle des Joy-

sticks in Maschinensprache geschrieben sind, kann man den Joystick aber auch in BASIC programmieren. Der wohl sichtbarste Unterschied zwischen diesen beiden Arten ist die Zeitdauer, die der Computer benötigt, um auf eine Joystickbewegung zu reagieren. Abgesehen davon gibt es im PET nichts, was in Maschinensprache geschrieben ist und auch nicht in BASIC gelöst werden könnte. Diese Beschreibung soll Ihnen zeigen, wie man beginnen kann, einen Joystick in BASIC zu programmieren.

Um die Stellung des Joysticks lesen zu können, muß die Adresse 59459 (dezimal) im PET auf 0 gesetzt werden. (5945 ist das Daten Richtungsregister und setzt fest, ob Daten auf dem Datenbus in den PET oder aus dem PET heraus fließen können).

Um dieses Register auf 0 zu setzen (00000000) wird der POKE-Befehl verwendet. Der Befehl POKE 59459,0 ermöglicht es dem PET, die Joysticksignale zu lesen.

Jetzt ist es auch wichtig, daß wir wissen, was in Adresse 59471 steht. Dort finden wir die Information, was unser Joystick gerade macht.

Beispiel: N = PEEK (59471)
führt zu der Variablen N als ein Ergebnis des momentanen Standes des Joysticks. Die Größe der Variablen N richtet sich wie folgt nach der Position des Joysticks. (N = PEEK (59471))

Wenn	N = 255,	Joystick ist nicht benutzt
	N = 254,	roter Knopf ist gedrückt
	N = 253,	nach Norden bewegen
	N = 251,	nach Süden bewegen
	N = 247,	nach Westen bewegen
	N = 245,	nach Nordwesten beweg.
	N = 243,	nach Südwesten bewegen
	N = 239	nach Osten bewegen
	N = 237	nach Nordosten bewegen
	N = 235	nach Südosten bewegen

Wenn der Joystick in eine der oben aufgeführten Richtungen bewegt wird, und gleichzeitig der rote Knopf gedrückt wird, wird N um eins verschoben.

Beispiel: Wenn der Knopf gedrückt ist und der Hebel nach Norden bewegt wird, ist N = 252 anstelle von 253.

Um genau zu sehen, was vorgeht, geben Sie einmal das folgende Programm ein:

```
10 POKE59459,0
15 PRINT PEEK (59471)
20 GO TO 15
```

Schließen Sie den Joystick an und geben Sie RUN ein. Sie sollten jetzt eine Spalte mit den Zahlen 255 auf der linken Bildschirmhälfte sehen. Sie wandert nach unten. Nun drücken Sie den roten Knopf. Die Zahlen sollten sich jetzt in 254 verwandeln. Nun lassen Sie den Knopf wieder los und drücken den Hebel nach oben. (Norden) (Ausgangsstellung des Joysticks ist immer roter Knopf rechts hinten).

Jetzt sollte die Zahl 253 erscheinen. Sie können so jetzt ein wenig experimentieren und die Zahlen der vorangegangenen Liste kontrollieren. Auch sollten Sie hin und wieder den roten Knopf drücken und nachsehen, ob auch wirklich die angezeigte Zahl um eins erniedrigt wird. Um das Programm zu stoppen, muß die RUN/STOP Taste gedrückt werden.

Wenn Sie die Cursor-Befehle des PETs verwenden, können Sie Ihren Joystick so programmieren, daß eine Figur auf dem Bildschirm umhergeschoben werden kann.

Beispiel: Wir wollen ein Grafiksymbol auf dem Bildschirm nach rechts (Osten) schieben.

```
10 POKE59459,0
15 N = PEEK (59471)
20 IF N = 239 THEN PRINT„0";
25 GOTO 15
```

In diesem Beispiel ist das Grafiksymbol der Buchstabe „ 0 " (für Osten). Zeile 10 ermöglicht dem PET den Joystickstand zu lesen. Zeile 15 liest dann den Joystick. Zeile 20 prüft, ob sich der Joystick nach Osten (rechts) bewegt. Wenn ja, dann wird der Buchstabe 0 ausgedruckt. Das Semicolon sagt, daß das 0 in der gleichen Zeile bleiben soll, da das BASIC den ausgedruckten Buchstaben sonst in die darauffolgende Zeile setzen würde. Zeile 25 bringt das Programm wieder zu Zeile 15, damit der Joystick erneut abgefragt werden kann.

Um das Programm zu starten, geben Sie RUN ein. Um das Programm zu stoppen, kann die RUN/STOP-Taste gedrückt werden. Am Schluß wäre noch folgendes zu bemerken.

Wenn Sie mit dem Buchstaben „ E " am rechten

Bildschirmrand angelangt sind und den Hebel noch weiter nach rechts gedrückt halten, läuft der Buchstabe in der nächsten Zeile wieder nach links. Wenn Sie haben wollen, daß der Buchstabe am rechten Ende stehen bleibt, muß dies in das Programm eingebaut werden. Auch wird das Programm nur auf die Befehle in Richtung Osten reagieren. Wollen Sie mehr, muß dies in das Programm eingebaut werden.

Wenn Sie alle Richtungen programmieren wollen, und diese in Ihren eigenen BASIC-Programmen verwenden wollen, muß dies entsprechend eingebaut werden.
Der Joystick kann auch in Maschinensprache programmiert werden, welches eine sehr viel schnellere Ausführung erlaubt und nur 1/8 an Speicherplatz benötigt. Wie wir wissen, ist die Programmierung in Maschinensprache jedoch recht aufwendig, so daß wir dem PET-Benutzer empfehlen, seine Joystickprogrammierung in BASIC vorzunehmen.

FAIRCHILD JOYSTICKs

Wenn Fairchild Joysticks geliefert werden,(s. Bild), gelten die folgenden Bedingungen:

Wenn der Jostick nicht bewegt wird gilt: N = 255

a) Joystick, hochgezogen = 255–1 (eins von 255 abziehen)
b) Joystick, nach Norden = 255–2
c) Joystick, nach Süden = 255–4
d) Joystick, nach West = 255–8
e) Joystick nach Osten = 255–16
f) Joystick nach unten gedrückt = 255–32
g) Joystick n. links gedr. = 255–64
h) Joystick n. rechts gedr.= 255–128

Wenn zwei dieser Bewegungen gleichzeitig durchgeführt werden, addieren sich die Abzugszahlen. Z. B., wenn der Joystick gleichzeitig gedrückt, nach Norden geschoben und noch nach links gedreht wird, ergibt sich ein Wert:

N = 255 –2 (Norden) –64 (links drehen)
N = 255 –66 = 189

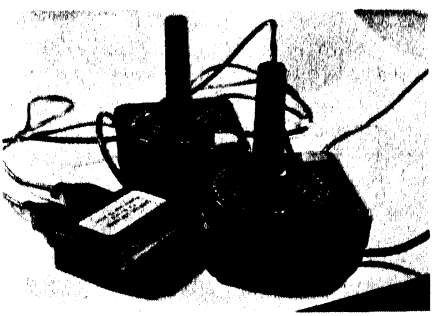
Dual-Joysticks für den PET

Dual Joystick

Doppelte Joysticks für PET
In einer der letzten Ausgaben haben wir Ihnen ausführlich einen Steuerknüppel für den PET beschrieben. Heute wollen wir Ihnen zwei Steuerknüppelsysteme vorstellen, bei denen mit zwei Joysticks gleichzeitig gearbeitet werden kann.
Das erste Doppelsystem besteht aus zwei ATARI-Joysticks. (Wie bereits in ELCOMP 2/78 besprochen)

Das System (Bild 1) hat einen gemeinsamen Adapter, der in den parallelen USER PORT des PETs eingesteckt wird.

Der rechte Steuerknüppel wird rechts und der linke Steuerknüppel links mit dem Interface verbunden (PET von vorne gesehen)
Mit diesen beiden Joysticks können Sie interessante Spiele gegen einen anderen Partner oder einzeln, oder gemeinsam gegen den Computer spielen. Wie man einen solchen Dual-Joystick programmiert, wollen wir Ihnen im folgenden Artikel zeigen.



Für jeden Steuerknüppel (rechts und links) kann der PET folgende Richtungen abtasten: Norden, Süden, Osten, Westen, Nordosten, Nordwesten, Südosten, Südwesten. Zusätzlich nimmt der PET eine Abtastung beim Drücken des roten Feuer-Knopfes von jedem der beiden Steuerknüppel vor. Wenn jedoch der rote Feuer-Knopf bei beiden Steuerknüppeln gedrückt wird, so verliert der PET die Fähigkeit, entweder die Ost- oder die Westrichtungen abzutasten. Zuerst ist jedoch die Erklärung nötig, wie der PET einige Bewegungsrichtungen abtasten kann. Für jedes Programm, bei welchem die Steuerknüppel verwendet werden, sollte immer die Anweisung: POKE 59459,0 zur Initialisierung der Parallelen E/A-"Ports" benutzt werden. Nachdem dies getan ist, wird der PET nun dazu befähigt, die Bewegungen der Steuerknüppel mit Hilfe einer einfachen Subtraktion abzutasten. Zur Darstellung des Geschehens sollte das Programm die parallelen E/A-Ports mit der folgenden Anweisung: N = PEEK (59471) auslesen. Der Wert von: N enthält jetzt die Bewegungen der Steuerknüppel. Wie kommt der Wert von N zustande?

Indem die zugehörige Zahl von 255 subtrahiert wird; das Programm weiß, „ was die Steuerknüppel machen"! Der Wert von 255 für „N" bedeutet, daß die Steuerknüppel noch nicht benutzt wurden. Einige Bewegungen der Steuerknüppel werden diesen Wert wie folgt verändern:

Wenn der rechte Steuerknüppel nach NORDEN bewegt wird, so müssen Sie eine 1 vom Wert N abziehen.
Wenn der rechte Steuerknüppel nach SÜDEN bewegt wird, so müssen Sie eine 2 vom Wert N abziehen.
Wenn der rechte Steuerknüppel nach WESTEN bewegt wird, so müssen Sie eine 4 vom Wert N abziehen.

Wenn der rechte Steuerknüppel nach OSTEN bewegt wird, so müssen Sie eine 8 vom Wert von N abziehen.
Wenn der linke Steuerknüppel nach NORDEN bewegt wird, so müssen Sie eine 16 vom Wert N abziehen.
Wenn der linke Steuerknüppel nach SÜDEN bewegt wird, so müssen Sie eine 32 vom Wert N abziehen.
Wenn der linke Steuerknüppel nach WESTEN bewegt wird, so müssen Sie eine 64 vom Wert N abziehen.

Wenn der linke Steuerknüppel nach OSTEN bewegt wird, so müssen Sie eine 128 vom Wert N abziehen.
Damit dies überprüft werden kann, müssen Sie die Schnittstelle anschließen und dann das folgende Programm schreiben:

```
10 POKE 59459,0
15 N = PEEK (59471)
20 PRINT N
25 GOTO 15
```

Als nächstes verbinden Sie die Steuerknüppel und geben RUN. Sie sollten dann eine Zahlenkolonne mit der Zahl 255 (der laufende Wert von „N") sehen, welche auf der linken Seite des PET-Bildschirmes von oben nach unten durchläuft. Nehmen Sie jetzt den rechten Steuerknüppel und bewegen Sie ihn nach Norden. Die Zahlenreihe ändert ihren Wert auf 254, (d. h. 1 wird vom laufenden Wert „N" abgezogen). Als nächstes bewegen Sie den rechten Steuerknüppel nach Süden und beobachten dabei die Zahlenkollonne, welche nun auf den Wert 253 wechselt. Wenn Sie alle aufgeführten Bewegungen versucht haben, sollte es Ihnen möglich sein, die „Antwort" auf jede Bewegung des Steuerknüppels zu bestätigen.

Stoppen Sie das Programm jetzt noch nicht! Versuchen Sie beide Steuerknüppel, den linken und den rechten, gleichzeitig nach Norden zu drücken. Der angezeigte Wert sollte 238 sein, da Sie 1 für den rechten Steuerknüppel und 16 für den linken Steuerknüppel abziehen müssen, so daß N = 255 minus 1, minus 16 = 238 ist. Sie sollten dies mit verschiedenen Steuerknüppel-Bewegungskombinationen versuchen, damit Sie mit den Regeln der Berechnung für den Wert von „N" vollkommen vertraut werden.

Während das Programm noch weiterläuft, drükken Sie nun auf den roten Feuerknopf auf dem rechten Steuerknüppel. Der angezeigte Wert sollte 243 sein. Beachten Sie, daß dieses der gleiche Wert wäre, als würden Sie den rechten Steuerknüppel gleichzeitig nach Westen und nach Osten bewegen. Da der Steuerknüppel jedoch nicht gleichzeitig nach beiden Seiten, Westen und Osten bewegt werden kann, weiß das Programm, daß der „Feuer-Knopf" gedrückt wurde. Nehmen Sie bitte zur Kenntnis, daß das Programm irgendeine dieser beiden Richtungen nicht länger abtasten kann, während der Feuer-Knopf gedrückt wird.

Um dies zu veranschaulichen, müssen Sie den Feuer-Knopf gedrückt halten und den Steuerknüppel, nach Osten bewegen; der Wert von „N" wird sich dabei nicht verändern. Gleicherweise, wenn Sie den Steuerknüppel nach Westen bewegen, ändert sich der Wert von „N" noch immer nicht.

Angenommen das Programm läuft noch, so drücken Sie den Feuer-Knopf auf dem linken Steuerknüppel, der Wert von „N" wird nun mit 63 angezeigt. Der linke Steuerknüppel-Kopf wirkt in einer ähnlichen Weise wie der des rechten Steuerknüppels. Nämlich das Drücken des Knopfes auf dem linken Steuerknüppel hat natürlich dieselbe Wirkung, als würde der Steuerknüppel gleichzeitig nach Osten und Westen bewegt werden.

Auch hierbei „bemerkt" das Programm, daß der Feuer-Knopf des linken Steuerknüppels gedrückt wurde, da diese Bewegung unmöglich ist.

Zusammenfassend: Ein Programm kann bestimmen, ob der Feuer-Knopf auf einem der beiden Steuerknüppel gedrückt wurde, indem es berechnet, ob sich der Steuerknüppel gleichzeitig in beide Richtungen Ost und West zu bewegen scheint. Ist dies der Fall, so wurde der Knopf gedrückt.

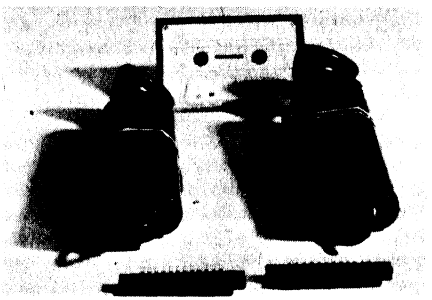
Während das Programm läuft, sollten Sie versuchen, den rechten Steuerknüppel in die nordöstliche Richtung zu bewegen. Der angezeigte Wert von „N" sollte dabei 246 sein. Dies entspricht dem Fall, in welchem beide Richtungen Norden (1 abziehen) und Osten (8 abziehen) gleichzeitig auftreten. Sie sollten alle diagonalen Richtungen (Nordost, Nordwest, Südwest und Südost) ausprobieren, damit Sie wissen, wie die Werte von „N" berechnet werden müssen, wenn die Steuerknüppel auf diese Art bewegt werden.

Ob Sie es glauben oder nicht, dies ist alles, was zur Programmierung der Schnittstelle mit zwei Steuerknüppeln erforderlich ist. Genau so wie bei den anderen Fähigkeiten des PETs wird es eine Weile dauern, bis Sie mit den Einzelheiten der Programmierung der beiden Steuerknüppel völlig vertraut sind. Wenn Sie jedoch einmal gelernt haben, werden Sie in der Lage sein, zusätzliche Spiele und Programme, welche Sie bereits besitzen, lustiger und interessanter zu gestalten. Zum Beispiel können Sie die Schnittstelle in einem Programm verwenden, welches erfordert, daß Sie einzelne Tasten zur Bestimmung der Bewegung einer Figur oder graphischer Zeichen benutzen sollen. Versuchen Sie es mit einigen der Programme, welche in den von Commodore an Sie gelieferten Mitteilungen für den PET enthalten sind.

Ein weiteres Dual-Joystick System

Neben den bekannten ATARI-Joysticks gibt es heute noch ein Dual-Joysticksystem mit dem Steuerknüppel von Fairchild. (Fairchild-Video-Spiele) (Bild 2)

Dieses System wird als Bausatz geliefert. Es besteht aus zwei Steuerknüppeln, zwei Steckern und einer Cassette mit drei Spielprogrammen und einem Utility-Programm für die Joysticks. Das Utilityprogramm kann Ihnen bei der Eigenentwicklung von Programmen später gute Dienste erweisen.



Zusammenbau der Steuerknüppel:

Sie benötigen dazu eine Drahtabziehzange, einen Lötkolben und normales Lötzinn. Verbinden Sie die farbig gekennzeichneten Drähte, welche vom Steuerknüppel kommen sorgfältig mit den mit Buchstaben bezeichneten Stiften des Steckers, wie unten gezeigt. Achten Sie sorgfältig darauf, daß keine unerwünschten Verbindungen durch verstreutes Lötzinn entstehen können. Die farbig gekennzeichneten Drähte des Steuerknüppels sind wie folgt mit den mit Buchstaben bezeichneten Stiften zu verbinden:

1. Steuerknüppel	Paralleler „Port"
Farbe d. Drahtes	Buchstabe auf dem Stecker für den USER PORT
Weiß	A
Schwarz	C
Gelb	D
Grün	E
Blau	F
Grau	H
Braun	J
Orange	K
Rot	L
TON–EINGANG (frei wählbar)	M
TON–MASSE(Erde)	N

Zwischen den Anschlüssen N und M kann ein Verstärkereingang angeschlossen werden. Bei einigen mitgelieferten Programmen können dann die Spiele mit Toneffekten untermalt werden.

2. Steuerknüppel	IEEE488 Bus
Farbe des Drahtes	Code des Stiftes für den IEE488 Stecker
Schwarz	1
Gelb	2
Grün	3
Blau	4
Grau	A
Braun	B
Orange	C
Rot	D
Weiß	E

Beachten Sie bitte wieder: Der Stecker mit den zwölf Stiften hat zwei parallele Stift-Reihen: Die obere Reihe ist mit Zahlen und die untere Reihe ist mit Buchstaben bezeichnet. Wenn Sie

das Gerät mit Ihrem PET verbinden, so müssen Sie darauf achten, daß sich die Reihe mit den nummerierten Stiften oben und die mit Buchstaben bezeichneten Stifte unten befinden. Eine gute Idee ist es, wenn Sie die Oberseite des Steckers, als Hinweis für Sie mit „Oben" kennzeichnen.

Anschluß an Ihren PET
Nach dem Zusammenbau wird der Stecker mit den 12 Stiften in ihren PET gesteckt und dort mit den parallelen Anwender- und IEEE488 „Ports", welche sich auf der Rückseite der Maschine befinden, verbunden. Im Hinblick auf die Rückseite der Maschine, werden Sie den mittleren der drei Ports, welchen Sie sehen können, für den ersten Steuerknüppel und den äußerst rechts liegenden Stecker für den zweiten Steuerknüppel benutzen.

Schalten Sie den PET aus, bevor Sie den Steuerknüppel an ihn anschließen! Sie sollten niemals ein Peripherie-Gerät mit Ihrem Computer verbinden, solange dieser eingeschaltet ist, damit vermeiden Sie, daß an der Maschine ein dauerhafter Schaden entsteht.

Ausführung der Spiele:
Alle Spielmuster erklären sich von selbst. Damit das Programm abläuft, müssen Sie die „SHIFT"-Taste nach unten halten und auf „RUN/STOP" drücken. Das Programm, welches zuerst geladen wird, nennt sich „LOGO". Es erklärt, wie das nachfolgende Programm ablaufen soll. Als Schutz gegen das Urheberrechts-Patent wurden die Programme so entworfen, daß der „LIST"-Befehl nicht normal arbeitet.

Wenn Sie die LIST-Routine ausprobieren und Ihr Bildschirm Ihnen eine unerwartete Antwort gibt, so denken Sie daran, daß sich Ihre Maschine im richtigen Arbeitszustand befindet.

Prüfung Ihrer Steuerknüppel

Laden Sie das Programm für den Steuerknüppel in der Original-Ausführung. Überprüfen Sie jede der acht Richtungen (rückwärts, vorwärts, rechts, links, nach oben ziehen, nach unten drükken, nach links drehen und nach rechts drehen) und der PET wird entsprechend darauf antworten. Beachten Sie, daß der Steuerknüppel in vier Richtungen gleichzeitig bewegt werden kann.

Schreiben Sie Ihre eigenen Programme:
Wenn Sie die Utilityprogramme von Microtronix verwenden, können Sie Ihre eigenen Programme für den Steuerknüppel entwickeln.

Befolgen Sie dabei diese Schritte:

1.) Für das Spiel mit einem Steuerknüppel müssen Sie die Zeilen von 1040 bis 2000 löschen. Dann gehen Sie zu Schritt 4 über.

2,) Für das Spiel mit zwei Steuerknüppel und zwei abwechselnden Spielern müssen die Zeilen 1040, 1050 und 1080 bis 2000 gelöscht werden. Anschließend machen Sie mit Schritt 4 weiter.

3.) Zur Auswahl des Steuerknüppels für den Anwender setzen Sie die Variable U1 auf 0 (Null) für den linken oder auf 1 für den rechten Steuerknüppel. (Diese Wahl wird unter Programm-Kontrolle ausgeführt) und danach löschen Sie nur die Zeilen 1060 und 1070.

4.) Geben Sie Ihren Programm-Anfang mit der Zeilennummer 1101 oder einer höheren Nummer ein.

5.) Beenden Sie ihr Programm mit „GOTO 1030". Zur schnelleren Beantwortung können Sie die Verzögerungs-Schleife in der Anweisung 1030 löschen.

6.) Ersetzen Sie die Anweisungen 110, 210 u. s. w. durch die Programm-Kalkulationen, Ausdrucken u. s. w., falls Sie sehen wollen, wenn der Steuerknüppel in der durch PRINT-Anweisung angezeigten Richtung bewegt wird. Zur Erhöhung der Reaktion des Steuerknüppels für diejenigen Fälle, in welchen Ihre Berechnungen länger als ein oder zwei Codezeilen sind, müssen

Sie eine „GOSUB"–Anweisung bei 110, 210 usw. eingeben, damit ein Sprung in Ihr Programm und die Rückkehr daraus ermöglicht wird. Falls Sie eine weitere Erklärung benötigen, schauen Sie am besten in einem Buch über „BASIC" nach. Setzen Sie keine Anweisung ein, deren Nummerierung kleiner als 50 ist!

Programme für Haushalt und private Finanzen

Nachfolgend geben wir Ihnen einige Beschreibungen von sehr leistungsfähigen Programmen für Haushalt und private Finanzen. Die Programme sind sehr sehr lang und werden deshalb auf Cassette geliefert. Sie erhalten sie im Fach- und Buchhandel unter folgenden Bestellnummern:

1. Haushaltsfinanzprogramme Teil 1 und Teil 2 auf einer Cassette
Best.-Nr. P11/P22 **DM 138,--**

2. Haushaltsutility I, Best.-Nr. P9 **DM 29,--**
3. Haushaltsutility II, Best.-Nr. P10 **DM 29,--**
4. Math Practice, Best.-Nr. P60 **DM 19,80**

Die Haushalts-Finanz-Programme Teil 1 und 2 sind zur allgemeinen Verwendung für die Aufzeichnung und Untersuchung von Ausgaben und Einkommen im Haushalt entwickelt worden. Teil: 1, "Aufzeichnung von Einkommen und Ausgaben", wird zur Eingabe und Auflistung von Haushaltsplan-Posten verwendet. Sie liefert außerdem eine Band-Aufnahme aller Posten. Teil 2, "Untersuchung finanzieller Anlagen" faßt die Datenaufnahmen von Teil 1 zusammen und stellt die gesamten Ausgaben-Anlagen dar. Die Namen der Programme auf dem Band lauten: "HF1" und "HF2".

Teil 1: Aufzeichnung von Einkommen und Ausgaben
Beschreibung und Anleitungen.
Setzen Sie die Cassette in das Bandgerät ein, spulen Sie diese zurück und schreiben sie „LOAD". Wenn das Programm „geladen" ist, so schreiben Sie RUN. Nachdem die Einleitungs-Anzeige etwa fünf Sekunden gelaufen ist, wird das Programm sechzehn Kategorien aufzeigen und Sie nach dem Monat und dem Jahr fragen, für welches Sie die Posten aufnehmen wollen. Die Antwort sollte in der folgenden Form erfolgen "JANUAR, 1979". Das Komma ist dabei wichtig. Das Programm fragt Sie anschließend, ob Sie eine neue, monatliche Aufzeichnung machen; wenn Sie eine alte Aufnahme auf den neuesten Stand bringen wollen, so antworten Sie mit „N" auf diese Frage.

Ihre alten Aufnahmen werden dann automatisch eingelesen, um auf den neuesten Stand gebracht zu werden. Der Rest von Teil 1 erklärt sich im allgemeinen von selbst. Bei der Verwendung des Programms ist jedoch eine gewisse Vorsicht erforderlich. Von großer Wichtigkeit ist dabei, daß man für jede der monatlichen Aufzeichnungen eine getrennte Cassette verwendet. Da der PET keine eingebaute Datei-Zähleinrichtung besitzt, beginnt er einfach mit dem Schreiben, wenn er angewiesen wird, eine Datei zu schreiben, obwohl das Band eine alte Information enthalten könnte. Dies funktioniert, wenn Sie eine alte Datei auf diesem Band auf den neuesten Stand bringen wollen. Wenn Sie jedoch mehr als eine Datei auf diesem Band haben, können Sie dieses überschreiben. Es ist möglich, ein BASIC-Programm zu schreiben, mit dem Sie die Dateien zählen können, jedoch würde hierzu wertvoller Speicherplatz benötigt werden. Wenn vom Programm die Anweisung kommt, die Cassette einzulesen, so überzeugen Sie sich, daß sie auch zurückgespult ist, bevor Sie mit der nächsten Schreib-Stufe auf dem Band weiterfahren. Das Band muß auch völlig zurückgespult sein, bevor es abgelesen wird, da sonst Lesefehler auftreten könnten.

Folgende Vorsichtsmaßnahmen sind bei der Benutzung dieses Bandes zu befolgen:

1. Um den Inhalt des Speichers zu sichern, ist die Beschreibung auf 29 Zeichen beschränkt (einschließlich der Zwischenräume). Zeichen, die über diese Anzahl hinausgehen, werden ausgelassen! Außerdem fügen Sie keine Kommas in die Beschreibung ein, denn dies Bedeutet für den Computer, daß sich die Beschreibung nur bis zum ersten Komma erstreckt und er ignoriert dann den Rest.
2. Verwenden Sie die Kommas immer so, wie sie im Programm angezeigt werden, da Sie sonst nicht das schreiben können, was sie wollen.
3. Der PET besitzt eine begrenzte Speicherfähigkeit. Wenn Ihre Beschreibungen dazu neigen, zu lange zu werden, so können Sie nur 40–50 Posten eingeben, bevor die Kapazität des Speichers überschritten wird. Sie erhalten eine Mitteilung, wenn die Speicherkapazität ausgelastet ist.

Falls Sie diese Mitteilung jedoch nicht beachten und weiterschreiben, stoppt das Programm und Sie werden die Posten wahrscheinlich neu eingeben müssen. Sie können maximal 100 Posten zur glei-

chen Zeit eingeben. Allerdings können Sie pro Monat mehrere Bänder verwenden. Teil 2 läßt diese Tatsache gelten und zieht dies während der Ausführung seiner Berechnungen in Rechenschaft. Die Namen der Kategorien können zur Anpassung Ihrer personellen Erfordernisse umgeändert werden. Es wird jedoch empfohlen, daß Sie die Kategorie I „INCOME" (Einkommen) nicht abändern, da in Teil 2 bestimmte Berechnungen vorkommen, in welchen diese Kategorie nicht benutzt wird. Die Zeilen 1020 – 1030 enthalten die Kategorien und ihre Buchstaben;Kennzeichnungen. Wenn Sie diese Zeilen ändern, so müssen Sie auch die Zeilen 105-107 in Teil 2 für Ihre neue Liste abändern.

Teil 2: Untersuchung finanzieller Anlagen
Beschreibung und Anleitungen.
Setzen Sie die Cassette in das Bandgerät ein, spulen sie zurück und schreiben LOAD HF2. Nachdem das Programm geladen ist, schreiben Sie RUN. Genauso wie in Teil 1 fordert Sie das Programm auf, den Monat und das Jahr einzugeben, für welche Sie die Zusammenfassung wünschen. Vergessen Sie dabei nicht, den gleichen Monatsnamen zu benutzen, wie in Teil 1. Z. B. können Sie im Teil 1 nicht die Abkürzung „JAN" verwenden und dann im Teil 2 „JANUAR" ausschreiben. Sie können die Monate in einer Reihenfolge eingeben. Wenn Sie zwei Bänder für einen Monat besitzen, ist es nicht nötig, diese aufeinanderfolgend einzugeben. Außerdem können Sie jederzeit einen alten Monat überprüfen oder sich ein Jahr zur Daten-Zusammenstellung ansehen.

Beachten Sie, daß bei dem Versuch, den gleichen Monat für zwei verschiedene Jahre einzugeben (z.B. April 1978 und April 1979) das Programm die Jahre ignoriert und die beiden April-Aufzeichnungen, als eine einzige Aufzeichnung zusammenfaßt.

Der Rest des Programms ist einfach und erklärt sich von selbst. Vergessen Sie nicht, die Kommas so zu setzen, wie sie im Programm angegeben sind.

Achtung: Wenn Sie zufällig ein Programm ausgelassen haben und dieses neu eingeben wollen, ohne daß dabei Ihre schon vorher eingegebenen Daten zerstört werden, so ist dies möglich. Für „HF1" schrei-

ben Sie einfach GOTO 470 und für „HF2" schreiben Sie GOTO 245. Beide Programme sind auf einer Band-Cassette enthalten und entsprechend mit „HF1" und „HF2" bezeichnet.
VERWENDEN SIE NUR GROSSBUCHSTABEN!

Mathematische Praktiken
aus Creative Software. Das Programm „Math. Practice" wurde für Kinder im Alter von ca. 8 — 12 Jahren entwickelt, um ihnen bei den Routine-Aufgaben und dem Auswendiglernen von Multiplikations- und Divisions-Tabellen behilflich zu sein. Es übt auf das Kind eine sofortige Rückwirkung in Form von ermutigenden Mitteilungen aus, — beglückwünschend, wenn das Problem richtig gelöst wurde und leicht anspornend, wenn das Problem falsch gelöst wurde. Ein ausgelassenes Problem wird zu einem späteren Zeitpunkt wiederholt, um dem Kind mehr Praxis für jene Zahlen zu geben und zu vermitteln, die es als schwierig empfindet. Eine laufende Benotung, sowie falsche Antworten werden auf der rechten Seite festgehalten, wonach das Kind jederzeit anfragen kann. Die Grenzwerte von den Divisoren und Multiplikatoren müssen festgelegt werden, bevor das Programm mit der Ausgabe der Probleme beginnt. Dadurch ist eine konzentrierte Problemstellung für dasjenige Gebiet gegeben, für welches das Kind die meiste Arbeit benötigt. Es kann eine Gesamtzahl von 100 Problemen gegeben werden. Am Ende der Praxis wird eine Abschlußnote gegeben und der Computer liefert eine kurze Bewertung dazu, wie das Kind die Aufgabe gelöst hat.

Wenn das Kind alle Aufgaben richtig gelöst hat, erscheint eine spezielle, spiralförmige Graphik in der Anzeige. Um die Arbeit zu intensivieren, ist der Name des Kindes am Anfang des Programmes einzugeben und während des gesamten Programmes konstant zu verwenden.

Dadurch wird die gesamte Praxis mehr persönlicher gehalten und macht damit dem Kind auch mehr Spaß, im Gegensatz zu dem langweiligen Drill des Auswendiglernens, welcher gewöhnlich zur Erlernung von Multiplikations- und Divisions-Tabellen erforderlich ist.

Wenn das Programm zufällig ausgelassen wird, (z.B. wenn das Programm zurückgespult wird, ohne daß zuerst die Antwort auf das

Problem eingegeben wurde), kann dies an dem Punkt wieder neu eingegeben werden, bei dem es verloren ging. Dazu gibt man den Befehl GOTO 220 ein. Dies bringt den Programmablauf genau an den Punkt, bei welchem das Programm verloren ging, wobei die laufende Benotung sowie der Name des Kindes usw. bewahrt bleiben.

GAMEPAC 1 (Spiele-Paket)
Aus Creative Software
Das Spiele-Paket 2 enthält weitere fünf spannende Spiele, welche für den PET in BASIC geschrieben wurden.
Es sind:
1. Pfeilwerfen. In diesem Spiel können bis zu 20 Spieler Pfeile an eine Zielscheibe mit 10, 20, 30 und 40 Punktezonen werfen. Der Spieler, der zuerst 200 Punkte erreicht, hat gewonnen. Sie haben eine Auswahl von drei Wurfmethoden, jede davon hat eine andere Erfolgswahrscheinlichkeit. Probieren Sie alle aus, und prüfen Sie, welche Ihnen am besten zusagt!
2. NIMBLE (Auffassungsgabe). Machen Sie in diesem spannenden Zahlenspiel zum Spaß einen Wettstreit mit dem Computer.

Household Utility
1. Kaufen oder Mieten
Zum Laden stecken Sie die Cassette in das Bandgerät und schreiben „LOAD BUY OR RENT". Obwohl all diese Anleitungen im Programm selbst enthalten sind, wird vom Programm die folgende Information verwendet, welche beim Ablauf des Programms zur Verfügung stehen sollte:

1. Der Preis des Hauses, welches Sie für den Kauf in Betracht ziehen.
2. Der Prozentsatz oder der wirkliche Betrag, den Sie für das Haus hinterlegen.
3. Die Laufzeit (in Jahren) und die jährliche Prozentsatzrate, für die Beleihung der Hypothek.
4. Die Vermögenssteuer für das Haus.
5. Ihren ungefähren Einkommenssteuersatz (in Prozent). Wenn Sie diese Posten jedoch nicht alle wissen, so erstellt das Programm vernünftige Vermutungen an und setzt die entsprechenden Werte ein.

Das Programm stellt eine monatliche Kostenrechnung für den Besitz eines Hauses auf und verwendet die Inflations-Rate zur Berich-

tigung dieses Preises. Sie können die Inflationsrate selbst berücksichtigen, ansonsten nimmt das Programm eine Inflationsrate von 8 % pro Jahr an. Gewöhnlich werden durch diese Berechnung die Kosten für den Besitz eines Hauses sehr gering und die Endkosten pro Monat werden Sie ohne Zweifel überraschen. Sie sollten dabei jedoch bedenken, daß die zusätzliche Schätzung aufgrund der Inflation nur einen Gewinn auf dem Papier darstellt, solange Sie nicht verkaufen oder ein zweites Darlehen auf das Haus aufnehmen. Somit berechnet das Programm auch die monatlichen Kosten, welche nur auf den zusätzlichen Abzügen beruhen. Der Eigentum eines Hauses ermöglicht es Ihnen, Ihre Einkommenssteuer zurückzufordern. Im Hinblick auf den Zahlenablauf ist diese letzte „Form", welche auf einer Kauf- oder Nichtkauf-Entscheidung basiert, wirklichkeitsnaher. Das Programm weist auch auf den Betrag der verlorenen Zinsen (6% pro Jahr) bei Barzahlung hin. Bei der Berechnung der monatlichen Kosten wird dieser Betrag jedoch nicht verwendet.

2. Darlehen

Zum Laden geben Sie „LOAD LOANS" ein. Auch hier sind wieder alle Anweisungen im Programm selbst enthalten.
Damit Sie dieses Programm verwenden können, müssen Sie zwei von den drei folgenden Posten bereithalten:

1. Die Laufzeit (in Monaten) des Darlehens
2. Den Betrag, welchen Sie ausleihen wollen.
3. Den Betrag, den Sie sich zur monatlichen Rückzahlung leisten können

Zusätzlich werden Sie immer die jährliche Prozentsatzrate des Darlehens angeben müssen.

Wenn Sie keinen genauen Wert haben, so nimmt das Programm 10 % pro Jahr an.

3. KALENDER

Zum Laden der Cassette geben Sie „LOAD CALENDAR" ein. Befolgen Sie die in dem Programm enthaltenen Anweisungen. Für die Zeit vom 1. März 1900 bis zum 1. März 2100 berechnet das Programm folgendes:

1. Den Wochentag, welcher zu einem eingegebenen Datum gehört
2. Das zukünftige Datum für ein gegebenes früheres Datum und die Anzahl der Tage bis zu diesem zukünftigen Datum.
3. Das frühere Datum zu einem gegebenen Datum und die Anzahl der Tage bis zu diesem früheren Datum.
4. die Anzahl der Tage zwischen irgenwelchen zwei Daten.

Beachten Sie, daß die Daten in der folgenden Form eingegeben werden müssen:

Month (Monat) , Day (Tag), Year (Jahr).

Die Kommas sind dabei wichtig! Das Programm prüft nur die ersten drei Buchstaben des Monats, so daß sie evtl. einen Monatsnamen auf die ersten drei Buchstaben abkürzen können.

Household Utility 2
Dieses Paket enthält drei Programme:
1. Zinseszins
2. Tilgung
3. Kosten für einen Wagen

Obwohl jedes Programm die gesamte Anweisung innerhalb des Hauptprogrammes selbst enthält, wird eine allgemeine Information, wie diese Programme anzuwenden sind, von Nutzen sein.

Das Programm Nr. 1 „Compount Interest (Zinseszinsen)" löst drei Problemarten für die Anlage eines Sparkontos.

Wenn Sie einen vorhandenen Betrag (der sich auf einem Sparkonto befindet) eingeben, so sagt Ihnen das Programm, wieviel Geld Sie an einem in der Zukunft liegenden Datum haben werden. (genannt: der Zukunftswert des Kontos). Es berechnet auch, wieviel Sie auf Ihrem Konto benötigen, damit Sie einen speziellen Zukunftswert erreichen.

Schließlich berechnet das Programm noch, wie lange Sie einen vorhandenen Betrag auf einem Sparkonto belassen müssen, damit dieser auf einen speziellen Zukunftswert ansteigt. Damit all diese Berechnungen ausgeführt werden können, müssen Sie dem Programm die

jährliche Prozentrate des Sparkontos angeben. Das Programm fordert diese Information zusammen mit den Daten des Jahres, in welchem der Zins fällig ist an, sobald sie benötigt wird.

(Bei den meisten Sparkassen ist der Zins täglich fällig, oder 365 mal pro Jahr)

Um das Programm zu laden, geben Sie „LOAD COMPOUND INTEREST" ein. Nachdem das Programm geladen ist, schreiben Sie RUN. Die meisten Informationen, welche das Programm von Ihnen anfordert, sollten eingegeben und danach die Return-Taste gedrückt werden. Einige Fragen, die mit ja oder nein beantwortet werden müssen, können mit der Eingabe von Y oder N beantwortet werden. Zur Beantwortung dieser Fragen darf die Return-Taste **nicht** gedrückt werden, da sonst der Programmablauf gestoppt wird und Sie erneut anfangen müssen. Außerdem sollte die jährliche Zinsrate besser in Prozent, anstelle der gleichwertigen Dezimalzahl eingegeben werden.
z.B.: 5,25 anstelle von 0,0525

Das Programm Nr. 2 **Amortization (Tilgung)** druckt auf dem Bildschirm eine Tabelle für die gewöhnlichen Kosten aus, welche im Zusammenhang mit einem Hypotheken-Darlehen entstehen. Das Programm wird den laufenden Betrag für die bezahlten Zinsen, das bezahlte Kapital, den Gesamtbetrag der bezahlten Zinsen und den verbleibenden Kontostand des Darlehens gesondert berechnen. Es wird diese Werte für einige Zahlungen auf dem Schirm anzeigen (z.B. von der 3. bis zur 8. Zahlung). Falls Sie die Zahlungen am Ende der Laufzeit wissen wollen, so kann das Programm eine Weile dazu benötigen, diese Zahlen zu berechnen. Wenn Sie z.B. diese Werte von den letzten zehn Zahlungen für eine 30-jährige Haus-Hypothek (Zahlungsnummer 351-360) wissen wollen, so benötigt das Programm etwa 25 Sekunden, um diese Zahlungen zu berechnen. Seien Sie deshalb nicht ungeduldig, wenn Sie hohe Zahlungsnummern eingeben. Ebenso, wenn der Betrag des Darlehens die Zahlungen pro Monat u.s.w., sehr groß sind, so werden die Werte in der Tabelle auf mehr als nur einer Zeile auf dem Schirm des PET zusammengestellt. Obgleich die Berechnungen auch weiterhin korrekt sind, werden die richtigen Zahlen etwas schwieriger abzulesen sein, da die Tabelle für kleinere Darlehenspreise nicht

mehr übersichtlich genug ist. Dies tritt nur bei Darlehensbeträgen von über 100.000,-- DM und monatlichen Zahlungen über DM 1.000,-- ein.

Zum Laden des Programms müssen Sie „LOAD AMORTIZATION" schreiben. Nachdem das Programm geladen ist, geben Sie RUN ein. Das Programm wird Sie nach allen speziellen Informationen fragen, welche es benötigt.
Das Programm Nr. 3 **Car Costs (Kosten für ein Auto)** berechnet die Kosten pro Meile, welche beim Fahren eines Autos entstehen. Bevor Sie das Programm verwenden, ist es nützlich, daß Sie die folgenden Informationen bereithalten.

1. Den Betrag, welchen Sie für den Wagen bezahlt haben.
2. Alle Gebühren und Steuern für die Zeit, in welcher Sie den Wagen besessen haben
3. Die Routineinstandsetzungs- und Reparaturkosten
4. Die Versicherungskosten
5. Die Anzahl der gefahrenen Meilen, seitdem Sie den Wagen gekauft haben
6. Den Betrag, welchen Sie für Benzin ausgegeben haben und
7. Den annähernden Wiederverkaufswert.

Das Programm kann Ihnen bei der Berechnung der Posten 6 und 7 behilflich sein. Es stützt sich dabei darauf, wie weit der Wagen gefahren wurde und wie alt der Wagen ist. Die Posten 1 – 5 müssen jedoch zur Eingabe in das Programm von Ihnen selbst berechnet oder geschätzt werden.

Um die Cassette zu laden, stecken Sie diese in das Bandgerät und schreiben dann „LOAD CAR COSTS". Nach dem Laden des Programms geben Sie RUN ein. Das Programm fragt nach allen Informationen, welche es benötigt und wird Sie über jene Posten informieren, bei welchen Ihre Schätzungen helfen können.

Musik mit dem PET

PET-Computer spielt Akkorde

Bis zu vier Tönen gleichzeitig kann man mit der nachfolgenden Zusatzplatine und dem PET-Computer erzeugen. Diese Zusatzeinheit ist nichts anderes als ein einfacher 8 Bit Digital/Analog-Wandler (MUP). Die zugehörige Software ist ein Unterprogramm, welches die eingegebenen Noten (1 – 4 gleichzeitig) und den gewünschten Notenwert über den Ausgangsport angibt. Der Digital-Analogwandler setzt dann die Impulse und deren Änderungen in zugehörige Töne um.

Was brauchen Sie, um diese Platine anschließen zu können?

Als erstes brauchen wir natürlich einmal den Commodore PET-Computer sowie die Musikplatine für den PET (MUP). Weiterhin benötigen wir ein Anschlußkabel, einen Verstärker und einen Lautsprecher. Verstärker und Lautsprecher können auch Ihre Stereoanlage oder Ihr Rundfunkgerät etc. sein.

Anschluß der MUP-Platine

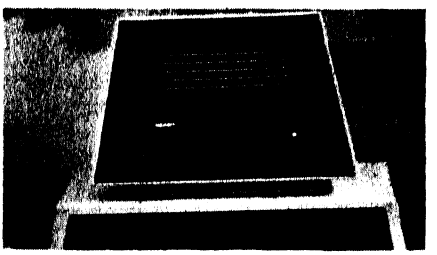
Sehen Sie sich zuerst einmal die Rückseite Ihres PETs an. Da sind zunächst einmal zwei Anschlußleisten mit 12 Polen und eine Anschlußleiste mit 6 Polen. Die Anschlußleiste mit den 6 Polen ist zum Anschluß eines zweiten Cassettenrecorders bestimmt. Gleich rechts daneben ist der 12polige Anschluß. Das ist der Steckanschluß für die Benutzer Ein-/Ausgabeschnittstelle. (User Port Connector) Die MUP-Platine wird auf diese beiden Anschlußleisten aufgesteckt und nutzt auf diese Weise die Stromversorgung für den zweiten Cassettenrecorder.

Vor dem Anstecken der Platine, zuerst den PET ausschalten. Nun kann die Zusatzplatine (Bauteileseite nach oben) auf die beiden Anschlüsse aufgesteckt werden. Jetzt kann das Kabel angeschlossen und die Verbindung zum Verstärker-Lautsprecher hergestellt werden.

Alles was jetzt noch zu tun ist, ist das Einlesen der Cassette, welche die nötige Betriebssoftware enthält. Diese Software ist soweit entwickelt, daß sie alle notwendigen Änderungen im Computer erledigt. Nach Einlesen des Programmes wird RUN eingegeben und es erscheint ein „MENUE".(Dies ist eine Zusammenstellung der möglichen Musikspielarten.)
Bis zu vier Töne können mit diesem Programm und der kleinen Zusatzplatine gleichzeitig generiert werden. (Akkorde)

1. Auf einer Notenlinie
2. Durch die Eingabe von Zahlen (jede Zahl entspricht einem bestimmten Ton).
3. Spielen von Zufallsnoten
4. Darstellung der Ausgangswellenform auf dem PET-Bildschirm
5. Spielen von Tönen über die Tastatur
6. Stop und Rückkehr

Nachdem die Auswahlliste auf dem Bildschirm erschienen ist, drücken Sie die Nummern Ihrer Wahl.
Beispiel:
Angenommen, Sie haben sich für Punt 1 entschieden.

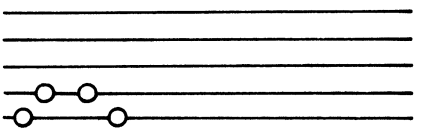


Dieser Programmteil zeichnet auf dem Bildsschirm eine fünfzeilige Notenlinie. Sie können

jetzt mit dem Cursor an die gewünschte Stelle fahren, wo Sie eine Note platzieren wollen. Drücken Sie nicht RETURN, bevor der Positionierungsvorgang und das Setzen aller Noten beendet ist.

0 = (Shift W) für eine ganze Note (nicht 0)
# für einen Halbton

Eine Note kann ausgelöscht werden, indem ein Leerzeichen darüber getippt wird. In vertikaler Richtung können bis zu vier Noten positioniert werden. Sie werden dann auch gleichzeitig gespielt.



Die horizontalen Linien auf dem PET-Bildschirm können ignoriert werden. Sie dienen nur zu Ihrer Orientierung. Wenn Sie beginnen wollen, die Noten zu spielen, drücken Sie die HOME-Taste. Achten Sie aber darauf, daß Sie nicht gleichzeitig SHIFT drücken. Dies würde den gesamten Bildschirm löschen.

Der Cursor muß auf RUN 5 stehen. Nun braucht nur noch die RETURN Taste gedrückt werden und die Musik beginnt. Das Programm spielt jede Note (bzw. Noten untereinander) von links nach rechts. Die gespielte Note wird durch einen Pfeil angezeigt. Wenn der Pfeil auf ein Asterik trifft, oder das rechte Bildende erreicht hat, beginnt das Spiel wieder von vorne.

Wenn Sie die gerade gespielte Musik ändern wollen, drücken Sie die STOP-Taste und benutzen Sie den Cursor für Ihre Änderungen. Ist die Änderung erfolgt, drücken Sie die HOME-Taste und anschließend RETURN. Dies können Sie so oft wiederholen, wie Sie wollen.

Um zum MENUE zurückzukehren, drücken Sie die STOP-Taste, löschen den Bildschirm CRL und geben RUN mit anschließendem RETURN ein.

2. Eingabe von Zahlen erzeugt Musik

Die Zahl 2 wird eingegeben. Jetzt können Sie Zahlen zwischen Null und Fünfzig eingeben. Je nachdem, wieviele Zahlen zwischen 1 und 50 Sie hintereinander eingeben, ohne diese durch ein Pluszeichen (+) zu trennen, werden diese Zahlen (entspr. Noten) auf einmal gespielt. Bei einstelligen Zahlen muß auf jeden Fall die 0 davor gesetzt werden. Um z. B. 3 Noten hintereinander zu spielen, müssen die Zahlen

22 + 24 + 26 eingegeben werden

Wird jedoch 22 24 26 hintereinander, ohne + dazwischen eingegeben, so werden diese drei Noten zusammen gespielt. Es können immer zwei Reihen auf dem Bildschirm eingegeben werden. Die Noten werden so lange gespielt, bis die STOP-Taste gedrückt wird.

Wollen Sie dies so abändern, daß nach Beendigung der zwei Zeilen die Tonfolge abgebrochen wird, brauchen Sie nur die Zeile 515 im Programm von THEN 510 in THEN RETURN umzuschreiben.

3. Spielen von Zufallsnoten

Einen sehr interessanten Effekt erreichen Sie durch das Spielen von Zufallsnoten. Hier ergeben sich Geräusche wie in Science Fiction Filmen und in Krimi-Untermalungen. Das Programm läuft so lange, bis die STOP-Taste gedrückt wird. Die Zahlentasten 1 – 9 bestimmen die Tondauer.

4. Darstellung der Ausgangswellenform auf dem Bildschirm

Die Wahl dieses Punktes aus dem MENUE erlaubt Ihnen das Betrachten der Wellenform, die der PET zur Musikerzeugung produziert. Die Signalform läuft auf dem Bildschirm von oben nach unten durch.

5. Spielen von Tönen über die Tastatur

Die Wahl dieses Punktes aus dem MENUE verwandelt Ihren PET in eine „elektronische Orgel". Das Drücken der Tasten A – Z bringt eine Reihe von ansteigenden Tönen. Über die Zahlentastatur 0 – 9 kann die Tonlänge programmiert werden. 0 = kurze Noten, 9 = lange Noten, 1 und 8 liegen dazwischen.

Programmier-Hinweise

Wenn Sie den Musik-Generator selbst programmieren wollen, soll Ihnen die nachfolgende Information etwas helfen.

Ein Assembler-Unterprogramm, welches Rechtecksignale am Pin 7 des parallelen Ein-/Ausgabeports erzeugt, kann durch SYS (832) aufgerufen werden. (832 dezimal = 340 Hex)

Die Tonhöhe (sie entspricht der Anzahl von Zyklen, die von einer Versorgungsschleife erzeugt werden) wird durch einen 16 Bit Zähler an den Plätzen 826 und 827 ein Programm erzeugt. Alle Nullen erzeugen einen hohen Ton, alle Zahlen über 2 an der Stelle 826 erzeugen einen sehr tiefen Ton. Grundsätzlich sollte man die Stelle 826 auf Null setzen und 827 sollte zwischen 20 und 255 verändert werden. Der Platz 825 enthält die Tondauer oder die Anzahl der Zyklen pro Note.

Beachten Sie auch, daß bei Erhöhen der Tonhöhe auch der Tondauerzähler erhöht werden muß.

Das Register, welches im PET die Betriebsrichtung für den Ein-/Ausgabe-Port festlegt, muß auf Ausgang gesetzt werden. Dies geschieht durch den Befehl

POKE 59459,255

Beispielprogramm: Spielen von 255 Noten

```
10  POKE 59459, 255
20  FOR I = 0 TO 255
30  POKE 827, I
40  SYS (832)
50  NEXT
```

Spielen von Zufallsmusik

```
10  POKE 59459, 255
20  FOR I = 1 TO 1000
30  POKE 827, RND(I)* 20 + 20
40  SYS (832)
50  NEXT
```

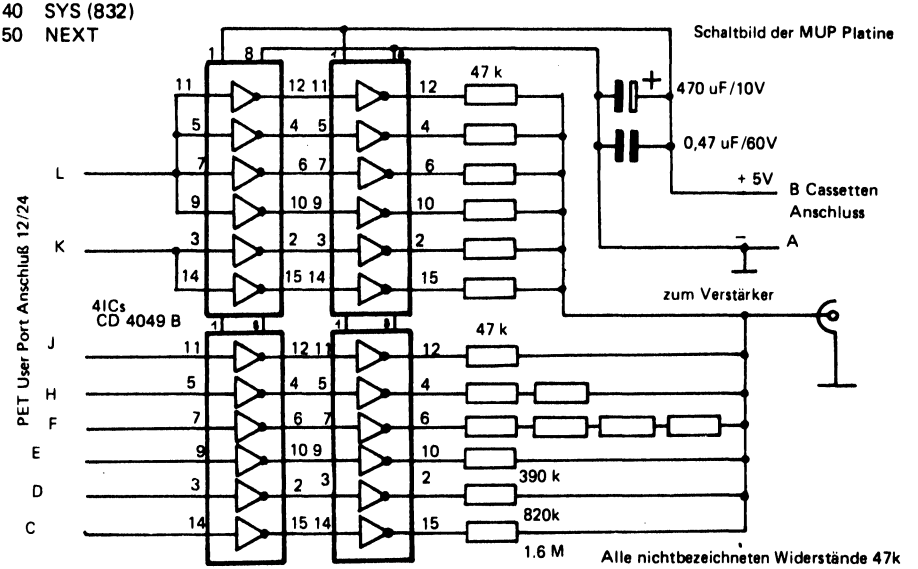
Um Noten ohne die Assembler-Routine zu spielen, geben Sie einfach Folgendes ein:

```
10  POKE 59459
20  FOR I = 1 TO 100 : POKE 59457, I:NEXT
```

Diese Routine POKED eine ansteigende Zahlenreihe von 1 bis 100 in den PET-Ausgabeport. Da jede Zahl den Status des PET invertiert, wird ein Ton generiert. Der Ton ist mehr ein Piepsen, da BASIC eine beträchtliche Zeit benötigt, um diese „Töne" zu produzieren.

Laden der Assembler-Unterroutine

Die Assembler-Routine wird über GOSUB 700 aus dem MAESTRO-Programm (Cassette zur Platine) geladen. Das Programm ist im zweiten Cassettenpuffer abgespeichert und ist dort recht gut geschützt.



PET MUSIC (Musikprogramme)

Das Programm ‚MUSIC' ist das Hauptprogramm. Es spielt Musikstücke in drei verschiedenen Versionen.

1. Nach Notenzeilen (Staff music)

Sie geben die Noten in eine Notenzeile ein und sagen dem PET, daß er sie spielen soll. Ein Pfeil unter den Notenzeilen zeigt auf die gerade zu spielende Note. Es kann nur eine Note zur gleichen Zeit gespielt werden. Auch hat sie immer die gleiche Zeitdauer.

2. Auf einem numerischen Tastenfeld (Numeric Notes)

Dies ermöglicht eine mehrfache Notendauer.

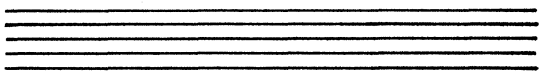
3. Zufallsnoten (RANDOM MUSIC)

Der PET spielt nun Noten entsprechend einem internen Zufallsgenerator. Die Geschwindigkeit richtet sich nach der zuletzt eingegebenen numerischen Taste.

Beschreibung der einzelnen Arten:

Zu 1.: Musik über Notenzeilen

A Geben Sie RUN 200 ein, um den Bildschirm zu löschen und zeichnen Sie eine neue Notenzeile, bestehend aus fünf einzelnen Zeilen.



B Geben Sie die Noten in die Notenzeilen nach Ihrem Wunsche ein. Benutzen Sie den Kreis (Shift w) für Noten ohne Vorzeichen (ganze Noten) und # für halbe Noten.

Die Noten reichen vom Ton ‚D' (erste Note unter der Grundlinie) bis zur Note ‚G' über der obersten Notenlinie. Sie geben die Buchstaben mit Hilfe der PET Cursor-Kontrolle ein - nicht mit Hilfe des Programmes.

C Bringen Sie den Cursor an eine freie Stelle auf dem Bildschirm und geben Sie ‚RUN' ein und drücken Sie ‚RETURN'. Es wird jetzt etwas dauern, weil das Programm nun die Musikroutinen in Assemblersprache lädt. Das Programm wird das gleiche Musikstück so lange wiederholen, bis die STOP-Taste gedrückt wird.

Um wieder zu starten, geben Sie wieder RUN 200 ein.

Zu 2.: Numerische Noten (Numeric Notes)

A Geben Sie RUN 600 ein. Der PET antwortet ‚ENTER MUSIC STRING?'

B Nun können Sie folgendes eingeben:

B 1 Eine Zahl von 0 - 9. Dies erzeugt eine Note.
B 2 Sie können sofort wieder eine weitere Note eingeben. Dies führt dazu, daß die Note als Viertelnote aufgefaßt wird.
B 3 Wenn Sie eine Klammer (.) nach einer Zahl eingeben, wird die Note doppelt so lang gespielt (halbe Note).
B 4 Wenn Sie ein O (alphabetisches ‚O' - nicht die Null eingeben) eigeben, wird die Note als ganze Note gespielt.

Sie können bis zu zwei Zeilen Eingabe gehen. Sie können auch mit der Cursor zu einer vorhergehenden Zeile gehen, editieren und die Return-Taste zur Wiederholung drücken.

Beispiel: 111.111.123456789.9.8.7.6.5.4.3.2.1.0.203040506070809000

Zu 3.: Zufallsnoten (Random Music)

Geben Sie RUN 400 ein, um dieses Programm zu starten. Der PET wird nun einige Zufallsnoten spielen, um anschließend die Tonleiter herauf- und herunterzuspielen.

Wenn Sie eine niedrige Zahl der numerischen Tastatur drücken, wird das Tempo steigen, bei einer hohen Zahl wird es sich verlangsamen.

Wenn der PET nicht arbeiten will, dann haben Sie bestimmt vergessen, die Assembler-Routinen neu zu laden. Um sie zu laden, geben Sie einfach RUN ein und es wird geladen.

Anwendungsbeispiel:

Die folgende Routine spielt ein recht interessantes Geräusch auf Ihrem PET (Achten Sie darauf, daß die MUSIC-Software auch vorher geladen wurde).

For I=1 to 100000:Poke 827, RND(1)*PITCH+OFF SET: SYS(832):NEXT

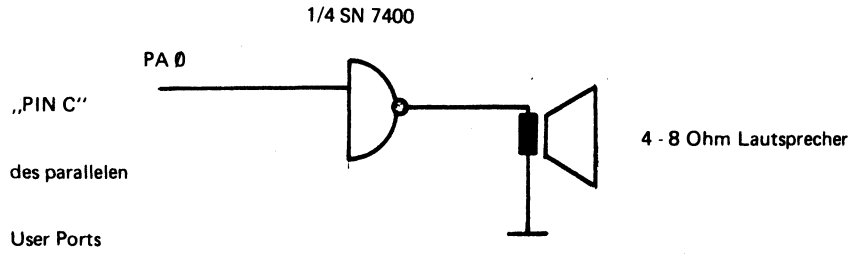
Wobei PITCH eine Zahl zwischen 2 und 255 ist und den Bereich festlegt.

OFF SET begrenzt die hohen Noten und muß entsprechend der PITCH-Eingabe gewählt werden. PITCH + OFFSET muß kleiner als 255 sein.

Diese Programme werden in Zukunft noch verbessert und können vom Benutzer nach Belieben erweitert werden. Vorschläge und Anregungen bitte schriftlich an den Verlag. Rückporto nicht vergessen!

MUSIC-GENERATOR

Sie können den PET zum Musikspielen verwenden, wenn Sie die untenstehende Schaltung an den parallelen Ausgangsport Ihres PETs anschließen.



Ein Programm mit dem Namen ‚TONE' muß in den zweiten Cassetten-Speicher geladen werden. Es erzeugt die Rechteckwellen am Ausgangsport. Das Programm wird durch den Befehl SYS (832) in BASIC ausgeführt. Vorher müssen PITCH und DURATION (Tondauer) wie folgt festgelegt werden:

POKE 825,D D legt die Tondauer fest, 255 ist eine brauchbare Zahl

POKE 826,0

POKE 827,P P = Pitch des Tones

POKE 59459,255 ermöglicht die Aktivierung der PORTs zu Ausgängen.

Um zu sehen, wie der Tongenerator arbeitet, geben Sie bitte folgendes ein:

```
FOR I=1 TO 10000:POKE 827,255*RND(1):SYS(832):NEXT
```

Diese Eingabe wird 10000 Zufallsnoten erzeugen. Achten Sie darauf, daß vorher POKE 826,0 eingegeben wurde.

Geschäfts- und Buchhaltungsprogramme für den Kleinbetrieb, Rechtsanwalt und Hausverwalter

Im nachfolgenden Teil bringen wir Ihnen die Originalbeschreibungen von fünf sehr interessanten Programmen. Wir haben die Beschreibungen bewußt im Original belassen (wie auch die Datenblätter), da es bei so wichtigen Daten und Informationen oft besser ist, die Originalversion zu benutzen.

Die Programme können vom Fachhandel oder im Buchhandel unter den folgenden Bestellnummern bezogen werden.:

1. General Ledger-Hustler 1, Buchhaltungsprogramm
   Best.-Nr. P37 DM 69,--

2. Checking Account (Kontenführung)
   Best.-Nr. P38 DM 79,--

3. Trust Account, Datenbank und Abrechnung für Vermögensverwalter sowie Kassen und Besitzverwalter, Vereine etc.
   Best.-Nr. P 39 DM 69,--

4. Legal Diary, Tagebuch und Abrechnung für Rechtsanwälte
   Best.-Nr. P40 DM 69,--

5. RENT ACCOUNT (für Hausverwalter)
   Best.-Nr. P41 DM 69,--

(S. auch Prospekt am Ende dieses Buches)

"GENERAL LEDGER-HUSTLER 1"

"Just LOAD and GO"-ONE OF THE "HUSTLER" SERIES OF PRE-RECORDED PROGRAMS

THE "GENERAL LEDGER-HUSTLER 1" PRE-RECORDED COMPUTER PROGRAM WRITTEN FOR THE COMMODORE PET* COMPUTER BY K.W. HARRIS

1. This pre-recorded cassette program tape is ready to use. No computer or programming experience is required. No changes or additions are necessary for immediate use of the GENERAL LEDGER-HUSTLER 1. Each program is recorded twice. Just LOAD and GO.

2. The program will hold up to 100 DATA lines with the standard 8 K user memory, and some 340 DATA lines with the optional additional 8 K user memory. When the maximum DATA line capacity is reached, it is a simple matter to start another block of memories, using the same record tapes or new ones.
Provisions are made to carry over to the new records the figures for BANK BAL-START OF PERIOD, WITHDRAWALS-NOT EXPENSE, EXPENSES, DEPOSITS MADE, and TOTAL INCOME. This gives continuity to the new records.

3. Entries are made in the order WITHDRAWALS and DEPOSITS are made in the account. The account checkbook and the bank records are the basis for this record keeping system. Each entry is coded as it is made into any of 8 types of deposits, any of 27 EXPENSE accounts, or any of 3 NON-EXPENSE withdrawal accounts.

4. Making the initial DATA entries is very easy to do, additional DATA is just as easy to enter into the computer.

5. The GENERAL LEDGER-HUSTLER 1 program is MULTI-REPORT:

   Mode #1- displays all WITHDRAWALS made, showing date, check number, amount, and a short description of the item.

   - displays all DEPOSITS made, with date, a code number that identifies the DEPOSIT as income or non-income, the amount, and a short description of the item.

   - ends the report by displaying the accumulated totals for: BANK BAL-START OF PERIOD, TOTAL EXPENSES, WITHDRAWALS-NOT EXPENSE, TOTAL INCOME RECD, TOTAL DEPOSITS MADE, and the CURRENT BANK BALANCE.

   Mode #2- on user request by CODE number, all entries with that CODE # are displayed line-by-line. When all the items have been displayed, the account title and the total amount of the account is shown. The date to start, and the date to end the report can be specified by the operator.

   Mode #3- calculates and displays, in turn, the detailed entries made for each DEPOSIT account, each EXPENSE account, and each NON-EXPENSE withdrawal account. The account items are completely shown in detail, then the account title and its total amount. The print-out starts with the lowest account number and shows each one in turn until the entire account list has been displayed.

6. In Mode #1, the user can start the print-out at any desired check number. It is not necessary to print-out the entire list of DATA entries to obtain the CURRENT BANK BALANCE, or the other accumulated totals to date.

*Trademark-Commodore Business Machines

GENERAL LEDGER-HUSTLER 1

INSTRUCTIONS - DEMONSTRATION PROGRAM

1. Install DEMO tape into recorder and REWIND-
2. Load tape into memory by pressing SHIFT key, holding it down while pressing RUN key.
3. Press PLAY key on recorder. Screen will display OK, SEARCHING, then FOUND, LOADING.
4. When tape is loaded, press REWIND key on recorder, then the STOP key.
5. If screen displays "? LOAD ERROR", repeat the loading steps.
6. When loading is complete, screen will display:
   GENERAL LEDGER
   ENTER 1- ENTRIES
   ENTER 2- ACCTS BY CODE #
   ENTER 3- ACCTS LISTED
7. Enter #1, hit RETURN key
8. Screen will display "ENTER CHECK # TO START PRINT-OUT"
9. Enter 1691 (this is the first check in the sample DATA), hit RETURN key.
10. Program will list out all the DATA lines entered, in order of entry, then the totals:

BANK BAL-START OF PERIOD		4567.99
DEPOSITS-INCOME	17880.00	
DEPOSITS-NOT INCOME	6125.00	
TOTAL DEPOSITS		24005.50
WITHDR'LS-EXPENSE	4121.27	
WITHDR'LS-NOT EXPENSE	2965.50	
TOTAL WITHDRAWALS		7086.77
CURRENT BANK BALANCE		21486.72

11. Program then resents, Enter #1, enter check number 1700, hit RETURN. DEPOSITS will be shown, then checks starting with the number requested.
12. Program again resets. Enter #2, hit RETURN key.
13. Screen will display "ACCT # WANTED- 90 ENDS THE REPORTS."
14. Enter #1 (ALL INCOME EXCEPT 2,3,4), hit RETURN key
15. Screen will display "DATE TO START REPORT". Enter August 1 (801), hit RETURN key.
16. Screen will display "DATE TO END REPORT". Enter October 1 (1001), hit RETURN key.
17. Screen will display all items in Account #1 for the period requested, then total 8060.00
18. For a display of ALL items entered in Account #1 for the entire period to date on the taped record, hit the Ø key on the computer. This skips "date to end report", the screen will display ALL entries in the account, then the total 17880.00
19. Enter #6 and the period desired. (Sample DATA runs from 728 to 1222) Hit RETURN
20. Enter #14 (RENTS PAID), hit RETURN key. For period 801 to 1130 total will be 987.90
21. Other accounts with DATA entries in the DEMO program are: (try various periods)
    #18 TAXES-STATE INC&MISC- (total for entire period- 1682.41)
    #23 ADVERTISING- (total for entire period- 429.61)
    #30 OFFICE EXPENSE- (total for entire period- 362.75)
    #37 LOANS MADE BY COMPANY- (total for entire period- 2500)
22. Type 50. This clears the screen and resets the program for further instructions.
23. Enter #3, hit RETURN key.
24. Screen will display the line-by-line details of each account with entries, starting with the lowest CODE number. When all lines of that account have been displayed, the screen will show the account title and the total amount in the account. The program then moves to the next account with entries, prints its DATA lines, then the title and total amount. The program does this until all accounts have been displayed on the screen. READY is then displayed.

The DEMO program DATA entries are too few to really demonstrate well the advantages of being able to choose the period to be covered in Mode #2. Month-by-month expense comparisons, receipts taken in over a specified period of time, taxes paid during a certain period, etc. are just a few of the possible uses. The user will find this a very valuable and much used capability of the GENERAL LEDGER-HUSTLER 1 program.

"GENERAL LEDGER-HUSTLER 1"

INSTRUCTIONS

A. Each DATA entry for this program is made in a single line "in residence" in the program. The DATA line consists of the line number, 4 "variables" and 1 "string variable". The computer REQUIRES that the "variables" be separated by commas, the "string variable" MUST start with a quotation marks. See B.6. below.

Each DATA line consists of:

Var. 1- the line number followed by the word DATA and the account CODE #(chosen from any of 8 DEPOSIT catagories,27 EXPENSE catagories,or 3 NON-EXPENSE withdrawal catagories. See CODE # LISTING- Page 6

Var. 2- the DATE in numbers (month,day of month) Example: Jan.5 would be 105

Var. 3- Either - the CHECK # (up to 4 digits) if an EXPENSE or NON-EXPENSE

Or - a SUB-CODE number if the entry is a DEPOSIT
SUB-CODE 9001 is used for INCOME DEPOSITS
SUB-CODE 9002 is used for NON-INCOME DEPOSITS

- if an EXPENSE item is a service charge, or some other bank charge that effects the bank balance,enter SUB-CODE 1111 for the item, (the CODE # would be #33- OTHER EXPENSES)

Var. 4- the AMOUNT in whole numbers and decimals. Example- 450.67

Var. 5- a short description starting with a quotation mark. Example: "OFFICE RENT the DESCRIPTION must be 16 characters or less. Count 1 for a space between words. Can be a combination of letters and figures.

B. ORIGINAL START-UP

1. Install MASTER program tape (Side 1) into recorder,press recorder REWIND button. LOAD program into computer memory by pressing SHIFT key,hold it down and press RUN key. Screen will tell you to press PLAY button on the recorder.

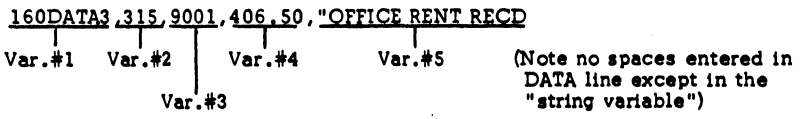
2. When loading is complete,screen will display the first part of the program, and instructions. Remove the MASTER tape,install #1 Record tape and REWIND.

3. Recall line 7 by typing LIST 7, hit RETURN key

4. Rewrite line 7 by typing 7 K= (enter the starting bank balance-numbers only), hit RETURN key. This places the beginning bank balance in computer memory.

5. Recall line 160 by typing LIST 160,hit RETURN key.

6. Rewrite line 160 using the first DATA to be entered. Example:



B. ORIGINAL START-UP (Cont'd)

7. Enter the remaining original DATA lines, using consecutive line numbers 161,162, 163,164 ect., until the first group of entries is complete.

8. RUN program- Mode #1, Mode #2, Mode #3

9. RECORD and VERIFY #1 Record tape by typing SAVE, then hit RETURN key. Screen will display PRESS PLAY & RECORD ON TAPE #1

10. When recording is complete, rewind tape, type VERIFY, then press PLAY on recorder.

11. Tape will run, then stop. Hit STOP button on recorder. VERIFIED will appear on screen. Remove #1 Record tape from the recorded.

12. Install #2 Record tape, RECORD and VERIFY by repeating steps 9,10, and 11.

ALWAYS make and check the duplicate tape. If a power failure occurs during recording, or #1 tape is damaged or lost, having a duplicate will permit the user to LOAD the previous entries that have been made, re-enter the new DATA lines again and repeat the recording steps. If #2 Record tape is used, be sure to re-record #1 Record tape so you still have two records.

C. ADDITIONAL ENTRIES:

1. Install #1 Record tape into recorder and rewind. LOAD into computer memory.

2. LIST to the end of the program.

3. Type in the next following line number and the first additional DATA

4. Type in the balance of the new entries, using consecutive line numbers.

5. LIST, then RUN in Mode #1 to accumulate the new totals.

6. RECORD and VERIFY #1 Record tape. Rewind and remove from recorder.

7. RECORD and VERIFY #2 Record tape. Rewind and remove from recorder.

D. STARTING A NEW BLOCK OF RECORDS:

New records may be made several ways.

1. A complete new set of Record tapes can be started, transferring the totals from the original records by using the method outlined for the NAMED block, described in detail in the following directions.

2. A NAMED block can be recorded on the original tape, following after the first block recorded. See detailed directions.

3. Side #2 of the original Record tapes can be used in the same manner as new tapes. This method is recommended because of the time saved in loading.

NEW RECORDS USING NAMED BLOCKS:

1. LOAD and RUN #1 Record tape, Mode #1. DO NOT REWIND TAPE

2. Make a written record of the amounts displayed for BANK BAL-START OF PERIOD, TOTAL EXPENSES-PERIOD, WITHDRAWALS-NOT EXPENSE, TOTAL DEPOSITS-PERIOD, and TOTAL INCOME-PERIOD

D. NEW RECORDS USING NAMED BLOCKS:

3. RUN the program again, this time use Mode #2

4. Add the figures found for CODES #5,6,7,and 8 to obtain a total figure for all NON-INCOME DEPOSITS that were made during the record period.

5. Remove the #1 Record tape from the recorder, DO NOT REWIND

6. LOAD #2 Record tape,DO NOT REWIND. Remove #2 Record tape from recorder.

7. Install MASTER program tape into recorder and rewind.

8. Type in LOAD"(the name you want for the new block)"-Example:LOAD"2",hit RET.

9. The MASTER program, now NAMED ,will LOAD into computer memory. Enter the totals from the first block as follows,using the figures from 2 and 4:

   a. Type LIST 7, hit RETURN key
   b. Type 7K=(enter the BANK BAL-START OF PERIOD amount),hit RETURN key
   c. Type 9H=(enter the TOTAL EXPENSES amount): W=(enter the WITHDRAWALS -NOT EXPENSE amount) hit RETURN key
   d. Type 10 I=(enter the TOTAL DEPOSITS amount)hit RETURN key
   e. Type 11 F=(enter total NON-INCOME amount from 4 above),hit RETURN
   f. Type 160DATA35,1,1,0,"A" (creates a dummy entry), hit RETURN key

10. RUN the program in Mode #1, using Check #1. The totals entered will be displayed

11. Remove the MASTER program tape and install #1 Record tape- DO NOT REWIND

12. Type in the new DATA entries, beginning with line number 160.

13. When entries are completed,type SAVE"(the name of the new block)",hit RETURN

14. RECORD,then rewind the tape all the way back to its beginning. Type VERIFY"(the name used for the new block)"- When VERIFIED appears on screen, rewind the tape.

15. Remove #1 Record tape and install #2 Record tape. DO NOT REWIND

16. Type SAVE"(the name used for the new block)",hit RETURN key

17. RECORD, then rewind the tape all the way back to its beginning. Type VERIFY"(the name used for the new block)"-When VERIFIED appears on screen, rewind the tape.

RECORDS ON NEW TAPES: Take the original record totals as described in 2. and 4. and perform the steps called for in 9. above,except the block has no name. Perform step 10. RECORD and VERIFY #3 Record tape. RECORD and VERIFY #4 Record tape.

CASSETTE TAPES made specifically for computers should ALWAYS be used. The MASTER PROGRAM tape and the two blank Record tapes supplied are this quality. Your computer demands perfect recordings. Proper tape magnetic characteristics and consistant quality help give you the needed excellent recordings. These cassettes are available from COMPUTERS ONE for $5.00 for two tapes, plus $ .75 shipping and handling,prepaid.

To order tapes by mail, write: COMPUTERS ONE
#306 Kahala Office Tower
4211 Waialae Ave.
Honolulu, Hawaii 96816

REMEMBER-ALWAYS RECORD TWO TAPES-EVERY TIME-NO EXCEPTIONS

CODE LIST FOR "GENERAL LEDGER-HUSTLER 1"

DEPOSITS:

CODE # - VARIABLE 1	SUB-CODE # VARIABLE 3
1- ALL INCOME (EXCEPT 2,3,4)	9001
2- INTEREST INCOME	9001
3- RENTS REC'D	9001
4- OTHER INCOME	9001
(Deposits Not Income) 5- MISC.PYTS TO COMPANY	9002
6- LOANS TO COMPANY	9002
7- LOAN REPAYMENTS	9002
8- OTHER DEPOSITS	9002
FOR EXPENSE ITEMS WITH NO CHECK NUMBER (BANK CHGS,ECT)	1111

NON-EXPENSE WITHDRAWALS:

CODE # - VARIABLE 1

36- CAPITAL EQUIPMENT

37- LOANS MADE BY COMPANY

38- WITHDRAWALS-NOT EXPENSE

EXAMPLE- DATA LINE ENTRY

203 DATA 3,315,9001,406.50,"RENT RECD"

Line #

Key Word

CODE #

DATE

Sub-code # for DEPOSIT
Check # for WITHDRAWAL

AMOUNT

Description in quotes, must be 16 or less characters, figures, or spaces

If a mistake is made in entering DATA, hit RETURN and retype the ENTIRE DATA line.

EXPENSES:

CODE # - VARIABLE 1

9- CASH REFUNDS/ALLOWANCES

10- COMPENSATION-OFFICERS

11- SALARIES,WAGES,COMMISSIONS

12- REPAIRS

13- BAD DEBTS

14- RENTS PAID

15- TAXES-FED CORP INCOME

16- TAXES-FED SOC. SECURITY

17- TAXES-FED WITHHOLDING/OTHER

18- TAXES-STATE CORP INC & MISC.

19- TAXES-STATE WITHHOLDING

20- TAXES-STATE UNEMP/OTHER

21- INTEREST PAID

22- COST OF GOODS SOLD

23- ADVERTISING

24- DUES & SUBSCRIPTIONS

25- AUTO EXPENSE

26- EQUIPMENT RENTAL

27- SHIPPING-CHARGES & MATRLS

28- INSURANCE-CORP

29- INSURANCE-WORKER COMP & HEALTH

30- OFFICE EXPENSE

31- TELEPHONE & TELEGRAMS

32- LICENSES

33- OTHER EXPENSES

34- ACCT #1- EXPENSE

35- ACCT #2- EXPENSE

SUGGESTIONS FOR "CUSTOM" MODIFICATIONS TO "GENERAL LEDGER" PROGRAM

Business enterprises have many different kinds of record keeping needs. The GENERAL LEDGER "Card of Accounts" on the CODE LIST, page 6, was set up for a typical small business. Some users of the program like to modify the "Card of Accounts" with account names that best suits their particular needs. It is a simple operation to change some or all of the account names, but it is ABSOLUTELY NECESSARY that the newly named account (s) be kept in the same catagory groups as the original account(s). The total number in each group MUST be kept the same.

The record keeping capability furnished by this program falls into FOUR groups:

Group A- DEPOSITS that are INCOME (Usually taxable income)
Accounts #1 thru #4, a total of 4 Accounts
These accounts ADD to the bank balance and accumulate as DEPOSITS-INCOM

Group B- DEPOSITS that are NOT INCOME (Usually not taxable as income)
Accounts #5 thru #8 inclusive, a total of 4 accounts
These accounts ADD to the bank balance and accumulate as DEPOSITS-NOT INCOI

Group C- WITHDRAWALS that are EXPENSE ITEMS to the business
Accounts #9 thru #35, a total of 27 accounts
These accounts SUBTRACT from the bank balance and accumulate as WITHDRAWALS-EXPENSE

Group-D- WITHDRAWALS that are NOT EXPENSE items to the business
Accounts #36 thru #38, a total of 3 accounts
These accounts SUBTRACT from the bank balance and accumulate as WITHDRAWALS-NOT EXPENSE

Program lines #121 thru #158 are the print-out lines for Accounts #1 thru #38. If the program user desires to change an account "NAME" or "NAMES", and the account (s) to be re-named fall into the same catagory as the original accounts, all that is required is to find the line number of the "NAME" to be revised, list that line number as a check, and RETYPE the ENTIRE LINE exactly, but now using the new account "NAME". EXACT is the key word. ALL punctuation marks ect. in the original line must be re-entered exactly, with the only change being the new "NAME".

EXAMPLE: Say Account #10- "COMPENSATION-OFFICERS is not used, and it is desired to change this account to "BONUS PAYMENTS":

1. List line # 130
2. Re-type: 130?"BONUS PAYMENTS";K$:?:GOTO65
3. Hit ENTER key

When the program is recorded, the new "NAME" will be incorporated from then on.

DISK OPERATION:

The Commodore PET(tm) disk system is not available at this writing. However, based on experience with the TRS-80(tm) disk, a disk system works fine.

In operation, the MASTER program is loaded into machine memory from the cassette. DATA lines are entered using exactly the same method used for tape records, then "SAVED" on the disk instead of the record cassette. TWO disks should be made, as usual. From then on, the disks are used exclusively, keeping the MASTER cassette to start new disks.

With this type of program, the number of DATA lines are limited by the capacity of the machine memory. 16 K is about the lowest practical capacity, while 48 K is probable the highest practical limit.

In a test we made, a program with DATA that required one minute 20 seconds to load from cassette loaded from a disk in just 9 seconds! For long records, this is an obvious advantage, and there are others beside the speed of operation.

CHECKING ACCOUNT-HUSTLER 1

"Just LOAD and GO"- ONE OF THE "HUSTLER" SERIES OF PRE-RECORDED PROGRAMS

WRITTEN FOR COMMODORE PET 2001(tm) BY K.W.HARRIS

INTRODUCTION:

This program is designed to keep accurate records of the users personal checking account. DEPOSIT records are kept in three catagories,SALARY & WAGES, OTHER INCOME, and DEPOSITS-NOT INCOME. WITHDRAWAL records are kept in eighteen different catagories.

Not only does the program keep track of the details of each of the three DEPOSIT catagories and the eighteen WITHDRAWAL catagories, it also keeps a running total for: DEPOSITS-INCOME,DEPOSITS-OTHER,TOTAL DEPOSITS,WITHDRAWALS-TAX DEDUCTIBLE, TOTAL WITHDRAWALS, and the CURRENT BANK BALANCE. The program accumulates all DEPOSITS,deducts all WITHDRAWALS, and displays the CURRENT BANK BALANCE.

As featured in all pre-recorded programs in the "HUSTLER" series, the complete financial details on any particular catagory can be recalled and displayed in just seconds, as well as the previously mentioned grand total amounts.

PROGRAM FEATURES:

1. The pre-recorded MASTER cassette tape is ready to use as received. No computer or programming experience is required. Follow the simple instructions. No additions or changes are required to start immediate use of the computer. Just LOAD and GO. Each program is recorded twice.

2. As supplied, the program will hold up to 145 check entries before a new record tape must be started. If an optional 8 K of user memory is added to the standard 8 K bytes, up to 370 checks can be accomodated. When the memory capacity is reached, or the desired record period is completed, it is a very simple matter to start another record tape, carrying over the grand total amounts to the new records if the user desires.

3. Entries are made in random order just as money is deposited and withdrawn from the account. Making the initial DATA entries is very simple, and additional entries as checks are written and deposits made is just as easy. Ease of making entries is a planned feature of this program.

4. Program is MULTI-REPORT:

   Mode #1- displays each DATA line in the order it was entered, showing the date,check number,amount,and a short description of entry

   - displays all deposits including date,the code number that tells if the deposit is income or non-income, the amount, and a short identifying description

   - after the detailed DATA lines are displayed on the screen, it then displays the following totals:

     BANK BAL-START OF PERIOD-
     DEPOSITS-INCOME-
     DEPOSITS-OTHER-
     TOTAL DEPOSITS-
     WITHDRAWALS-TAX DEDUCTIBLE-
     TOTAL WITHDRAWALS-
     CURRENT BANK BALANCE-

4. Program is MULTI-REPORT: (Cont'd)

   Mode #2- on request by user for a specific CODE number, DATA entry lines for the requested catagory are sorted out of the random entry listing and displayed line-by-line. When all entered line items for that CODE number have been displayed, the catagory title and the sub-total amount for the catagory are displayed.

   Mode #3- separates and displays each of the DATA entries made for each catagory as described for Mode #2, including the catagory total at the conclusion of the line-by-line display. When the first catagory is completed (the program starts with the lowest CODE number),the next following account detailed DATA lines are displayed line-by-line, and the catagory title and total amount are displayed Then the program moves to the next catagory. This operation is repeated until all catagories with entries are displayed.

5. In Mode #1, the user can request the display of the check details to start with a designated check number. It is not necessary to display the entire DATA list to get the details on a particular check. This feature can be tried out on the demonstration program tape.

6. The record breakdown can be made just as elaborate or as simple as the user desires. There is no requirement that all the catagories have entries made in them. If no entry is made in a certain catagory, the program just ignores it and continues on to the next one that does have entries.

   Account #1 and Account #2 are provided for use for any special expense catagory that the user may want to keep track of for his or her information.

7. PROPERTY TAXES and CONTRIBUTIONS that are deductible on income taxes are accumulated and displayed as a separate figure. Amounts entered for these items are still deducted from the bank balance as are all other withdrawals.

CHECKING ACCOUNT-HUSTLER 1

DEMONSTRATION TAPE:

The DEMONSTRATION tape program was designed to show how easy it is to use the computer to recall entered DATA, and the speed with which the detailed information appears on the screen. Typical entries have been made in the three DEPOSIT catagories, and twelve of the eighteen WITHDRAWAL catagories.

DIRECTIONS:

1. Load demo tape into computer by pressing SHIFT key,hold it down and press RUN key.
2. Press PLAY key on recorder.
3. Screen will display OK, SEARCHING, then FOUND, LOADING
4. When loading is complete, press rewind key on recorder, then STOP
5. If screen displays "? LOAD ERROR", repeat the loading operation.
6. When loading is completed, screen will display:

   READY
   RUN
   CHECKING ACCOUNT
   ENTER 1- ENTRIES
   ENTER 2- ACCTS BY CODE #
   ENTER 3- ACCTS LISTED

7. Press the number 1 on keyboard, hit RETURN key.
8. Screen will display "ENTER CHECK # TO START PRINT-OUT"
9. Key in number 379 (the first check number in sample DATA), hit RETURN
10. Program will list out, a line at a time, all DATA line entries made, in order of entry. When all lines are displayed, the screen will display the totals:

BANK BAL-START OF PERIOD-	4567.99
DEPOSITS-INCOME-	5250.00
DEPOSITS-NOT INCOME-	450.00
TOTAL DEPOSITS MADE-PERIOD-	5700.00
WITHDRAWALS-TAX DEDUCTIBLE-	432.72
TOTAL WITHDRAWALS-	4731.10
CURRENT BANK BALANCE-	5536.89

    It is not necessary to display all checks. Type RUN, then hit RETURN,and enter 1 again from the keyboard. Enter the number 386, hit RETURN key. All calculations and accumulations will be made,but display will start with the check number requested.
11. After a few seconds delay, the program resets waiting for the next instruction.
12. Press the number 2 on keyboard, hit RETURN key.
13. Screen will display "ACCT # WANTED-RETURN ENDS PROGRAM"
14. Type in the number 1 (for SALARY & WAGES) from the keyboard, hit RETURN key
15. Screen will display complete data on SALARY & WAGES entries, then the total- 3750.00
16. Type in the number 8 (PROPERTY TAXES) fron the keyboard, hit RETURN key
17. Screen will display complete data on PROPERTY TAX entries,then the total- 392.72
18. Other accounts with DATA entries are: 2,3,4,5,6,7,9,10,11,13,14,16,17,19. Try them out. Note the speed of response to your request for the information. Use any number you desire, it is not necessary to go in any particular rotation.
19. Type RUN on keyboard, hit RETURN
20. Press the number 3 on keyboard, hit RETURN key
21. Screen will display the line-by-line record of each catagory that has an entry, starting with the lowest CODE number. After the sub-total is displayed, the program will move to the next catagory, display it's DATA lines and sub-total, and then on to the next one until all catagories have been displayed.

CHECKING ACCOUNT-HUSTLER 1

INSTRUCTIONS

A. Each DATA entry for this program is made in a single line "in residence" in the program. Each line consists of 4 "variables" and 1 "string variable". The computer requires that the "variables" be separated by commas. The "string variable" must start with a quotation mark. See Example in B.6. below.

Each DATA line consists of:

Var. 1- the line number, the key word DATA, the category CODE # (chosen from any of 3 DEPOSIT categories or any of the 18 WITHDRAWAL categories. See CODE # LISTING - Page 7

Var. 2- the DATE in numbers (month, day of month) Example: Jan. 5 would be 105

Var. 3- Either- the CHECK number (up to 4 digits) if item is a WITHDRAWAL

or- a SUB-CODE number if the entry is a DEPOSIT
SUB-CODE 9001 is used for INCOME DEPOSITS
SUB-CODE 9002 is used for NON-INCOME DEPOSITS

- if the WITHDRAWAL is a bank service charge against the account, enter SUB-CODE 1111 for this "variable"

Var. 4- the AMOUNT in numbers and decimals. Example: 450.67

Var. 5- a short DESCRIPTION starting with a quotation mark. Example: "MARCH SAL the description should be 16 characters or less, including spaces between words, can be a combination of letters and figures.

Line 7 is reserved for the beginning bank balance.

Lines 150 on are reserved for DATA entries.

B. ORIGINAL START-UP:

Before starting to enter DATA, LIST the DEMO program tape again and examine closely the DATA entry lines starting at line 150. After examination, type NEW, hit RETURN.

1. Load the MASTER tape into the computer by pressing SHIFT key, hold it down and press RUN key. Then press PLAY button on the recorder.
2. When loading is complete, the screen will display the first part of the program and the program instructions. Remove MASTER tape, install #1 Record tape, rewind.
3. Recall line 7 by typing LIST 7, then hit RETURN key.
4. Rewrite line 7 by typing 7P=(enter the beginning bank balance, numbers only), hit RETURN key.
5. Recall line 150 by typing LIST 150, then hit RETURN key.
6. Rewrite line 150 using the first DATA to be entered. Example:

```
150DATA1,315,9001,1875.00,"MARCH SAL
          |   |    |      |
Var.#1    |   |    |  Var.#4  Var.#5
      Var.#2  |    |
              |    |
          Var.#3
```

(Note that no spaces are entered in DATA line except in the "string variable")

B. ORIGINAL START-UP (Cont'd)

7. Enter the remaining original DATA lines,using consecutive line numbers 151,152, 153,154 ect., until the first group of entries is complete.

8. RUN program- Mode #1, Mode #2, Mode #3

9. RECORD and VERIFY #1 Record tape by typing SAVE, then hit RETURN key. Screen will display PRESS PLAY & RECORD ON TAPE #1

10. When recording is complete, rewind tape, type VERIFY,then press PLAY onrecorder.

11. Tape will run, then stop. Hit STOP button on recorder. VERIFIED will appear on screen. Remove #1 Record tape from the recorder.

12. Install #2 Record tape,RECORD and VERIFY by repeating steps 9, 10, and 11.

ALWAYS make and check the duplicate tape. If a power failure occurs during recording, or #1 tape is damaged or lost, having a duplicate will permit the user to LOAD the previous entries that have been made, re-enter the new DATA lines again and repeat the recording steps. If #2 tape is used, be sure to re-record #1 Record tape so you still have two records.

C. ADDITIONAL ENTRIES:

1. Install #1 Record tape into recorder and rewind. LOAD into computer memory.

2. LIST to the end of the program. Note the final previous DATA line number.

3. Type in the next following line number and complete the additional DATA line.

4. Type in the balance of the new entries,using consecutive line numbers.

5. LIST, then RUN in Mode #1 to accumulate the new totals.

6. RECORD and VERIFY #1 Record tape. Rewind and remove from recorder.

7. RECORD and VERIFY #2 Record tape. Rewind and remove from recorder.

D. STARTING A NEW BLOCK OF RECORDS:

New records may be made several ways.

1. A complete new set of Record tapes can be started, transferring the totals from the original records by using the method outlined for the NAMED block, described in detail in the following directions.

2. A NAMED block can be recorded on the original tape, following after the first block recorded. See detailed directions.

3. Side #2 of the original Record tapes can be used in the same manner as new tapes. This method is recommended because of the time saved in loading.

NEW RECORDS USING NAMED BLOCKS:

1. LOAD and RUN #1 Record tape, Mode #1. DO NOT REWIND TAPE

2. Make a written record of the amounts displayed for BANK BAL-START OF PERIOD, DEPOSITS-NOT INCOME,TOTAL DEPOSITS MADE-PERIOD,WITHDRAWALS-TAX DEDUCTIBLE, and TOTAL WITHDRAWALS. (for re-entry on new tape record)

CHECKING ACCOUNT-HUSTLER 1

NEW RECORDS USING NAMED BLOCKS: (Cont'd)

3. Remove the #1 Record tape from the recorder, NOT REWOUND

4. LOAD #2 Record tape into computer, DO NOT REWIND, Remove tape from recorder.

5. Install MASTER program tape into recorder and rewind.

6. Type in LOAD"(the name you want for the new block)"- Example: LOAD"2"- RETURN

7. LOAD the MASTER program (now named) into the computer by pressing PLAY button on the recorder. Enter the figures to be carried over to the new records, using the amounts found in Step 2, as follows:

   a. Type LIST 7, hit RETURN key
   b. Type 7P=(enter the BANK BAL-START OF PERIOD amount), hit RETURN key
   c. Type 8R=(enter the DEPOSITS-NOT INCOME amount): I=(enter the TOTAL DEPOSITS amount): U=(enter the WITHDRAWALS-TAX DEDUCTIBLE amount): T=(enter the TOTAL WITHDRAWALS amount), then hit RETURN key
      Example: 8R=4567.99:I=7200.00:U=432.76:T=4734.14 (NOTE: Each of the statements is separated from the one following by a colon)
   d. Type 150DATA1,1,1,0,"A (creates a dummy entry), hit RETURN key

8. RUN the program in Mode #1, using check #1. The totals entered will be displayed.

9. Remove the MASTER program tape and install the #1 Record tape- IMPORTANT-DO NOT REWIND AT THIS TIME.

10. Beginning with line number 150, type in the new DATA line entries.

11. When entries are completed, type SAVE"(the name of the new block)", hit RETURN

12. RECORD the #1 Record tape, then rewind the tape back to it's beginning. Type VERIFY"(the name used for the new block)". When VERIFIED appears, rewind tape.

13. Remove #1 Record tape , and install #2 Record tape. DO NOT REWIND.

14. Type SAVE"(the name used for the new block)", hit RETURN key.

15. RECORD the #2 Record tape, then rewind the tape back to it's beginning. Type VERIFY"(the name used for the new block)". When VERIFIED appears, rewind tape.

If the new records are to be made on a new set of tapes, or the #2 Side of the original tapes are to be used, it is not necessary to "NAME" the new block of records. In this case, obtain the totals as described in Step 2. Then carry out the steps called for in 7. above. Do Step 8. RECORD and VERIFY #3 Record tape, and #4 Record tape.

CASSETTE TAPES made specifically for computers should ALWAYS be used. The MASTER PROGRAM tape and the two blank Record tapes supplied are of this quality. The compute demands perfect recordings. Proper tape magnetic characteristics and consistant quality will give you the needed excellent recordings. These cassettes are available from COMPUTERS ONE for $5.00 for two tapes, plus $.75 shipping & handling, pre-paid. To order tapes by mail, write: COMPUTERS ONE
#306 Kahala Office Tower
4211 Waialae Ave.
Honolulu, Hawaii 96816

REMEMBER-ALWAYS RECORD TWO TAPES-EVERY TIME-NO EXCEPTIONS

CODE LIST FOR "CHECKING ACCOUNT-HUSTLER 1"

DEPOSITS:

CODE #- VARIABLE 1	SUB-CODE # VARIABLE 3
1- SALARY & WAGES	9001
2- OTHER INCOME	9001
3- DEPOSITS-(NOT INCOME)	9002
FOR EXPENSE ITEMS WITH NO CHECK NO.(BANK CHGS,ECT)	1111

WITHDRAWALS:

CODE #- VARIABLE 1

4- WITHDRAWALS-MISC
5- MORTGAGE OR RENT
6- FOOD & SUPPLIES
7- UTILITIES
8- PROPERTY TAXES
9- CONTRIBUTIONS
10- LIFE INSURANCE #1
11- LIFE INSURANCE #2
12- LIFE INSURANCE #3
13- MEDICAL INSURANCE
14- LOAN #1
15- LOAN #2
16- CREDIT CARD #1
17- CREDIT CARD #2
18- CREDIT CARD #3
19- BANK CHARGES (NO CHECK)
20- ACCT #1 (Any account user wants to keep a record on)
21- ACCT #2 " " " "

EXAMPLE: DATA LINE ENTRY

```
150DATA1,602,9001,1435.75,"JUNE SAL
```

Line Number

Key Word

CODE #

DATE

Check number for WITHDRAWAL
Sub-code # for DEPOSIT

AMOUNT

Description starting with quote, 16 or 'ess characters,figures,and spaces

NOTE: If a mistake is made in entering DATA, hit RETURN and retype the ENTIRE DATA line.

"TRUST ACCOUNTS-HUSTLER 1"

"Just LOAD and GO"-ANOTHER OF THE "HUSTLER" SERIES OF PRE-RECORDED BUSINESS PROGRAMS

The "TRUST ACCOUNT-HUSTLER 1" PRE-RECORDED COMPUTER PROGRAM - WRITTEN FOR THE COMMODORE PET* 2001 PERSONAL COMPUTER BY K.W. HARRIS

INTRODUCTION:

Lawyers handling estate settlements must keep accurate, detailed records to satisfy the requirements of the Probate Court. This record keeping is a time-consuming, expensive operation if there is more than a very small number of trust accounts to administer. Attorneys who do estate settlement work,(and this includes practically all practicing attorneys), will find the TRUST ACCOUNTS-HUSTLER 1 of great assistance in keeping track of these records.

Not only does the program keep track of the details of each individual account, it also keeps a running grand total on all of the major account amounts, so the lawyer knows at all times exactly where his trust account totals stand.

Five essential sub-totals are kept in full detail on each trust account: "ORIGINAL CASH ASSETS", "ADDITIONAL CASH RECEIVED", "PAYMENTS TO CLIENTS", "EXPENSES PAID", and "BALANCE REMAINING IN ACCOUNT".

As featured in all pre-recorded programs in the "HUSTLER" series, the complete financial details on any particular account can be recalled and displayed in just seconds, as well as the mentioned grand totals on the entire trust account phase of the lawyer's business.

PROGRAM FEATURES:

1. The pre-recorded MASTER cassette tape is ready to use as received. NO computer or programming experience is required. Absolutely NO additions or changes are necessary to permit immediate use of the "TRUST ACCOUNTS-HUSTLER 1". Just LOAD and GO. Each program is recorded twice.

2. As supplied, the program will hold up to 175 DATA entry lines before a new record tape (or an additional block on the original tape) must be started. If an additional 8 K of user memory (available as an optional extra) is added to the standard 8 K bytes, up to 410 entry lines can be accommodated.

   When the memory capacity is reached, it is very simple to start another block of records for the still active trust accounts.

   Provisions can be made to carry over the grand total amounts of "ORIGINAL CASH ASSETS", "ADDITIONAL CASH RECEIVED", "PAYMENTS TO CLIENTS", "EXPENSES PAID", and "BALANCE REMAINING IN ACCOUNT". Also, the sub-totals for the individual trust accounts are carried over to the new record tapes.

3. Entries are made in random order just as money is received and paid out. DATA entries are as simple to make as operating a modern cash register.

*Trade Mark Commodore Business Machines

PROGRAM FEATURES: (Cont'd)

4. Entering additional DATA is just as simple and easy. Ease of making entries is a planned major feature of the program.

5. Program is MULTI-REPORT:

Mode #1- displays each DATA entry in the order it was made, showing the ACCOUNT number, DATE, CODE number for the item, AMOUNT, and a short identifying description of the item entered.

- after the detailed DATA lines are printed out on the screen, it displays a running grand total-to-date for: ORIGINAL CASH ASSETS, ADDITIONAL CASH RECEIVED, PAYMENTS TO CLIENTS, EXPENSES PAID, and BALANCE IN ACCOUNT

Mode #2- on user request, a specific account's individual DATA entries are separated out of the random entry listing and displayed line-by-line, according to the account number requested. The account number is assigned at the opening of the account.

- after the detailed line-by-line DATA entries are completely displayed for that particular account, the figures for that account's ORIGINAL CASH ASSETS, ADDITIONAL CASH RECEIVED, PAYMENTS TO CLIENTS, EXPENSES PAID and BALANCE IN ACCOUNT are displayed on the screen. The program is reset, waiting for the next account number request, or the program to end.

Mode #3- separates and displays each of the DATA entries made for each TRUST ACCOUNT as described for Mode #2, including the account totals at the conclusion of the line-by-line display. When the first account is completed (always the lowest account number), the program moves to the next account, displays the DATA entries for that particular account and its totals, and then moves to the next account. This operation is repeated until all the accounts in the DATA have been displayed.

"TRUST ACCOUNT-HUSTLER 1"

"Just LOAD and GO"-ONE OF THE "HUSTLER"SERIES OF PRE-RECORDED BUSINESS PROGRAMS

DEMONSTRATION TAPE

This pre-recorded demonstration tape has 41 typical entries made in three typical trust accounts, with the entered DATA coded into four essential sub-accounts.

Ease of recalling DATA is illustrated by the almost instant display of the wanted information on the computer screen.

DIRECTIONS

1. Load demo tape into memory by pressing SHIFT key, hold it down and press RUN key.
2. Press PLAY key on recorder.
3. Screen will display OK, SEARCHING, then FOUND, LOADING
4. When tape is loaded, press STOP button on recorder, rewind the tape, and press the recorder STOP button again.
5. If screen displays ? LOAD ERROR, repeat the loading steps
6. When loading is complete, screen will instruct user to ENTER 1, ENTER 2, ENTER 3, or ENTER 4 for totals of all accounts.
7. Enter 1 from the keyboard, hit RETURN key
8. The program will list out, a line at a time in order of entry, the complete list of DATA lines entered on the tape. After the lines are printed, the screen will display the accumulated grand totals:

ORIGINAL CASH ASSETS	5904.00
ADDNL CASH RECD	64682.08
PYTS TO CLIENTS	9650.00
EXPENSES	2398.75
BAL IN ACCT	58537.33

Run #1 again, this time hit the STOP button before the program completely prints out. The scrolling display will stop, permitting the user to examine any of the DATA lines entered. Hit RETURN key, type CONT and hit RETURN key again. The program will start again and run to its conclusion.

9. After a few seconds delay, program resets, waiting for further instructions.
10. Enter 2 from the keyboard, hit RETURN key
11. Screen will display ENTER ACCT # WANTED. 75 ENDS REPORT
12. Accounts #1, #2 and #3 have entries. Enter any of these numbers, hit RETURN key
13. Screen will display all of the detailed DATA entries made for that specific account, then display the sub-totals for that particular client. Program then re-sets, waiting for the next account number to be requested, or the program ended.
14. Run the other accounts.
15. Type in RUN again, hit RETURN key
16. Enter 3 from the keyboard, hit RETURN key
17. Screen will display, line-by-line- the entries made for the account having the lowest number. On completion of the detailed DATA lines for that account, the screen will display the five sub-total amounts. When the first account is finished (#1 on the demo tape), the program will move to #2, display all the DATA lines and totals for this account, then move to #3. When #3 is completed, the program resets, waiting for further instructions.

The three Modes of display give the user much flexibility in recalling DATA entries. #2 Mode displays the complete details as well as the totals on each account as it is requested. Full information on any of the trust accounts in the records is made available in just seconds.

I N S T R U C T I O N S

"TRUST ACCOUNTS-HUSTLER 1"

A. DATA entries for this program are each a single line "in residence" in the program. Each line consists of 4 variables and 1 string variable. The variables are separated by a comma, the string variable must start with a quotation mark. See typical entries in 4. below. Note that no "spaces" are used in the DATA line. Each DATA line consists of:

Var. 1 - the line number, the word DATA, the trust accounts numbered consecutively 1 thru 994, (no space between line number, DATA, account number)

Var. 2 - the DATE in numbers (month, day of month) Ex: Jan. 5 would be 105

Var. 3 - the CODE number identifying the type of entry being made:

CODE #1 - Entry is for ORIGINAL CASH ASSETS

CODE #2 - Entry is for ADDITIONAL CASH RECEIVED

CODE #3 - Entry is for PAYMENTS TO CLIENTS

CODE #4 - Entry is for EXPENSES PAID

(Note: No entry is required for BALANCE IN ACCOUNT, this figure is calculated by the computer)

Var. 4 - the AMOUNT to be charged or credited. The figure is entered as whole numbers and decimals. Ex: 450.75

Var. 5 - a short DESCRIPTION starting with a quotation mark. (Do not exceed 16 letters and numbers). See exception below for the first 4 "label" lines.

B. ORIGINAL START-UP FOR #1 RECORD TAPE

1. Install MASTER tape in recorder, and REWind to start of tape.

2. Load Master tape into memory-press SHIFT key, hold it down and press RUN key. Then press PLAY key on recorder, tape will load-on completion press STOP.

3. Remove MASTER Program Tape from recorder, install #1 Record Tape and rewind

4. Type LIST 100, hit RETURN key. Rewrite line 100 by entering the initial DATA line for the first trust account. The program uses 4 DATA lines initially to provide a line for each of the CODES - #1, #2, #3, and #4. These lines are used to "label" the account, name, address, phone number, etc. If a mistake is made, hit RETURN and retype the entire DATA line. Assume the first trust account is a probate of an estate, Henry Miller died June 16, 1977, filed in Probate court Oct. 17, Court File #1374, Beginning assets were $4,728.50, trust account opened in Bank of Hawaii 11/5 #1 was assigned to this account by the attorney's secretary. Entries are:

```
Type-  100DATA1,1017,1,0,"PROBATE
Type-  101DATA1,1017,2,0,"CT # 1374
Type-  102DATA1,1017,3,0,"HENRY MILLER
Type-  103DATA1,1017,4,0,"DECSD 6/16/77
Type-  104DATA1,1105,1,4728.50,"B OF HAWAII
```

Note: The first 4 lines of a specific account will print out only the 'descriptive' part of the line data, so up to 30 letters and numbers can be entered in Variable 5 in each of these 4 lines.

B. ORIGINAL START-UP FOR #1 RECORD TAPE (Cont'd)

4. These 5 entries complete the initial DATA entries for TRUST ACCOUNT #1. Subsequent entries in the program will be made starting with line number 105, properly coded of course.

   IMPORTANT:
   EACH variable #1, #2, #3, #4, and #5 MUST ALWAYS have something entered. The entry can be 1 digit in each of the first 4 variables, and 1 letter enclosed in quotes for #5, but THERE MUST ALWAYS BE AN ENTRY IN EACH ONE.

5. The initial entries for the next account number are made in the same manner, again starting with 4 lines of initial entries to "label" the account. Later on, if the trust account goes over to the next block of entries on a new tape, the AMOUNT variable location in the "label" lines is used to carry over the figures for ORIGINAL ASSETS, ADDITIONAL CASH RECEIVED, PAYMENTS TO CLIENTS, and EXPENSES PAID amounts to the new records.

6. Enter the balance of the other trust account DATA entries to start the records, using consecutive line numbers for each DATA line, and starting off each new account with the initial account lines, one line each for CODE #1, CODE #2, CODE #3 and CODE #4.

7. RUN the program, Mode #1, Mode #2, Mode #3. When the final account has been displayed, hit STOP, then RETURN. Screen will display READY.

8. RECORD #1 Record tape by typing SAVE, then hit RETURN key. Screen will display PRESS PLAY & RECORD ON TAPE #1

9. When recording is complete, rewind tape, type VERIFY, then hit RETURN key. As instructed by screen, press PLAY button on recorder.

10. Tape will run, then stop. Press STOP button on recorder. VERIFIED will appear on screen. Remove #1 Record tape from recorder.

11. Install #2 Record tape, RECORD and VERIFY by repeating steps 8., 9, and 10.

C. It is strongly recommended that a duplicate tape ALWAYS be made. A power failure during recording, damage, or loss of the #1 Record tape will still permit the user to load the previous entries from the second tape, again enter the additional DATA entries, RUN the program, RECORD and VERIFY. If it is necessary to use the #2 tape, be sure to also RECORD the #1 tape.

D. MAKING ADDITIONAL ENTRIES

1. Load #1 Record tape into memory, rewind tape

2. LIST program.

3. Type in the line number following the last entry made, complete the new entry.

4. Type in the balance of the new entries, using consecutive line numbers.

5. LIST program so new entries can be inspected for accuracy, RUN in Mode #1

6. RECORD and VERIFY #1 Record tape, rewind and remove from recorder

7. RECORD and VERIFY #2 Record tape, rewind and remove from recorder

E. STARTING SUBSEQUENT RECORD TAPES

The program will hold up to 175 DATA entry lines with the standard 8 K user memory bytes, and up to 410 DATA entry lines with the optional 8 K additional memory. It is assumed the new records will be made on new Record Tapes #3 and #4.

1. LOAD and Run #1 Record Tape, Mode #1
2. Write down the grand total amounts displayed for ORIGINAL ASSETS, ADDITIONAL CASH RECEIVED, PAYMENTS TO CLIENTS, and EXPENSES PAID
3. RUN the program again, in Mode #2
4. Request the assigned account number for each account to be carried over to the new records, and write down the individual totals displayed for ORIGINAL ASSETS, ADDITIONAL CASH RECEIVED, PAYMENTS TO CLIENTS, and EXPENSES.
5. Remove the #1 Record Tape from the recorder and install the MASTER Program Tape, LOAD and VERIFY
6. Remove the MASTER Tape, install and rewind the new #3 Record Tape
7. The previous total amounts for each of the 4 categories are carried over to the new tape by re-typing DATA entries for lines 12,13,14 and 15.

   LIST 12-15, these lines will appear on the screen

   a. Type- 12DATA0,1,1, (the amount of ORIGINAL ASSETS), "ORIG CASH ASSETS"

   b. Type- 13DATA0,1,2, (the amount of the ADDITIONAL CASH RECD), "ADDNL CASH RCI

   c. Type- 14DATA0,1,3, (the amount of PAYMENTS TO CLIENTS), "PYTS TO CLIENTS"

   d. Type- 15DATA0,1,4, (the amount of EXPENSES), "EXPENSES PD"
8. Sub-totals for each of the trust accounts to be carried over to the new records are started out by re-entering four lines for each account as was done originally, except that this time the amount variable is entered with the total obtained in 4. above. The entries again start on line 100.

   Ex: Say the Miller case is still in liquidation as of Feb. 15, $22,750.75 was realized from sale of their home, $25,500.00 has been disbursed from the account to heirs, and 2,015.40 of expenses have been paid out. The initial new entries would be:

   ```
   100DATA1,215,1,4728.50,"B OF HAWAII
   101DATA1,215,2,22750.75,"CT # 1374
   102DATA1,215,3,25500.00,"HENRY MILLER
   103DATA1,215,4,2015.40,"DECSD 6/16/77
   ```

   The remaining active accounts from the previous records are entered in the same manner. The individual records plus the grand totals-to-date records entered in lines 12,13,14,and15 as instructed in 7. above, will start the new record tapes off completely up-to-date.
9. Complete all DATA entries for all of the trust accounts to be carried over, LIST and check for accuracy, then RECORD #3 Record Tape, and VERIFY
10. RECORD #4 Record Tape and VERIFY

11. COMPUTER TAPES:

The cassette tapes supplied with the program are made specifically for personal computer use. Your machine demands perfect recordings. Proper tape magnetic characteristics and consistant quality will give you the needed excellent recordings. The "polished" tape with no splices in the "Pilon-30"(tm) cassettes will help to increase recording head life. The fine mechanical construction will minimize jamming or tape stoppage during loading and recording.

These are important business records. ALWAYS use tapes made for computer use. Tapes are available from COMPUTERS ONE ,$5.00 for two tapes plus $.75 shipping and handling. To order tapes by mail:

COMPUTERS ONE
#306 Kahala Office Tower
4211 Waialae Ave.
Honolulu, Hawaii 96816

Tape orders must be pre-paid.

IMPORTANT
Clean the recording head, erase head and the capstan drive at least once a month, as described in your Owners Manual. This is very important to assure consistant results.

REMEMBER- ALWAYS RECORD TWO TAPES - EVERY TIME - NO EXCEPTIONS

"LEGAL DIARY-HUSTLER 1"

"Just LOAD and GO" - ANOTHER OF THE "HUSTLER" SERIES OF PRE-RECORDED BUSINESS PROGRAMS

THE "LEGAL DIARY-HUSTLER 1" PRE-RECORDED COMPUTER PROGRAM - WRITTEN FOR THE COMMODORE PET* 2001 PERSONAL COMPUTER BY K.W. HARRIS

INTRODUCTION:

A lawyer has two things to sell, his time and his skill at his profession. To properly charge his clients, most attorneys keep a running record of the time they spend on a particular case, and any out-of-pocket expenses they incur for that client. Also, they must keep track of what payments have been received, and any balance owed.

The lawyer keeps a daily diary that includes part of this information, his secretary or clerk keeps track of some of this data, so pulling it all together is quite an expensive, time consuming chore.

The "LEGAL DAIRY-HUSTLER 1" pre-recorded tape program replaces the time and financial part of the daily diary, and keeps these records more efficiently, at less cost and with great accuracy. All financial detail information on a client can be recalled in just seconds.

PROGRAM FEATURES:

1. The pre-recorded MASTER cassette tape is ready to use as received. NO computer or programming experience is required. Absolutely NO additions or changes are necessary to permit immediate use. Each program is recorded twice. Just LOAD and GO.

2. As supplied, the program will hold up to 175 DATA lines before a new record tape (or an additional block on the original tape) must be started. If an additional 8 K of user memory (available as an optional extra) is added to the standard 8 K bytes, up to 410 lines can be accommodated.

   When the memory capacity is used up, it is very simple to start another block of records for the active clients.

   Provisions are made in the program to carry over the grand total amounts of "TOTAL HOURS TO DATE", "CLIENTS COSTS TO DATE", "PAYMENTS RECEIVED TO DATE", and "LEGAL CHARGES TO DATE". Also, the sub-amount totals for the active clients are carried over to the new record tapes.

3. Entries are made in random order as time charges are accrued, checks received and deposited, and expenses are paid out. DATA entries are as simple to make as operating a modern cash register.

*Trade Mark Commodore Business Machines

"LEGAL DIARY-HUSTLER 1"

PROGRAM FEATURES: (Cont'd)

4. Entering additional DATA is just as simple and easy. Ease of making entries is a major feature of this program.

5. Program is MULTI-REPORT:

Mode #1- displays each DATA entry in the order it was made, showing the CLIENT number, DATE, CODE number of the entry, AMOUNT, and a short description of the item entered.

- after the detailed DATA lines are printed out on the screen, it displays a running grand total-to-date for: TOTAL HOURS TO DATE, CLIENTS COSTS TO DATE, PAYMENTS RECEIVED TO DATE, TOTAL CHARGES TO DATE, and the computer calculates and displays the BALANCE OWED

Mode #2- on user request, a specific client's individual DATA entries are brought out and displayed line-by-line, according to the client number requested. The client number is assigned at the opening of the account.

- after the detailed line-by-line DATA entries are completely displayed for that particular client account, the figures for TOTAL HOURS TO DATE, CLIENTS COSTS TO DATE, PAYMENTS RECD TO DATE, TOTAL CHARGES TO DATE, and BALANCE OWED on that specific client account are displayed. Then the program resets waiting for the next client number to be requested, or the program to be ended.

Mode #3- separates and displays each of the DATA entries made for each CLIENT account as described for Mode #2, including the account totals at the end of the line-by-line display.
At the conclusion of the display for the first account, the program then moves to the next CLIENT account, displays all the detail DATA entries for that particular account, displays the totals, and then moves on to the next account. This operation is repeated until all the accounts have been displayed.

"LEGAL DAIRY-HUSTLER 1"

"Just LOAD and GO"-ONE OF THE "HUSTLER" SERIES OF PRE-RECORDED BUSINESS PROGRAMS

DEMONSTRATION TAPE:

This pre-recorded demonstration tape has typical entries made in three legal accounts, with entered DATA coded into four essential sub-accounts.

Ease of recalling DATA is illustrated by the almost instant display of the wanted information on the computer screen.

DIRECTIONS:

1. Load demo tape into memory by pressing SHIFT key, hold it down and press RUN key.
2. Press PLAY key on recorder.
3. Screen will display OK, SEARCHING, then FOUND, LOADING.
4. When tape is loaded, press STOP on recorder, REWind, and press STOP again.
5. If screen displays "? LOAD ERROR", repeat the loading steps.
6. When loading is complete, screen will display:

```
READY
RUN
LEGAL ACCOUNTS
ENTER 1- ENTRIES
ENTER 2- CLIENTS BY CODE #
ENTER 3- CLIENT LISTING   and ENTER 4- TOTALS OF ACCOUNTS
```

7. Press the number 1 on keyboard, hit RETURN key
8. The program will list out, a line at a time in order of entry, the complete list of DATA lines entered to date. After the lines are printed out, the screen will display the accumulated grand totals:

```
TOTAL HRS TO DATE          30.00
CLIENTS COSTS TO DATE      59.25
PAYMENTS RECD TO DATE     895.00
TOTAL CHRGS TO DATE      1534.25
BALANCE OWED              639.25
```

9. After a few seconds delay, the program resets and is ready for the next instruction.
10. Press the number 2 on keyboard, hit RETURN key.
11. Screen will display ENTER ACCT # WANTED. 75 ENDS REPORT
12. Accounts #1, #2 and #3 have entries made, so enter any of these numbers, hit RETURN
13. Screen will display all of the detailed DATA entries made for that specific account, then display the sub-totals of the five major accounts for that particular client. Program then re-sets, waiting for the next client account number to be requested, or the program ended.
14. RUN the other accounts.
15. Type RUN again, hit RETURN key
16. Press the number 3 on keyboard, hit RETURN key
17. Screen will display, line-by-line, the entries made for the account having the lowest number. On completion of the detailed DATA lines for that account, the screen will display the sub-total amounts for the five categories.

    When the first account is finished (#1 on the demo tape), the program moves to #2, displays all the detailed DATA lines for this account, and the totals, then moves to #3. When #3 has been completely displayed, the program automatically resets after a short time delay.

The three Modes of display give the user great flexibility in recalling DATA entries. #2 Mode displays the complete details as well as the totals on each account as it is requested. Full information on the client's account is made available in just seconds.

I N S T R U C T I O N S

"LEGAL DIARY-HUSTLER 1"

A. DATA entries for this program are each a single line "in residence" in the program. Each consists of 4 variables and 1 string variable. The variables are separated by a comma, the string variable must start with a quotation mark. See typical entries in 4. below. Note that no "spaces" are used in the DATA line. Each DATA line consists of:

Var. 1 - the line number, the word DATA, clients numbered consecutively 1 through 994, Enter with no space between line number, DATA, client #

Var. 2 - the DATE in numbers (month, day of month) Ex: Jan. 5 would be 105

Var. 3 - the CODE number identifying the type of entry being made:

CODE #1 - Entry is a TIME charge to client's account

CODE #2 - Entry is a COST charge to client's account

CODE #3 - Entry is a PAYMENT RECEIVED to be credited to client's acc't

CODE #4 - Entry is a LEGAL CHARGE to client's account

Var. 4 - the AMOUNT to be charged or credited

- for CODE #1 this figure is hours, expressed as hours and portions of an hour as a decimal. Ex: 3 ½ hours would be entered 3.5
- for CODES #2, 3 and 4, the AMOUNT is entered as whole numbers and decimals. Ex: 450.72

Var. 5 - a short DESCRIPTION starting with a quotation mark (do not exceed 16 letters and numbers). See exception below for the first 4 "label" lines. Ex: "COPY DOCS

B. ORIGINAL START-UP FOR #1 RECORD TAPE

1. Install MASTER tape in recorder, and REWind to start of tape
2. Load MASTER tape into memory: press SHIFT key, hold it down and press RUN key. Then press recorder PLAY key, tape will load. When loaded press recorder STOP key.
3. Remove MASTER program tape, install #1 Record tape and rewind.
4. Type LIST100, hit RETURN key. Rewrite line 100 by entering the initial DATA line for the first client. This program uses 4 DATA lines initially to provide a line for each of the categories #1, #2, #3, and #4. These lines are used to enter the client's name, address, phone number.
If a mistake is made, hit RETURN and retype the entire DATA line.
Assume the first client is William Holden, 25 Crestview Dr., Honolulu, Hawaii 96821. He had his first meeting with Lawyer Adams on Feb. 5, and paid an initial retainer of $250.00. Holden was assigned Client #1.

```
Type-   100DATA1,205,1,0,"WILLIAM HOLDEN     hit RETURN
Type-   101DATA1,205,2,0,"25 CRESTVIEW        "     "
Type-   102DATA1,205,3,0,"DR HON HI           "     "
Type-   103DATA1,205,4,0,"96821               "     "
Type-   104DATA1,205,1,3.5,"0 CONFERENCE      "     "
Type-   105DATA1,205,3,250.00,"RETAINER       "     "
```

Note: The first 4 lines of a specific account will print out only the 'descriptive' part of the line data, so up to 30 letters and numbers can be entered in Variable 5 in each of these 4 lines.

B. ORIGINAL START-UP FOR #1 RECORD TAPE (Cont'd)

4. (Cont'd) These 6 entries complete the first records for Mr. Holden. Any subsequent entries in the program will be made starting with line number 106, properly coded of course.

IMPORTANT:
EACH variable #1, #2, #3, #4 and #5 MUST ALWAYS have something entered. The entry can be 1 digit in each of the first 4 variables, and 1 letter enclosed in quotes for #5, but THERE MUST ALWAYS BE AN ENTRY IN EACH ONE.

5. The initial entries for the next client are made the same way, again starting with 4 lines of initial entries to "label" the client in the display.

   Later on, if the client's account goes over to the next block of entries, the AMOUNT variable location can be used to carry over the TIME, CLIENTS COSTS, PAYMENTS RECEIVED, and the LEGAL CHARGES amounts to the new records by using the initial 4 lines originally set up for identification.

6. Enter the balance of the client DATA entries to start the records, using consecutive line numbers, and starting off each new account with the initial 4 account lines, one line each for CODE #1, CODE #2, CODE #3, CODE #4.

7. RUN the program - Mode #1
   - Mode #2
   - Mode #3

8. RECORD and VERIFY #1 Record tape by typing SAVE, then hit RETURN key Screen will display PRESS PLAY & RECORD ON TAPE #1

9. When recording is complete, rewind tape, type VERIFY, then press PLAY on recorder.

10. Tape will run, then stop. Press STOP button on recorder. VERIFIED will appear on screen. Remove #1 Record tape from recorder.

11. Install #2 Record tape, RECORD and VERIFY by repeating steps 8., 9, and 10.

C. DUPLICATE RECORD TAPES

It is strongly recommended that a duplicate tape ALWAYS be made. A power failure during recording, damage, or loss of the #1 Record tape will still permit the operator to LOAD the previous entries from the second tape, again enter the additional DATA entries, RUN the program and RECORD. If it is necessary to use the #2 tape, be sure to RECORD the #1 tape also.

D. ADDITIONAL ENTRIES

1. LOAD #1 Record Tape into memory, rewind tape
2. LIST program.
3. Type in the line number following the last entry made, complete the new entry
4. Type in the balance of the new entries, using consecutive line numbers.
5. LIST program so new entries can be inspected for accuracy, RUN in MODE #1
6. RECORD and VERIFY #1 Record Tape, rewind and remove from recorder
7. RECORD and VERIFY #2 Record Tape, rewind and remove from recorder.

E. STARTING SUBSEQUENT RECORD TAPES

The program will hold up to 175 DATA entry lines with the standard 8 K user memory bytes, and up to 410 DATA entry lines with the optional 8 K additional memory. It is assumed the new records will be made on new Record Tapes #3 and #4.

1. LOAD and RUN #1 Record Tape, Mode #1
2. Write down the grand total amounts displayed for "TOTAL HOURS T D", "CLIENTS COSTS T D", "PYTS RECD T D", "LEGAL CHGS T D"
3. RUN the program again in MODE #2
4. Enter the assigned number for each client to be carried over to the new records, and write down the individual totals displayed for the "TOTAL HRS T D", "CLIENTS COSTS T D", "PYTS RECD T D", "LEGAL CHGS T D"
5. Remove the #1 Record Tape from the recorder, and install the MASTER Program Tape, LOAD and VERIFY the MASTER Tape.
6. Remove the MASTER Tape, install the new #3 Record Tape and rewind.
7. The previous total amounts for each of the 4 categories are carried over to the new tape by DATA entries on lines 12,13,14,and 15.
   LIST 12-15 these lines will appear on the screen display
   a. Type 12DATA0,1,1, (the amount of the TOTAL HRS TD), "TOTAL HRS T D
   b. Type 13DATA0,1,2, (the amount of the CLIENTS COSTS TD), "CLNTS CSTS T D
   c. Type 14DATA0,1,3, (the amount of the PYTS RCD T D), "PYTS RCD T D
   d. Type 15DATA0,1,4, (the amount of LGL CHGS T D), "LGL CHGS T D
8. Sub-totals for each of the client accounts to be carried over to the new records are started out by re-entering four lines for each client as was done for the client originally, except that this time the AMOUNT variable is entered with the total obtained in 4. above. The entries again start on line 100.

   Mr. Holden, Client #1, is still an active case, so his initial entries in the new records might be:

```
100DATA1,912,1,47.5,"WILLIAM HOLDEN    (Lawyer Adams has charged 47 ½ hours)
101DATA1,912,2,97.50,"25 CRESTVIEW     (Holden's costs to date have been 97.50)
102DATA1,912,3,475.00,"DR HON HI       (Holden has paid 475 to date)
103DATA1,912,4,600.00,"96821           (Legal charges are 600 to date)
```

   The remaining active accounts from the previous record are entered the same as above. The individual records, plus the grand total-to-date entered in lines 12,13,14 and 15 as instructed in 6. above, will start the new record tapes off completely up-to-date.
9. After all DATA entries are completed, RECORD #3 Record Tape, VERIFY
10. RECORD #4 Record Tape and VERIFY.

11. COMPUTER TAPES:

The cassette tapes supplied with this program are made specifically for personal computer use. The machine demands perfect recordings. Proper tape magnetic characteristics and consistant quality will give the needed excellent recordings. The "polished" tape with no splices in the "Pilon-30" (tm) cassettes will help to increase recording head life. The fine mechanical construction will help to minimize jamming or tape stoppage during loading and recording.

These are important records. ALWAYS use tapes made for computer use. Tapes are available from COMPUTERS ONE, $5.00 for two tapes, plus $.75 shipping and handling. To order tapes by mail, write:

COMPUTERS ONE
#306 Kahala Office Tower
4211 Waialae Ave.
Honolulu, Hawaii 96816

Tape orders must be pre-paid

IMPORTANT

Clean the recording head, erase head and the capstan drive at least once a month, as described in your OWNERS MANUAL. This is very important to assure consistant results.

REMEMBER-ALWAYS RECORD TWO TAPES - EVERY TIME - NO EXCEPTIONS

"RENT ACCOUNTS-HUSTLER 1"

"Just LOAD and GO"-ONE OF THE "HUSTLER" Series of Pre-Recorded Business Programs

THE "RENT ACCOUNTS-HUSTLER 1" PRE-RECORDED COMPUTER PROGRAM- WRITTEN FOR THE COMMODORE PET* 2001 PERSONAL COMPUTER BY K.W. HARRIS

INTRODUCTION:

Record keeping for RENT ACCOUNTS is a time consuming operation if there is more than a very small number of accounts to administer. Real estate companies offering rental management services, as well as individuals owning several rental properties will find the RENT ACCOUNTS-HUSTLER 1 program very helpful.

Not only does it keep the complete, detailed financial records of each individual rental property, but it also keeps a running grand total of the major account totals, including any delinquencies in rent or other tenant charges.

Six essential accounts are kept with complete detail on each property: RENT DEPOSITS, RENTS RECEIVED, PAYMENTS MADE TO OWNER, EXPENSES (paid out and charged to that particular property), RENTS OWED, and BALANCE REMAINING IN ACCOUNT.

PROGRAM FEATURES:

1. The pre-recorded MASTER cassette tape is ready to use as received. No computer or programming experience is required. Absolutely NO additions or changes are necessary to permit immediate use. Each program is recorded twice. Just LOAD and GO.

2. As supplied, the program will hold some 175 DATA lines before a new record tape (or an additional block on the original tape) must be started. If an additional 8 K of user memory is added to the original 8 K bytes, some 410 lines can be accomodated by the program.

   When memory capacity is reached, it is very simple to start another block of records for the still active property accounts. Complete directions are included.

   Provisions are made to carry over the grand total amounts of "RENT DEPOSITS", RENTS RECEIVED", "PAYMENTS TO OWNERS", "EXPENSES", "RENTS OWED", and "BALANCE IN ACCOUNT". Also, the sub-total amounts for the individual rental properties can be carried over to the new record tapes.

3. Entries are made in random order, just as money is received and paid out. DATA entries are as simple to make as operating a modern cash register.

4. Entering additional DATA is just as simple and easy. Ease of making new entries was a planned major feature of the program.

*Trade Mark of COMMODORE BUSINESS MACHINES

"RENT ACCOUNTS-HUSTLER 1"

PROGRAM FEATURES: (Cont'd)

5. Program is MULTI-REPORT:

Mode #1- displays each DATA entry in the order it was made, showing the ACCOUNT number, DATE, CODE number for the entry, AMOUNT, ans a short identifying description of the item entered.

- after the detailed DATA lines are completely displayed, the screen displays a running grand total-to-date for: RENT DEPOSITS, RENTS RECEIVED, PAYMENTS TO OWNERS, EXPENSES, RENTS OWED, and BALANCE IN ACCOUNT

Mode #2- on operator request, each individual property has its DATA entries brought out and displayed line-by-line for that particular property, as requested for that particular ACCOUNT number. The property ACCOUNT number is assigned by the operator at the start of the ACCOUNT record.

- after the detailed line-by-line DATA entries are completely displayed for that ACCOUNT, the totals for RENT DEPOSITS, RENTS RECEIVED, PAYMENTS TO OWNERS, EXPENSES, RENTS OWED, andBALANCE IN ACCOUNT for that particular property are displayed. After a short timed delay, the program resets and waits for the next ACCOUNT number to be requested, or the reports ended.

Mode #3- separates and displays each of the DATA entries made for each property ACCOUNT as described for Mode #2, including the totals at the conclusion of the line-by-line display. When the first ACCOUNT is finished, the program then moves to the next property ACCOUNT, displays all the detail DATA entries for that particular property, displays the totals for the ACCOUNT, and then moves to the next ACCOUNT, repeating the operation until all accounts have been displayed. After a short timed delay, the program resets waiting for further instructions.

"RENT ACCOUNTS-HUSTLER 1"

"Just LOAD and GO"-ONE OF THE "HUSTLER" Series of Pre-Recorded Business Programs

DEMONSTRATION TAPE:

This pre-recorded demonstration tape has typical entries that might be required to keep the records on four rental properties. The records might be kept by private individuals for their own rental properties, or by a real estate company on properties managed by them. Ease of recalling DATA is illustrated by the almost instant display of the wanted information on the computer screen.

DIRECTIONS:

1. Load demo tape into memory by pressing SHIFT key, hold it down and press RUN key.
2. Press PLAY key on recorder.
3. Screen will display OK, SEARCHING, then FOUND, LOADING
4. When tape is loaded, press STOP on recorder, REWind, and press STOP again.
5. If screen displays "? LOAD ERROR", repeat the "loading" steps.
6. When loading is complete, the screen will display:

```
READY
RUN
RENT ACCOUNTS
ENTER 1- ENTRIES                    ENTER 3- ACCTS LISTED
ENTER 2- ACCTS BY CODE #            ENTER 4- TOTALS OF ACCOUNTS
```

7. Press the number 1 on keyboard, hit RETURN key
8. The program will list out, a line at a time in order of entry, the complete list of DATA lines entered to date. After the lines are completely printed out, the screen will display the accumulated grand totals:

```
RENT DEPOSITS-      2560.00
RENTS RECD-         3675.00
PYTS TO OWNERS-     1417.40
EXPENSES-            142.96
RENTS OWED-           75.00
BAL IN ACCT-        4174.64
```

Note: The demo program includes entries on delinquencies in ACCT #1 and #2, with entries showing partial repayment in ACCT #2. See detailed instructions on Page 6 for more information on handling delinquent accounts.

9. After a short timed pause, the program resets awaiting further instructions.
10. Press the number 2 on keyboard, hit RETURN key.
11. Screen will display 'ENTER ACCOUNT # WANTED. 75 ENDS REPORT'
12. ACCOUNTS #1,#2,#3, and #11 have entries in the demo program. Enter any of these numbers and hit RETURN key.
13. Screen will display all the detail DATA entries made for that specific account, then display the sub-totals of the six catagories for that particular account. After a short timed delay, program resets, waiting for the next ACCOUNT number to be requested, or the reports ended.
14. Run the other accounts.
15. Wait for the program to reset, then hit 75.
16. Press the number 3 on keyboard, hit RETURN key.
17. Starting with the lowest account number, the screen will display line-by-line the entries made for that account. When the detailed DATA lines are completed for that account, the screen will display the sub-total amounts for the account. When the first account is finished (#1 on the demo tape), the program moves to #2, display all the detail DATA entries and the totals for the account, then moves to #3. On completion of #3 account, #11 will display after a short delay, then program resets.

The three Modes of display give great flexibility in recalling DATA entries. Mode #2 displays the complete information on the specific property, as well as the total amounts for each of the six sub-accounts. The full and complete property information is made available in just seconds.

INSTRUCTIONS

"RENT ACCOUNTS-HUSTLER 1"

A. DATA entries for this program are each a single line "in residence" in the program. Each line consists of 4 "variables" and 1 "string variable". The variables are separated by a comma, the string variable must start with a quotation mark.

Each DATA line consists of:

Var. 1- the line number, the word DATA, the property accounts numbered consecutively 1 thru 994

Var. 2- the DATE in numbers (month,day of month) Ex. Jan. 5 would be 105

Var. 3- The CODE number identifying the type of entry being made:

CODE #1 - Entry is for RENT DEPOSITS

CODE #2 - Entry is for RENTS RECEIVED

CODE #3 - Entry is for PAYMENTS MADE TO OWNERS

CODE #4 - Entry is for EXPENSES PAID

CODE #5 - Entry is for RENTS OWED (also for charges to tenant)

Note: No entry is required for BALANCE IN ACCOUNT, this figure is calculated by the computer

Var. 4 - the AMOUNT to be charged or credited. The figure is entered as a whole number and a decimal. Ex. 450.72

Var. 5- a short DESCRIPTION starting with a quotation mark. Do not exceed 16 letters and numbers. (see exception below for the first 5 "label" lines)

B. ORIGINAL START-UP FOR #1 RECORD TAPE

1. Install MASTER tape into recorder, and REWind to beginning of tape.
2. To load MASTER tape into memory, press SHIFT key, hold it down and press RUN key. Then press recorder PLAY key, tape will load into machine memory.
3. Remove MASTER program tape, install #1 Record tape and rewind.
4. Type LIST100, hit RETURN key Rewrite line 100 by entering the initial DATA line for the first property. The program uses 5 DATA lines initially to provide a line for each of the catagories #1, #2, #3, #4 and #5. These lines are used to "label" the display with tenant name, address, phone number, monthly rental ect. As you noted in the DEMO program, the first 5 lines for a specific account will display only the 'descriptive' part of the line data entered, so up to 30 letters and numbers can be entered in Variable 5. for the first 5 lines of a new account. If a mistake is made, hit RETURN and retype the entire line.
As an example, assume the first property is a residence located at 25 Crestview, Honolulu, Hawaii 96816. Rent is 500.00 per month. A deposit of 500.00 was received Sept. 15, and rent was paid on November 1. #1 was assigned to account.

```
Type-      100DATA1,915,1,0,"RESIDENCE
Type-      101DATA1,915,2,0,"25 CRESTVIEW
Type-      102DATA1,915,3,0,"DR HON HI
Type-      103DATA1,915,4,0,"96816
Type-      104DATA1,915,5,0,"RENT 500/MO
Type-      105DATA1,915,1,500.00,"RENT DEP
Type-      106DATA1,1101,2,500.00,"NOV RENT
```

B. ORIGINAL START-UP FOR #1 RECORD TAPE (Cont'd)

4. These 7 entries complete the first records for Property #1. Any subsequent entries in the program will be made starting with line number 107, properly coded of course.

   IMPORTANT:
   EACH variable #1, #2, #3, #4 and #5 MUST ALWAYS have an entry. The entry can be 1 digit in each of the first 4 variables, and 1 letter enclosed in quotation marks for #5, but THERE MUST ALWAYS BE AN ENTRY IN EACH ONE.

5. The initial entries for the next property are made the same way, again starting with 5 lines of initial entries to "label" the property in the display. Later, if the property account carries over to the next block of entries on a new tape, the RENT DEPOSITS, RENTS RECEIVED, PAYMENTS TO OWNERS, RENTS OWED and EXPENSES PAID can be transferred over to the new records by using the initial 5 lines originally set up for identification.

6. Enter the balance of the property DATA lines to start the records, using consecutive line numbers, and starting off each new account with the initial 5 account lines, one line each for CODE 1, CODE 2, CODE 3, CODE 4 and CODE 5

7. RUN the program with entered DATA, Mode #1, Mode #2, Mode #3. When the final account has been displayed, hit STOP button, then RETURN. READY will display.

8. RECORD and VERIFY #1 Record tape by typing SAVE, then hit RETURN key. Screen will display PRESS PLAY & RECORD ON TAPE #1.

9. When recording is complete, rewind tape, type VERIFY, then press PLAY on the recorder.

10. Tape will run, then stop. Press STOP button on recorder. VERIFIED will appear on screen. Remove #1 Record tape from recorder.

11. Install #2 Record tape, RECORD and VERIFY by repeating steps 8., 9., and 10.

C. DUPLICATE RECORD TAPES

It is STRONGLY recommended that a duplicate record tape ALWAYS be made. A power failure during recording, damage, or loss of the #1 Record tape will still permit the user to load the previous entries from the second tape, again enter the additional DATA entries, RUN the program, RECORD and VERIFY. If the #2 Record tape is used, be sure and RECORD and VERIFY the #1 Record tape also.

D. ADDITIONAL ENTRIES

1. Load #1 Record tape into memory, rewind tape.
2. LIST program.
3. Type in the line number following the last entry made, complete the new entry.
4. Type in the balance of the new entries, using consecutive line numbers.
5. LIST program so new entries can be inspected for accuracy, RUN in Mode #1
6. RECORD and VERIFY #1 Record tape, rewind and remove from recorder
7. RECORD and VERIFY #2 Record tape, rewind and remove from recorder

E. DELINQUENT ACCOUNTS:

The program easily keeps accounts on delinquencies, late charges or other charges to be made against a tenant's account. A normal, timely rent payment for August, Account #2, rent 600.00 per month, would be entered as:

```
113DATA2,806,2,600.00,"AUG RENT
```

On Sept. 4, the tenant tells the manager he can pay only 525.00 against September rent. The following entry using CODE #2 will credit the tenant with his payment-

```
131DATA2,904,2,525.00,"SEPT RENT-PART
132DATA2,904,5,75.00,"SEPT DEL     (this entry starts accruing the delinquency)
```

On Oct. 5, the tenant pays 650.00,which includes 50.00 towards delinquency. Entries:

```
133DATA2,1005,2,650.00,"OCT RENT PYT   (this credits account with full payment)
134DATA2,1005,5,-50.00,"50 PYT ON DEL  (updates the delinquent am't still owed)
```

Line 134 entry requires some additional explanation. Note the payment to decrease the delinquency is CODE 5, but the amount repaid is entered as a negative number, -50.00. This allows the computer to make the account adjustment that decreases the delinquency by the amount of 50.00. Also note the amount to be credited is entered in the 'descriptive'item that starts with a quotation mark. This is necessary. While the arithmetic is done correctly, the program will not print out a number that includes a minus (-) sign. Therefore, to keep track of the amount that was actually credited toward decreasing the delinquency, it is necessary to enter this amount in the 'descriptive' part of the DATA entry line.

The above line numbers are those in the DEMO program. A careful examination of the entries made will show that the actual operation is quite simple.

TO SUMMERIZE:

Regular, timely rent payments- Enter as CODE 2, according to directions
Partial rent payment requires TWO entries:
(a) Whatever payment is made is entered as CODE 2
(b) The delinquent amount(and late charge ect. if applicable) is entered as CODE 5
Repayment of delinquency, late charge ect. is entered as CODE 5, but as a negative number,and the amount credited against delinquency is entered in the 'descriptive' part of the entry.

After the delinquency is fully paid up, the computer will automatically print out RENTS OWED 0.00, as long as no delinquency amount is entered as CODE 5.

F. STARTING SUBSEQUENT RECORD TAPES:

The program will hold up to 175 DATA entry lines with the standard 8 K user memory bytes, and up to 410 DATA entry lines with 8 K additional memory. The following assumes the new records will be made on new Record Tapes #3 and #4.

1. LOAD and RUN #1 Record tape, Mode #1.

2. Write down the grand total amounts displayed for RENT DEPOSITS,RENTS RECEIVED, PAYMENTS TO OWNERS,EXPENSES, and RENTS OWED (if any)

3. RUN the program again, in Mode #2.

4. Request the ACCOUNT # for each property that is to be carried over to the new records, and write down the individual account's totals displayed for the RENT DEPOSITS,RENTS RECEIVED,PAYMENTS TO OWNERS,EXPENSES and RENTS OWED.

5. Remove the #1 Record tape,install the MASTER program tape, LOAD MASTER PROGRAM into machine memory, rewind and verify.

6. Remove the MASTER tape from recorder,install the new #3 Record tape and rewind.

7. The grand total amounts for each of the 5 catagories are carried over by re-typing DATA lines 11,12,13,14 and 15, and inserting the amounts found in Step 2. If any accounts are to be deleted for the new records, the grand totals will have to corrected by subtracting the amounts in the individual deleted accounts from the grand totals.

   LIST 11-15, these lines will appear on the screen:

a. Type- 11DATAØ,1,1,(the amount of the RENT DEPOSITS),"RENT DEPOSITS

b. Type- 12DATAØ,1,2,(the amount of the RENTS RECEIVED),"RENTS REC'D

c. Type- 13DATAØ,1,3,(the amount of PYTS TO OWNERS),"PYTS TO OWNERS

d. Type- 14DATAØ,1,5,(the amount of RENTS OWED),"RENTS OWED

e. Type- 15DATAØ,1,4,(the amount of EXPENSES),"EXPENSES

8. Sub-totals for each individual property account to be carried over to the new records are started out by re-entering the 5 "label" lines for each property account as was done originally, except this time the AMOUNT variable (Var.4) is entered with the totals obtained in 4. above. The entries again start on line 100.

   Example: 25 Crestview, property #1 is still an active management account, so the initial entries in the new records might be:

```
100DATA1,912,1,500.00,"RESIDENCE
101DATA1,912,2,3600.00,"25 CRESTVIEW
102DATA1,912,3,3000.00,"DR HON HI
103DATA1,912,4,380.75,"96816
105DATA1,912,5,0,"RENTS OWED
```

   The remaining active accounts from the previous records are entered with their initial 5 "label" entries in the same manned as described above. The individual records, plus the grand totals entered in lines 11,12,13,14 and 15 complete the start-off entries.After all accounts are entered, RECORD #3 Record tape,VERIFY. Then RECORD #4 Record tape,and VERIFY.

G. REMOVING A RECORD WHEN AN ACCOUNT BECOMES INACTIVE:

The working Record tapes should be kept current at all times. Rent accounts become inactive for many reasons (sale of property, owner moves in, another property manager is appointed, ect.) or there is a tenant change. A new tenant must have an entirely new account number, and the old account closed out.

a. Install #1 Record tape and LOAD into machine memory.
b. LIST the program until the line is found where the initial RENT DEPOSIT (CODE #1) was entered into the account. Carefully write down the content of the entire DATA line, including the line number.
c. Re-write the DATA line as a REM line, with information on the date, check number, amount of RENT DEPOSIT returned, ect.

EXAMPLE: USING THE DEMO TAPE - ACCOUNT #2

Original DEPOSIT line was: 111DATA2,801,1,600.00,"DEPOSIT
Re-type 111 to read: 111REM"DEP RET 11/15-CHK#1701-600.00" (if entire deposit was returned to tenant by check # 1701)

d. Hit RETURN, then RUN program in Mode #1. The totals now reflect the subtraction of the rent deposit in both RENT DEPOSITS and BAL IN ACCOUNT. The REM line will not print out, but can be read by listing the program.

e. If 50.00 of the deposit was required for repairs, and tenant owed 25.00 back rent, four line entries are required to up-date and close out the account.

111REM"525.00 DEP RET 11/15 CHK # 1701(600.00-50.00 REP & 25.00 DEL R)

To correct the accounts and credit tenant with the 75.00 amount withheld, LIST to the end of the DATA lines and make 3 additional entries for ACCT #2:

144DATA2,1115,2,50.00,"FOR REP AT MOVE-OUT (credits tenant with payment)
145DATA2,1115,4,50.00,"1702-REPAIRS PD (corrects account for repairs made)
146DATA2,1115,5,-25.00,"PYT-DEL RENT (credits tenant with payment)

RUN program in Mode #1. All totals are corrected for the deletions and credits.

f. RUN program again in Mode #2. Request ACCT #2. There is still a balance in the account after the DEPOSIT was returned. This money belongs to the owner, LIST to the end of the program. Make a final DATA line entry after the last DATA line for the amount still in the account, as a final payment to owner. Use the ACCOUNT number 2 and the CODE number #3 as usual for the entry, and include the check number in the "description" item of the line.

RUN the program again, all accounts should balance, and the deleted account BAL IN ACCT should be 0.00. Note; the RENTS RECEIVED total amount was increased by the 50.00 withheld for repairs.

If actual payment to owner is not required, re-enter the amount of the BAL IN ACCT in the NEW tenant account as CODE #2 , RENTS RECEIVED, in the new second "initial entry" line for the new tenant.

The same account number should not be re-assigned to a new account as long as any record records exist as carry-overs, ect. to new record tapes.

After all changes are made, and accounts checked carefully, RECORD and VERIFY both Record tape #1 and Record tape #2.

NOTIZEN

DATENBLÄTTER

R6500 Mikroprozessor
R6500B (3MHz)
R6520 Peripherer Interfaceadapter
R6522 Versatile Interfaceadapter
Produktbeschreibung für 6522
R2114 RAM
SY2316 ROM
R6541 Programmierbarer Keyboard-Kontroller

NOTIZEN

DOCUMENT NO. 29000 D39
REVISION 2, AUGUST 1978

PART NUMBER
R650X and R651X

R6500 Microcomputer System

DATA SHEET

R6500 MICROPROCESSORS (CPU's)

SYSTEM ABSTRACT

The 8-bit R6500 microcomputer system is produced with N-Channel, Silicon Gate technology. Its performance speeds are enhanced by advanced system architecture. This innovative architecture results in smaller chips – the semiconductor threshold to cost-effectivity. System cost-effectivity is further enhanced by providing a family of 10 software-compatible microprocessor (CPU) devices, described in this document. Rockwell also provides memory and microcomputer system . . . as well as low-cost design aids and documentation.

R6500 MICROPROCESSOR (CPU) CONCEPT

Ten CPU devices are available. All are software-compatible. They provide options of addressable memory, interrupt input, on-chip clock oscillators and drivers. All are bus-compatible with earlier generation microprocessors like the M6800 devices.

The family includes six microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs, crystal or RC inputs provide the time base. The external clock versions are geared for multiprocessor system applications where maximum timing control is mandatory. All R6500 microprocessors are also available in a variety of packaging (ceramic and plastic), operating frequency (1 MHz and 2 MHz) and temperature (commercial, industrial and military) versions.

MEMBERS OF THE R6500 MICROPROCESSOR (CPU) FAMILY

Microprocessors with On-Chip Clock Oscillator

Model	Addressable Memory
R6502	65K Bytes
R6503	4K Bytes
R6504	8K Bytes
R6505	4K Bytes
R6506	4K Bytes
R6507	8K Bytes

Microprocessors with External Two Phase Clock Output

Model	Addressable Memory
R6512	65K Bytes
R6513	4K Bytes
R6514	8K Bytes
R6515	4K Bytes

FEATURES

- Single +5V supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- Use with any type of speed memory
- 8-bit Bidirectional Data Bus
- Addressable memory range of up to 65K bytes
- "Ready" input
- Direct Memory Access capability
- Bus compatible with M6800
- 1 MHz and 2 MHz operation
- Choice of external or on-chip clocks
- On-the-chip clock options
  - External single clock input
  - RC time base input
  - Crystal time base input
- Commercial, industrial and military temperature versions
- Pipeline architecture

Ordering Information

Order Number: R65XX _ _ _

- Temperature Range:
  - No suffix = 0°C to +70°C
  - E = −40°C to +85°C (Industrial)
  - MT = −55°C to +125°C (Military)
  - M = MIL-STD-883, Class B
- Package:
  - C = Ceramic
  - P = Plastic (Not Avaible for M or MT suffix)
- Frequency Range:
  - No suffix = 1 MHz
  - A = 2 MHz
- Model Designator:
  - XX = 02, 03, 04, ... 15

NOTE: Contact your local Rockwell Representative concering availability.

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

R6500 Signal Description

Clocks (ϕ_1, ϕ_2)

The R651X requires a two phase non-overlapping clock that runs at the V_{CC} voltage level.

The R650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled.

Address Bus (A0-A15)

These outputs are TTL compatible, capable of driving one standard TTL load and 130 pF.

Data Bus (D0-D7)

Eight pins are used for the data bus. This is a bidirectional bus, transferring data to and from the device and peripherals. The outputs are tri-state buffers capable of driving one standard TTL load and 130 pF.

Data Bus Enable (DBE)

This TTL compatible input allows external control of the tri-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation DBE would be driven by the phase two (ϕ_2) clock, thus allowing data output from microprocessor only during ϕ_2. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low.

Ready (RDY)

This input signal allows the user to halt or single cycle the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one (ϕ_1) will halt the microprocessor with the output address lines reflecting the current address being fetched. If Ready is low during a write cycle, it is ignored until the following read operation. This condition will remain through a subsequent phase two (ϕ_2) in which the Ready signal is low. This feature allows microprocessor interfacing with the low speed PROMS as well as fast (max. 2 cycle). Direct Memory Access (DMA).

Interrupt Request ($\overline{IRQ}$)

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, therefore transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)

A negative going edge on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state interrupt mask flag. The vector address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ register to V_{CC} for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during ϕ_2 (phase 2) and will begin the appropriate interrupt routine on the ϕ_1 (phase 1) following the completion of the current instruction.

Set Overflow Flag (S.O.)

A negative going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of ϕ_1 and must be externally synchronized.

SYNC

This output line is provided to identify those cycles in which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during ϕ_1 of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the ϕ_1 clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

Reset

This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

After V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/W and (SYNC) signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

ADDRESSING MODES

ACCUMULATOR ADDRESSING – This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

IMMEDIATE ADDRESSING – In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

ABSOLUTE ADDRESSING – In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

ZERO PAGE ADDRESSING – The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

INDEXED ZERO PAGE ADDRESSING – (X, Y indexing) – This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

INDEXED ABSOLUTE ADDRESSING – (X, Y indexing) – This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

IMPLIED ADDRESSING – In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

RELATIVE ADDRESSING – Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

INDEXED INDIRECT ADDRESSING – In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

INDIRECT INDEXED ADDRESSING – In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

ABSOLUTE INDIRECT – The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter.

INSTRUCTION SET – ALPHABETIC SEQUENCE

ADC Add Memory to Accumulator with Carry
AND "AND" Memory with Accumulator
ASL Shift left One Bit (Memory or Accumulator)

BCC Branch on Carry Clear
BCS Branch on Carry Set
BEQ Branch on Result Zero
BIT Test Bits in Memory with Accumulator
BMI Branch on Result Minus
BNE Branch on Result not Zero
BPL Branch on Result Plus
BRK Force Break
BVC Branch on Overflow Clear
BVS Branch on Overflow Set

CLC Clear Carry Flag
CLD Clear Decimal Mode
CLI Clear Interrupt Disable Bit
CLV Clear Overflow Flag
CMP Compare Memory and Accumulator
CPX Compare Memory and Index X
CPY Compare Memory and Index Y

DEC Decrement Memory by One
DEX Decrement Index X by One
DEY Decrement Index Y by One

EOR "Exclusive-or" Memory with Accumulator

INC Increment Memory by One
INX Increment Index X by One
INY Increment Index Y by One

JMP Jump to New Location
JSR Jump to New Location Saving Return Address

LDA Load Accumulator with Memory
LDX Load Index X with Memory
LDY Load Index Y with Memory
LSR Shift One Bit Right (Memory or Accumulator)

NOP No Operation

ORA "OR" Memory with Accumulator

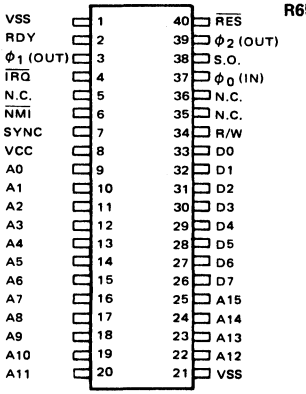
PHA Push Accumulator on Stack
PHP Push Processor Status on Stack
PLA Pull Accumulator from Stack
PLP Pull Processor Status from Stack

ROL Rotate One Bit Left (Memory or Accumulator)
ROR Rotate One Bit Right (Memory or Accumulator)
RTI Return from Interrupt
RTS Return from Subroutine

SBC Subtract Memory from Accumulator with Borrow
SEC Set Carry Flag
SED Set Decimal Mode
SEI Set Interrupt Disable Status
STA Store Accumulator in Memory
STX Store Index X in Memory
STY Store Index Y in Memory

TAX Transfer Accumulator to Index X
TAY Transfer Accumulator to Index Y
TSX Transfer Stack Pointer to Index X
TXA Transfer Index X to Accumulator
TXS Transfer Index X to Stack Register
TYA Transfer Index Y to Accumulator

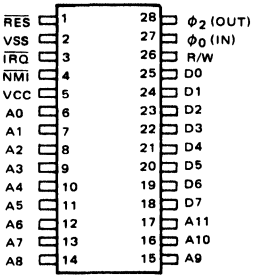
R6502 — 40 Pin Package



Features of R6502

- 65K Addressable Bytes of Memory (A0-A15)
- IRQ Interrupt
- On-the-chip Clock
  - TTL Level Single Phase Input
  - RC Time Base Input
  - Crystal Time Base Input
- SYNC Signal
  (can be used for single instruction execution)
- RDY Signal
  (can be used to halt or single cycle execution)
- Two Phase Output Clock for Timing of Support Chips
- NMI Interrupt

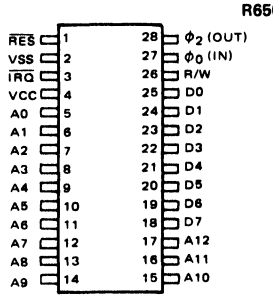
R6503 — 28 Pin Package



Features of R6503

- 4K Addressable Bytes of Memory (A0-A11)
- On-the-chip Clock
- IRQ Interrupt
- NMI Interrupt
- 8 Bit Bidirectional Data Bus

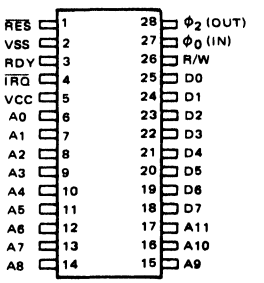
R6504 — 28 Pin Package



Features of R6504

- 8K Addressable Bytes of Memory (A0-A12)
- On-the-chip Clock
- IRQ Interrupt
- 8 Bit Bidirectional Data Bus

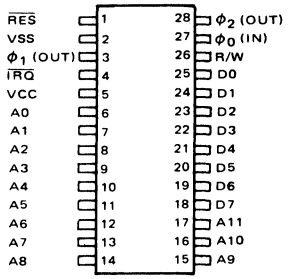
R6505 — 28 Pin Package



Features of R6505

- 4K Addressable Bytes of Memory (A0-A11)
- On-the-chip Clock
- IRQ Interrupt
- RDY Signal
- 8 Bit Bidirectional Data Bus

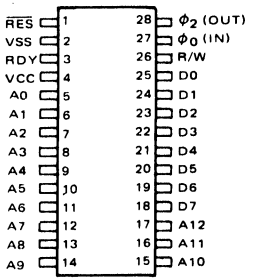
R6506 – 28 Pin Package



Features of R6506

- 4K Addressable Bytes of Memory (A0-A11)
- On-the-chip Clock
- $\overline{IRQ}$ Interrupt
- Two phase output clock for timing of support chips
- 8 Bit Bidirectional Data Bus

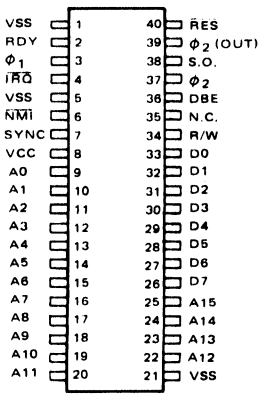
R6507 – 28 Pin Package



Features of R6507

- 8K Addressable Bytes of Memory (A0-A12)
- On-the-chip Clock
- RDY Signal
- 8 Bit Bidirectional Data Bus

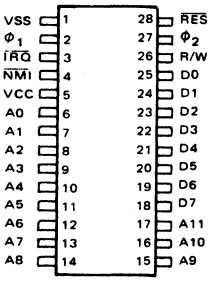
R6512 – 40 Pin Package



Features of R6512

- 65K Addressable Bytes of Memory (A0-A15)
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- RDY Signal
- 8 Bit Bidirectional Data Bus
- SYNC Signal
- Two phase clock input
- Data Bus Enable

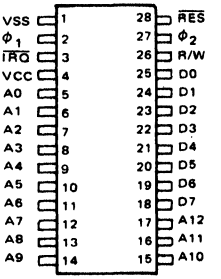
R6513 – 28 Pin Package



Features of R6513

- 4K Addressable Bytes of Memory (A0-A11)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- 8 Bit Bidirectional Data Bus

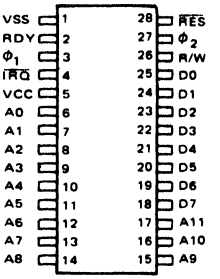
R6514 – 28 Pin Package



Features of R6514

- 8K Addressable Bytes of Memory (A0-A12)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- 8 Bit Bidirectional Data Bus

R6515 – 28 Pin Package



Features of R6515

- 4K Addressable Bytes of Memory (A0-A11)
- Two phase clock input
- $\overline{IRQ}$ Interrupt
- RDY Signal
- 8 Bit Bidirectional Data Bus

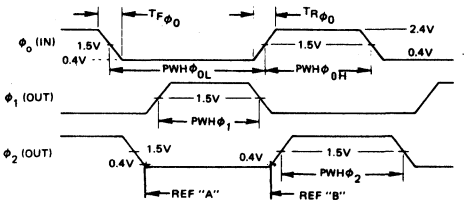
INSTRUCTION SET

MNEMONIC	OPERATION	IMMEDIATE OP	IMMEDIATE n	IMMEDIATE #	ABSOLUTE OP	ABSOLUTE n	ABSOLUTE #	ZERO PAGE OP	ZERO PAGE n	ZERO PAGE #	ACCUM OP	ACCUM n	ACCUM #	IMPLIED OP	IMPLIED n	IMPLIED #	(IND, X) OP	(IND, X) n	(IND, X) #	(IND), Y OP	(IND), Y n	(IND), Y #	Z. PAGE, X OP	Z. PAGE, X n	Z. PAGE, X #	ABS. X OP	ABS. X n	ABS. X #	ABS. Y OP	ABS. Y n	ABS. Y #	RELATIVE OP	RELATIVE n	RELATIVE #	INDIRECT OP	INDIRECT n	INDIRECT #	Z. PAGE, Y OP	Z. PAGE, Y n	Z. PAGE, Y #	PROCESSOR STATUS CODES 7 6 5 4 3 2 1 0 N V - B D I Z C	MNEMONIC
ADC	A + M + C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										N V · · · · Z C	ADC
AND	A ∧ M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										N · · · · · Z ·	AND
ASL	C ← 7 6 5 4 3 2 1 0 ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													N · · · · · Z C	ASL
BCC	BRANCH ON C = 0 (2)																															90	2	2								· · · · · · · ·	BCC
BCS	BRANCH ON C = 1 (2)																															B0	2	2								· · · · · · · ·	BCS
BEQ	BRANCH ON Z = 1 (2)																															F0	2	2								· · ⊕ · · · · ·	BEQ
BIT	A ∧ M				2C	4	3	24	3	2																															M_7 M_6 · · · · Z ·	BIT
BMI	BRANCH ON N = 1 (2)																															30	2	2								· · · · · · · ·	BMI
BNE	BRANCH ON Z = 0 (2)																															D0	2	2								· · · · · · · ·	BNE
BPL	BRANCH ON N = 0 (2)																															10	2	2								· · · · · · · ·	BPL
BRK	BREAK													00	7	1																									· · · · 1 · 1 · ·	BRK
BVC	BRANCH ON V = 0 (2)																															50	2	2								· · · · · · · ·	BVC
BVS	BRANCH ON V = 1 (2)																															70	2	2								· · · · · · · ·	BVS
CLC	0 → C													18	2	1																									· · · · · · · 0	CLC
CLD	0 → D													D8	2	1																									· · · · 0 · · ·	CLD
CLI	0 → I													58	2	1																									· · · · · 0 · ·	CLI
CLV	0 → V													B8	2	1																									· 0 · · · · · ·	CLV
CMP	A − M	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										N · · · · · Z C	CMP
CPX	X − M	E0	2	2	EC	4	3	E4	3	2																															N · · · · · Z C	CPX
CPY	Y − M	C0	2	2	CC	4	3	C4	3	2																															N · · · · · Z C	CPY
DEC	M − 1 → M				CE	6	3	C6	5	2																DE	7	3													N · · · · · Z ·	DEC
DEX	X − 1 → X													CA	2	1																									N · · · · · Z ·	DEX
DEY	Y − 1 → Y													88	2	1																									N · · · · · Z ·	DEY
EOR	A ∨ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										N · · · · · Z ·	EOR
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													N · · · · · Z ·	INC
INX	X + 1 → X													E8	2	1																									N · · · · · Z ·	INX
INY	Y + 1 → Y													C8	2	1																									N · · · · · Z ·	INY
JMP	JUMP TO NEW LOC				4C	3	3																												6C	5	3				· · · · · · · ·	JMP
JSR	JUMP SUB				20	6	3																																		· · · · · · · ·	JSR
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										N · · · · · Z ·	LDA
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	N · · · · · Z ·	LDX
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													N · · · · · Z ·	LDY
LSR	0 → 7 6 5 4 3 2 1 0 → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 · · · · · Z C	LSR
NOP	NO OPERATION													EA	2	1																									· · · · · · · ·	NOP
ORA	A ∨ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										N · · · · · Z ·	ORA
PHA	A → Ms S − 1 → S													48	3	1																									· · · · · · · ·	PHA
PHP	P → Ms S − 1 → S													08	3	1																									· · · · · · · ·	PHP
PLA	S + 1 → S Ms → A													68	4	1																									N · · · · · Z ·	PLA
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)	PLP
ROL	7 6 5 4 3 2 1 0 ← C				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													N · · · · · Z C	ROL
ROR	C → 7 6 5 4 3 2 1 0				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													N · · · · · Z C	ROR
RTI	RTRN INT													40	6	1																									(RESTORED)	RTI
RTS	RTRN SUB													60	6	1																									· · · · · · · ·	RTS
SBC	A − M − C → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										N V · · · · Z (3)	SBC
SEC	1 → C													38	2	1																									· · · · · · · 1	SEC
SED	1 → D													F8	2	1																									· · · · 1 · · ·	SED
SEI	1 → I													78	2	1																									· · · · · 1 · ·	SEI
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										· · · · · · · ·	STA
STX	X → M				8E	4	3	86	3	2																												96	4	2				· · · · · · · ·	STX
STY	Y → M				8C	4	3	84	3	2													94	4	2																	· · · · · · · ·	STY
TAX	A → X													AA	2	1																									N · · · · · Z ·	TAX
TAY	A → Y													A8	2	1																									N · · · · · Z ·	TAY
TSX	S → X													BA	2	1																									N · · · · · Z ·	TSX
TXA	X → A													8A	2	1																									N · · · · · Z ·	TXA
TXS	X → S													9A	2	1																									· · · · · · · ·	TXS
TYA	Y → A													98	2	1																									N · · · · · Z ·	TYA

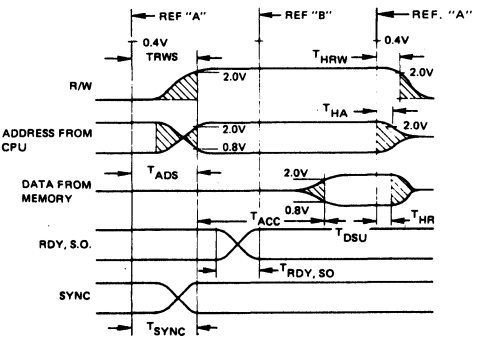
(1) ADD 1 to 'N' IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO 'N' IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO 'N' IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT = BORROW
(4) IF IN DECIMAL MODE, Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X	INDEX X	+	ADD
Y	INDEX Y	-	SUBTRACT
A	ACCUMULATOR	∧	AND
M	MEMORY PER EFFECTIVE ADDRESS	∨	OR
Ms	MEMORY PER STACK POINTER	∨	EXCLUSIVE OR
M_7	MEMORY BIT 7		
M_6	MEMORY BIT 6		
n	NO. CYCLES		
#	NO. BYTES		

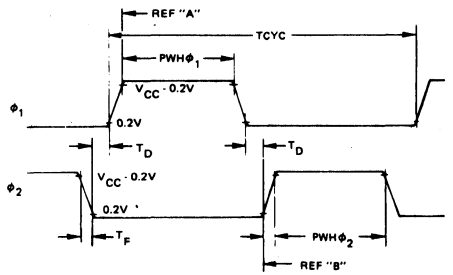
Clock Timing – R6502, 03, 04, 05, 06, 07



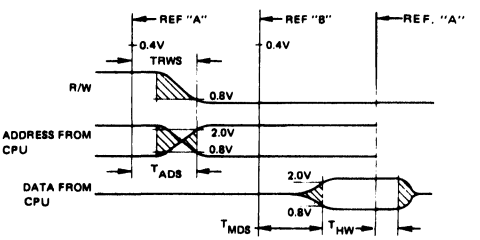
Timing for Reading Data from Memory or Peripherals



Clock Timing – R6512, 13, 14, 15

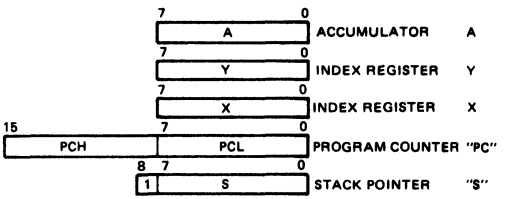


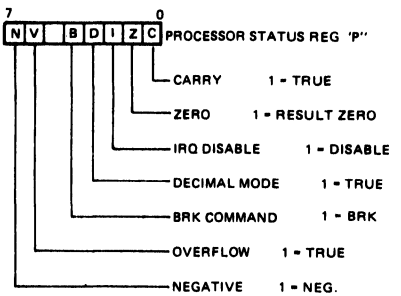
Timing for Writing Data to Memory or Peripherals



Note: "REF." means Reference Points on clocks.

PROGRAMMING MODEL





1 MHz Timing

Clock Timing – R6502, 03, 04, 05, 06, 07

Characteristic	Symbol	Min	Typ	Max	Units
Cycle Time	T_{CYC} *	1000	–	–	ns
$\phi_{0(IN)}$ Pulse Width (measured at 1.5V)	$PWH\phi_0$	460	–	520	ns
$\phi_{0(IN)}$ Rise, Fall Time	$TR\phi_0$, $TF\phi_0$	–	–	10	ns
Delay Time Between Clocks (measured at 1.5V)	T_D	5	–	–	ns
$\phi_{1(OUT)}$ Pulse Width (measured at 1.5V)	$PWH\phi_1$	$PWH\phi_{0L}$-20	–	$PWH\phi_{0L}$	ns
$\phi_{2(OUT)}$ Pulse Width (measured at 1.5V)	$PWH\phi_2$	$PWH\phi_{0H}$-40	–	$PWH\phi_{0H}$-10	ns
$\phi_{1(OUT)}$, $\phi_{2(OUT)}$ Rise, Fall Time (measured at 0.8V to 2.0V) (Load = 30 pF +1 TTL)	T_R, T_F	–	–	25	ns

2 MHz Timing

Clock Timing – R6502, 03, 04, 05, 06, 07

Characteristic	Symbol	Min	Typ	Max	Units
Cycle Time	T_{CYC} *	500	–	–	ns
$\phi_{0(IN)}$ Pulse Width (measured at 1.5V)	$PWH\phi_0$	240	–	260	ns
$\phi_{0(IN)}$ Rise, Fall Time	$TR\phi_0$, $TF\phi_0$	–	–	10	ns
Delay Time Between Clocks (measured at 1.5V)	T_D	5	–	–	ns
$\phi_{1(OUT)}$ Pulse Width (measured at 1.5V)	$PWH\phi_1$	$PWH\phi_{0L}$-20	–	$PWH\phi_{0L}$	ns
$\phi_{2(OUT)}$ Pulse Width (measured at 1.5V)	$PWH\phi_2$	$PWH\phi_{0H}$-40	–	$PWH\phi_{0H}$-10	ns
$\phi_{1(OUT)}$, $\phi_{2(OUT)}$ Rise, Fall Time (measured at 0.8V to 2.0V) (Load = 30 pF + 1 TTL)	T_R, T_F	–	–	25	ns

*The lowest operating frequency for the commercial temperature range CPU's is 100 KHz, which corresponds to a maximum cycle time (TCYC) of 10 μs. The lowest operating frequency for the industrial and military temperature range CPU's is 250 KHz, which corresponds to a maximum cycle time (T_{CYC}) of 4 μs.

Clock Timing – R6512, 13, 14, 15

Characteristic	Symbol	Min	Typ	Max	Units
Cycle Time	T_{CYC} *	1000	–	–	ns
Clock Pulse Width $\phi1$ (Measured at Vcc - 0.2V) $\phi2$	PWH $\phi1$ PWH $\phi1$	430 470	–	–	ns
Fall Time (Measured from 0.2V to Vcc - 0.2V)	T_F	–	–	25	ns
Delay Time between Clocks (Measured at 0.2V)	T_D	0	–	–	ns

Clock Timing – R6512, 13, 14, 15

Characteristic	Symbol	Min	Typ	Max	Units
Cycle Time	T_{CYC} *	500	–	–	ns
Clock Pulse Width $\phi1$ (Measured at Vcc - 0.2V) $\phi2$	PWH $\phi1$ PWH $\phi2$	215 235	–	–	ns
Fall Time (measured from 0.2V to Vcc - 0.2V)	T_F	–	–	12	ns
Delay Time between Clocks (measured at 0.2V)	T_D	0	–	–	ns

Read/Write Timing **

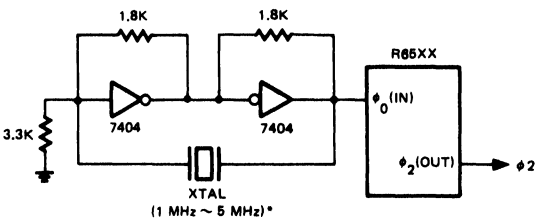
Characteristic	Symbol	Min	Typ	Max	Units
Read/Write Setup Time from R6500	T_{RWS}	–	100	225	ns
Address Setup Time from R6500	T_{ADS}	–	100	225	ns
Memory Read Access Time	T_{ACC}	–	–	650	ns
Data Stability Time Period	T_{DSU}	100	–	–	ns
Data Hold Time – Read	T_{HR}	10	–	–	ns
Data Hold Time – Write	T_{HW}	60	60	–	ns
Data Setup Time from R6500	T_{MDS}	–	150	175	ns
RDY, S.O. Setup Time	T_{RDY}	100	–	–	ns
SYNC Setup Time from R6500	T_{SYNC}	–	–	225	ns
Address Hold Time	T_{HA}	30	60	–	ns
R/W Hold Time	T_{HRW}	30	60	–	ns

Read/Write Timing **

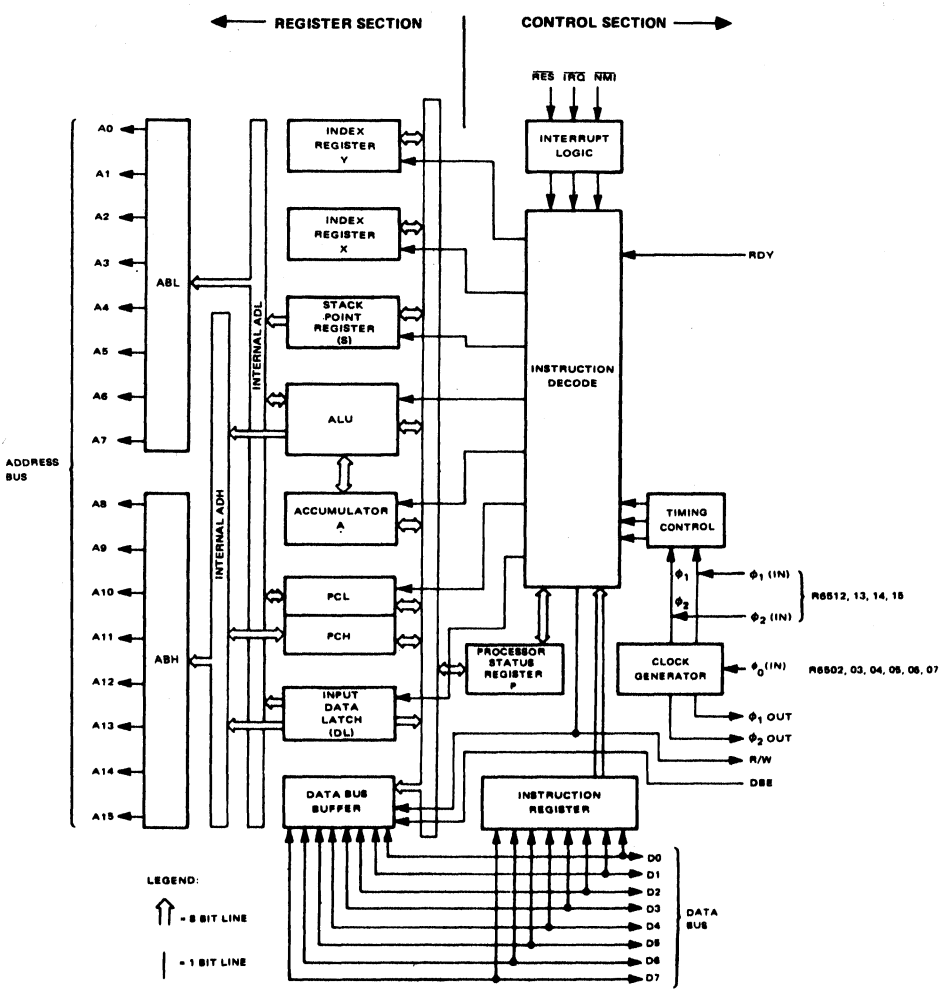
Characteristic	Symbol	Min	Typ	Max	Units
Read/Write Setup Time from R6500A	T_{RWS}	–	75	140	ns
Address Setup Time from R6500A	T_{ADS}	–	75	140	ns
Memory Read Access Time	T_{ACC}	–	–	310	ns
Data Stability Time Period	T_{DSU}	80	–	–	ns
Data Hold Time – Read	T_{HR}	10	–	–	ns
Data Hold Time – Write	T_{HW}	60	60	–	ns
Data Setup Time from R6500A	T_{MDS}	–	75	100	ns
RDY, S.O. Setup Time	T_{RDY}	60	–	–	ns
SYNC Setup Time from R6500A	T_{SYNC}	–	–	180	ns
Address Hold Time	T_{HA}	30	60	–	ns
R/W Hold Time	T_{HRW}	30	60	–	ns

** Load Conditions = 1 TTL Load + 130 pf

RECOMMENDED TIME BASE GENERATION



*CRYSTAL: CTS KNIGHTS MP SERIES, OR EQUIVALENT



Note: 1. Clock Generator is not included on R6512, 13, 14, 15
2. Addressing Capability and control options vary with each of the R6500 Products.

R6500 Internal Architecture

SPECIFICATIONS

Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature	T_A		°C
Commercial		0 to +70	
Industrial		-40 to +85	
Military		-55 to +125	
Storage Temperature	T_{STG}	-55 to +150	°C

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

Electrical Characteristics

(V_{CC} = 5.0±5%, V_{SS} = 0)

ϕ_1, ϕ_2 applies to R6512, 13, 14, 15, $\phi_{o(in)}$ applies to R6502, 03, 04, 05, 06 and 07.

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	V_{IH}				Vdc
Logic, $\phi_{o(in)}$		V_{SS} + 2.4	–	V_{CC}	
ϕ_1, ϕ_2		V_{CC} - 0.2	–	V_{CC} + 0.25	
Input Low Voltage	V_{IL}				Vdc
Logic, $\phi_{o(in)}$		V_{SS} - 0.3	–	V_{SS} + 0.4	
ϕ_1, ϕ_2		V_{SS} - 0.3	–	V_{SS} + 0.2	
Input High Threshold Voltage	V_{IHT}				Vdc
$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.		V_{SS} + 2.0	–	–	
Input Low Threshold Voltage	V_{ILT}				Vdc
$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.		–	–	V_{SS} + 0.8	
Input Leakage Current	I_{in}				μA
(V_{in} = 0 to 5.25V, V_{CC} = 0)					
Logic (Excl. RDY, S.O.)		–	–	2.5	
ϕ_1, ϕ_2		–	–	100	
$\phi_{o(in)}$		–	–	10.0	
Three-State (Off State) Input Current	I_{TSI}				μA
(V_{in} = 0.4 to 2.4V, V_{CC} = 5.25V)					
Data Lines		–	–	10	
Output High Voltage	V_{OH}				Vdc
(I_{LOAD} = -100 μAdc, V_{CC} = 4.75V)					
SYNC, Data, A0-A15, R/W, ϕ_1, ϕ_2		V_{SS} + 2.4	–	–	
Output Low Voltage	V_{OL}				Vdc
(I_{LOAD} = 1.6 mAdc, V_{CC} = 4.75V)					
SYNC, Data, A0-A15, R/W, ϕ_1, ϕ_2		–	–	V_{SS} + 0.4	
Power Dissipation	P_D				W
Commercial temp. versions		–	0.25	0.575	
Industrial and military temp. versions		–	0.25	0.700	
Capacitance at 25°C	C				pF
(V_{in} = 0, f = 1 MHz)					
Logic	C_{in}	–	–	10	
Data		–	–	15	
A0-A15, R/W, SYNC	C_{out}	–	–	12	
$\phi_{o(in)}$	$C_{\phi o(in)}$	–	–	15	
ϕ_1	$C_{\phi 1}$	–	30	50	
ϕ_2	$C_{\phi 2}$	–	50	80	

Note: $\overline{IRQ}$ and $\overline{NMI}$ require 3K pull-up resistors.

DOCUMENT NO. 29000 D39S1
OCTOBER 1978

PART NUMBER
R65XXB

R6500 Microcomputer System
DATA SHEET SUPPLEMENT

R6500B SERIES (3 MHZ) MICROPROCESSORS

The Rockwell R6500B series is a high-performance addition to the advanced architecture R6500 8-bit microprocessor family. The family includes 10 microprocessor (CPU) devices – six CPUs have on-chip clock oscillators and drivers, four CPUs are driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single-phase inputs, crystal or RC inputs provide the time base. The external clock versions are used in multi-processor system applications. All members of the R6500 family are totally software compatible.

The R6500B series microprocessors operate at a 3 MHz clock rate, providing an instruction cycle of less than 0.7 microseconds. The R6500B microprocessors are available in ceramic and molded plastic packages (order R65XXBC or R65XXBP, respectively) and operate at 0°C to +70°C.

The common characteristics for each microprocessor in the R6500 family are contained in the R6500 Microprocessor Data Sheet, Document Number 29000 D39. Specifications unique to the R6500B series are listed in this document.

FEATURES

- 3 MHz clock rate
- Single +5V supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- 8-bit Bidirectional Data Bus
- Addressable memory range of up to 65K bytes
- "Ready" input
- Direct Memory Access Capability
- Bus compatible with M6800
- Choice of external or on-chip clocks
- Pipeline architecture
- 0°C to +70°C operation

R6500 CPU OPTIONS

	40-Pin DIP		18-Pin DIP				
	R6502	R6512	R6503 R6513	R6504 R6514	R6505 R6515	R6506	R6507
Memory Address Space	65K	65K	4K	8K	4K	4K	8K
Interrupts – Maskable – Non-Maskable	Yes Yes	Yes Yes	Yes Yes	Yes No	Yes No	Yes No	No No
SYNC – Output indicates op code fetch cycle	Yes	Yes	No	No	No	No	No
RDY – Single step and slow memory synchronization	Yes	Yes	No	No	Yes	No	Yes
Ø1 Clock Output	Yes	Yes	No	No	No	Yes	No
DBE – Extended Data Bus Hold Time	No	Yes	No	No	No	No	No

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

SPECIFICATIONS

Maximum Ratings

V_{CC}, Supply Voltage	-0.3 to +7.0 Vdc
V_{IN}, Input Voltage	-0.3 to +7.0 Vdc
T_A, Operating Temperature	0 to +70°C
T_{STG}, Storage Temperature	-55 to +150°C

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

Electrical Characteristics (V_{CC} = 5.0V ±5%, V_{SS} = 0, T_A = 0°C 70°C) ϕ_1, ϕ_2 applies to R6512, 13, 14, 15, $\phi_{0(IN)}$ applies to R6502, 03, 04, 05, 06, and 07

Symbol	Characteristic	Min.	Typ.	Max.	Unit
V_{IH}	Input High Voltage Logic, $\phi_{0(IN)}$ ϕ_1, ϕ_2	V_{SS} +2.4 V_{CC} -0.2	– –	V_{CC} V_{CC} +0.25	Vdc
V_{IL}	Input Low Voltage Logic, $\phi_{0(IN)}$ ϕ_1, ϕ_2	V_{SS} -0.3 V_{SS} -0.3	– –	V_{SS} +0.4 V_{SS} +0.4	Vdc
V_{IHT}	Input High Threshold Voltage $\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.	V_{SS} +2.0	–	–	Vdc
V_{ILT}	Input Low Threshold Voltage $\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O.	–	–	V_{SS} +0.8	Vdc
I_{IN}	Input Leakage Current (V_{IN} = 0 to 5.25V, V_{CC} = 0) Logic (Excl. RDY, S.O.) ϕ_1, ϕ_2 $\phi_{0(IN)}$	– – –	– – –	2.5 100 10.0	μA μA μA
I_{TSI}	Three-State (Off State) Input Current (V_{IN} = 0.4 to 2.4V, V_{CC} = 5.25V) Data Lines	–	–	10	μA
V_{OH}	Output High Voltage (I_{LOAD} = -100 μAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W	V_{SS} +2.4	–	–	Vdc
V_{OL}	Output Low Voltage (I_{LOAD} = 1.6 mAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W	–	–	V_{SS} +0.4	Vdc
P_D	Power Dissipation	–	50	80	W
C C_{IN} C_{OUT} $C_{\phi 0(IN)}$ $C_{\phi 1}$ $C_{\phi 2}$	Capacitance (V_{IN} = 0, T_A = 25°C, f = 1 MHz) Logic Data A0-A15, R/W, SYNC $\phi_{0(IN)}$ ϕ_1 ϕ_2	– – – – – –	– – – – 30 50	10 15 12 15 50 50	pF

NOTE: $\overline{IRQ}$ and $\overline{NMI}$ require 3K pull-up resistors.

Clock Timing – R6512, 13, 14, 15, 16

Symbol	Characteristic	Min.	Typ.	Max.	Unit
T_{CYC}	Cycle Time	333	–	–	nsec
PWH ϕ_1 PWH ϕ_2	Clock Pulse Width ϕ_1 (measured at V_{CC} -0.2V) ϕ_2	150 160	–	–	nsec
T_F, T_R	Fall Time, Rise Time (measured from 0.2V to V_{CC} -0.2V)	–	–	15	nsec
T_D	Delay Time between Clocks (measured at 0.2V)	0	–	–	nsec

Clock Timing – R6502, 03, 04, 05, 06, 07

Symbol	Characteristic	Min.	Typ.	Max.	Unit
T_{CYC}	Cycle Time	333	–	–	ns
PWH ϕ_0	$\phi_{0(IN)}$ Pulse Width (measured at 1.5V)	160	–	170	ns
TR ϕ_0, TF ϕ_0	$\phi_{0(IN)}$ Rise, Fall Time	–	–	10	ns
T_D	Delay Time between Clocks (measured at 1.5V)	5	–	–	ns
PWH ϕ_1	$\phi_{1(OUT)}$ Pulse Width (measured at 1.5V)	PWH ϕ_{0L} -20	–	PWH ϕ_{0L}	ns
PWH ϕ_2	$\phi_{2(OUT)}$ Pulse Width (measured at 1.5V)	PWH ϕ_{0H} -40	–	PWH ϕ_{0H} -10	ns
T_R, T_F	$\phi_{1(OUT)}$, $\phi_{2(OUT)}$ Rise, Fall Time (Load = 30 pF + 1 TTL) (measured at 0.8V to 2.0V)	–	–	15	ns

Read/Write Timing (Load = 1 TTL + 130 pF)

Symbol	Characteristic	Min.	Typ.	Max.	Unit
T_{RWS}	Read/Write Setup Time from R6500B	–	80	110	ns
T_{ADS}	Address Setup Time from R6500B	–	80	110	ns
T_{ACC}	Memory Read Access Time	–	–	170	ns
T_{DSU}	Data Stability Time Period	50	–	–	ns
T_{HR}	Data Hold Time – Read	10	–	–	ns
T_{HW}	Data Hold Time – Write	30	–	–	ns
T_{MDS}	Data Setup Time from R6500B	–	60	75	ns
T_{RDY}	RDY, S.O. Setup Time	35	–	–	ns
T_{SYNC}	SYNC Setup Time from R6500B	–	–	100	ns
T_{HA}	Address Hold Time	15	30	–	ns
T_{HRW}	R/W Hold Time	15	30	–	ns

ROCKWELL INTERNATIONAL - MICROELECTRONIC DEVICES

REGIONAL SALES OFFICES

HOME OFFICE*
Rockwell International Corp.
Microelectronic Devices
P.O. Box 3669
Anaheim, Ca. 92803
U.S.A.
Phone: (714) 632-0950
TWX: 910-591-1698

Also Applications Centers

CENTRAL REGION, U.S.A.
Contact Robert O. Whitesell & Associates
6691 East Washington Street
Indianapolis, Indiana 46219
(317) 356-9263 Attn: Mill Gamble, Mgr

EASTERN REGION, U.S.A.*
Carolier Office Building
850-870 U.S. Route 1
North Brunswick, New Jersey 08902
Phone: (201) 246-9630

MIDWEST REGION, U.S.A.
1011 E. Touhy Avenue, Suite 245
Des Plaines, IL 60018
Phone: (312) 297-8867

WESTERN REGION, U.S.A.
3310 Miraloma Avenue
P.O. Box 3669
Anaheim, Ca. 92803
Phone: (714) 632-0950

EUROPE
Rockwell International GmbH
Microelectronic Devices
Fraunhoferstrasse 11
D-8033 Munchen-Martinsried
Germany
Phone: (089) 859-9575
Telex: 0521/2650

FAR EAST
Rockwell International Overseas Corp.
Ichiban-cho Central Building
22-1 Ichiban-cho, Chiyoda-ku
Toyko 102, Japan
Phone: 265-8606
Telex: J22198

YOUR LOCAL REPRESENTATIVE

DOCUMENT NO. 29000 D40
REVISION 2, OCT. 1978

PART NUMBER
R6520

R6500 Microcomputer System DATA SHEET

PERIPHERAL INTERFACE ADAPTER (PIA)

DESCRIPTION

The R6520 Peripheral Interface Adapter is designed to solve a broad range of peripheral control problems in the implementation of microcomputer systems. This device allows a very effective trade-off between software and hardware by providing significant capability and flexibility in a low cost chip. When coupled with the power and speed of the R6500 family of microprocessors, the R6520 allows implementation of very complex systems at a minimum overall cost.

Control of peripheral devices is handled primarily through two 8-bit bidirectional ports. Each of these lines can be programmed to act as either an input or an output. In addition, four peripheral control/interrupt input lines are provided. These lines can be used to interrupt the processor or for "hand-shaking" data between the processor and a peripheral device.

Ordering Information

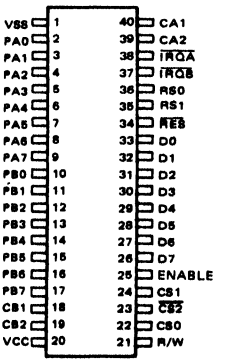
Order Number: R6520 — — —

- Temperature Range:
  - No suffix = 0°C to +70°C
  - E = -40°C to +85°C (Industrial)
  - MT = -55°C to +125°C (Military)
  - M = MIL-STD-883, Class B
- Package:
  - C = Ceramic
  - P = Plastic (Not Available for M or MT suffix)
- Frequency Range:
  - No suffix = 1 MHz
  - A = 2MHz

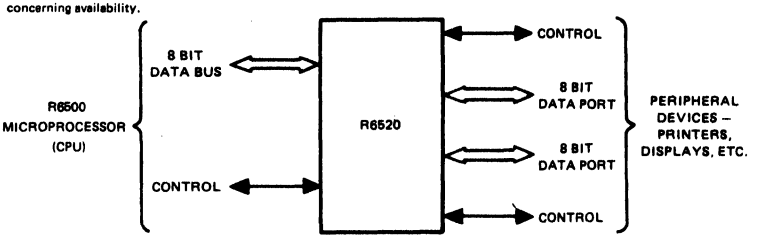
NOTE: Contact your local Rockwell Representative concerning availability.

FEATURES

- High performance replacement for Motorola/AMI/MOSTEK/ Hitachi peripheral adapter.
- N channel, depletion load technology, single +5V supply.
- Completely Static and TTL compatible.
- CMOS compatible peripheral control lines.
- Fully automatic "hand-shake" allows positive control of data transfers between processor and peripheral devices.
- Commercial, industrial and military temperature range versions.



Pin Configuration



Basic R6520 Interface Diagram

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

SUMMARY OF R6520 OPERATION

See Rockwell Microcomputer Hardware Manual for detailed description of R6520 operation.

CA1/CB1 Control

CRA (CRB)		Active Transition of Input Signal*	IRQA (IRQB) Interrupt Outputs
Bit 1	Bit 0		
0	0	Negative	Disable — remain high
0	1	Negative	Enable — goes low when bit 7 in CRA (CRB) is set by active transition of signal on CA1 (CB1)
1	0	Positive	Disable — remain high
1	1	Positive	Enable — as explained above

*Note: Bit 7 of CRA (CRB) will be set to a logic 1 by an active transition of the CA1 (CB1) signal. This is independent of the state of Bit 0 in CRA (CRB).

CA2/CB2 Input Modes

CRA (CRB)			Active Transition of Input Signal*	IRQA (IRQB) Interrupt Output
Bit 5	Bit 4	Bit 3		
0	0	0	Negative	Disable — remains high
0	0	1	Negative	Enable — goes low when bit 6 in CRA (CRB) is set by active transition of signal on CA2 (CB2)
0	1	0	Positive	Disable — remains high
0	1	1	Positive	Enable — as explained above

Note: Bit 6 of CRA (CRB) will be set to a logic 1 by an active transition of the CA2 (CB2) signal. This is independent of the state of Bit 3 in CRA (CRB).

CA2 Output Modes

CRA			Mode	Description
Bit 5	Bit 4	Bit 3		
1	0	0	"Handshake" on Read	CA2 is set high on an active transition of the CA1 interrupt input signal and set low by a microprocessor "Read A Data" operation. This allows positive control of data transfers from the peripheral device to the microprocessor.
1	0	1	Pulse Output	CA2 goes low for one cycle after a "Read A Data" operation. This pulse can be used to signal the peripheral device that data was taken.
1	1	0	Manual Output	CA2 set low
1	1	1	Manual Output	CA2 set high

CB2 Output Modes

CRB			Mode	Description
Bit 5	Bit 4	Bit 3		
1	0	0	"Handshake" on Write	CB2 is set low on microprocessor "Write B Data" operation and is set high by an active transition of the CB1 interrupt input signal. This allows positive control of data transfers from the microprocessor to the peripheral device.
1	0	1	Pulse Output	CB2 goes low for one cycle after a microprocessor "Write B Data" operation. This can be used to signal the peripheral device that data is available.
1	1	0	Manual Output	CB2 set low
1	1	1	Manual Output	CB2 set high

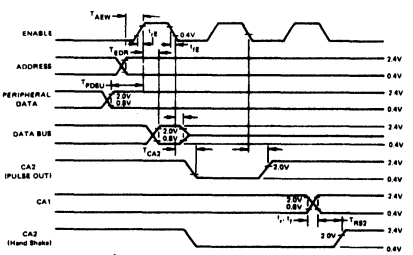
A.C. CHARACTERISTICS

Read Timing Characteristics (Loading 130 pF and one TTL load)

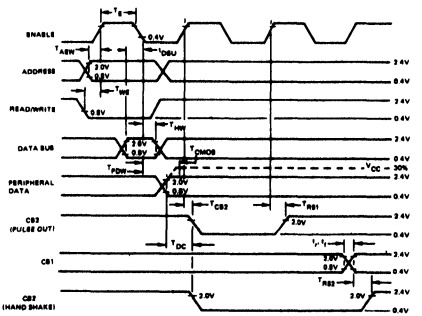
Characteristics	Symbol	1 MHz		2 MHz		Unit
		Min	Max	Min	Max	
Delay Time, Address valid to Enable positive transition	T_{AEW}	180	–	90	–	ns
Delay Time, Enable positive transition to Data valid on bus	T_{EDR}	–	395	–	190	ns
Peripheral Data Setup Time	T_{PDSU}	300	–	150	–	ns
Data Bus Hold Time	T_{HR}	10	–	10	–	ns
Delay Time, Enable negative transition to CA2 negative transition	T_{CA2}	–	1.0	–	0.5	µs
Delay Time, Enable negative transition to CA2 positive transition	T_{RS1}	–	1.0	–	0.5	µs
Rise and Fall Time for CA1 and CA2 input signals	t_r, t_f	–	1.0	–	0.5	µs
Delay Time from CA1 active transition to CA2 positive transition	T_{RS2}	–	2.0	–	1.0	µs
Rise and Fall Time for Enable input	t_{rE}, t_{fE}	–	25	–	25	ns

Write Timing Characteristics

Characteristics	Symbol	1 MHz		2 MHz		Unit
		Min	Max	Min	Max	
Enable Pulse Width	T_E	0.470	25	0.235	25	µs
Delay Time, Address valid to Enable positive transition	T_{AEW}	180	–	90	–	ns
Delay Time, Data valid to Enable negative transition	T_{DSU}	300	–	150	–	ns
Delay Time, Read/Write negative transition to Enable positive transition	T_{WE}	130	–	65	–	ns
Data Bus Hold Time	T_{HW}	10	–	10	–	ns
Delay Time, Enable negative transition to Peripheral Data valid	T_{PDW}	–	1.0	–	0.5	µs
Delay Time, Enable negative transition to Peripheral Data valid CMOS (V_{CC} - 30%) PA0-PA7, CA2	T_{CMOS}	–	2.0	–	1.0	µs
Delay Time, Enable positive transition to CB2 negative transition	T_{CB2}	–	1.0	–	0.5	µs
Delay Time, Peripheral Data valid to CB2 negative transition	T_{DC}	0	1.5	0	0.75	µs
Delay Time, Enable positive transition to CB2 positive transition	T_{RS1}	–	1.0	–	0.5	µs
Rise and Fall Time for CB1 and CB2 input signals	t_r, t_f	–	1.0	–	0.5	µs
Delay Time, CB1 active transition to CB2 positive transition	T_{RS2}	–	2.0	–	1.0	µs



Read Timing Characteristics



Write Timing Characteristics

SPECIFICATIONS

Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature Range	T		°C
Commercial		0 to +70	
Industrial		-40 to +85	
Military		-55 to +125	
Storage Temperature Range	T_{STG}	-55 to + 150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages, however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this circuit.

Static D.C. Characteristics

(V_{CC} = 5.0V ±5%, V_{SS} = 0, T_A = 25°C unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage (Normal Operating Levels)		V_{IH}	+2.0	–	V_{CC}	Vdc
Input Low Voltage (Normal Operating Levels)		V_{IL}	-0.3	–	+0.8	Vdc
Input Threshold Voltage		V_{IT}	0.8	–	2.0	Vdc
Input Leakage Current		I_{in}				μAdc
V_{in} = 0 to 5.0 Vdc R/W, $\overline{Reset}$, RS0, RS1, CS0, CS1, $\overline{CS2}$, CA1, CB1, Φ2			–	±1.0	±2.5	
Three-State (Off State Input Current)		I_{TSI}				
(V_{in} = 0.4 to 2.4 Vdc, V_{CC} = max)	D0-D7, PB0-PB7, CB2		–	±2.0	±10	μAdc
Input High Current		I_{IH}				
(V_{IH} = 2.4 Vdc)	PA0-PA7, CA2		-100	-250	–	μAdc
Input Low Current		I_{IL}				
(V_{IL} = 0.4 Vdc)	PA0-PA7, CA2		–	-1.0	-1.6	mAdc
Output High Voltage		V_{OH}				
(V_{CC} = min, I_{Load} = -100 μAdc)			2.4	–	–	Vdc
Output Low Voltage		V_{OL}				
(V_{CC} = min, I_{Load} = 1.6 mAdc)			–	–	+0.4	Vdc
Output High Current (Sourcing)		I_{OH}				
(V_{OH} = 2.4 Vdc)			-100	-1000	–	μAdc
(V_O = 1.5 Vdc, the current for driving other than TTL, e.g., Darlington Base)	PB0-PB7, CB2		-1.0	-2.5	–	mAdc
Output Low Current (Sinking)		I_{OL}				
(V_{OL} = 0.4 Vdc)			1.6	–	–	mAdc
Output Leakage Current (Off State)	$\overline{IRQA}$, $\overline{IRQB}$	I_{off}	–	1.0	10	μAdc
Power Dissipation		P_D	–	200	500	mW
Input Capacitance		C_{in}				pF
(V_{in} = 0, T_A = 25°C, f = 1.0 MHz)						
D0-D7, PA0-PA7, PB0-PB7, CA2, CB2			–	–	10	
R/W, $\overline{Reset}$, RS0, RS1, CS0, CS1, $\overline{CS2}$,			–	–	7.0	
CA1, CB1, Φ2			–	–	20	
Output Capacitance		C_{out}				
(V_{in} = 0, T_A = 25°C, f = 1.0 MHz)			–	–	10	pF

NOTE: Negative sign indicates outward current flow, positive indicates inward flow.

ROCKWELL INTERNATIONAL - MICROELECTRONIC DEVICES

REGIONAL SALES OFFICES

HOME OFFICE*
Rockwell International Corp.
Microelectronic Devices
P.O. Box 3669
Anaheim, Ca. 92803
U.S.A.
Phone: (714) 632-0960
TWX: 910-591-1698

* Also Applications Centers

CENTRAL REGION, U.S.A.
Contact Robert O. Whitesell & Associates
6691 East Washington Street
Indianapolis, Indiana 46219
(317) 359-9283 Attn: Milt Gamble, Mgr

EASTERN REGION, U.S.A.*
Caroller Office Building
950-970 U.S. Route 1
North Brunswick, New Jersey 08902
Phone: (201) 246-3630

MIDWEST REGION, U.S.A.
1011 E. Touhy Avenue, Suite 245
Des Plaines, IL 60018
Phone: (312) 297-8867

WESTERN REGION, U.S.A.
3310 Miraloma Avenue
P.O. Box 3669
Anaheim, Ca. 92803
Phone: (714) 632-0960

EUROPE
Rockwell International GmbH
Microelectronic Devices
Freunhoferstrasse 11
D-8033 Munchen-Martinsried
Germany
Phone: (089) 859-9675
Telex: 0521/2650

FAR EAST
Rockwell International Overseas Corp.
Ichiban-cho Central Building
22-1 Ichiban-cho, Chiyoda-ku
Toyko 102, Japan
Phone: 265-2605
Telex: J22198

YOUR LOCAL REPRESENTATIVE

DOCUMENT NO. 29000 D47
REVISION 1, OCT. 1978

PART NUMBER
R6522

R6500 Microcomputer System
DATA SHEET
VERSATILE INTERFACE ADAPTER (VIA)

SYSTEM ABSTRACT

The 8-bit R6500 microcomputer system is produced with N-channel, silicon-gate, depletion-load technology. Its performance speeds are enhanced by advanced system architecture. Its innovative architecture results in smaller chips – the semiconductor threshold to cost-effectivity. System cost-effectivity is further enhanced by providing a family of 10 software-compatible microprocessor (CPU) devices, memory and I/O devices . . . as well as low-cost design aids and documentation.

DESCRIPTION

The R6522 VIA adds two powerful, flexible Interval Timers, a serial-to-parallel/parallel-to-serial shift register and input latching on the peripheral ports to the capabilities of the R6520 Peripheral Interface Adapter (PIA) device. Handshaking capability is expanded to allow control of bidirectional data transfers between VIAs in multiple processor systems and between peripherals.

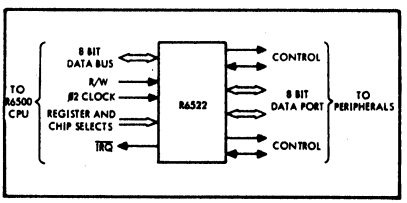
Control of peripherals is primarily through two 8-bit bidirectional ports. Each of these ports can be programmed to act as an input or an output. Peripheral I/O lines can be selectively controlled by the Interval Timers to generate programmable-frequency square waves and/or to count externally generated pulses. Positive control of VIA functions is gained through its internal register organization: Interrupt Flag Register, Interrupt Enable Register, and two Function Control Registers.

FEATURES

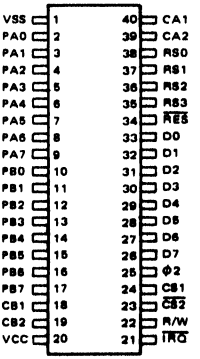
- Organized for simplified software control of many functions
- Compatible with the R650X and R651X family of microprocessors (CPUs)
- Bi-directional, 8-bit data bus for communication with microprocessor
- Two Bi-directional, 8-bit input/output ports for interface with peripheral devices
- CMOS and TTL compatible input/output peripheral ports
- Data Direction Registers allow each peripheral pin to act as either an input or an output
- Interrupt Flag Register allows the microprocessor to readily determine the source of an interrupt and provides convenient control of the interrupts within the chip
- Handshake control logic for input/output peripheral data transfer operations
- Data latching on peripheral input/output ports
- Two fully-programmable interval timers/counters
- Eight-bit Shift Register for serial interface
- Forty-pin plastic or ceramic DIP package.

Ordering Information

Order Number	Package Type	Frequency	Temperature Range
R6522P	Plastic	1 MHz	0°C to +70°C
R6522AP	Plastic	2 MHz	0°C to +70°C
R6522C	Ceramic	1 MHz	0°C to +70°C
R6522AC	Ceramic	2 MHz	0°C to +70°C
R6522PE	Plastic	1 MHz	-40°C to +85°C
R6522APE	Plastic	2 MHz	-40°C to +85°C
R6522CE	Ceramic	1 MHz	-40°C to +85°C
R6522ACE	Ceramic	2 MHz	-40°C to +85°C
R6522CMT	Ceramic	1 MHz	-55°C to +125°C



Basic R6522 Interface Diagram



Pin Configuration

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

OPERATION SUMMARY

Register Select Lines (RS0, RS1, RS2, RS3)

The four Register select lines are normally connected to the processor address bus lines to allow the processor to select the internal R6522 register which is to be accessed. The sixteen possible combinations access the registers as follows:

RS3	RS2	RS1	RS0	Register	Remarks	RS3	RS2	RS1	RS0	Register	Remarks
L	L	L	L	ORB		H	L	L	L	T2L-L	Write Latch
L	L	L	H	ORA	Controls Handshake					T2C-L	Read Counter
L	L	H	L	DDRB		H	L	L	H	T2C-H	Triggers T2L-L/T2C-L Transfer
L	L	H	H	DDRA		H	L	H		SR	
L	H	L	L	T1L-L	Write Latch	H	L	H	H	ACR	
				T1C-L	Read Counter	H	H	L	L	PCR	
L	H	L	H	T1C-H	Trigger T1L-L/T1C-L Transfer	H	H	L	H	IFR	
L	H	H	L	T1L-L		H	H	H	L	IER	
L	H	H	H	T1L-H		H	H	H	H	ORA	No Effect on Handshake

Note: L = 0.4V DC, H = 2.4V DC.

Timer 2 Control

RS3	RS2	RS1	RS0	R/W = L	R/W = H
H	L	L	L	Write T2L-L	Read T2C-L Clear Interrupt flag
H	L	L	H	Write T2C-H Transfer T2L-L to T2C-L Clear Interrupt flag	Read T2C-H

Writing the Timer 1 Register

The operations which take place when writing to each of the four T1 addresses are as follows:

RS3	RS2	RS1	RS0	Operation (R/W = L)
L	H	L	L	Write into low order latch
L	H	L	H	Write into high order latch Write into high order counter Transfer low order latch into low order counter Reset T1 interrupt flag
L	H	H	L	Write low order latch
X	H	H	H	Write high order latch Reset T1 interrupt flag

Reading the Timer 1 Registers

For reading the Timer 1 registers, the four addresses relate directly to the four registers as follows:

RS3	RS2	RS1	RS0	Operation (R/W = H)
L	H	L	L	Read T1 low order counter Reset T1 interrupt flag
L	H	L	H	Read T1 high order counter
L	H	H	L	Read T1 low order latch
L	H	H	H	Read T1 high order latch

Timer 1 Operating Modes

Two bits are provided in the Auxiliary Control Register to allow selection of the T1 operating modes. These bits and the four possible modes are as follows:

ACR7 Output Enable	ACR6 "Free-Run" Enable	Mode
0	0	Generate a single time-out interrupt each time T1 is loaded
0	1	Generate continuous interrupts
1	0	Generate a single interrupt and an output pulse on PB7 for each T1 load operation
1	1	Generate continuous interrupts and a square wave output on PB7

FUNCTION CONTROL

Control of the various functions and operating modes within the R6522 is accomplished primarily through two registers, the Peripheral Control Register (PCR), and the Auxiliary Control Register (ACR). The PCR is used primarily to select the operating mode for the four peripheral control pins. The Auxiliary Control Register selects the operating mode for the Interval Timers (T1, T2), and the Serial Port (SR).

Peripheral Control Register

The Peripheral Control Register is organized as follows:

Bit #	7	6	5	4	3	2	1	0
Function	CB2 Control			CB1 Control	CA2 Control			CA1 Control

Typical functions are shown below:

PCR3	PCR2	PCR1	Mode
0	0	0	Input mode – Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register.
0	0	1	Independent interrupt input mode – Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
0	1	0	Input mode – Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	Independent interrupt input mode – Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
1	0	0	Handshake output mode – Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	Pulse output mode – CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	Manual output mode – The CA2 output is held low in this mode.
1	1	1	Manual output mode – The CA2 output is held high in this mode.

Auxiliary Control Register

Many of the functions in the Auxiliary Control Register have been discussed previously. However, a summary of this register is presented here as a convenient reference for the R6522 user. The Auxiliary Control Register is organized as follows:

Bit #	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

Shift Register Control

The Shift Register operating mode is selected as follows:

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled.
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in under control of system clock.
0	1	1	Shift in under control of external clock pulses.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of the system clock.
1	1	1	Shift out under control of external clock pulses.

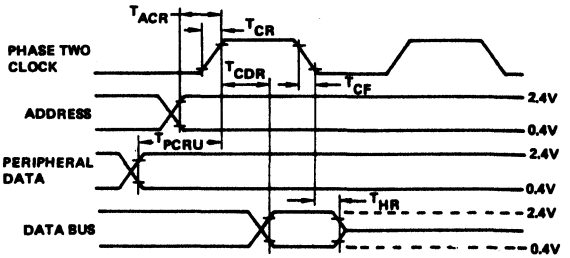
T2 Control

Timer 2 operates in two modes. If ACR5 = 0, T2 acts as an interval timer in the one-shot mode. If ACR5 = 1, Timer 2 acts to count a predetermined number of pulses on pin PB6.

TIMING CHARACTERISTICS

Read Timing Characteristics (loading 130 pF and one TTL load)

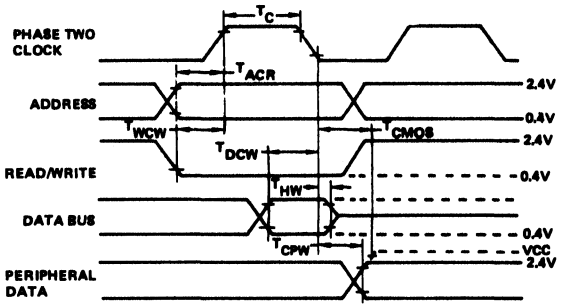
Parameter	Symbol	Min	Typ	Max	Unit
Delay time, address valid to clock positive transition	T_{ACR}	180	–	–	nS
Delay time, clock positive transition to data valid on bus	T_{CDR}	–	–	395	nS
Peripheral data setup time	T_{PCR}	300	–	–	nS
Data bus hold time	T_{HR}	10	–	–	nS
Rise and fall time for clock input	T_{RC} T_{RF}	–	–	25	nS



Read Timing Characteristics

Write Timing Characteristics

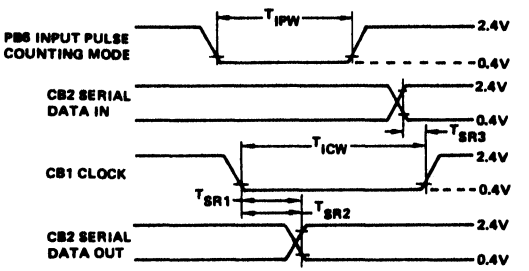
Parameter	Symbol	Min	Typ	Max	Unit
Enable pulse width	T_C	0.47	–	25	μS
Delay time, address valid to clock positive transition	T_{ACW}	180	–	–	nS
Delay time, data valid to clock negative transition	T_{DCW}	300	–	–	nS
Delay time, read/write negative transition to clock positive transition	T_{WCW}	180	–	–	nS
Data bus hold time	T_{HW}	10	–	–	nS
Delay time, Enable negative transition to peripheral data valid	T_{CPW}	–	–	1.0	μS
Delay time, clock negative transition to peripheral data valid CMOS (VCC - 30%)	T_{CMOS}	–	-	2.0	μS



Write Timing Characteristics

I/O Timing Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Rise and fall time for CA1, CB1, CA2 and CB2 input signals	T_{RF}	–	–	1.0	μs
Delay time, clock negative transition to CA2 negative transition (read handshake or pulse mode)	T_{CA2}	–	–	1.0	μs
Delay time, clock negative transition to CA2 positive transition (pulse mode)	T_{RS1}	–	–	1.0	μs
Delay time, CA1 active transition to CA2 positive transition (handshake mode)	T_{RS2}	–	–	2.0	μs
Delay time, clock positive transition to CA2 or CB2 negative transition (write handshake)	T_{WHS}	–	–	1.0	μs
Delay time, peripheral data valid to CB2 negative transition	T_{DC}	0	–	1.5	μs
Delay time, clock positive transition to CA2 or CB2 positive transition (pulse mode)	T_{RS3}	–	–	1.0	μs
Delay time, CB1 active transition to CA2 or CB2 positive transition (handshake mode)	T_{RS4}	–	–	2.0	μs
Delay time, peripheral data valid to CA1 or CB1 active transition (input latching)	T_{IL}	300	–	–	ns
Delay time CB1 negative transition to CB2 data valid (internal SR clock, shift out)	T_{SR1}	–	–	300	ns
Delay time, negative transition of CB1 input clock to CB2 data valid (external clock, shift out)	T_{SR2}	–	–	300	ns
Delay time, CB2 data valid to positive transition of CB1 clock (shift in, internal or external clock)	T_{SR3}	–	–	300	ns
Pulse Width – PB6 Input Pulse	T_{IPW}	2	–	–	μs
Pulse Width – CB1 Input Clock	T_{ICW}	2	–	–	μs
Pulse Spacing – PB6 Input Pulse	I_{IPS}	2	–	–	μs
Pulse Spacing – CB1 Input Pulse	I_{ICS}	2	–	–	μs



I/O Timing Characteristics

SPECIFICATIONS

Maximum Ratings

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{IN}	-0.3 to +7.0	Vdc
Operating Temperature Range	T		°C
Commercial		0 to +70	
Industrial		-40 to +85	
Military		-55 to +125	
Storage Temperature Range	T_{STG}	-55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages.

Electrical Characteristics

(VCC = 5.0V ±5%, VSS = 0)

Characteristic	Symbol	Min	Max	Unit
Input high voltage (normal operation)	V_{IH}	+2.4	VCC	Vdc
Input low voltage (normal operation)	V_{IL}	-0.3	+0.8	Vdc
Input leakage current – V_{in} = 0 to 5 Vdc R/W, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, $\overline{CS2}$, CA1, Ø2	I_{IN}	–	±2.5	µAdc
Off-state input current – V_{in} = 0.4 to 2.4V VCC = Max, D0 to D7	I_{TSI}	–	±10	µAdc
Input high current – V_{IH} = 2.4V PA0-PA7, CA2, PB0-PB7, CB1, CB2	I_{IH}	-100	–	µAdc
Input low current – V_{IL} = 0.4 Vdc PA0-PA7, CA2, PB0-PB7, CB1, CB2	I_{IL}	–	-1.6	mAdc
Output high voltage VCC = min, I_{load} = -100 µAdc PA0-PA7, CA2, PB0-PB7, CB1, CB2	V_{OH}	2.4	–	Vdc
Output low voltage VCC = min, I_{load} = 1.6 mAdc	V_{OL}	–	+0.4	Vdc
Output high current (sourcing) V_{OH} = 2.4V V_{OH} = 1.5V, PB0-PB7, CB1, CB2	I_{OH}	 -100 -1.0	 – –	 µAdc mAdc
Output low current (sinking) V_{OL} = 0.4 Vdc	I_{OL}	1.6	–	mAdc
Output leakage current (off state) $\overline{IRQ}$	I_{off}	–	10	µAdc
Input Capacitance – T_A = 25°C, f = 1 MHz R/W, $\overline{RES}$, RE0, RS1, RS2, RS3, CS1, $\overline{CS2}$, D0-D7, PA0-PA7, CA2, PB0-PB7, CB1, CB2 Ø2 input	C_{in}	 – – –	 7.0 10 20	pF
Output capacitance – T_A = 25°C, f = 1 MHz	C_{out}	–	10	pF
Power dissipation	P_d	–	750	mW

ROCKWELL R6500 MICROCOMPUTER FAMILY

PRODUCT DESCRIPTION
DOC NO. 29650 N40
AUGUST 1977
REVISION: 0

VERSATILE INTERFACE ADAPTER (VIA) PART NO. R6522

ABSTRACT

This document details the functions and operating characteristics of the Versatile Interface Adapter device. This device interfaces the R650X microcomputer with a wide variety of peripheral devices providing both parallel and serial data transfer functions.

Rockwell International

© Rockwell International Corporation 1977
All Rights Reserved
Printed in U.S.A.

Rockwell Microelectronic Devices
P.O. Box 3669
Anaheim, Calif. 92803

R6522 VERSATILE INTERFACE ADAPTER

INTRODUCTION

The R6522 Versatile Interface Adapter (VIA) provides important additional capabilities to those of the R6520 Peripheral Interface Adapter (PIA). Because Designers will, in most cases, want to consider both devices for particular applications, the R6522 is described in relation to the R6520. Also, the R6522 is a new device. It is fully described in the Rockwell R6500 Hardware Manual, but not in earlier editions of 6500 Manuals. Hence this Rockwell Product Description is provided so that Designers will not have to discard their personally annotated 6500 Manuals.

DESCRIPTION

The R6522 VIA adds two powerful, flexible Interval Timers, a serial-to-parallel/parallel-to-serial shift register and input latching on the peripheral ports to the capabilities of the R6520 PIA device. Handshaking capability is expanded to allow control of bidirectional data transfers between VIAs in multiple processor systems and between peripherals.

Control of peripherals is primarily through two 8-bit bidirectional ports -- see Figure 2. Each of these ports can be programmed to act as an input or an output. Peripheral I/O lines can be selectively controlled by the Interval Timers to generate programmable-frequency square waves and/or to count externally generated pulses. Positive control of VIA functions is gained through its internal register organization: Interrupt Flag Register, Interrupt Enable Register, and two Function Control Registers.

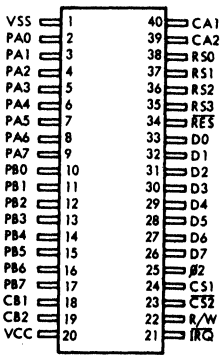


Figure 1. PIN CONFIGURATION

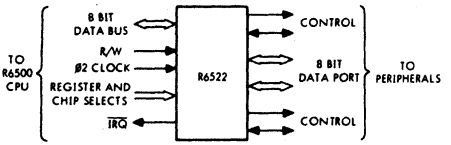


Figure 2. INTERFACE DIAGRAM

FEATURES

- Organized for simplified software control of many functions
- Compatible with the R650X and R651X family of microprocessors (CPUs)
- Bidirectional, 8-bit data bus for communication with microprocessor
- Two bidirectional, 8-bit input/output ports for interface with peripheral devices
- CMOS and TTL compatible input/output peripheral ports
- Data Direction Registers allow each peripheral pin to act as either an input or an output
- Interrupt Flag Register allows the microprocessor to readily determine the source of an interrupt and provides convenient control of the interrupts within the chip
- Handshake control logic for input/output peripheral data transfer operations
- Data latching on peripheral input/output ports
- Two fully-programmable interval timer/counters
- Eight-bit Shift Register for serial interface
- Forty-pin plastic or ceramic DIP package

A block diagram of the R6522 Versatile Interface Adapter is shown in Figure 3.

ORDERING INFORMATION

Order Number	Package Type	Temperature Range
R6522P	Plastic	0°C to 70°C
R6522C	Ceramic	0°C to 70°C

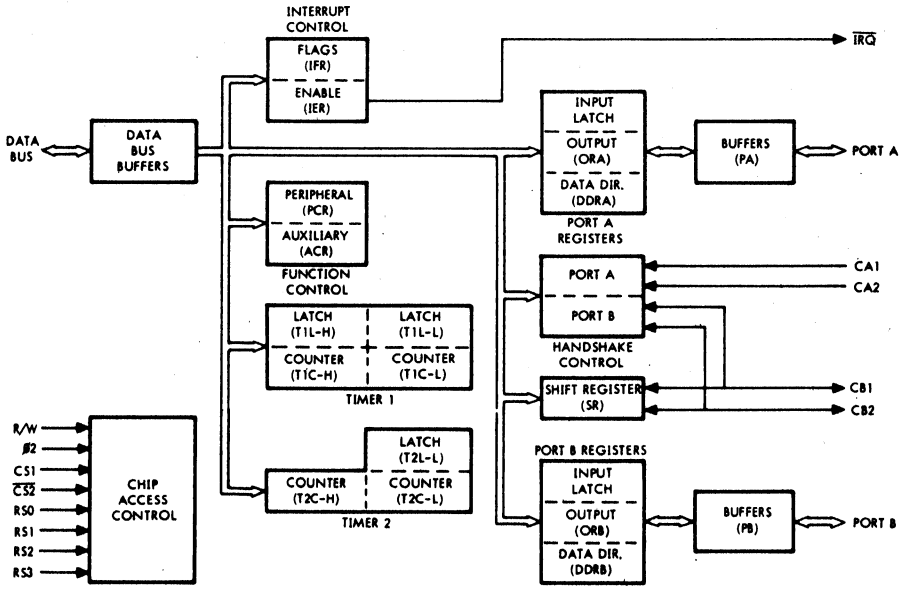


Figure 3. R6522 BLOCK DIAGRAM

PROCESSOR INTERFACE

This section contains a description of the buses and control lines which are used to interface the R6522 to the system processor. AC and DC parameters associated with this interface are specified on pages 19 through 22 of this document.

PHASE TWO CLOCK (Ø2)

Data transfers between the R6522 and the system processor take place only while the Phase Two Clock is high. In addition, Ø2 acts as the time base for the various timers, shift registers, etc. on the chip.

CHIP SELECT LINES (CS1, $\overline{CS2}$)

The two Chip select inputs are normally connected to processor address lines either directly or through decoding. The selected R6522 register will be accessed when CS1 is high and $\overline{CS2}$ is low.

REGISTER SELECT LINES (RS0, RS1, RS2, RS3)

The four Register select lines are normally connected to the processor address bus lines to allow the processor to select the internal R6522 register which is to be accessed. The sixteen possible combinations access the registers as follows.

RS3	RS2	RS1	RS0	Register	Remarks
L	L	L	L	ORB	
L	L	L	H	ORA	Controls Handshake
L	L	H	L	DDRB	
L	L	H	H	DDRA	
L	H	L	L	T1L-L T1C-L	Write Latch Read Counter
L	H	L	H	T1C-H	Trigger T1L-L/ T1C-L Transfer
L	H	H	L	T1L-L	
L	H	H	H	T1L-H	
H	L	L	L	T2L-L T2C-L	Write Latch Read Counter
H	L	L	H	T2C-H	Triggers T2L-L/ T2C-L Transfer
H	L	H	L	SR	
H	L	H	H	ACR	
H	H	L	L	PCR	
H	H	L	H	IFR	
H	H	H	L	IER	
H	H	H	H	ORA	No Effect on Handshake
NOTE: L = 0.4V DC, H = 2.4V DC.					

READ/WRITE LINE (R/W)

The direction of data transfers between the R6522 and the system processor is controlled by the R/W line. If R/W is low, data will be transferred out of the processor into the selected R6522 register (write operation). If R/W is high and the chip is selected, data will be transferred out of the R6522 (read operation).

DATA BUS (DB0-DB7)

The 8 bi-directional data bus lines are used to transfer data between the R6522 and the system processor. The internal drivers will remain in the high-impedance state except when the chip is selected (CS1 = 1, $\overline{CS2}$ = 0), Read/Write is high and the Phase Two Clock is high. At this time, the contents of the selected register are placed on the data bus. When the chip is selected, with Read/Write low and $\phi 2$ = 1, the data on the data bus will be transferred into the selected R6522 register.

RESET ($\overline{RES}$)

The Reset input clears all internal registers to logic 0 (except T1, T2 and SR). This places all peripheral interface lines in the input state, disables the timers, shift register, etc. and disables interrupting from the chip.

INTERRUPT REQUEST ($\overline{IRQ}$)

The Interrupt Request output goes low whenever an internal interrupt flag is set and the corresponding interrupt enable bit is a logic 1. This output is "open-drain" to allow the interrupt request signal to be "wire-or'ed" with other equivalent signals in the system.

PERIPHERAL INTERFACE

This section contains a brief description of the buses and control lines which are used to drive peripheral devices under control of the internal R6522 registers.

PERIPHERAL A PORT (PA0-PA7)

The Peripheral A port consists of 8 lines which can be individually programmed to act as an input or an output under control of a Data Direction Register. The polarity of output pins is controlled by an Output Register and input data can be latched into an internal register under control of the CA1 line. All of these modes of operation are controlled by the system processor through the internal control registers. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode.

PERIPHERAL A CONTROL LINES (CA1, CA2)

The two peripheral A control lines act as interrupt inputs or as handshake outputs. Each line controls an internal interrupt flag with a corresponding interrupt enable bit. In addition, CA1 controls the latching of data on Peripheral A

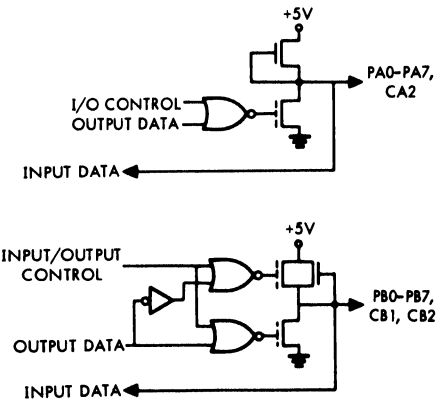


Figure 4. PERIPHERAL OUTPUT BUFFERS

Port input lines. The various modes of operation are controlled by the system processor through the internal control registers. CA1 is a high-impedance input only while CA2 represents one standard TTL load in the input mode. CA2 will drive one standard TTL load in the output mode.

PERIPHERAL B PORT (PB0-PB7)

The Peripheral B port consists of 8 bi-directional lines which are controlled by an output register and a data direction register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. In addition, they are capable of sourcing 3.0 ma at 1.5 VDC in the output mode to allow the outputs to directly drive Darlington transistor switches.

PERIPHERAL B CONTROL LINES (CB1, CB2)

The Peripheral B control lines act as interrupt inputs or as handshake outputs. As with CA1 and CA2, each line controls an interrupt flag with a corresponding interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. In addition, they are capable of sourcing 3.0 ma at 1.5 VDC in the output mode to allow the outputs to directly drive Darlington transistor switches.

R6522 OPERATION

This section contains a discussion of the various blocks of logic shown in Figure 3. In addition, the internal operation of the R6522 is described in detail.

DATA BUS BUFFERS (DB), PERIPHERAL A BUFFERS (PA), PERIPHERAL B BUFFERS (PB)

The characteristics of the buffers which provide the required voltage and current drive capability were discussed in the previous section. AC and DC parameters for these buffers are specified on pages 19 through 22 of this document.

CHIP ACCESS CONTROL

The Chip Access Control contains the necessary logic to detect the chip select condition and to decode the Register Select inputs to allow accessing the desired internal register. In addition, the R/W and Ø2 signals are utilized to control the direction and timing of data transfers. When writing into the R6522, data is first latched into a data input register during Ø2. Data is then transferred into the desired internal register during Ø2 · Chip Select. This allows the peripheral I/O line to change without "glitching". When the processor reads the R6522, data is transferred from the desired internal register directly onto the Data Bus during Ø2.

PORT A REGISTERS, PORT B REGISTERS

Three registers are used in accessing each of the 8-bit peripheral ports. Each port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A "1" causes the pin to act as an output.

Each peripheral pin is also controlled by a bit in the Output Register (ORA, ORB) and an Input Register (IRA, IRB). When the pin is programmed to act as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register. A "1" in the Output Register causes the pin to go high, and a "0" causes the pin to go low. Data can be written into Output Register bits corresponding to pins which are programmed to act as inputs; however, the pin will be unaffected.

Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the data on the PA pins. With input latching enabled, IRA will reflect the contents of the Port A prior to setting the CA1 Interrupt Flag (IFR1) by an active transition on CA1.

The IRB register operates in a similar manner. However, for output pins, the corresponding IRB bit will reflect the contents of the Output Register bit instead of the actual pin. This allows proper data to be read into the processor if the output pin is not allowed to go to full voltage. With input latching enabled on Port B, setting CB1 interrupt flag will cause IRB to latch this combination of input data and ORB data until the interrupt flag is cleared.

HANDSHAKE CONTROL

The R6522 allows positive control of data transfers between the system processor and peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

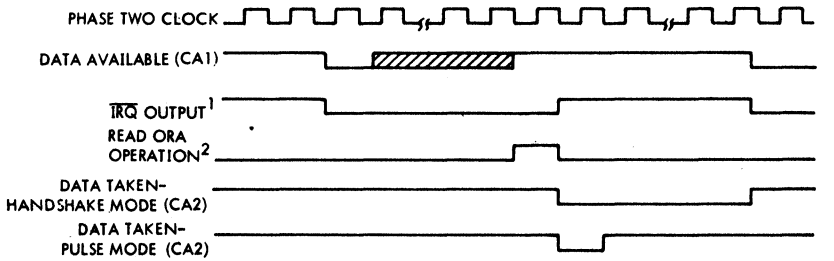
Read Handshake

Positive control of data transfers from peripheral devices into the system processor can be accomplished effectively using "Read" handshaking. In this case, the peripheral device must generate "Data Ready" to signal the processor that valid data is present on the peripheral port. This signal normally interrupts the processor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the R6522, automatic "Read" handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the "Data Ready" signal and CA2 generates the "Data Taken" signal. The Data Ready signal will set an internal flag which may interrupt the processor or which can be polled under software control. The Data Taken signal can be either a pulse or a DC level which is set low by the system processor and is cleared by the Data Ready signal. These options are shown in Figure 5 which illustrates the normal Read Handshaking sequence.

Write Handshake

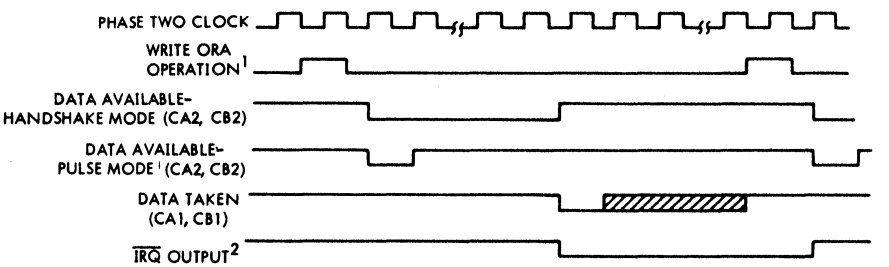
The sequence of operations which allows handshaking data from the system processor to a peripheral device is similar to that described in Section A for Read Handshaking. However, for "Write" handshaking, the processor must generate the "Data Ready" signal (through the R6522) and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the R6522. CA2 or CB2 acts as a Data Ready output in either the DC level or pulse mode and CA1 or CB1 accepts the "Data Taken" signal from the peripheral device, setting the interrupt flag and clearing the "Data Ready" output. This sequence is shown in Figure 6.



NOTES:

1. Signals "Data Available" to the system processor.
2. R/W = 1, $\overline{CS2}$ = 0, CS1 = 1, RS3 = 0, RS2 = 0, RS1 = 0, RS0 = 1.

Figure 5. READ HANDSHAKE TIMING SEQUENCE



NOTES:

1. R/W = 0, $\overline{CS2}$ = 0, CS1 = 1, RS3 = 0, RS2 = 0, RS1 = 0, RS0 = 1.
2. Signals "Data Taken" to the system processor.

Figure 6. WRITE HANDSHAKE TIMING SEQUENCE

TIMER 1

Introduction

Interval Timer T1 consists of two 8-bit latches and a 16-bit counter. The latches are used to store data which is to be loaded into the counter. After loading, the counter decrements at system clock rate, i.e., under control of the clock applied to the Phase Two input pin. Upon reaching zero, an interrupt flag will be set, and $\overline{IRQ}$ will go low. The timer will then disable any further interrupts, or will automatically transfer the contents of the latches into the counter and will continue to decrement. In addition, the timer can be instructed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is discussed separately below.

Writing the Timer 1 Registers

The operations which take place when writing to each of the four T1 addresses are as follows:

RS3	RS2	RS1	RS0	Operation (R/W = L)
L	H	L	L	Write into low order latch
L	H	L	H	Write into high order latch Write into high order counter Transfer low order latch into low order counter Reset T1 interrupt flag
L	H	H	L	Write low order latch
L	H	H	H	Write high order latch Reset T1 interrupt flag

Note that the processor does not write directly into the low order counter (T1C-L). Instead, this half of the counter is loaded automatically from the low order latch when the processor writes into the high order counter. In fact, it may not be necessary to write to the low order counter in some applications since the timing operation is triggered by writing to the high order counter.

The second set of addresses allows the processor to write into the latch register without affecting the count-down in progress. This is discussed in detail below.

Reading the Timer 1 Registers

For reading the Timer 1 registers, the four addresses relate directly to the four registers as follows:

RS3	RS2	RS1	RS0	Operation (R/W = H)
L	H	L	L	Read T1 low order counter Reset T1 interrupt flag
L	H	L	H	Read T1 high order counter
L	H	H	L	Read T1 low order latch
L	H	H	H	Read T1 high order latch

Timer 1 Operating Modes

Two bits are provided in the Auxiliary Control Register to allow selection of the T1 operating modes. These bits and the four possible modes are as follows:

ACR7 Output Enable	ACR6 "Free-Run" Enable	Mode
0	0	Generate a single time-out interrupt each time T1 is loaded. PB7 disabled.
0	1	Generate continuous interrupts. PB7 disabled.
1	0	Generate a single interrupt and an output pulse on PB7 for each T1 load operation.
1	1	Generate continuous interrupts and a square wave output on PB7.

Timer 1 One-Shot Mode

The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval timer, the delay between the "write T1C-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7 = 1) a "write T1C-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single programmable width pulse.

NOTE

PB7 will act as an output if DDRB7 = 1 or if ACR7 = 1. However, if both DDRB7 and ACR7 are logic 1, PB7 will be controlled from Timer 1 and ORB7 will have no effect on the pin.

In the one-shot mode, writing into the high order latch has no effect on the operation of Timer 1. However, it will be necessary to assure that the low order latch contains the proper data before initiating the count-down with a "write T1C-H" operation. When the processor writes into the high order counter, the T1 interrupt flag will be cleared, the contents of the low order latch will be transferred into the low order counter, and the timer will begin to decrement at system clock rate. If the PB7 output is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, the $\overline{IRQ}$ pin will go low (interrupt enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the system processor to read the contents of the counter to determine the time since interrupt. However, the T1

interrupt flag cannot be set again unless it has been cleared as described under the paragraph entitled "Interrupt Flag Register" later in this document.

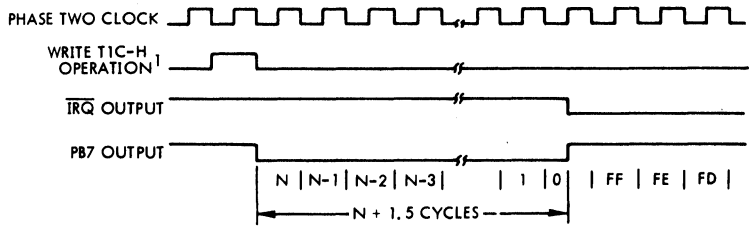
Timing for the R6522 interval timer one-shot modes is shown in Figure 7.

Timer 1 Free-Running Mode

The most important advantage associated with the latches in T1 is the ability to produce a continuous series of evenly spaced interrupts and the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

In the free-running mode (ACR6 = 1), the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of continuing to decrement from zero after a time-out, the timer automatically transfers the contents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing T1C-H, by reading T1C-L, or by writing directly into the flag as described below. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out.

All interval timers in the R6500 family devices are "retriggerable". Rewriting the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high order counter (T1C-H). However, by loading the latches only, the processor can access the timer during each down-counting operation without affecting the time-out in process. Instead, the data loaded into the latches will determine the length of the next time-out period. This capability is particularly valuable in the free-running mode with the output enabled. In this mode, the signal on PB7 is inverted and the interrupt flag is set with each time-out. By responding to the interrupts with new data for the latches, the processor can determine the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex waveforms can be generated. Timing for the free-running mode is shown in Figure 8.



NOTES:

1. R/W = L, $\overline{CS2}$ = L, CS1 = H, RS3 = L, RS2 = H, RS1 = L, RS0 = H.

Figure 7. INTERVAL TIMER "ONE-SHOT" MODE TIMING SEQUENCE

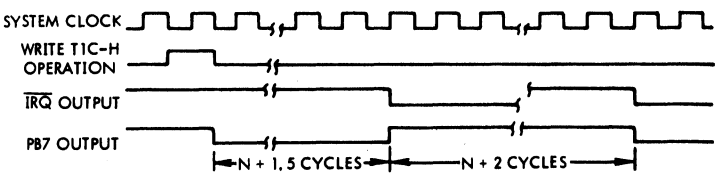


Figure 8. TIMER 1 "FREE-RUNNING" MODE

TIMER 2

Timer 2 operates as an interval timer (in the "one-shot" mode only), or as a counter for counting negative pulses on the PB6 peripheral pin. A single control bit is provided in the Auxiliary Control Register to select between these two modes. This timer is comprised of a "write-only" low-order latch (T2L-L), a "read-only" low-order counter and a read/write high-order counter. The counter registers act as a 16-bit counter which decrements at Ø2 rate.

Timer 2 addressing can be summarized as follows:

RS3	RS2	RS1	RS0	R/W = 0	R/W = 1
H	L	L	L	Write T2L-L	Read T2C-L Clear Interrupt flag
H	L	L	H	Write T2C-H Transfer T2L-L to T2C-L Clear Interrupt flag	Read T2C-H

Timer 2 Interval Timer Mode

As an interval timer, T2 operates in the "one-shot" mode similar to Timer 1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation. After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H. Timing for this operation is shown in Figure 7.

Timer 2 Pulse Counting Mode

In the pulse counting mode, T2 serves primarily to count a predetermined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing the T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessary to rewrite T2C-H to allow the interrupt flag to set on subsequent down-counting operations. Timing for this mode is shown in Figure 9.

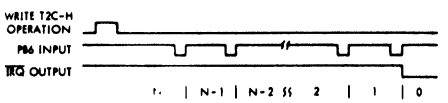


Figure 9. TIMER 2 PULSE COUNTING MODE

SHIFT REGISTER

The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling shifting in external devices.

The control bits which allow control of the various shift register operating modes are located in the Auxiliary Control Register. These bits can be set and cleared by the system processor to select one of the operating modes discussed in the following paragraphs.

Shift Register Input Modes

Bit 4 of the Auxiliary Control Register selects the input or output modes. There are three input modes and four output modes, differing primarily in the source of the pulses which control the shifting operation. With ACR4 = 0 the input modes are selected by ACR3 and ACR2 as follows:

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in at System Clock Rate
0	1	1	Shift in under control of external input pulses

Mode 000 - Shift Register Disabled

The 000 mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR, but the shifting operation is disabled and operation of CB1 and CB2 is controlled by the appropriate bits in the Peripheral Control Register (PCR). In this mode the SR Interrupt Flag is disabled (held to a logic 0).

Mode 001 - Shift In Under Control of Timer 2

In this mode the shifting rate is controlled by the low order 8 bits of T2. Shift pulses are generated on the CB1 pin to control shifting in external devices. The time between transitions of this output clock is a function of the system clock period and the contents of the low order T2 latch.

The shifting operation is triggered by writing or reading the shift register. Data is shifted first into the low order bit of SR and is then shifted into the next higher order bit of the shift register on the trailing edge of each clock pulse. As shown in Figure 10, the input data should change before the leading edge of the clock pulse. This data is loaded into the shift register during the system clock cycle following the trailing edge of the clock pulse. After 8 clock pulses, the shift register interrupt flag will be set and $\overline{IRQ}$ will go low.

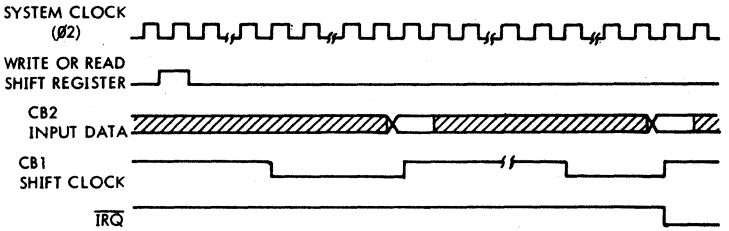


Figure 10. SHIFTING IN UNDER CONTROL OF T2

Mode 010 - Shift in at System Clock Rate

In this mode the shift rate is a direct function of the system clock frequency. CB1 becomes an output which generates shift pulses for controlling external devices. Timer 2 operates as an independent interval timer and has no effect on SR. The shifting operation is triggered by reading or writing the Shift Register. Data is shifted first into bit 0 and is then shifted into the next higher order bit of the shift register on the trailing edge of each clock pulse. After 8 clock pulses, the shift register interrupt flag will be set, and the output clock pulses on CB1 will stop.

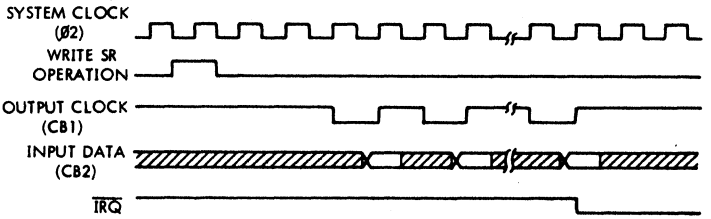


Figure 11. TIMING SEQUENCE FOR SHIFTING IN AT SYSTEM CLOCK RATE

Mode 011 - Shift in Under Control of External Clock

In this mode CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation; it acts simply as a pulse counter. Reading or writing the Shift Register resets the interrupt flag and initializes the SR counter to count another 8 pulses.

Note that data is shifted during the first system clock cycle following the trailing edge of the CB1 shift pulse. For this reason, data must be held stable during the first full cycle following CB1 going high. Timing for this operation is illustrated in Figure 12.

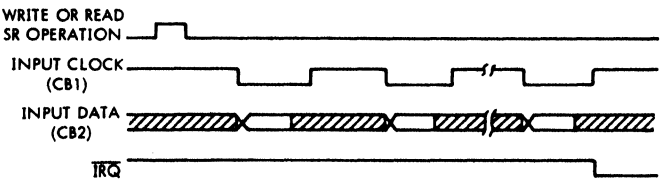


Figure 12. TIMING SEQUENCE FOR SHIFTING IN UNDER CONTROL OF EXTERNAL CLOCK

Shift Register Output Modes

The four Shift Register Output Modes are selected by setting the Input/Output Control Bit (ACR4) to a logic 1 and then selecting the specific output mode with ACR3 and ACR2. In each of these modes the Shift Register shifts data out of bit 7 to the CB2 pin. At the same time the contents of bit 7 are shifted back into bit 0. As in the input modes, CB1 is used either as an output to provide shifting pulses out or as an input to allow shifting from an external pulse. The four modes are as follows:

ACR4	ACR3	ACR2	Mode
1	0	0	Shift out - Free-running mode. Shift rate controlled by T2.
1	0	1	Shift out - Shift rate controlled by T2. Shift pulses generated on CB1.
1	1	0	Shift out at system clock rate.
1	1	1	Shift out under control of an external pulse.

Mode 100 - Free-Running Output

This mode is very similar to mode 101 in which the shifting rate is set by T2. However, in mode 100 the SR Counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into bit 0, the 8 bits loaded into the shift register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled.

Mode 101 - Shift Out under Control of T2

In this mode the shift rate is controlled by T2 (as in the previous mode). However, with each read or write of the shift register the SR Counter is reset and 8 bits are shifted onto CB2. At the same time, 8 shift pulses are generated on CB1 to control shifting in external devices. After the 8 shift pulses, the shifting is disabled, the SR Interrupt flag is set and CB2 goes to a state determined by the CB2 Control bit (PC5) in the Peripheral Control Register.

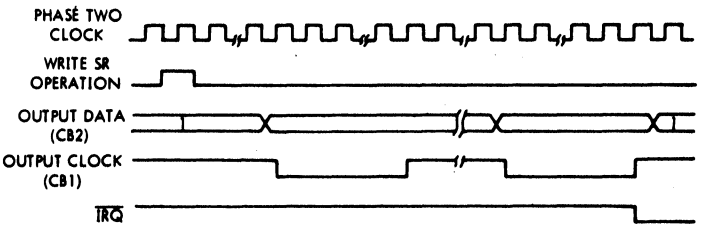
The CB2 control bits (PC7, PC6 and PC5) should be used to set CB2 to a manual output selecting either a high or low polarity. If the shift register is reloaded before the last time-out, the shifting will continue. This sequence is illustrated in Figure 13.

Mode 110 - Shifting Out at System Clock Rate

In this mode the shift register operation is similar to that shown in Figure 13. However, the shifting rate is a function of the system clock on the chip enable pin (Ø2) and is independent of T2. Timer 2 resumes its normal function as an independent interval timer. Figure 14 illustrates the timing sequence for mode 110.

Mode 111 - Shift Out Under Control of an External Pulse

In this mode, shifting is controlled by pulses applied to the CB1 pin by an external device. The SR Counter sets the SR Interrupt flag each time it counts 8 pulses but it does not disable the shifting function. Each time the microprocessor writes or reads the shift register, the SR Interrupt flag is reset and the SR Counter is initialized to begin counting the next 8 shift pulses on pin CB1. After 8 shift pulses, the interrupt flag is set. The microprocessor can then load the shift register with the next byte of data.



NOTES:

1. Data out determined by CB2 control in PCR.

Figure 13. SHIFTING OUT UNDER CONTROL OF T2

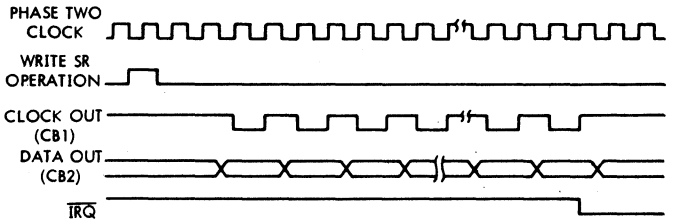


Figure 14. SHIFTING OUT UNDER CONTROL OF SYSTEM CLOCK

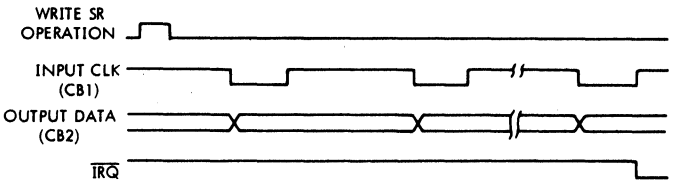


Figure 15. SHIFTING OUT UNDER CONTROL OF EXTERNAL CLOCK

INTERRUPT CONTROL

Controlling interrupts within the R6522 involves three principal operations. These are flagging the interrupts, enabling interrupts and signalling to the processor that an active interrupt exists within the chip. Interrupt flags are set by interrupting conditions which exist within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor must examine these flags in order from highest to lowest priority. This is accomplished by reading the flag register into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt.

Associated with each interrupt flag is an interrupt enable bit. This bit can be set or cleared by the processor to enable interrupting the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output ($\overline{IRQ}$) will go low. $\overline{IRQ}$ is an "open-collector" output which can be "wire OR'ed" with other devices in the system to interrupt the processor.

In the R6522, all the interrupt flags are contained in one register. In addition, bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This allows very convenient polling of several devices within a system to locate the source of an interrupt.

	7	6	5	4	3	2	1	0
Interrupt Flag Register	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
Interrupt Enable Register	Set/ clear control	T1	T2	CB1	CB2	SR	CA1	CA2

Interrupt Flag Register

The IFR is a read/bit-clear register. When the proper chip select and register signals are applied to the chip, the contents of this register are placed on the data bus. Bit 7 indicates the status of the $\overline{IRQ}$ output. This bit corresponds to the logic function: IRQ = IFR6 X IER6 + IFR5 X IER5 + IFR4 X IER4 + IFR3 X IER3 + IFR2 X IER2 + IFR1 X IER1 + IFR0 X IER0. Note: X = logic AND, + = logic OR.

Bits six through zero are latches which are set and cleared as follows:

Bit #	Set By	Cleared By
0	Active transition of the signal on the CA2 pin	Reading or writing the A Port Output Register (ORA) using address 0001
1	Active transition of the signal on the CA1 pin	Reading or writing the A Port Output Register (ORA) using address 0001
2	Completion of eight shifts	Reading or writing the Shift Register
3	Active transition of the signal on the CB2 pin	Reading or writing the B Port Output Register
4	Active transition of the signal on the CB1 pin	Reading or writing the B Port Output Register
5	Time-out of Timer 2	Reading T2 low order counter. Writing T2 high order counter.
6	Time-out of Timer 1	Reading T1 low order counter. Writing T1 high order counter.

The IFR bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts as discussed in the next section.

The Interrupt Enable Register is used to prevent interrupt conditions from generating an interrupt. Once any enabled interrupt is generated, the Interrupt Flag Register contains all interrupt bits, and is not masked by the Interrupt Enable Register.

Interrupt Enable Register (IER)

For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register. The system processor can set or clear selected bits in this register to facilitate controlling individual interrupts without affecting others. This is accomplished by writing to address 1110 (IER address). If bit 7 of the data placed on the system data bus during this write operation is a 0, each 1 in bits 6 through 0 clears the corresponding bit in the Interrupt Enable Register. For each zero in bits 6 through 0, the corresponding bit is unaffected.

Setting selected bits in the Interrupt Enable Register is accomplished by writing to the same address with bit 7 in the data word set to a logic 1. In this case, each 1 in bits 6 through 0 will set the corresponding bit. For each zero, the corresponding bit will be unaffected. This individual control of the setting and clearing operations allows very convenient control of interrupts during system operation.

In addition to setting and clearing IER bits, the processor can read the contents of this register by placing the proper address on the register select and chip select inputs with the R/W line high. Bit 7 will be read as a logic 0.

FUNCTION CONTROL

Control of the various functions and operating modes within the R6522 is accomplished primarily through two registers, the Peripheral Control Register (PCR), and the Auxiliary Control Register (ACR). The PCR is used primarily to select the operating mode for the four peripheral control pins. The Auxiliary Control Register selects the operating mode for the interval timers (T1, T2), and the serial port (SR).

Peripheral Control Register

The Peripheral Control Register is organized as follows:

Bit #	7	6	5	4	3	2	1	0
Function	CB2 Control			CB1 Control	CA2 Control			CA1 Control

Each of these functions is discussed in detail below.

CA1 Control

Bit 0 of the Peripheral Control Register selects the active transition of the input signal applied to the CA1 interrupt input pin. If this bit is a logic 0, the CA1 interrupt flag will be set by a negative transition (high to low) of the signal on the CA1 pin. If PCR0 is a logic 1, the CA1 interrupt flag will be set by a positive transition (low to high) of this signal.

CA2 Control

The CA2 pin can be programmed to act as an interrupt input or as a peripheral control output. As an input, CA2 operates in two modes, differing primarily in the methods available for resetting the interrupt flag. Each of these two input modes can operate with either a positive or a negative active transition as described above for CA1.

In the output mode, the CA2 pin combines the operations performed on the CA2 and CB2 pins of the R6520. This added flexibility allows processor to perform a normal "write" handshaking in a system which uses CB1 and CB2 for the serial operations described above. The CA2 operating modes are selected as follows:

PCR3	PCR2	PCR1	Mode
0	0	0	Input mode - Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register.
0	0	1	Independent interrupt input mode - Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
0	1	0	Input mode - Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	Independent interrupt input mode - Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag.
1	0	0	Handshake output mode - Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	Pulse Output mode - CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	Manual output mode - The CA2 output is held low in this mode.
1	1	1	Manual output mode - The CA2 output is held high in this mode.

In the independent input mode, writing or reading the ORA register has no effect on the CA2 interrupt flag. This flag must be cleared by writing a logic 1 into the appropriate IFR bit. This mode allows the processor to handle interrupts which are independent of any operations taking place on the peripheral I/O ports.

The handshake and pulse output modes have been described previously. Note that the timing of the output signal varies slightly depending on whether the operation is initiated by a read or a write.

CB1 Control

Control of the active transition of the CB1 input signal operates in exactly the same manner as that described above for CA1. If PCR4 is a logic 0, the CB1 interrupt flag (IFR4) will be set by a negative transition of the CB1 input signal and cleared by a read or write of the ORB register. If PCR4 is a logic 1, IFR4 will be set by a positive transition of CB1.

If the Shift Register function has been enabled, CB1 will act as an input or output for the shift register clock signals. In this mode the CB1 interrupt flag will still respond to the selected transition of the signal on the CB1 pin.

CB2 Control

With the serial port disabled, operation of the CB2 pin is a function of the three high order bits of the PCR. The CB2 modes are very similar to those described previously for CA2. These modes are selected as follows.

PCR7	PCR6	PCR5	Mode
0	0	0	Interrupt input mode - Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the Peripheral B Output Register.
0	0	1	Independent interrupt input mode - Set IFR3 on a negative transition of the CB2 input signal. Reading or writing ORB does not clear the interrupt flag.
0	1	0	Input mode - Set CB2 interrupt flag on a positive transition of the CB2 input signal. Clear the CB2 interrupt flag on a read or write of ORB.
0	1	1	Independent input mode - Set IFR3 on a positive transition of the CB2 input signal. Reading or writing ORB does not clear the CB2 interrupt flag.
1	0	0	Handshake output mode - Set CB2 low on a write ORB operation. Reset CB2 high with an active transition of the CB1 input signal.
1	0	1	Pulse output mode - Set CB2 low for one cycle following a write ORB operation.
1	1	0	Manual output mode - The CB2 output is held low in this mode.
1	1	1	Manual output mode - The CB2 output is held high in this mode.

Auxiliary Control Register

Many of the functions in the Auxiliary Control Register have been discussed previously. However, a summary of this register is presented here as a convenient reference for the R6522 user. The Auxiliary Control Register is organized as follows:

Bit #	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

PA Latch Enable

The R6522 provides input latching on both the PA and PB ports. In this mode, the data present on the peripheral A input pins will be latched within the chip when the CA1 interrupt flag is set. Reading the PA port will result in these latches being transferred into the processor. As long as the CA1 interrupt flag is set, the data on the peripheral pins can change without affecting the data in the latches. This input latching can be used with any of the CA2 input or output modes.

It is important to note that on the PA port, the processor always reads the data on the peripheral pins (as reflected in the latches). For output pins, the processor still reads the latches. This may or may not reflect the data currently in the ORA. Proper system operation requires careful planning on the part of the system designer if input latching is combined with output pins on the peripheral ports.

Input latching is enabled by setting bit 0 in the Auxiliary Control Register to a logic 1. As long as this bit is a 0, the latches will directly reflect the data on the pins.

PB Latch Enable

Input latching on the PB port is controlled in the same manner as that described for the PA port. However, with the peripheral B port the input latch will store either the voltage on the pin or the contents of the Output Register (ORB) depending on whether the pin is programmed to act as an input or an output. As with the PA port, the processor always reads the input latches.

Shift Register Control

The Shift Register operating mode is selected as follows:

ACR4	ACR3	ACR2	Mode
0	0	0	Shift Register Disabled
0	0	1	Shift in under control of Timer 2
0	1	0	Shift in under control of system clock
0	1	1	Shift in under control of external clock pulses
1	0	0	Free-running output at rate determined by Timer 2
1	0	1	Shift out under control of Timer 2
1	1	0	Shift out under control of the system clock
1	1	1	Shift out under control of external clock pulses

T2 Control

Timer 2 operates in two modes. If ACR5 = 1, T2 acts as an interval timer in the one-shot mode. If ACR5 = 0, Timer 2 acts to count a predetermined number of pulses on pin PB6.

T1 Control

Timer 1 operates in the one-shot or free-running mode with the PB7 output control enabled or disabled. These modes are selected as follows:

ACR7	ACR6	Mode
0	0	One-shot mode - Output to PB7 disabled
0	1	Free-running mode - Output to PB7 disabled
1	0	One-shot mode - Output to PB7 enabled
1	1	Free-running mode - Output to PB7 enabled

APPLICATION OF THE R6522

The R6522 represents a significant advance in general-purpose microprocessor I/O. Unfortunately, its many powerful features, coupled with a set of very flexible operating modes, cause this device to appear to be very complex at first glance. However, a detailed analysis will show that the VIA is organized to allow convenient control of these powerful features. This section seeks to assist the system designer in his understanding of the R6522 by illustrating how the device can be used in microprocessor based systems.

CONTROL OF R6522 INTERRUPTS

Organization of the R6522 interrupt flags into a single register greatly facilitates the servicing of interrupts from this device. Since there is only one IRQ output for the seven possible sources of interrupt within the chip, the processor must examine these flags to determine the cause of an interrupt. This is best accomplished by first transferring the contents of the flag register into the accumulator. At this time it may be necessary to mask off those flags which have been disabled in the Interrupt Enable Register. This is particularly important for the edge detecting inputs where the flags may be set whether or not the interrupting function has been enabled. Masking off those flags can be accomplished by performing an AND operation between the IER and the accumulator or by performing an "AND IMMEDIATE". The second byte of this AND # instruction should specify those flags which correspond to interrupt functions which are to be serviced.

If the N flag is set after these operations, an active interrupt exists within the chips. This interrupt can be detected with a series of shift and branch instructions.

Clearing interrupt flags is accomplished very conveniently by writing a logic 1 directly into the appropriate bit of the Interrupt Flag Register. This can be combined with an interrupt enable or disable operation as follows:

```
LDA # @10010000 ; initialize accumulator
STA IFR         ; clear interrupt flag
STA IER         ; set interrupt enable flag
```

or:

```
LDA # @00001000 ; initialize accumulator
STA IFR         ; clear interrupt flag
STA IER         ; disable interrupt
```

Another very useful technique for clearing interrupt flags is to simply transfer the contents of the flag register back into this register as follows:

```
LDA IFR ; transfer IFR to accumulator
STA IFR ; clear flags corresponding to active
          interrupts
```

After completion of this operation the accumulator will still contain the interrupt flag information. Most important, writing into the flag register clears only those flags which are already set. This eliminates the possibility of inadvertently clearing a flag while it is being set.

USE OF TIMER 1

Timer 1 represents one of the most powerful features of the R6522. The ability to generate very evenly spaced interrupts and the ability to control the voltage on PB7 makes this timer particularly valuable in various timing, data detection and waveform generation applications.

Time-of-Day Clock Applications

An important feature of many systems is the time-of-day clock. In microprocessor-based systems the time of day is usually maintained in memory and is updated in an interrupt service routine. A regular processor interrupt will then assure that this time of day will always be available when it is needed in the main program.

Generating very regular interrupts using previously available timers presented difficulties because of the need to reload the timer for each interrupt. Unfortunately, the time between the interrupts will fluctuate due to variations in the interrupt response time. This problem is eliminated in the Timer 1 "free-running" mode. The accuracy of these "free-running" interrupts is only a function of the system clock and is not affected by interrupt response time.

Asynchronous Data Detection

The extraction of clock and data information from serial asynchronous ASCII signals or from any single channel data recording device relies on the ability to establish accurate strobes. As discussed previously, the period of these strobes can be seriously affected by the interrupt response time using conventional timers. However, T1 again allows generation of very accurate interrupts. The processor responds to these interrupts by strobing the input data. The ability to reload the T1 latches without affecting the count-down in progress is very useful in this application. This allows the strobe time to be doubled or halved during data detection. This sequence of operations is illustrated in Figure 16.

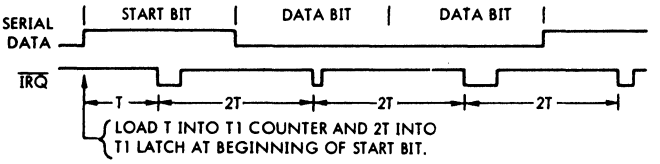


Figure 16. DETECTING ASYNCHRONOUS DATA USING TIMER 1

Waveform Generation with Timer 1

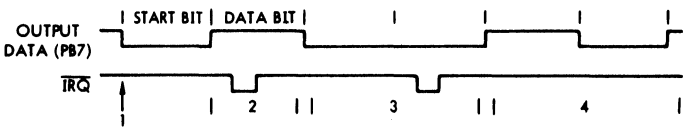
In addition to generating processor interrupts, Timer 1 can be used to control the output voltage on peripheral pin PB7 (output mode). In this mode a single negative pulse can be generated on PB7 (one-shot mode) or, in the free-running mode, a continuous waveform can be generated. In this latter mode the voltage on PB7 will be inverted each time T1 times out.

A single solenoid can be triggered very conveniently in the one-shot mode if the PB7 signal is used to control the solenoid directly. With this configuration the solenoid can be triggered by simply writing to T1C-H.

Generating very complex waveforms can be a simple problem if T1 is used to control PB7 in the free-running mode. During any count-down process the latches can be loaded to determine the length of the next count-down period. Figure 17 shows this timing sequence for generating ASCII serial data.

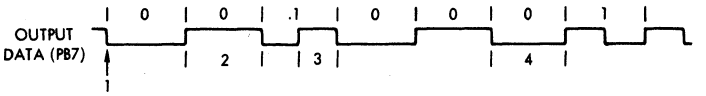
An application where this mode of operation is also very powerful is in the generation of bi-phase encoded data for tape or disk storage. This encoding technique and the sequence of operations which would take place are illustrated in Figure 18.

These applications represent only a tiny portion of the potential T1 applications. Some other possibilities are pulse width modulation waveforms, A/D techniques requiring very accurate pulse widths, and waveform synthesis in electronic games.



1. Load T into T1 counter and latch. Load T into T2 to trigger T1 latch reload.
2. Load 2T into T1 latch during this bit time.
3. Load T into T1 latch anytime during this period. Load NT into T2. N = number of 1's or 0's which follow.
4. A series of 1's and 0's will be generated until the T1 latch is again changed. Note that the use of T2 to control reloading the T1 latch eliminates the need to interrupt on each transition.

Figure 17. ASCII SERIAL DATA GENERATION USING T1



1. Load T1 counter and latch.
2. Shift T1 latch one bit to the right during this period.
3. Shift T1 latch left during this period.
4. Shift T1 latch right during this period.

Note that T1 must be accessed only when the output data changes. A string of 1's or 0's can be generated without processor intervention.

Figure 18. GENERATING BI-PHASE ENCODED DATA

Using the R6522 Shift Register

The Shift Register in the R6522 is designed primarily as a synchronous serial communications port for distributed systems. These systems can be either single-processor with distributed peripheral controllers or distributed processor systems. The most important characteristic of the Shift Register in these applications is its ability to transfer information at relatively slow data rates to allow the use of R-C noise suppression techniques. This transfer can be accomplished while the processor is servicing other aspects of the system. An example of a simple 2-processor distributed system is shown in Figure 19. Use of the R6522 Shift Register allows effective communication between the two systems without the use of relatively complex asynchronous communications techniques.

In a system with distributed peripherals, the Shift Register can be used to transfer data to the peripheral interface devices. This is illustrated in Figure 20 for a system with a number of distributed status displays. These displays are serviced by stand-alone controllers which actuate the lamps in the status displays through simple drivers. The data and clock lines are wired in parallel to each unit. In addition, a single R6522 peripheral port output allows selection of the display to be loaded. These select lines can be eliminated if all displays are to contain the same information. With the system shown, the status display can be updated at any time by simply selecting the desired display and then writing to the Shift Register.

Remote input devices can be serviced in much the same manner by shifting data into the Shift Register under control of a peripheral port output as shown in Figure 20. Each set of input switches can be polled by first selecting the set to be polled and then triggering the shifting operation with a Shift Register read operation. A shift register interrupt can be used to cause the processor to read the resulting input information after shifting is complete.

The techniques described above can be utilized to expand I/O capability in a microprocessor based system. In a system with many status lamps or many input switches, simple TTL shift registers will provide the necessary I/O in a very cost effective manner. This is illustrated in Figure 21.

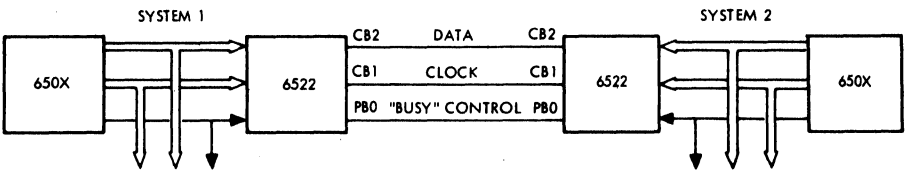


Figure 19. USING SHIFT REGISTER FOR INTER-SYSTEM COMMUNICATION

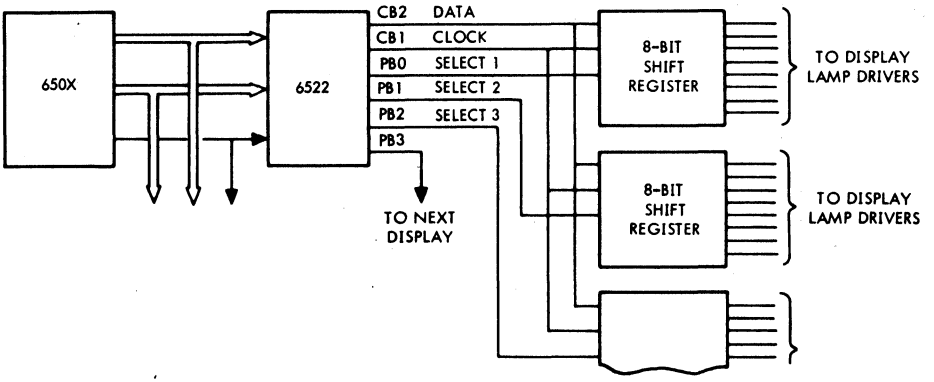


Figure 20. USING THE SHIFT REGISTER FOR SERVICING REMOTE STATUS DISPLAYS

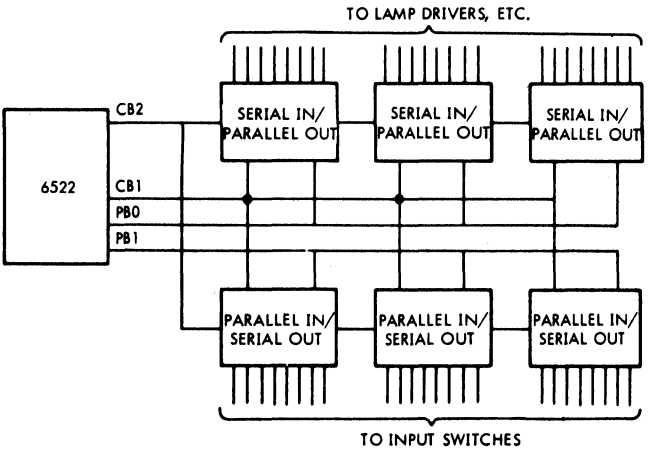
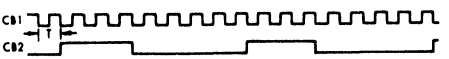


Figure 21. EXPANDING SYSTEM I/O USING SHIFT REGISTER

Clock Generation Using the Shift Register

In all output modes the data shifted out of bit 7 will also be shifted into bit 0. For this reason the Shift Register need not be reloaded if the same data is to be shifted out each time. A Shift Register read operation can be used to trigger the shifting operation.

This capability is very useful for generating peripheral clocks in the continuous output mode. This mode allows an 8-bit pattern to be shifted out continuously. This is illustrated in Figure 22. Note that in this mode the shifting operation is controlled by Timer 2. A single bit time can therefore be up to 256 clock cycles in length.



NOTES:

1. Shift Register loaded with 1110 0000_2 initially.
2. T determined by Timer 2.

Figure 22. CLOCK GENERATION USING SR FREE-RUNNING MODE

MAXIMUM RATINGS

	Symbol	Value	Unit
Supply Voltage	VCC	-0.3 to +7.0	VDC
Input Voltage	V_{IN}	-0.3 to +7.0	VDC
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{STG}	-55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages. However, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages.

STATIC DC CHARACTERISTICS

(VCC = 5.0V ± 5%, VSS = 0, T_A = 0 to +70°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input high voltage (normal operation)	V_{IH}	+2.4	-	VCC	VDC
Input low voltage (normal operation)	V_{IL}	-0.3	-	+0.4	VDC
Input leakage current: V_{IN} = 0 to 5 VDC R/W, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, $\overline{CS2}$, CA1, Ø2	I_{IN}	-	±1.0	±2.5	μADC
Off-state input current: V_{IN} = .4 to 2.4V VCC = Max, D0 to D7	I_{TSI}	-	±2.0	±10	μADC
Input high current: V_{IH} = 2.4V PA0-PA7, CA2, PB0-PB7, CB1, CB2	I_{IH}	-100	-250	-	μADC
Input low current: V_{IL} = 0.4 VDC PA0-PA7, CA2, PB0-PB7, CB1, CB2	I_{IL}	-	-1.0	-1.6	mADC
Output high voltage VCC = Min, I_{LOAD} = -100 μADC PA0-PA7, CA2, PB0-PB7, CB1, CB2	V_{OH}	2.4	-	-	VDC
Output low voltage VCC = Min, I_{LOAD} = 1.6 mADC	V_{OL}	-	-	+0.4	VDC
Output high current (sourcing) V_{OH} = 2.4V V_{OH} = 1.5V, PB0-PB7, CB1, CB2	I_{OH}	 -100 -3.0	 -1000 -5.0	 - -	 μADC mADC
Output low current (sinking) V_{OL} = 0.4 VDC	I_{OL}	1.6	-	-	mADC
Output leakage current (off state) $\overline{IRQ}$	I_{OFF}	-	1.0	10	μADC
Input Capacitance: T_A = 25°C, f = 1 MHz R/W, $\overline{RES}$, RS0, RS1, RS2, RS3, CS1, CS2 D0-D7, PA0-PA7, CA2, PB0-PB7, CB1, CB2 Ø2 input	C_{IN}	 - - -	 - - -	 7.0 10 20	 pF pF pF
Output Capacitance: T_A = 25°C, f = 1 MHz	C_{OUT}	-	-	10	pF
Power dissipation	P_D	-	-	1000	MW

AC CHARACTERISTICS

Read Timing Characteristics (Figure 23, loading 130 pF and one TTL load)

Characteristic	Symbol	Min	Typ	Max	Unit
Delay time, address valid to clock positive transition	T_{ACR}	180	-	-	nS
Delay time, clock positive transition to data valid on bus	T_{CDR}	-	-	395	nS
Peripheral data setup time	T_{PCR}	300	-	-	nS
Data bus hold time	T_{HR}	10	-	-	nS
Rise and fall time for clock input	T_{RC} T_{RF}	-	-	25	nS

Write Timing Characteristics (Figure 24)

Characteristic	Symbol	Min	Typ	Max	Unit
Enable pulse width	T_C	0.47	-	25	µS
Delay time, address valid to clock positive transition	T_{ACW}	180	-	-	nS
Delay time, data valid to clock negative transition	T_{DCW}	300	-	-	nS
Delay time, read/write negative transition to clock positive transition	T_{WCW}	180	-	-	nS
Data bus hold time	T_{HW}	10	-	-	nS
Delay time, Enable negative transition to peripheral data valid	T_{CPW}	-	-	1.0	µS
Delay time, clock negative transition to peripheral data valid CMOS (VCC - 30%)	T_{CMOS}	-	-	2.0	µS

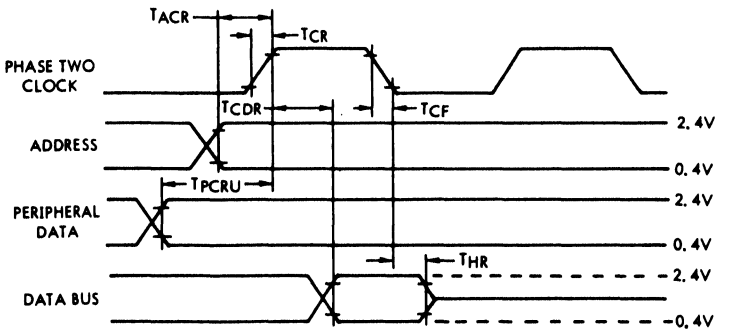


Figure 23. READ TIMING CHARACTERISTICS

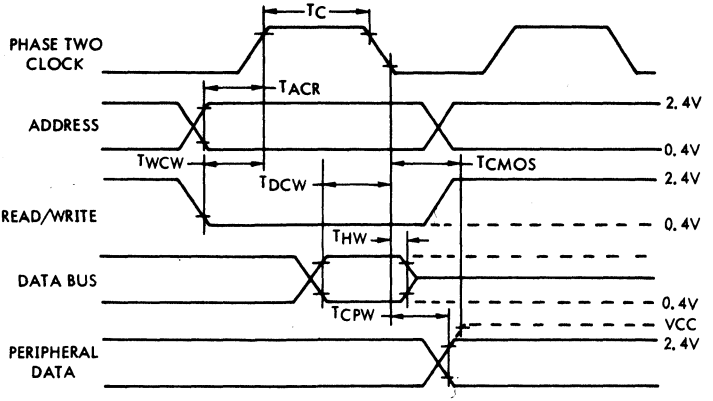


Figure 24. WRITE TIMING CHARACTERISTICS

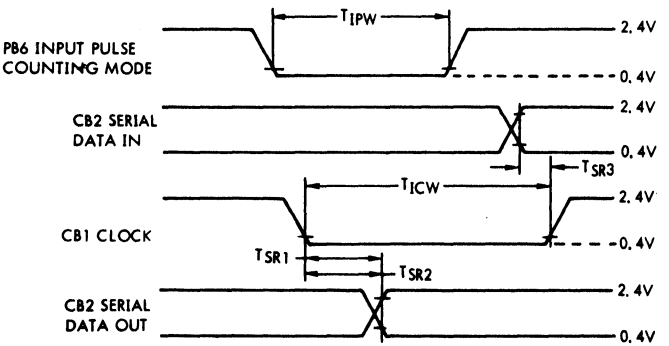


Figure 25. I/O TIMING CHARACTERISTICS

PERIPHERAL INTERFACE CHARACTERISTICS

Characteristic	Symbol	Min	Typ	Max	Unit
Rise and fall time for CA1, CB1, CA2 and CB2 input signals	T_{RF}	-	-	1.0	µs
Delay time, clock negative transition to CA2 negative transition (read handshake or pulse mode)	T_{CA2}	-	-	1.0	µs
Delay time, clock negative transition to CA2 positive transition (pulse mode)	T_{RS1}	-	-	1.0	µs
Delay time, CA1 active transition to CA2 positive transition (handshake mode)	T_{RS2}	-	-	2.0	µs
Delay time, clock positive transition to CA2 or CB2 negative transition (write handshake)	T_{WHS}	-	-	1.0	µs
Delay time, peripheral data valid to CB2 negative transition	T_{DC}	0	-	1.5	µs
Delay time, clock positive transition to CA2 or CB2 positive transition (pulse mode)	T_{RS3}	-	-	1.0	µs
Delay time, CB1 active transition to CA2 or CB2 positive transition (handshake mode)	T_{RS4}	-	-	2.0	µs
Delay time, peripheral data valid to CA1 or CB1 active transition (input latching)	T_{IL}	300	-	-	ns
Delay time, CB1 negative transition to CB2 data valid (internal SR clock, shift out)	T_{SR1}	-	-	300	ns
Delay time, negative transition of CB1 input clock to CB2 data valid (external clock, shift out)	T_{SR2}	-	-	300	ns
Delay time, CB2 data valid to positive transition of CB1 clock (shift in, internal or external clock)	T_{SR3}	-	-	300	ns
Pulse Width - PB6 Input Pulse	T_{IPW}	2	-	-	µs
Pulse Width - CB1 Input Clock	T_{ICW}	2	-	-	µs
Pulse Spacing - PB6 Input Pulse	I_{IPS}	2	-	-	µs
Pulse Spacing - CB1 Input Pulse	I_{ICS}	2	-	-	µs

DOCUMENT NO. 29000 D44
REVISION 2, OCT. 1978

PART NUMBER
R2114

R6500 Microcomputer System
DATA SHEET

1024 X 4 STATIC RANDOM ACCESS MEMORY

SYSTEM ABSTRACT

The' 8-bit R6500 microcomputer system is produced with N-Channel, Silicon-Gate technology. Its performance speeds are enhanced by advanced system architecture. Its innovative architecture results in smaller chips – the semiconductor threshold to cost-effectivity. System cost-effectivity is further enhanced by providing a family of software-compatible microprocessor memory, and I/O devices. . .as well as low-cost design aids and documentation.

DESCRIPTION

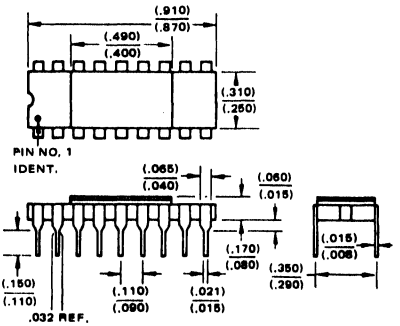
The R2114 is a 4096-Bit static Random Access Memory organized 1024 words by 4-bits. It is designed using fully DC stable (static) circuitry in both the memory array and the decoding and therefore requires no clocks or refreshing to operate. Address setup times are not required and the data is read out nondestructively with the same polarity as the input data. Common Input/Output pins are provided to simplify design of bus oriented systems, and can drive 2 standard TTL loads.

The R2114 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. It is totally TTL compatible in all respects: inputs, outputs, and the single +5V supply. A separate Chip Select ($\overline{CS}$) input allows easy selection of an individual device when outputs are or-tied.

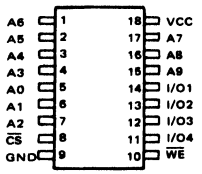
The R2114 is packaged in an 18-pin DIP for the highest possible density and is fabricated with N-channel Ion Implanted, Silicon-Gate technology – a technology providing excellent performance characteristics as well as protection against contamination allowing the use of low cost packaging techniques.

FEATURES

- 450 ns Maximum Access
- Low Operating Power Dissipation 0.1 mW/Bit Typical
- No Clocks or Strobes Required
- Identical Cycle and Access Times
- Single +5V Supply
- Totally TTL Compatible:
  All Inputs, Outputs, and Power Supply
- Common Data I/O
- 400 mv Noise Immunity
- High Density 18 Pin Package



Package Dimensions



Pin Configuration

Ordering Information

Order Number	Package Type	Access Time	Temperature Range
R2114C	Ceramic	450 nsec	0°C to +70°C
R2114P	Plastic	450 nsec	0°C to +70°C

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

DATA STORAGE

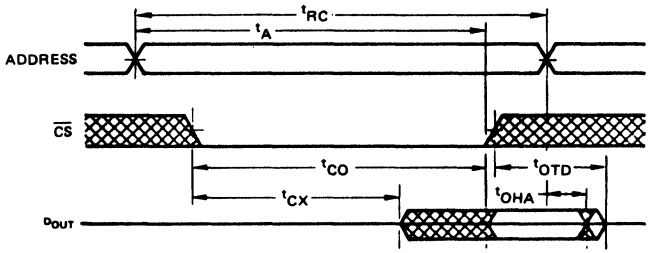
When $\overline{WE}$ is high, the data input buffers are inhibited to prevent erroneous data from being written into the array. As long as $\overline{WE}$ remains high, the data stored cannot be affected by the Address, Chip Select, or Data I/O logic levels or timing transitions.

Data storage also cannot be affected by $\overline{WE}$, Addresses, or the I/O ports as long as $\overline{CS}$ is high. Either $\overline{CS}$ or $\overline{WE}$ or both can prevent extraneous writing due to signal transitions.

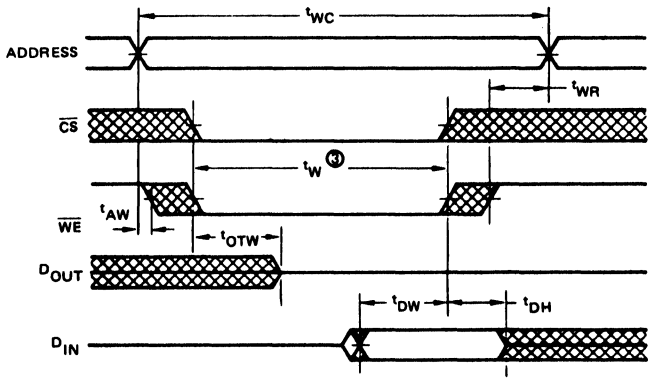
Data within the array can only be changed during Write time – defined as the overlap of $\overline{CS}$ low and $\overline{WE}$ low. The addresses must be properly established during the entire Write time plus t_{WR}.

Internal delays are such that address decoding propagates ahead of data inputs and therefore no address setup time is required. If the Write time precedes the addresses, the data in previously addressed locations, or some other location, may be changed. Addresses must remain stable for the entire Write cycle but the Data Inputs may change in the beginning of the cycle. The data which is stable for t_{DH} at the end of the Write time will be written into the addressed location.

Read Cycle ②



Write Cycle



Notes:

② $\overline{WE}$ is high for a Read Cycle

③ t_W is measured from the latter of $\overline{CS}$ or $\overline{WE}$ going low to the earlier of $\overline{CS}$ or $\overline{WE}$ going high.

Timing Diagrams

SPECIFICATIONS

Absolute Maximum Ratings

Temperature Under Bias	-10°C to 80°C
Storage Temperature	-65°C to 150°C
Voltage on Any Pin with Respect to Ground	-0.5V to +7V
Power Dissipation	1.0W

Comment

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Characteristics

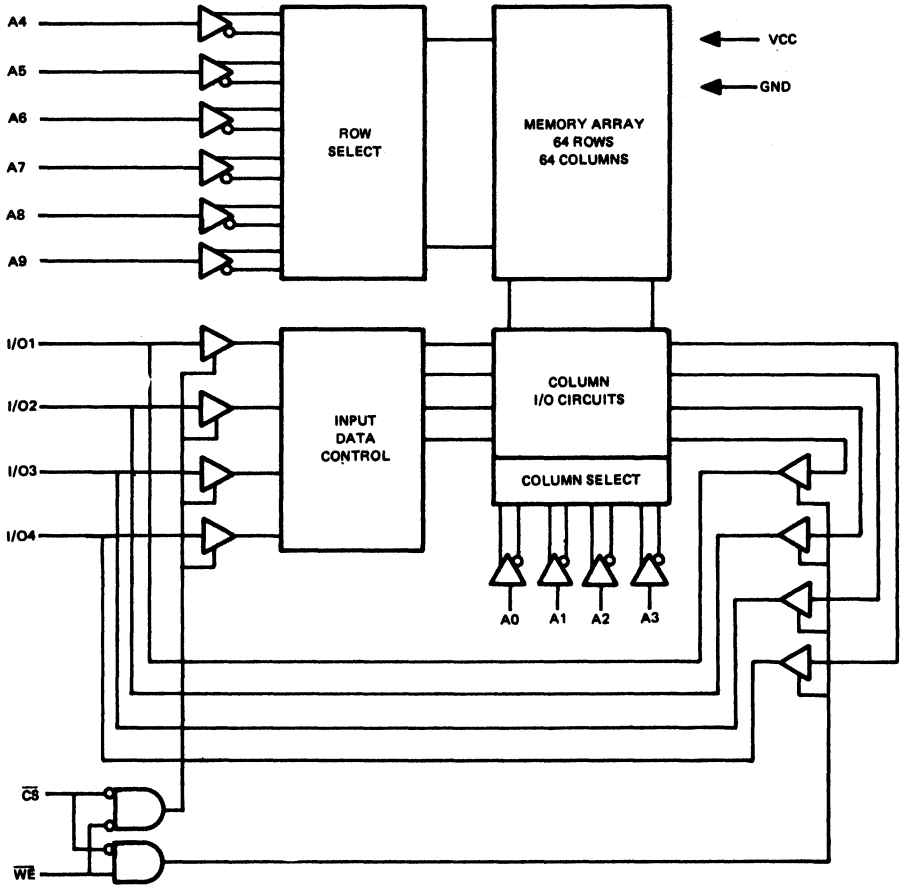
T_A = 0°C to +70°C, V_{CC} = 5V ±5% (Unless Otherwise Specified)

Symbol	Parameter	Min	Typ	Max	Units	Conditions
ILI	Input Current			10	µA	V_{IN} = 0 to +5.25V
ILO	I/O Leakage Current			10	µA	V_{IO} = 0.45V to 4.0V, $\overline{CE}$ = 2.0V
ICC1	Supply Current		80		mA	V_{CC} = 4.75V, I_{IO} = 0 mA, T_A = 25°C
ICC2	Supply Current				mA	V_{CC} = 4.75V, I_{IO} = 0 mA, T_A = 0°C
VIL	Input Low Voltage	-0.5		0.8	V	
VIH	Input High Voltage	2.0		V_{CC}	V	
VOL	Output Low Voltage			0.4	V	I_{OL} = 3.2 mA
VOH	Output High Voltage	2.4				I_{OH} = -1.0 mA
READ CYCLE						
t_{RC}	Read Cycle Time	450			ns	
t_A	Access Time			450	ns	
t_{CO}	Chip Select to Output Valid			150	ns	
t_{CX}	Chip Select to Output Enabled	0				
t_{OTD}	Chip Deselect to Output Off			150	ns	
t_{OHA}	Output Hold From Address Change	50			ns	
WRITE CYCLE						
t_{WC}	Write Cycle Time	450			ns	
t_{AW}	Address to Write Setup Time	0			ns	
t_W	Write Pulse Width	250			ns	
t_{WR}	Write Release Time	50			ns	
t_{OTW}	Write to Output Off			150	ns	
t_{DW}	Data to Write Overlap	250			ns	
t_{DH}	Data Hold	0				
CI/O	Input/Output Capacitance at 25°C			10	pF	f = 1 MHz
CIN	Input Capacitance at 25°C			8	pF	f = 1 MHz

NOTE: This parameter is periodically sampled and not 100% tested.

A.C. Test Conditions

Input Pulse Levels. . . . . 0.8V to 2.0V
Input Rise and Fall Time. . . . . 10 nsec
Timing Measurement Levels: Input 1.5V
Output . . . . . 0.8 and 2.0V
Output Load. . . . . 1 TTL Gate and 50 pF



Block Diagram

ROCKWELL INTERNATIONAL - MICROELECTRONIC DEVICES

REGIONAL SALES OFFICES

HOME OFFICE*
Rockwell International Corp.
Microelectronic Devices
P.O. Box 3669
Anaheim, Ca. 92803
U.S.A.
Phone: (714) 632-0960
TWX: 910-591-1698

CENTRAL REGION, U.S.A.
Contact Robert O. Whitesell & Associates
6691 East Washington Street
Indianapolis, Indiana 46219
(317) 356-9263 Attn: Mel Gamble, Mgr

EASTERN REGION, U.S.A.*
Caroller Office Building
990-970 U.S. Route 1
North Brunswick, New Jersey 08902
Phone: (201) 246-3630

MIDWEST REGION, U.S.A.
1011 E. Touhy Avenue, Suite 245
Des Plaines, IL 60018
Phone: (312) 297-8867

WESTERN REGION, U.S.A.
3310 Miraloma Avenue
P.O. Box 3669
Anaheim, Ca. 92803
Phone: (714) 632-0960

* Also Applications Centers

EUROPE
Rockwell International GmbH
Microelectronic Devices
Fraunhoferstrasse 11
D-8033 Munchen-Martinsried
Germany
Phone: (089) 859-9575
Telex: 0521/2650

FAR EAST
Rockwell International Overseas Corp.
Ichiben-cho Central Building
22-1 Ichiben-cho, Chiyoda-ku
Toyko 102, Japan
Phone: 265-8606
Telex: J22198

808

YOUR LOCAL REPRESENTATIVE

2048x8 Static Read Only Memory

SY2316A SY2316B

MEMORY PRODUCTS

Synertek®

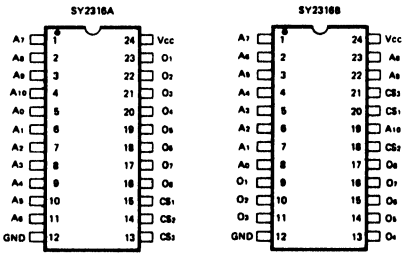
- 2048x8 Bit Organization
- Single +5 Volt Supply
- Metal Mask Programming
- Two Week Prototype Turnaround
- Access Time–550ns /450ns (max.)
- Totally Static Operation
- Completely TTL Compatible
- Three-State Outputs for Wire-OR Expansion
- Three Programmable Chip Selects
- SY2316A – Replacement for Intel 2316A
- SY2316B – Pin Compatible with 2708 EPROM
  – Replacement for Two 2708s

The SY2316A and SY2316B high performance read only memories are organized 2048 words by 8 bits with access times of less than 550 ns and 450 ns. These ROMs are designed to be compatible with all microprocessor and simllar applications where high performance, large bit storage and simple interfacing are important design considerations. These devices offer TTL input and output levels with a minimum of 0.4 Volt noise immunity in conjunction with a +5 Volt power supply.

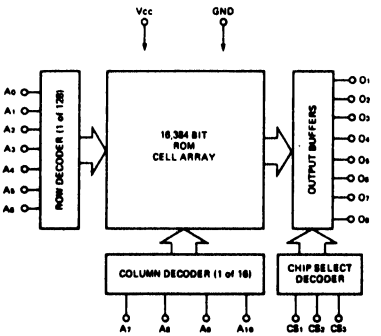
The SY2316A/B operate totally asynchronously. No clock input is required. The three programmable Chip Select inputs allow eight 16K ROMs to be OR-tied without external decoding. Both devices offer three-state output buffers for memory expansion.

Designed to replace two 2708 8K EPROMs, the SY2316B can eliminate the need to redesign printed circuit boards for volume mask programmed ROMs after prototyping with EPROMs.

PIN CONFIGURATION



BLOCK DIAGRAM



ORDERING INFORMATION

Order Number	Package Type	Access Time	Temperature Range
SYC2316A	Ceramic	550ns	0°C to +70°C
SYP2316A	Plastic	550ns	0°C to +70°C
SYC2316B	Ceramic	450ns	0°C to +70°C
SYP2316B	Plastic	450ns	0°C to +70°C

A custom number will be assigned by Synertek.

ABSOLUTE MAXIMUM RATINGS*

Ambient Operating Temperature	0° to +70°C
Storage Temperature	−65°C to +150°C
Supply Voltage to Ground Potential	−0.5V to +7.0V
Applied Output Voltage	−0.5V to +7.0V
Applied Input Voltage	−0.5V to +7.0V
Power Dissipation	1.0W

COMMENT*

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those indicated in the operational sections of this specification is not implied and exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

T_A = 0°C to +70°C, Vcc = 5.0V ± 5% (unless otherwise specified)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{OH}	Output HIGH Voltage	2.4	Vcc	Volts	Vcc = 4.75V, I_{OH} = −200 μA
V_{OL}	Output LOW Voltage		0.4	Volts	Vcc = 4.75V, I_{OL} = 2.1 mA
V_{IH}	Input HIGH Voltage	2.0	Vcc	Volts	
V_{IL}	Input LOW Voltage	−0.5	0.8	Volts	See Note 1
I_{LI}	Input Load Current		10	uA	Vcc = 5.25V, 0V ≤ V_{in} ≤ 5.25V
I_{LO}	Output Leakage Current		10	uA	Chip Deselected V_{out} = +0.4V to Vcc
Icc	Power Supply Current		98	mA	Output Unloaded Vcc = 5.25V, V_{in} = Vcc

Note 1: Input levels that swing more negative than −0.5V will be clamped and may cause damage to the device.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C, Vcc = 5.0V ± 5% (unless otherwise specified)

Symbol	Parameter	SY2316B		SY2316A		Units	Test Conditions
		Min.	Max.	Min.	Max.		
t_{ACC}	Address Access Time		450		550	ns	Output load: 1 TTL load
t_{CO}	Chip Select Delay		250		300	ns	and 100 pf
t_{DF}	Chip Deselect Delay		250		300	ns	Input transition time: 20ns
t_{OH}	Previous Data Valid After Address Change Delay	20		20		ns	Timing reference levels: Input: 1.5V Output: 0.8V and 2.2V

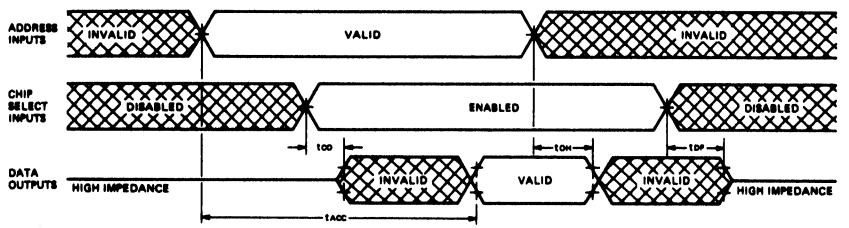
CAPACITANCE

t_A = 25°C, f = 1.0MHz, See Note 2

Symbol	Parameter	Min.	Max.	Units	Test Conditions
C_I	Input Capacitance		7	pF	All pins except pin under
C_O	Output Capacitance		10	pF	test tied to AC ground

Note 2: This parameter is periodically sampled and is not 100% tested.

TIMING DIAGRAM



PROGRAMMING INSTRUCTIONS

All Synertek read only memories utilize computer aided techniques to manufacture and test custom bit patterns. The custom bit pattern and address information is supplied on standard 80 column computer cards in the format described below.

All addresses and related output patterns must be completely defined. Each deck of cards defining a specific ROM bit pattern consists of 1) four Title Cards and 2) address and bit pattern Data Cards. Positive logic is generally used on all input cards: a logic "1" is the most positive or HIGH level, and a logic "0" is the most negative or LOW level. Synertek can also accept ROM data in other formats, compatible with most microprocessors and PROMS. Consult your Synertek representative for details.

TITLE CARDS

A set of four Title Cards should accompany each data deck. These cards give our computer programs additional information necessary to accurately produce high density ROMS. These four Title Cards must contain the following information:

	COLUMN	INFORMATION
First Card	1–30	Customer name
	31–50	Customer part number
	60–72	Synertek part number (punch "2316A" or "2316B")
Second Card	1–30	Customer contact (name)
	31–50	Customer telephone number
Third Card	1–6	Leave blank – pattern number to be assigned by Synertek
	30	$CS_3/\overline{CS_3}$ chip select logic level (if LOW selects chip, punch "0"; if HIGH selects chip, punch "1")
	31	$CS_2/\overline{CS_2}$ chip select logic level.
	32	$CS_1/\overline{CS_1}$ chip select logic level.
Fourth Card	1–8	Data Format. Synertek, or Intel data card format may be used. Specify format by punching "Synertek," or "Intel" starting in column one.
	15–28	Logic format; punch "POSITIVE LOGIC" or "NEGATIVE LOGIC."
	35–57	Truth table verification code; punch either "VERIFICATION HOLD" (manufacturing starts after customer approval of bit pattern data supplied by Synertek) or "VERIFICATION NOT NEEDED" (manufacturing starts immediately upon receipt of customer card deck)

SYNERTEK DATA CARD FORMAT

All addresses are coded in decimal form (0 through 2047). All output words are coded both in binary and octal forms. Output 8 (O_8) is the MSB, and Output 1 (O_1) is the LSB.

	COLUMN	INFORMATION
Data Cards	1–4	Decimal address
	6–13	Output (MSB-LSB)
	15–17	Octal equivalent of output data
	22–25	Decimal address
	27–34	Output (MSB-LSB)
	36–38	Octal equivalent of output data
	43–46	Decimal address
	48–55	Output (MSB-LSB)
	57–59	Octal equivalent of output data
	64–67	Decimal address
	69–76	Output (MSB-LSB)
	78–80	Octal equivalent of output data

INTEL DATA CARD FORMAT

Output data is punched as either a "P" or an "N"; a "P" is defined as a HIGH, and an "N" is defined as a LOW. Output 8 (O_8) is the MSB and Output 1 (O_1) is the LSB. The four Title Cards listed above must accompany the Intel card deck.

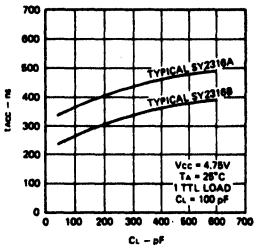
	COLUMN	INFORMATION
Data Cards	1–5	Punch the 5-digit decimal equivalent of the binary coded address which begins each card. This is the initial input address. The address is right justified, i.e. 00000, 00008, 00016, etc.
	7–14	Output data (MSB-LSB) for initial input address.
	16–23	Output data for initial input address +1
	25–32	Output data for initial input address +2
	34–41	Output data for initial input address +3
	43–50	Output data for initial input address +4
	52–59	Output data for initial input address +5
	61–68	Output data for initial input address +6
	70–77	Output data for initial input address +7
	79–80	ROM pattern number (may be left blank)

Send bit pattern data to the following special address:

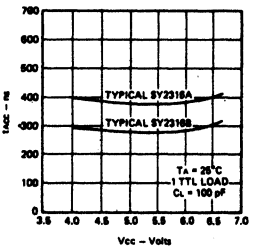
Synertek – ROM
P.O. Box 562
3050 Coronado Drive
Santa Clara, CA 95051

TYPICAL CHARACTERISTICS

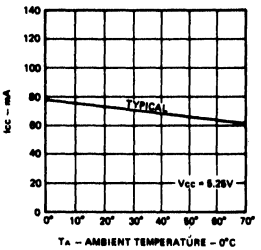
ACCESS TIME VS. CAPACITIVE LOAD



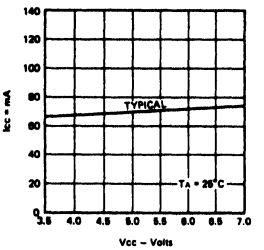
ACCESS TIME VS. SUPPLY VOLTAGE



SUPPLY CURRENT VS. AMBIENT TEMPERATURE

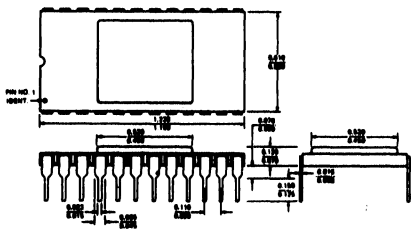


SUPPLY CURRENT VS. SUPPLY VOLTAGE

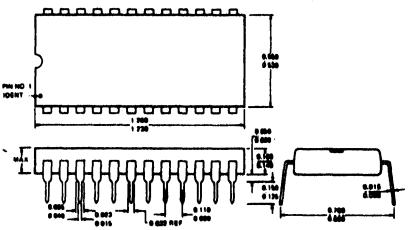


PACKAGING DIAGRAM

CERAMIC PACKAGE



PLASTIC PACKAGE



DOCUMENT NO. 29000 P54
AUGUST 1978

PART NUMBER
R6541

R6500 Microcomputer System
PRODUCT PREVIEW

PROGRAMMABLE KEYBOARD/DISPLAY CONTROLLER (PKDC)

DESCRIPTION

The R6541 Programmable Keyboard/Display Controller (PKDC) is a general purpose keyboard and segmented display interface device designed for use with 8-bit microprocessors. The R6541 adds an advanced keyboard scan and display refresh capability to the established and expanding line of R6500 products. The 8-bit R6500 microcomputer system is produced with N-channel, silicon gate, depletion load technology. The autonomous operation of the R6541 relieves the CPU from performing high demand periodic keyboard/display service functions.

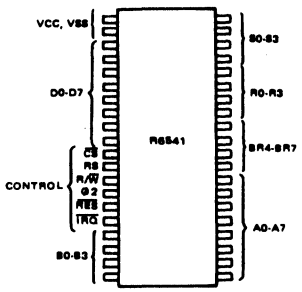
The R6541 has two sections: keyboard and display. The keyboard portion can scan up to 128 matrix type key switches. It can also interface with an array of 64 sensors (static switches) or a strobed interface keyboard. Key depression can be N-key rollover type. Key entries are debounced and stored in an 8-character First In First Out (FIFO)/Sensor RAM. A programmable length interrupt prevents FIFO/Sensor RAM overflow.

The display section provides a buffered scanned display interface with LED, fluorescent, Burroughs SELF-SCAN® display, and other display technologies. Numeric and alphanumeric displays may be directly driven as well as simple indicators. The R6541 has two CPU addressable 16 x 8 display RAM's.

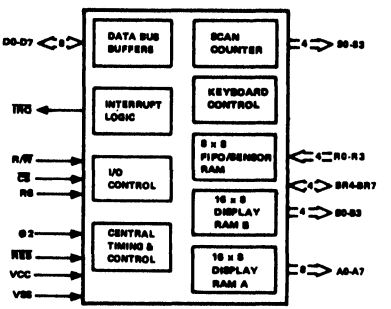
A unique R6541 feature is the 4-line bidirectional BR Display/Return bus. This flexible sensor input or display output capability allows the R6541 to be easily configured to specific applications. When the BR lines are used as outputs, two independent 8 segment, 16-character displays or one 16 segment, 16 character display can be directly driven along with scanning a 64 key encoded keyboard. When the BR lines are used as input, a 12 segment, 16 character display can be directly driven along with scanning a non-encoded 16 x 8 keyboard or sensor matrix. Many other application configurations are possible.

FEATURES

- Compatible with 8-bit microprocessors
- 8-Character FIFO/Sensor RAM with programmable interrupt
- N-Key rollover
- 12, 16 or dual 8 segment display outputs
- Flexible 16 x 8 keyboard matrix scan modes
- Flexible 8 x 8 sensor matrix scan modes
- Programmable keyboard/sensor mode partitioning
- Strobed output entry mode
- Fully programmable timing
  - Key scan
  - Debounce
  - Display scan
  - Digit and segment
- 2 MHz or 1 MHz operation
- Single +5V ± 10% power supply
- 40-pin plastic or ceramic DIP



R6541 Pin Configuration



R6541 Interface Diagram

© Rockwell International Corporation 1978
All Rights Reserved
Printed in U.S.A.

Specifications subject to change without notice

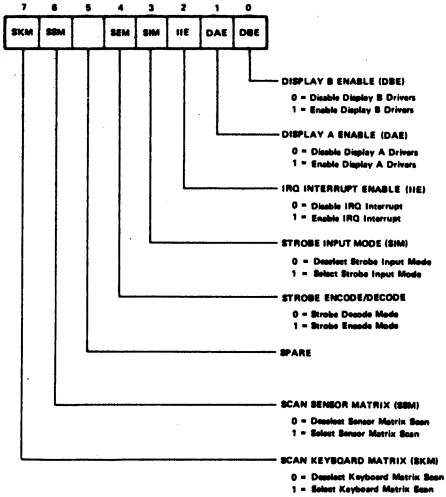
R6541 INTERFACE SUMMARY

Pin Name	No.	Description	Type
S0-S3	4	Scan Outputs	Output
R0-R3	4	Return Inputs	Input
BR4-BR7	4	Display B Outputs/ Return Inputs	Bidirectional
B0-B3	4	Display B Outputs	Output
A0-A7	8	Display A Outputs	Output
D0-D7	8	Data Bus	Bidirectional
$\overline{CS}$	1	Chip Select	Input
RS	1	Register Select	Input
R/$\overline{W}$	1	Read/$\overline{Write}$	Input
Ø2	1	Phase 2 Clock	Input
$\overline{RES}$	1	Reset	Input
$\overline{IRQ}$	1	Interrupt Request	Output
VCC, VSS	2	Power	Input

INTERFACE SIGNAL DESCRIPTION

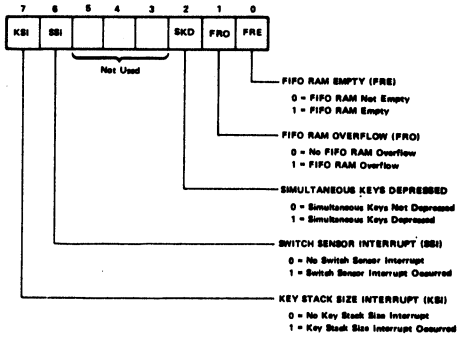
Pin Name	No. of Lines	Description
S0-S3	4	Scan Outputs. These lines scan, or strobe, the key switch or sensor matrix and display digits. The lines can be either encoded (1 of 16) or decoded (1 of 4).
R0-R3	4	Return Inputs. These lines input the data from matrix type keys, static switches, or a strobed keyboard. The lines can be used as a 4-bit input or as part of an 8-bit input in conjunction with BR4-BR7. Each line has an active pullup resistance to keep it high until a switch closure pulls it low.
BR4-BR7	4	Display B Outputs/Return Inputs. These bidirectional lines may be used either as an extension of the Return Inputs or as an extension of the Display B outputs. The lines are assigned as either inputs or outputs at system definition time; therefore, they may not be used as both inputs and outputs in a given application. These lines may also be used as an independent 4-bit output or an independent 4-bit input. Internal pullups are provided for use as inputs. In a scanned keyboard matrix application, BR6 and BR7 may be used as CONTROL and SHIFT, respectively. When used as outputs, these lines are driven from the 16 x 8 Display RAM B. When used as outputs, these lines are synchronized to the scan-outputs (S0-S3) for multiplexed digit displays.
B0-B3	4	Display B Outputs. These lines are outputs from the 16 x 8 Display RAM B. The lines may be used as a 4-bit output port or may be considered as part of an 8-bit output port in conjunction with BR4-BR7. When used as outputs, these lines are synchronized to the scan outputs (S0-S3) for multiplexed digit displays.
A0-A7	8	Display A Outputs. These lines are the outputs from the 16 x 8 Display RAM A. The lines may be used as an 8-bit output or two 4-bit outputs. When used as an output, these lines are synchronized to the scan outputs (S0-S3) for multiplexed digit displays.
D0-D7	8	Data Bus. Eight bidirectional tri-state lines used to transfer all the data between the CPU and the R6541.
$\overline{IRQ}$	1	Interrupt Request. This Interrupt Request output is used in the Keyboard or Strobe Input Modes to interrupt the CPU when driven to the low state. An external pullup resistor is required.
$\overline{RES}$	1	Reset. This high impedance input line resets the R6541 to the power-on initialization condition when driven to the low state.
$\overline{CS}$	1	Chip Select. This high impedance input line selects the R6541 when in the low state.
RS	1	Register Select. This high impedance input line is used to select specific R6541 registers. When low, the address Pointer can be written and the Status Register read. When high, the Address Pointer identifies the specific register to be used for data transfer.
R/$\overline{W}$	1	Read/$\overline{Write}$. This high impedance line controls the transfer of data between the CPU and the R6541. When high (read), data is transferred from the R6541 to the CPU. When low (write), data is transferred from the CPU to the R6541.
Ø2	1	Phase 2 Clock. This clock input signal is used to synchronize internal logic and generate internal timing.
VCC	1	Power. This input line supplies +5V ±10% operating power.
VSS	1	Signal Ground. This line is power and signal return.

The Control Register allows the CPU to select display and keyboard functions.

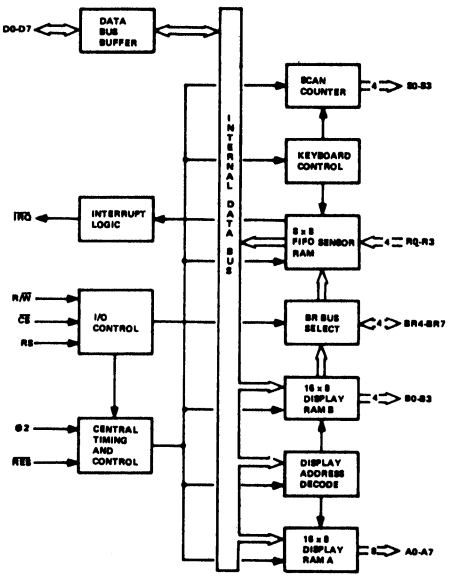


R6541 Control Register

The Status Register reports the status of various conditions and interrupts.



R6541 Status Register

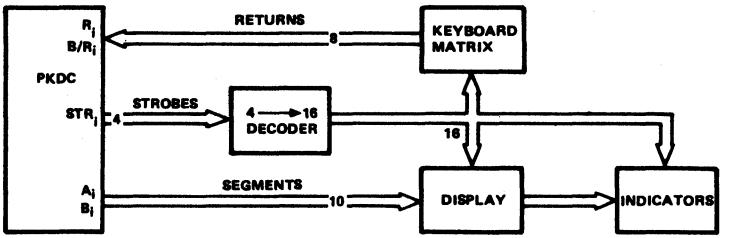


R6541 Block Diagram

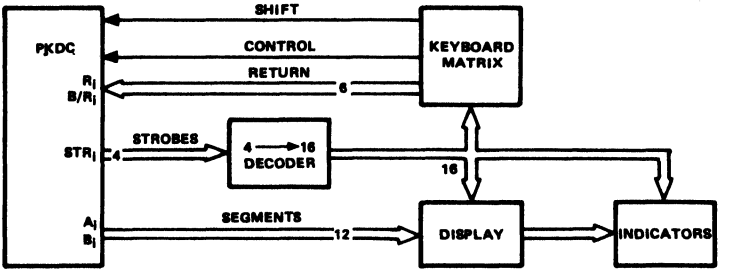
Address Pointer
Status Register
RAM A0 Display
↓
RAM A15 Display
RAM B0 Display
↓
RAM B15 Display
FIFO/SENSOR RAM
FIFO SENSOR Address Pointer
FIFO/SENSOR Key Stack Size
Internal Clock
Keyboard Scan Time
Keyboard Debounce Time
Key Scan Time
Display Scan Time
Digit "On" Time
Segment "On" Time
Control Register

R6541 Register Organization

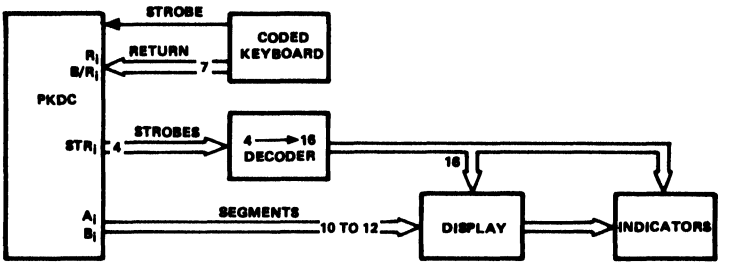
APPLICATION EXAMPLES



128 KEY POSITIONS AND
16 CHARACTER (10 SEGMENT) DISPLAY



96 KEY POSITIONS WITH SHIFT, CONTROL AND
16 CHARACTER (12 SEGMENT) DISPLAY



KEYBOARD CODED INPUT AND
16 CHARACTER (10-12 SEGMENT) DISPLAY

ROCKWELL INTERNATIONAL - MICROELECTRONIC DEVICES

REGIONAL SALES OFFICES

HOME OFFICE*
Rockwell International Corp.
Microelectronic Devices
P.O. Box 3669
Anaheim, Ca. 92803
U.S.A.
Phone: (714) 632-3960
TWX: 910-591-1698

* Also Applications Centers

CENTRAL REGION, U.S.A.
Contact Robert O. Whitesell & Associates
6681 East Washington Street
Indianapolis, Indiana 46219
(317) 359-9263 Attn. Milt Gamble, Mgr.

EASTERN REGION, U.S.A.*
Caroller Office Building
560-570 U.S. Route 1
North Brunswick, New Jersey 08902
Phone: (201) 246-3630

MIDWEST REGION, U.S.A.
1011 E. Touhy Avenue, Suite 245
Des Plaines, IL 60018
Phone: (312) 297-8862

WESTERN REGION, U.S.A.
3310 Miraloma Avenue
P.O. Box 3669
Anaheim, Ca. 92803
Phone: (714) 632-3960

EUROPE
Rockwell International GmbH
Microelectronic Devices
Fraunhoferstrasse 11
D-8033 Munchen-Martinsried
Germany
Phone: (089) 859-9575
Telex: 0521/2650

FAR EAST
Rockwell International Overseas Corp.
Ichiban-cho Central Building
22-1 Ichiban-cho, Chiyoda-ku
Toyko 102, Japan
Phone: 265-3506
Telex: J22198

YOUR LOCAL REPRESENTATIVE

808

NOTIZEN

NOTIZEN

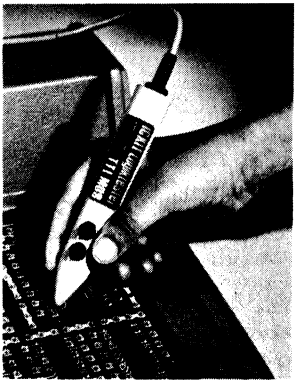
NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

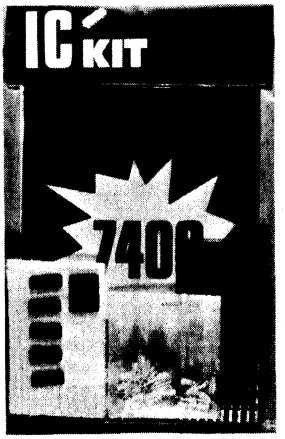


LOGIKTESTER FÜR TTL u. CMOS

Ein praktischer Prüfstift für logische Pegel in Digitalschaltung. Geringste Stromaufnahme. Die Anzeige erfolgt durch zwei LEDs. Bausatz komplett mit Gehäuse, Kabel, Stecker, genauer Bauanleitung und einem Experimentierheft mit vielen Schaltgn.

Best.Nr. 55 Bausatz DM 24.80

Best.Nr.56 fertig montiertes Gerät zusammen mit Buch DM 39.80



DIGITALTECHNIK EXPERIMENTIERSATZ (Bild rechts) TTL

TTL Experimentiersatz. Eine kleine Einführung in die Digitaltechnik und TTL -Technik. Büchlein zusammen mit fünf TTL integrierten Schaltkreisen. Experimentierplatine für drei ICs (14 oder 16 polig) zwei IC Sockel und Steckermaterial und Stifte.
Best. Nr. 105 nur DM 19.80

CMOS Experimentiersatz. Wie Nr.105, jedoch mit CMOS Bausteinen und CMOS - Einführungs und Schaltungsbuch. Der ideale CMOS Kurs. Best.Nr. 106 nur DM 19.80

Hiermit gebe ich folgende Bestellung unter Anerkennung Ihrer Lieferungs- und Zahlungsbedingungen auf.

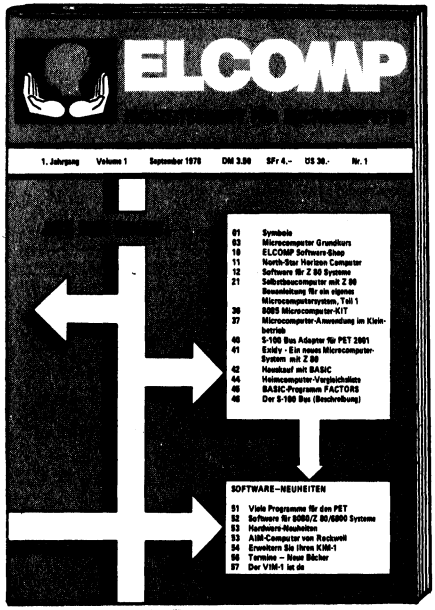
ELCOMP

Menge	Artikelbezeichnung	Einzelpreis

Menge	Artikelbezeichnung	Einzelpreis

Unterschrift (für Jugendliche unter 18 J. der Erziehungsberechtigte) **Datum**

ELCOMP
Die Fachzeitschrift für MICROCOMPUTER
Eine unentbehrliche Informationsquelle für alle Elektroniker



- Microcomputer-Anwendungsbeispiele
- Künstliche Intelligenz
- Block-Strukturierte Programme
- Datenverarbeitung im Kleinbetrieb
- Club-Neuheiten
- Computer und Kunst
- Musik mit dem Computer
- Monitore für 8080, 6800, 6502, Z 80, SC/MP, 2650, 1802
- Eigenbau-Computersysteme
- Interface-Techniken
- Microcomputer KITs
- Neue Produkte
- Betriebssysteme für Floppys
- Programmiertechniken
- Software-Quellen
- Programmierbeispiele
- Soziale Aspekte der Microcomputer-technik
- Technologische Neuheiten
- Anwendungen in der Meß- und Regel-technik
- Anwendungen bei Funk-Amateuren

Für Ihre Abonnementbestellung verwenden Sie bitte das Formular auf Seite 194. Der Preis beträgt pro Jahr DM 59,-- incl. Versand, Porto und Verpackung. 10 Hefte pro Jahr, wobei für Juli/August und November/Dezember Doppelhefte geliefert werden. Abonnieren Sie noch heute.

Abs.

Name

_____

Straße:

_____

Ort:

_____

Bestellkarte

Den Gesamtbetrag für die umseitig bestellten Bücher bitte ich per Nachnahme zu berechnen/habe ich heute auf Ihr Postscheckkonto München 15994-807 überwiesen. Nichtzutreffendes bitte streichen.

○ Bitte liefern Sie mir auch die Experimentierplatine IC - KIT WH-1 g für 14,16,24,28 und 40 polige DIL IC-Gehäuse. Bausatz komplett: DM 79.—

○ Bitte liefern Sie von jedem neu erscheinenden Buch ein Muster per Nachnahme. (Rückgaberecht)

Ing.
W. Hofacker GmbH
Verlag
Postfach 437
8000 München 75

PET Business Software auf Cassette

General Ledger-Hustler 1 (PET, 8K)
Ein Multi-Report Programm
Es zeigt alle Abhebungen vom Konto mit Datum, Scheck-Nr., Betrag und Kurzbeschreibung sowie alle Scheckeinreichungen und Einzahlungen. Report über alle Summen
Codierte Eingänge werden gespeichert und können aufaddiert und angezeigt werden. Führt Berechnungen durch und zeigt Ausgaben/Einnahmen etc. an.
Best.-Nr. P37 **DM 69,--**

Checking Account(PET, 8K)
Bis zu 140 Scheck-Eingaben pro Speicherzyklus (18 verschiedene Gruppierungen)
Eingänge und Ausgänge auf einem Konto können genau verfolgt werden. Zwischensummen und Endsummen. Drei verschiedene Kontengruppen sind möglich.
Best.-Nr. P38 **DM 79,--**

Trust Account
Ein Programm für den Rechtsanwalt. Datenablage, Speicherung und Aufruf, Verwaltung von Kundenkonten.
Best.-Nr. P39 **DM69,--**

Legal Diary
Ein Programm für den Rechtsanwalt. Es führt Buch über die verwendete Zeit pro Kunde und Zeitraum. Es addiert die Kosten auf und erstellt Zwischensummen. Multi-Report-System.
Best.-Nr. P40 **DM 69,--**
Rent Account, Best.Nr. P41 **DM 69,--**
Dual Joystick, Fairchild
Jetzt können Sie mit zwei Steuerknüppeln gleichzeitig Symbole und Buchstaben oder sogar Bilder auf dem Bildschirm verschieben. Ideal als externe Eingabe oder Steuerung Ihres PETs. Auch für Spiele bestens geeignet. Sie können jetzt gegen einen Partner oder auch zusammen oder alle gegen den Computer spielen. Zwei Joysticks mit Stecker und Softwarecassette. Utility Programm, drei sehr interessante Spiele mit Toneffekten (über Verstärker + Lautsprecher)
Best.-Nr. P42 **DM 279,--**

Dual Joystic, ATARI
Ähnlich wie P42. Beide Joysticks arbeiten über den User-Port. Incl. Software auf Cassette und Programmieranleitung.
Best.-Nr. 43 **DM 279,--**

Music-Box
Ein Verstärker mit Lautsprecher in einem schönen Gehäuse. Anschlußkabel und Stecker. Keine externe Stromversorgung nötig. Sehr laut. Komplett mit Software und Programmanleitung.
Best.-Nr. P44 **DM 199,--**

VOICE SET
Jetzt kann Ihr PET endlich hören und Töne von sich geben. Spracheingabe und Tonausgabe für Ihren PET. Komplett montierte Eingabe-Electronic mit Lautsprecher, Gehäuse, Verstärker, zwei Cassetten mit Software, VOICETRAP und SOUNDBREAK. Manuals
P45 **DM 249,--**

Druckerinterface für RS232
vom IEC-Bus auf RS232. Ohne Netzteil (± 12V/20 mA werden benötigt) ohne Software
P46 **DM 669,--**
P26 zugehörige Software (word processing) **DM 96,--**

Computerspiele
Eine neue Cassette mit lustigen und sehr interessanten Spielprogrammen. (Schießbude, Flugzeugabschießen, Tic, Tac, Toe, Grundsätzliches über Spielprogramme mit Anleitungstext.
P47 **DM 19,80**

Casino und Spielesimulationen
Die ideale Programmcassette für den leidenschaftlichen Spieler. Simulationen und Spiele, die sich ums Roulett und Lotto drehen. Auch für jeden Lotto- und Totospieler bestens geeignet. Vielleicht bringt es Ihnen den Haupttreffer?
P48 **DM 19,80**

Programmieren in Maschinensprache
Diese Cassette braucht jeder, der den PET in Maschinensprache pro-

grammieren möchte. Viele nützliche Routinen, einfacher Assembler, Tricks u. v. a. mehr

P49 **DM 39,80**

Ein-/Ausgabeprogrammierung mit PET

Wer hier einmal angefangen hat, mit dem PET die Außenwelt zu verbinden, wird begeistert sein. Steuerungen, externe Schaltungen, Schaltuhren, Lauflicht u. v. a.

P50 **DM 49,80**

Advanced BASIC

Ein BASIC-Kurs für Fortgeschrittene. Tricks und viele Hinweise. Komplizierte Programmiertechniken auf einfache Weise erklärt und nähergebracht.

P51 **DM 69,--**

PILOT

Eine Programmsprache für Kinder (6 — 12) , ideal für den Anfänger. Cassette mit Manual

P52 **DM 49,--**

Diät- und Gesundheitsprogramme

Der PET im Dienste Ihrer Gesundheit. Kalorien, Gewichtsabnahme etc

P53 **DM 19,80**

220V/50 Hz Schaltinterface (Bausatz)

Ein Schaltinterface für Ihren PET. Sie können per Programm einen Wechselspannungsverbraucher 220V/1,5A schalten. (Stereoanlage, Cassettenrecorder etc.). Mit Software und Gehäuse, Stecker, Bauanleitung, Listing auf Cassette u. Papier.

P54 **DM 169,--**

Externe Experimentierplatine für PET

Eine externe Experimentierplatine. Jetzt können Sie externe Hardwareerweiterungen über den USER-Port selbst aufbauen.

Die externen Bauelemente können gesteckt werden (kein Löten). Per Programm können die gewünschten Signale auf das Board gegeben werden. Mit Kabel, Stecker, Beschreibung und Grundsoftware auf Cassette.

P55 **DM 199,--**

Analog Digital/Digital Analog Wandler Platine
Ideal für jeden, der den PET zum Messen, Prüfen, Musikerzeugen und Ein-/Ausgabe von Analogsignalen benötigt. Linearität D/A ± 1/2 LSB. ADC-Wandler. Successive Aproximation, 4µs Einstellzeit. Komplett mit phantastischer Software.
P56 **DM 499,--**

Microcomputer Magazin ELCOMP
Jeden Monat viele Programme und Anleitungen.
Einzelpreis **DM 4,50**
Jahresbezugspreis, incl. **DM 59,--**

Audiocassette mit PET-Musikdemonstrationen
P57 **DM 9,80**

Cassette mit Computer Musik (Musik der Zukunft)
Ein phantastischer Sound, Stereo, Bach, Paul Linke u. v. a.
P58, Cassette **DM 19,80**
P59, Stereoschallplatte **DM 19,80**

TRS-80 Software auf Cassette

Stimulating Simulations
Viele Spiele und Simulationen, Cassette mit Manual
Best.-Nr. TR1 **DM 39,80**

Graphik für TRS-80
Graphik in Vollendung. Viele Programme und Anregungen
TR2 **DM 39,80**

MICHROCHESS für TRS-80
Für Level I und Level II, 4 K RAM
TR3 **DM 79,--**

TRS-80 Programmbibliothek (100 Programme) Level II
Eine Sammlung von Programmen aus Finanzwesen, Spiele, Personal Finanzen, Graphik Business, Education, Mit Manual. Fünf Cassetten im Ordner.
TR4 **DM 199,--**

Software für Apple II

Professionelle Sekretärin
Ein Programmpaket für alle, die Ihre Zeit sinnvoll planen und nutzen müssen. Ideal für den Rechtsanwalt, Steuerberater, Manager etc. Das File System sorgt für die Einhaltung Ihrer Termine und speichert alle Termine für das ganze Jahr. Über eine besondere Kartei (File) können wichtige Telefonnummern abgerufen werden
Best.-Nr. A001, Cassette **DM 299,--**

Executive Management Übersicht
Dieses Programm wurde speziell für leitende Angestellte, Geschäftsführer und Direktoren entwickelt. Es können Verkaufszahlen, Kosten, maximale Lagerbestände und Gewinne über einen Zeitraum von 25 Monaten oder zwei Quartale aufgezeichnet werden. Die Darstellung erfolgt in Farbgraphik. 4K RAM sind erforderlich.
Best.-Nr. A004, Cassette **DM 149,--**

DATA Management
Eine universell einsetzbare Datenbank zur Speicherung von Daten aller Art. (Rezepte, Technische Berichte etc) bis hin zu Buchhaltungsdaten)
Best.-Nr. A006, Cassette **DM 78,--**

Inventur Programm
Dieses Programm wurde so entwickelt, daß Sie einen ständigen Überblick über Ihren Warenbestand haben. (16KRAM)
Best.-Nr. A017 **DM 299,--**

The BASIC-Teacher

Ein Lehr- und Lernprogramm für den Apple. 12 Lektionen, Apple Tricks, Graphik, Ton und Musik, PEEK und POKE u. v. a.
Best.-Nr. A014, Cassette **DM 84,--**

Invoicing
Ein Programm für Kleinbetriebe. Rechnungen und Lieferscheine werden geschrieben und gleichzeitig Rückstandsbearbeitungen durchgeführt. (16 K RAM)
Best.-Nr. A011, Cassette **DM 189,--**

Private Sekretärin
Die Privat-Datenbank für den Alltagsgebrauch. Telefonnummern, Adressen, Termine etc. (4–8K RAM)
Best.-Nr. A002, Cassette **DM 189,--**

Programmierte Gymnastik
Dieses Programm gibt Ihnen genaue Anleitungen über Ihre täglichen Übungen, Schwierigkeitsgrade sind programmierbar. Ein Programm mit Bewegung.
Best.-Nr. A007, Cassette **DM 63,--**

Billing Management
Ein Fakturierprogramm für kleine Betriebe. Das System bereitet die Rechnungserstellung vor, schreibt Lieferscheine, erstellt monatliche Zusammenfassungen und eine Aufstellung pro Konto und Monat.
(16K RAM)
Best.-Nr. A015, Cassette **DM 189,--**

Retail Management
Ein Programm zur Unterstützung eines Einzelhandels-Kaufmannes. Einkauf, Preisgestaltung und Kredit werden vom Computer unterstützt. (16 K RAM-Bedarf)
Best.-Nr. A016, Cassette **DM 189,--**

Asset Record Program
Ein Programm zur Erfassung des Inventurwertes des eingesetzten Kapitals im Kleinbetrieb.
Best.-Nr. A010, Cassette **DM 189,--**

Best.Nr.

1

TBB - Handbuch Band 1 , W. Hofacker

Transistor Berechnungs- und Bauanleitungshandbuch Band 1

Das Handbuch für jeden Elektroniker. Rechenbeispiele, Berechnungsgrundlagen, Bauanleitungen, Nomogramme, Tabellen und Vergleichslisten aus den wichtigsten Bereichen der Elektronik. Ein Buch zum Einarbeiten in die Elektronik. Ein Buch zum Nachschlagen. Grundlagen Digitaltechnik, Netzgeräte und Transformatorenberechnung, Berechnung von Multivibratoren, Schmitt Trigger u. v. a. Über 130 Seiten. **DM 19,80**

2

TBB - Handbuch Band 2 , W. Hofacker

Transistor Berechnungs- und Bauanleitungshandbuch Band 2

Dieses Buch ist die Fortsetzung des erfolgreichen Handbuches TBB-Handbuch Band 1. Ein Buch, das sich in der Hand des Praktikers bestens bewährt hat. Weitere neueste Schaltbeispiele und Berechnungsgrundlagen. Experimentier- und Versuchsbeschreibungen. Integrierte Spannungsregler, Wärmeableitung, Operationsverstärker Einführung, RC-Zeitglieder, Transistortester u. v. a. **DM 19,80**

3

Elektronik im Auto , H. Gebauer

Ein Buch für jeden technisch interessierten Autofahrer. Es zeigt Ihnen die vielen Möglichkeiten zur Verbesserung von Sicherheit, Leistung und Fahrkomfort in Ihrem Auto. Thyristorzündung, Drehzahlmesser, Beschleunigungsmesser, Geschwindigkeitswarner, Batterieladegerät u. v. a. Tips, genaue Beschreibungen, Bauanleitungen. **DM 9,80**

4

IC - Handbuch , C. Lorenz

Handbuch für digitale und lineare integrierte Schaltungen.

Sensationelle Neuheit. Ein Handbuch für digitale und lineare Schaltkreise. Daten- und Auswahllisten, Vergleichslisten, Gehäuseformen, Grundlagen, Einführungsbeschreibungen, viele Schaltbeispiele, Bauanleitungen, Printvorlagen, u. v. a.
Alles über TTL-Technik, C MOS, MOS-Schaltungen, Uhren ICs, lineare Schaltungen, ein Chip-Rechner, integrierte NF-Verstärker u. v. a. **DM 19,80**

5

IC - Datenbuch , D. Steinbach

Daten- und Auswahllisten der gebräuchlichsten integrierten Schaltkreise. Digital und analog. Gerade bei ICs ist es wichtig die Anschlußfolgen genau zu kennen. Auf über 55 Seiten finden Sie: Die wichtigsten TTL-Schaltkreise, C-Mos Serie, lineare Schaltungen wie Operationsverstärker, Komparatoren, NF-Verstärker, Spannungsregler, Triggerschaltungen, Impulsgeber u. v. a. Weiterhin finden Sie eine C Mos-Vergleichsliste sowie Kurzdaten und logisches Verhalten dieser C MOS Elemente. Das IC-Datenbuch wird auch Ihnen ein unentbehrlicher Begleiter bei allen Arbeiten mit integrierten Schaltungen sein. **DM 9,80**

6

IC - Schaltungen , D. Steinbach

Hier finden Sie eine gelungene Zusammenstellung der wichtigsten Anwendungsbeispiele aus dem Bereich der integrierten Schaltungen. TTL – C MOS – Linear. Alle Schaltungen sind übersichtlich und klar dargestellt und mit einer kurzen, jedoch sehr genauen Beschreibung versehen. Viele Schaltungen sind Grundschaltungen, die man beim Umgang mit integrierten Schaltungen immer wieder benötigt. Tastenentprellung, Zähler, Impulsgeber, Codierer, Dekodierer, Datenübertragung, Serien-Parallel-Wandler, Digitalvoltmeter u. v. a. **DM 9,80**

7

Elektronik Schaltungen,
4. völlig neu überarbeitete Auflage, W. Hofacker

Die ideale Schaltungssammlung zum Basteln und Experimentieren. Schaltungen mit Operationsverstärkern, Spannungsreglern, TTL, C-MOS Schaltkreisen. MOS Uhr mit Wecker, elektronischer Würfel, Musik Synthisizer, Timer 555 Anwendungen, Experimentieranleitungen und viele andere hochinteressante Schaltbeispiele tlw. mit Printvorlage. 64 Seiten Inhalt. **DM 5,–**

8

IC - Bauanleitungs -Handbuch -IC -KIT , C. Lorenz

Ein Bauanleitungsbuch mit vielen hochinteressanten Bauanleitungen aus dem Bereich der LSI Schaltungstechnik. Schaltbeispiele mit Printvorlagen zum Selbstherstellen der Leiterplatten mit genauesten Beschreibungen. Hochaktuell und brandneu: Funktionsgenerator XR 2206, MOS-Uhr mit Wecker, Schlummerautomatik und programmierbarem Wecktongenerator, Sensortastenwahl, IC-Netzteil, Funktionsgenerator 8038 neuartige Transistorzündung, 35 W NF-Verstärker, Experimentieranleitung und Grundkurs über Flip Flops, Experimente mit Digitalschaltungen u. v. a. Zu allen Schaltungen finden Sie Platinvorlagen oder Sie können die Experimentierschaltungen auf der Experimentierplatine WH-1 g durchführen. Über 125 Seiten. **DM 19,80**

41

Experimentierplatine mit Sockel, Stecker und Füßen Typ WH-1g für 40,28,24,16 und 14 polige DIL -Gehäuse **DM 79,–**

9

Feldeffekttransistoren, C. Lorenz

Der Feldeffekttransistor (FET) gehört heute zu den interessantesten Bauteilen überhaupt. Wie man damit experimentiert, wie man seine Funktion versteht und wie man damit brauchbare und hochinteressante Schaltungen aufbauen kann, zeigt Ihnen dieses Buch. Grundlagen, Kennlinienfelder, Tabellen, Berechnungsgrundlagen, Rechenbeispiele, Anschlußbilder und eine Vergleichsliste für Feldeffekttransistoren bilden den Kern dieser umfangreichen Darstellung. Alles in allem finden Sie hier eine praxisnahe und komplette Arbeitsunterlage, mit der Sie im Beruf und auch im Hobby erfolgreich arbeiten können. Über 45 Seiten. **DM 5,–**

10

Elektronik und Radio, C. Lorenz
4. Auflage. Völlig neu bearbeitet und stark erweitert.

Eine Einführung in die Radiotechnik, wie man sie nicht alle Tage findet. Eine sehr geschickt gemachte Einführung mit vielen Schaltungen, Bauanleitungen und genauesten Funktionsbeschreibungen. Vom einfachen Diodenempfänger (Detektor) bis zu interessanten Sender- und Empfängerschaltungen. (Minispione) Viele hundert Bilder zeigen Ihnen genau, wie Sie beim Experimentieren vorgehen müssen. IC-Radio, IC-Sender, Antennen, Berechnungsgrundlagen, Tabellen u. v. a. Über 150 Seiten **DM 19,80**

11

IC -Niederfrequenzverstärker , C. Lorenz

Grundlagen der integrierten NF-Verstärker, Berechnung von kompletten IC-NF-Verstärkerstufen. Anwendungsbeispiele mit den interessantesten und gebräuchlichsten Standard IC-NF-Verstärkern wie TBA 800, TBA 830, usw. Printvorlagen, Auswahltabellen, Experimentieranleitungen und Anschlußbilder machen dieses Buch zu einem unentbehrlichen Begleiter für alle, die sich mit NF-Verstärkern beschäftigen wollen. Über 65 Seiten. **DM 9,80**

12

BIS BUCH, Beispiele integrierter Schaltungen, H. Bernstein

Auf über 130 Seiten Anwendungsbeispiele mit integrierten Schaltkreisen. Zeitgeber 555, Funktionsgenerator ICL 8038, Opto Elektronik, Operationsverstärker, Analogschalter, Digital-Analog-Wandler, Analoge Rechenbausteine, Schreib-Lese-Speicher (RAM), Festwertspeicher (ROM), Speicherschaltungen, Uhrenbausteine u. v. a. **DM 19,80**

13

HEH, Hobby Elektronik Handbuch , C. Lorenz

Das Schaltungsbuch für jeden Hobbyelektroniker. Schaltbeispiele und Bauanleitungen aus dem gesamten Hobbybereich. Lichtorgeln, Eiswarngerät fürs Auto, Alarmanlagen, Metallsuchgerät, PLL-Schaltungen, Logik-Tester, Funktionsgeneratoren u. v. a. Über 55 Seiten. **DM 9,80**

14

IC - Vergleichsliste , C. Lorenz

Vergleichsliste für digitale und lineare integrierte Schaltkreise.

Standard TTL, Low Power Schottky TTL, C MOS, Triacs Thyristoren, Optoelektronik, Operationsverstärker, Spannungskomparatoren, Spannungsregler, NF-Verstärker u. v. a. Funktionsvergleichsliste CMOS zu TTL. Vergleichstabelle für Transistoren und Dioden sowie Darlingtontransistoren. Eine Vergleichsliste, die man immer wieder braucht.

DM 29,80

15

Opto -Handbuch, Handbuch für Optoelektronik , C. Lorenz

Das Handbuch für die gesamte Optoelektronik. Eine Einführung und ein ideales Nachschlagwerk. Grundlagen, Definitionen aller Kenngrößen, Opto-Lexikon, Berechnungsgrundlagen, Rechenbeispiele, Schaltbeispiele: Lichtsender, Lichtempfänger, Anzeigen, Infrarot Detektoren, Lichtmeßgerät, Optokoppler, Pegelschalter, Opto-Vergleichsliste. Anschlußbilder wichtiger 7-Segment-Anzeigen u. v. a. Über 106 Seiten. **DM 19,80**

16

C MOS Einführung, Entwurf, Schaltbeispiele, Teil 1 , H. Bernstein

Vom C MOS Gatterbaustein über Schieberegister und Zähler bis hin zum C MOS Schreib-Lesespeicher. Insgesamt werden neunzehn interessante und bekannte C MOS Schaltkreise beschrieben. Zu jedem Bauelement sind genaue Daten, Schaltbild und Anwendungsbeispiele angegeben. Im großen Applikationsteil finden Sie: C MOS-Kippstufen, Addierwerke und Rechenschaltungen, Digital Analog Wandler, Schieberegister für analoge Spannungen, Multiplexsysteme für analoge Signale u. v. a. Eine komplette Einführung und gut geeignet für das Selbststudium der C MOS Technik. Über 140 Seiten. **DM 19,80**

17

C MOS Entwurf und Schaltbeispiele, Teil 2 , H. Bernstein

Fortsetzung von Teil 1. Anwendungsbeispiele mit genauen Schaltungsbeschreibungen und Bauelementeunterlagen. Daten, Anschlußbelegungen weiterer wichtiger hochintegrierter C MOS Elemente. Ein komplettes Arbeits- und Experimentierbuch. C-MOS Uhrenschaltungen, Schieberegisterschaltungen, Parallel-Serien Umsetzung, statische und dynamische Speicherschaltungen, Zählschaltungen, Digital Analog Wandler, Analog Digital Wandler, Digital Voltmeter, I/O Registerschaltungen, Codier und Dekodierschaltungen. RAM und ROM Anwendungen. Über 140 Seiten. **DM 19,80**

18

C MOS Entwurf und Schaltbeispiele, Teil 3, H. Bernstein

Fortsetzung von Teil 2. Eine sehr umfangreiche Applikationssammlung mit hochintegrierten C MOS Elementen. Rechnerschaltungen, Speicher- und Steuerschaltungen, Multiplex- und Datenbussysteme, Uhrenschaltungen, PLL-Schaltungen, Liquid Cristal Anzeigen und deren Treiberschaltungen, Optoelektronik in Verbindung mit C MOS. Grundlagen, Aufbau und Wirkungsweise der Prozeßrechentechnik, Arithmetische Logische Einheiten (ALU) und andere wichtige Funktionen aus der Prozeßrechentechnik. RAMs, ROMs und FIFO-Speicherschaltungen. Über 140 Seiten. **DM 19,80**

19

IC Experimentier Handbuch - IC -EX, C. Lorenz C. Lorenz

Eine sehr umfangreiche Schaltungssammlung und Bauanleitunssammlung mit neuesten integrierten Bausteinen. Neue, jedoch beim Fachhandel erhältliche Standard ICs. Rechnerschaltungen, Mikroprozessoren, I/O Schaltungen, druckende und anzeigende Rechner, Stoppuhren, Zählerschaltungen, Digitalvoltmeter, professioneller Synthisizer, Hilfsschaltungen für den Elektronik Experimenter, Analog Digital Wandler, Frequenzzähler u. v. a. hochinteressante Bauanleitungen. Viele Schaltungen können auf der IC KIT Experimentierplatine WH—1g aufgebaut werden. Über 120 Seiten. **DM 19,80**

20

Operationsverstärker, Grundlagen und Schaltbeispiele, C. Lorenz

Dieses Buch umfaßt das gesamte Gebiet der linearen Schaltungstechnik und stellt ein in dieser Preislage bisher noch nie dagewesen Nachschlagwerk und Einführungshandbuch dar. Bestens geeignet für das Selbststudium. Nach einer pädagogisch geschickt gemachten Einführung folgen theoretische Arbeitsunterlagen und die zugehörigen Schaltbeispiele mit Daten und Gehäuseanschlüssen. Dieses wertvolle Buch dürfte seinen Platz auch bei Ihren Arbeitsunterlagen finden, und wird dann immer von Nutzen sein, wenn es um die Lösung von nicht routinemäßigen Aufgaben geht. **DM 19,80**

21

Digitaltechnik Grundkurs (TTL -C MOS - MOS und Software), C. Lorenz

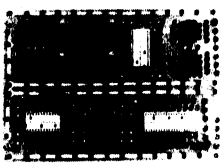
Ein Einführungskurs in die Digitaltechnik für Anfänger und Fortgeschrittene. Ein Fachbuch für den programmierten Selbstunterricht. Der ideale Kurzlehrgang für das Selbststudium. Der Kurs vermittelt Ihnen alle wichtigen Grundkenntnisse vom TTL-Gatter bis zum Mikroprozessor und Lösung von Schaltungsaufgaben durch Software. Viele Versuchsaufbauten und Experimente aus diesem Kurs können auf der IC-KIT Platine WH—1g durchgeführt werden. Grundlagen, Gatter, Zähler, programmierbare Zähler, IC-Tester, Schieberegister, Speicher, Mikroprozessoren u. v. a. Über 130 Seiten. **DM 19,80**

41

Experimentierplatine WH-1g dazu, Best.Nr. 41 DM 79,–

Experimentierplatine WH - 1 g
Abmessungen: 210 mm x 150 mm

Experimentierplatine WH - 1 g
fertig aufgebaut und mit Sockeln bestückt.



22

Mikroprozessoren, Eigenschaften und Aufbau Teil 1 , H. Bernstein

Grundlagen, Eigenschaften und Aufbau von Mikroprozessoren. Organisation von Recheneinheiten und Mikroprogrammen. Programmierung und Klassifizierung von Mikroprozessoren. Ablaufdiagramm, Flußdiagramm. Ein Chip-Technik und Multi Chip-Technik, Transfer- und Sprungfunktionen. Speichertechnik: RAMs, ROMs, FIFO, FILO. Programmierbare logische Arrays (PLA)
Anwendungsbeispiele und Anwendungsbereiche. Über 120 Seiten. **DM 19,80**

23

Elektronik Grundkurs (Kurzlehrgang Elektronik) , C. Lorenz

Eine leichtverständliche und pädagogisch geschickt gemachte Einführung in die Technik der elektronischen Schaltungen. Ein Kurzlehrgang und Schnellkurs zugleich. Aber auch ein recht brauchbares Nachschlagewerk für den fortgeschrittenen Elektroniker. Mit wenig Mühe können Sie sich hier die Grundkenntnisse der elektronischen Schaltungspraxis aneignen. Das Buch schafft die Voraussetzungen für ein erfolgreiches und sicheres Arbeiten mit interessanten Schaltkreisen modernster Technologien. Unentbehrlich für das Experimentieren mit den heutigen modernen hochintegrierten Schaltkreisen. Über 150 Seiten.
DM 9,80

24

Mikrocomputer Technik, Hans Peter Blomeyer-Bartenstein

In diesem Buche finden Sie eine umfassende, einführende und weiterführende Hilfe zum Einstieg in die Mikrocomputertechnik mit vielen Schalt- und Programmierbeispielen. Als praktische Betrachtungsgrundlage dient das supermoderne Microcomputerkonzept Z80A von ZILOG. Das Buch geht auf alle wichtigen Zusammenhänge ein und erklärt diese dem Leser so ausführlich, daß kaum noch Fragen offen bleiben. Über 240 Seiten **DM 29,80**

25

Hobby Computer Handbuch, C. Lorenz

Eine leicht verständliche Einführung in die Microcomputertechnik

Diese sehr umfangreiche Einführung in die Microcomputertechnik dürfte zu diesem Preise einmalig sein. Auf über 450 Seiten finden Sie: Grundlagen der Computer- und Microcomputer-Technik, Was ist ein Microcomputer?, Microcomputer KITs, Einplatinencomputer, KIM, MIKIT, Z80 KIT, 6800 KIT, NEC 8080 KIT u. v. a. Genaue Beschreibungen der wichtigsten Mikroprozessortypen. Zusammenstellung und Beschreibung der modernen Personal Computer. (IMSAI, CROMEMCO, CAT, OSI, POLY 88 u. v. a.) Kompaktcomputer wie PET und TR-8. Interfactechniken, Ein/Ausgabegeräte, ROMs, RAMs, Programmiergeräte für PROMs. Löschgeräte für EPROMs u. v. a. mehr. Das ideale allumfassende Buch für den Microcomputertechniker. Für Industrieanwendung ebenso geeignet, wie für den Hobby-Computer Fan. Über 450 Seiten **DM 29,80**

26

Mikroprozessor Teil 2, H. Bernstein

Die Fortsetzung unseres ersten so erfolgreichen Buches über Mikroprozessoren. Technologie von Mikroprozessor- und Speicherbausteinen. Festwertspeicher, PROM, REPROM, FIFO, Schieberegister, MPR-Register, ARL-Register, SAR-Register. Aufbau eines Mikroprozessorsystems mit 8080, RAM- und ROM Schnittstellen. Befehlssatz 8080. Über 120 Seiten. **DM 19,80**

27

Mikroprozessor Software Handbuch MSH, C. Lorenz

Grundlagen und Einführung in die Mikroprogrammierung. Grundlagen und Einführung in die wichtigsten Programmiersprachen (BASIC, FORTRAN, ASSEMBLER-Sprachen) Zusammenstellung der wichtigsten Befehlslisten: 8080, Z80, M 6800, National, Fairchild, etc.
Ein Software Handbuch für jeden der mit Mikroprozessoren oder Mikrocomputern zu tun hat. Über 200 Seiten. **Preis DM 29,80**

28

Lexikon und Wörterbuch für Elektronik und Mikroprozessortechnik, LEM, C. Lorenz

Ein Hilfs- und Arbeitsbuch für jeden der sich heute mit der modernsten Elektronik beschäftigt. Viele engl. Ausdrücke werden heute in der Elektronik, Computer- und Mikroprozessortechnik verwendet und oft fehlt uns eine genaueste und präzise Erklärung. Dieses Buch übersetzt Ihnen den englischen Fachausdruck und gibt Ihnen zusätzlich noch eine deutsche Erläuterung und Erklärung dieses Begriffes und was es damit auf sich hat. Ein Lexikon und Wörterbuch in einem einzigen Buch vereinigt. Das Buch, das Sie schon lange gesucht haben. Ca. 250 Seiten. Lieferbar Herbst 1978 **Preis DM 29,80**

29

Mikrocomputer Datenbuch, C. Lorenz

Zusammenstellung der wichtigsten Mikroprozessordaten.

Eine übersichtliche und sehr informative Zusammenstellung der wichtigsten Mikroprozessorbausteine auf dem Markt. 8080A, 8085, 8048, Z80, Z8, 6500, 6800, 2650, 1802, F8, 3870, SC/MP, PACE u. v. a. Daten, Anschlußbilder, wichtige technische und elektrische Daten, Architektur, grundlegende Eigenschaften. Zu jedem Mikroprozessor werden dann auch noch die peripheren Bausteine sowie RAM und ROM Elemente behandelt. Das ideale Handbuch für jeden modernen Elektroniker. Lieferbar Ende 78 **DM 49,–**

30

Aktivtraining - Mikrocomputer (Sammelordner)

Ein Mikrocomputerkurs in Form einer Lehrbriefsammlung.

Das Aktivtraining-Seminar erscheint in einzelnen Lehrbriefen, die in einem formschönen praktischen Plastik-Sammelordner geliefert werden. Die Grundausstattung besteht aus 10 Lehrbriefen und reicht von den elementaren Grundlagen bis hin zu praktischen Beispielen. Weiterhin ist ausführliches Mikrocomputer Datenmaterial im Ordner enthalten. Die Lehrbriefe werden regelmäßig erscheinen und können jederzeit nachbezogen werden. Lieferbar Ende 78 **DM 49,–**

31

57 Programme in BASIC, C. Lorenz

Ein Buch mit technisch-wissenschaftlichen Programmen und einer großen Anzahl von Spielprogrammen in BASIC. (Games) Ein Buch für jeden, der sich mit dem faszinierenden Hobby der Mikrocomputertechnik befassen will. Alle Listings sind in BASIC und können auf den meisten Personal Computer Systemen gefahren werden. Lieferbar Ende 78 **DM 39,–**

32

Circuits Digital Et Pratique, C. Lorenz

Dieses Buch ist eine französische Übersetzung unserer beiden Bücher Nr. 6 und Nr. 13 (IC-Schaltungen und Hobby Elektronik Handbuch HEH)
Ca. 100 Seiten

DM 19,80
FF 39,90

33

Mikrocomputer Programmierbeispiele für 2650, Dr. J. Hatzenbichler

Eine Einführung in die Programmierung von Mikrocomputern an Hand des Prozessors 2650 von Signetics. Viele Programmierbeispiele in Maschinensprache, die Sie auf einem preiswerten Mikroprozessorsystem MIKIT 2650-P2 ausführen können. Zeitschleifenprogramme, Blinkschaltung, Lauflicht, Stufenzähler, Elektronischer Würfel, Stoppuhr, Reaktionszeittester, Computer Musik Programm u. v. a. Jeder Befehl wird genau erläutert und an Hand eines ausführlichen Flußdiagrammes erklärt. Jedes Programm liegt in Form eines Computer-Listings vor. Jedes Programm ist genauestens beschrieben. Sie können so auf einfache Weise die Zusammenhänge erkennen und erwerben damit die Grundkenntnisse zur Erstellung Ihrer eigenen Programme. Sie erkennen, wie man mit nur einer Schaltung (Mikrocomputer) unendlich viele praktische Anwendungsschaltungen realisieren kann. Zu diesem Buche ist auch ein komplett aufgebautes und getestetes Mikrocomputersystem erhältlich, auf dem Sie alle beschriebenen Programme selbst ausführen können. Über 120 Seiten

DM 19,80

105
1

TTL - Experimentierbuch

Eine kleine Einführung in die Digitaltechnik

Grundlagen der Digitaltechnik kurz erklärt. Bauanleitung für einen praktischen Logik-Tester. Viele Experimente und Anwendungsschaltungen mit dem TTL Gatterbaustein 7400. Das ideale Einführungsbuch in die Digitaltechnik.

DM 5,–

106
1

CMOS - Experimentierbuch

Eine kleine Einführung in die CMOS Schaltungstechnik

CMOS Grundlagen, Behandlungshinweise für CMOS Bausteine, Zusammenstellung der wichtigsten CMOS Bausteine und deren Anschlußbilder. Viele praktische CMOS Schaltbeispiele. Ideal für jeden Elektroniker.

DM 5,–

IC KIT Experimentier-platine

Mit der IC-KIT Experimentierplatine wird jetzt das Aufbauen von Schaltungen mit ICs, Transistoren, Widerständen und Kondensatoren zum Kinderspiel. Einfach, schnell und ohne Mühe lassen sich mit dieser hochwertigen Epoxy Versuchsplatine selbst die komplizierten Versuchsschaltungen aufbauen.
Die IC KIT Experimentierplatine Typ WH-1 g eignet sich zum Aufbau von Uhren-, Rechner- und Mikroprozessorschaltungen besonders gut, da sie neben 14,16,24 und 28 poligen DIL-Sockeln auch einen 40 poligen Stecksockel enthält.

Der Einsatz von Adaptern ermöglicht auch das Stecken von diskreten Bauelementen. Weiterhin können Sie sich mit diesen Adaptern selbst kleine Funktionsgruppen aufbauen, die Sie immer wieder verwenden können, zum Beispiel Taktgeneratoren, Logiktester, Tastenentprellschaltungen etc. Der Schaltungsaufbau wird durch das Stecken aller Bauteile und Verbindungsleitungen wesentlich vereinfacht. Die einzelnen Elemente werden nicht beschädigt und bleiben frei von Lötzinn. Sie können immer wieder verwendet werden. Sie sparen Zeit und Geld!

41

Der Bausatz IC KIT WH-1 g enthält: 1 Experimentierplatine in stabiler Epoxy-Ausführung mit den Abmessungen 210 x 150 mm fertig gebohrt und Leiterbahnen verzinnt. Die Bestückungsseite ist mit einem kratzfesten Lack überzogen und bedruckt. Sie können so die gesuchten Anschlüsse beim Experimentieren immer leicht und schnell finden. 340 Messingbuchsen, verschiedenfarbige Stecker mit Querloch (2,6 mm ϕ), Metallfüße und sämtliche Sockel.

Bestell Nr. 41 — Preis DM 79.–

41/1

Adapter zum Stecken von diskreten Bauelementen, 5 Stück in einem Beutel verpackt. Die diskreten Bauteile werden hier einmal angelötet und können dann immer wieder verwendet werden. Einfachste Handhabung.

Bestell Nr. 41/1 — Preis DM 14,90

48

5V Netzgerät stabilisiert für die IC KIT Experimentierplatine Typ WH-1 g. Bausatz komplett mit Netztransformator, Platine, allen Bauelementen, Netzkabel etc. Maximale Welligkeit 1 %, TK = 0,5 mV pro °C, Strombegrenzung. Maximaler Ausgangsstrom 650 mA. Ideal für alle TTL und CMOS Experimente.
Bestell Nr. 48 — Preis DM 39.–

MICROPROCESSOR / MICROPROGRAMMING HANDBOOK, Brice Ward

Ein praktisches Handbuch für jeden, der sich mit Microprocessoren beschäftigen möchte. Auf über 290 Seiten finden Sie Grundlagen, alles über Programmierung und Anwendungsbeispiele über die wohl interessantesten Integrierten Schaltungen unserer Zeit.
Das Buch sagt Ihnen auf einfache Weise, was ein Microprocessor ist, wie sie arbeiten, wie man sie anwendet, wo man sie anwendet und wie Sie diese für Ihre eigenen Problemlösungen einsetzen können.
Weiterhin finden Sie dazu wie man Microprogramme schreibt, wie man sämtliche anfallenden Arbeiten in und um den Microprocessor anfaßt. Ausführliche Beschreibung der Grundlagen der Microprogrammierung sowie deren grundsätzlichen Techniken (Programmschleifen, Subroutinen, Interrupt usw.)
Inhalt: Einführung in die Microprocessortechnik, grundlegende Computer-Funktion. Der Microprocessor von innen. Das moderne Microprocessorsystem. MCS4, MCS40, MCS80, Speichersysteme, Einführung in die Microprogrammierung, Microprogrammierung in Maschinensprache, Microprogrammierung in Assembler-Sprache.
294 Seiten, 170 Bilder in englischer Sprache, Best.Nr. 785 DM 35,--

MODERN GUIDE TO DIGITALLOGIC, United Technical Publications (Processors, Memories + Interfaces)
In diesem Buche wurde zum ersten Male das Problem ausführlich behandelt, wie man schnelle Logik (ECL) mit langsamer Logik kombiniert. Weiterhin finden Sie alles über schnelle Speichersysteme. Neben vielen Beschreibungen über moderne Interfacetechniken finden Sie alles was Sie zum Arbeiten mit Schottky TTL, High Speed ECL, C MOS, störsicherer Logik und vielen Microprocessoren benötigen.
Bipolare und MOS-Speicherschaltungen werden anhand von Beispielen ausführlich besprochen.
294 Seiten, 222 Bilder in englischer Sprache, Best.Nr. 709 DM 35,--

BEGINNER's GUIDE TO COMPUTER PROGRAMMING, Brice Ward

Eine ideale Einführung in die Programmierung von Computern (Microprocessoren). Das Buch beginnt mit der Entwicklung einer einfachen Programmiersprache für den eigenen Bereich und zum Selbststudium. Dann geht der Autor auf die anderen Sprachen über.
Inhalt: Grundlegende Programmkonzepte, I/O-Schaltungen, Flußdiagramme, Programmtest, Schleifen, Indexregister, Programmiersprachen verschiedener Ebenen, Compiler, Cobol u. v. a.
480 Seiten, 364 Bilder in englischer Sprache, Best.Nr. 574 DM 39,--

COMPUTER PROGRAMMING HANDBOOK, Peter Stark

Eine vollständige Einführung in die Programmierung und Daten-Verarbeitung, mit komplett ausgeführten Beispielen. Ein extrem zusammengefaßtes Werk über die gesamte Programmierung und EDV. Es sind keine Kenntnisse aus der höheren Mathematik zum Verständnis notwendig.
Das Buch beinhaltet alle drei Arten der Computersprachen - Maschinensprachen - symbolische Sprachen, problemorientierte Sprachen (wie FORTRAN IV im besonderen).

Jede Sprache wird bis ins Detail analysiert und mit anschaulichen Beispielen verständlich gemacht.
Dieses Buch kann man als den "Ein-Buch-Computerkurs" bezeichnen und sollte in keinem Bücherschrank fehlen.
518 Seiten, 114 Bilder in englischer Sprache, Best.Nr. 752 DM 45,--

PROGRAMMING MICROPROCESSORS, M. W. McMurran

Dieses brandneue Buch reicht von der grundlegenden Microprocessororganisation über Zahlensystem, Flußdiagramme, Adressierung, Assemblierung über Subroutinenbeschreibungen, Programmierhilfen, Datenaustausch, Compilern bis hin zu speziellen Programmiertechniken.
Ein modernes, ausführliches Werk über Microprocessoren mit Schwergewicht auf der Software.
270 Seiten, in englischer Sprache, Best.Nr. 985 DM 35,--

MICROPROCESSOR PROGRAMMING FOR THE COMPUTER HOBBYIST

Dieses Buch ist speziell für den Computer Hobbyisten geschrieben, der sich bereits mit weiterführenden Programmiertechniken und Datenstrukturen beschäftigen möchte. Es wurde so verfaßt, daß es mit seinem Inhalt dort beginnt, wo die meisten Handbücher der Microprocessor-Hersteller aufhören. Nach einer kurzen Einführung wird sofort mit höheren Programmiersprachen begonne. (PL/M, PL/1) Großen Wert legte man auf die ausführliche Behandlung der arithmetischen Funktionen, da die meisten Processoren nur auf 8 Bit oder 16 Bit Basis arbeiten. Suchprogramme werden im Zusammenhang mit dem Schachspielproblem behandelt.
Über 380 Seiten, in englischer Sprache, Best.Nr. 952 DM 39,--

57 PRACTICAL PROGRAMS + GAMES IN BASIC, Ken Tracton

Programmbeispiele aus allen Bereichen. Vom Krieg der Sterne bis zu Black Jack und von hyperbolischen Funktionen bis zur linearen Interpolation. Ein Buch mit Programmierbeispielen für den Hobby-Computer-Besitzer oder jeden, der Zugriff zu einem Computersystem mit Standard BASIC hat. Die meisten Programme sind so geschrieben, daß sie sogar auf vereinfachten BASIC-Versionen gefahren werden können. Die Programme wurden nicht nur nach dem Gesichtspunkt der praktischen Verwendbarkeit ausgesucht, sondern bilden auch bei der Ausbildung eine wertvolle Lehr- und Lernhilfe. Aus dem Inhalt: Space Wars, One Arm Bandit, Black Jack, Number Guess Game, Mathematikprogramme u. v. a.
Über 200 Seiten, in englischer Sprache, Best.Nr. 1000 DM 35,--

MASTER HANDBOOK OF 1001 ELECTRONIC CIRCUITS, K. W. Sessions

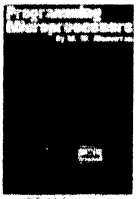
In diesem umfangreichen Buch finden Sie Schaltbeispiele mit integrierten Schaltkreisen und Transistoren aus praktisch allen Gebieten der Elektronik.
Den einzelnen Schaltungen sind alle wichtigen Daten beigefügt, die Sie benötigen, um die Schaltung zum Funktionieren zu bringen.
Das ideale Nachschlagewerk für alle aktiven Elektroniker, Ingenieure, Hobbyelektroniker, Amateure und für jeden, der in einer unvorhergesehenen Situation eine ganz bestimmte Schaltung sucht. Neben einem 22-seitigen Inhaltsverzeichnis sind die Schaltbeispiele nach klassischen Anwendungen geordnet. Die einzelnen Kapitel sind alphabetisch geordnet. Dies ermöglicht Ihnen ein schnelles Auffinden der gesuchten Schaltung. Am Schluß finden Sie noch eine IC-Vergleichsliste mit den wichtigsten Daten und Anschlußbelegungen. Dieses Buch sollte jeder Elektroniker besitzen.
602 Seiten, über 1.250 Bilder, in englischer Sprache, Best.Nr. 800 DM 49,--

SWITCHING REGULATORS + POWER SUPPLIES, Irving Gottlieb

Das Buch beschreibt ausführlich die Funktionsweise der modernen, getakteten Netzgeräte. Theorie, Berechnungsgrundlagen, ausführliche Besprechung der erforderlichen Bauelemente, Schaltbeispiele und genaue Anleitungen zum Entwurf eigener Schaltregler.
(DC/PC, AC/DC, DC/AC und AC/AC)
252 Seiten, 128 Bilder, in englischer Sprache, Best.Nr. 828 DM 35,--

Es gelten unsere Lieferungs- und Zahlungsbedingungen

Neue Bücher über Mikro-prozessoren



Programming Microprocessors

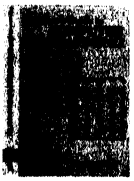
Zahlensysteme, Flußdiagramme, Adressierungsarten, Assemblersprachen, Programmierhilfen u. v. a. Das ideale Microprozessorbuch mit Schwerpunkt Software. Über 270 Seiten.

Best. Nr. 985 DM 35,–

Microprocessor/Microprogramming Handbook.

Das praktische Handbuch für jeden der sich mit Mikroproz. beschäftigen will. Über 290 Seiten.

Best. Nr. 785 DM 35,–



Beginners Guide to Computer-Programming.

Die ideale Einführung in die Computerprogrammierung. Gut geeignet für das Selbststudium. Über 480 Seiten.

Best. Nr. 574 DM 39,–

Computer Programming Handbook

Eine genaue Einführung in die Programmierung und EDV mit ausgeführten Beispielen. Der ideale "Ein Buch Computerkurs". Über 510 Seiten.

Best. Nr. 752 DM 45,–

Modern Guide to Digital Logic Processors, Memories, Interfaces.

Speicherschaltungen, Interfaceschaltungen, ECL, Schottky TTL u. v. a. mehr. Über 290 Seiten.

Best. Nr. 709 DM 35,–

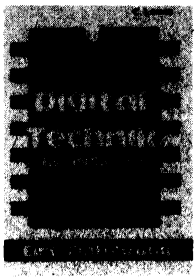
Master Handbook of 1001 practical Electronic Circuits.

Ein Schaltungsbuch mit über 1000 Schaltbeispielen mit kurzen Beschreibungen. Über 600 Seiten.

Best. Nr. 800 DM 49,–

Best. Nr.	Stck.	BUCHTITEL BEZEICHNUNG	a DM	ges. DM
1		TBB – Handbuch Band 1	19.80	
2		TBB – Handbuch Band 2	19.80	
3		Elektronik im Auto	9.80	
4		IC – Handbuch, TTL, CMOS, Lin.	19.80	
5		IC – Datenbuch, TTL, CMOS, Lin.	9.80	
6		IC – Schaltungen, TTL, CMOS	9.80	
7		Elektronik Schaltungen	5.–	
8		IC – Bauanleitungshandbuch	19.80	
9		Feldeffekttransistoren	5.–	
10		Elektronik und Radio 4. Auflage	19.80	
11		IC – NF – Verstärker	9.80	
12		Beispiele integrierter Schaltungen	19.80	
13		HEH, Hobby Elektronik Handbuch	9.80	
14		IC – Vergleichsliste	29.80	
15		Optoelektronik Handbuch	19.80	
16		CMOS Teil 1, Einführung	19.80	
17		CMOS Teil 2, Schaltbeispiele	19.80	
18		CMOS Teil 3, Schaltbeispiele	19.80	
19		IC – Experimentier Handbuch	19.80	
20		Operationsverstärker	19.80	
21		Digitaltechnik Grundkurs	19.80	
22		Mikroprozessoren 2. Auflage	19.80	
23		Elektronik Grundkurs	9.80	
24		Mikrocomputer Technik	29.80	
25		Hobby Computer Handbuch	29.80	
26		Mikroprozessor Teil 2	19.80	
27		Mikroprozessor Software Handbuch	29.80	
28		Lexikon + Wörterbuch	29.80	
29		Microcomputer Datenbuch	49.80	
30		Aktivtraining Mikrocomputer	49.80	
31		57 Programme in BASIC	39.–	
32		Circuits Digital Et Pratique	19.80	
33		Microcomputer Programmierbeispiele	19.80	
41		IC-KIT Experimentierplatine WH-1 g	79.–	
105/1		TTL-Experimentierbuch	5.–	
106/1		CMOS-Experimentierbuch	5.–	

Kurzlehrgang TTL CMOS



21/41

TTL – Kurzlehrgang
Der ideale Einführungskurs für das Selbststudium, den Unterricht, die Ausbildung und für den Amateur. Das Buch Nr. 21 zusammen mit unserem IC-KIT Experimentierplatinenbausatz WH-1g bilden die ideale Grundlage für diese TTL-Einführung. Kreuzen Sie auf der Bestellkarte einfach die Bestellnummern 21 und 41 an, Sie erhalten dann umgehend diesen wertvollen Kurzlehrgang zugeschickt.
Best. Nr. 21/41 DM 98,80

CMOS – Kurzlehrgang
Der ideale Einführungskurs in die moderne C-MOS Technik. Grundlagen, Entwurf, Versuchsschaltungen. Das Buch Nr. 16 zusammen mit der IC-KIT Experimentierplatine WH-1g bilden die ideale Grundlage für diesen interessanten Kurs. Kreuzen Sie auf der Bestellkarte einfach die Bestellnummern 16 und 41 an. Sie erhalten dann umgehend diesen C MOS-Kurs.
Best. Nr. 16/41 DM 98,80

16/41

Abs.
Name

_____

Straße:

_____

Ort:

_____

Bestellkarte

○ Bitte liefern Sie mir auch die Experimentierplatine IC - KIT WH-1 g für 14,16,24,28 und 40 polige DIL IC-Gehäuse. Bausatz komplett: DM 79.–

○ Bitte liefern Sie von jedem neu erscheinenden Buch ein Muster per Nachnahme. (Rückgaberecht)

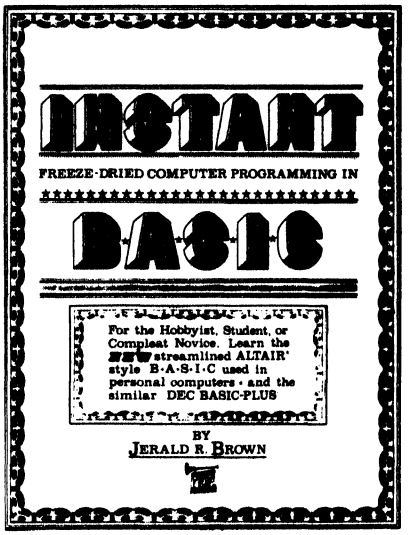
PROGRAMM BIBLIOTHEK

BASIC

Application Software!

WEITERE BÜCHER ÜBER BASIC

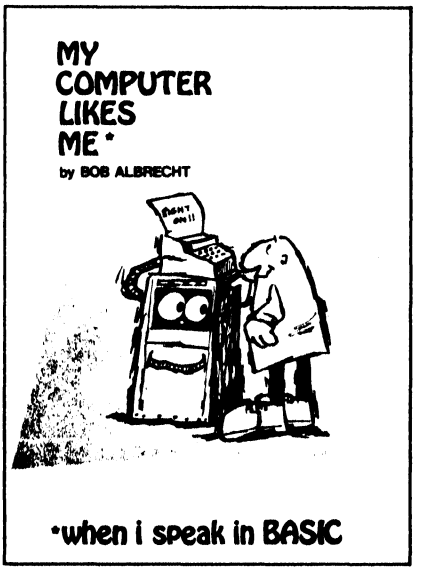
Microprocessor Software Handbuch, C. Lorenz
Neben vielen Tabellen, Daten, Befehlslisten und Grundlagen der Programmierung von Microcomputern finden Sie eine deutschsprachige Einführung in die BASIC-Programmiersprache. Zahlen, Variable, Konstanten, Besprechung der wichtigsten BASIC-Anweisungen mit vielen einfachen Programmierbeispielen. NIBL-BASIC, ein TINY-BASIC für den INS 8060 von National. Eine Einführung mit genauen Erklärungen und vielen Programmbeispielen.
Best.Nr. 27 DM 29,80



Instant BASIC, Jerald R. Brown

Die ideale und vollständige BASIC-Einführung für den Microcomputer-Anwender. Leicht zu verstehen, einfach anzuwenden. Der Autor führt Sie durch seine besonders leichte Schreibweise und dem nicht-mathematischen Context fast spielerisch in die BASIC-Programmiersprache ein. Lustige Bilder machen das Buch zu einem wahrhaften Lernspaß. Das Buch wurde speziell für den Heimcomputer-Anwender gedacht.
Über 160 Seiten (DIN A 4)
Best.Nr. 80/58 DM 24,80

My Computer Likes Me ... When I Speak In BASIC, Bob Albrecht
Eine leicht verständliche und im Konversationsstil geschriebene Einführung in die BASIC-Programmiersprache. Das ideale Lern- und Programmierbuch für jung und alt. Dieses Buch ermöglicht auch dem Leser ohne Vorkenntnisse eine schnelle und leichte Einarbeitung in die BASIC-Programmiersprache.
Best.Nr. 80/56, 64 Seiten DM 9,80



57 Praktische Programme und Spiele in BASIC
Ein Buch mit Programmierbeispielen für den Hobby-Computer-Besitzer oder jeden, der Zugriff zu einem Computersystem mit Standard BASIC hat. Die meisten Programme laufen auf TINY-BASIC. Inhalt: Mathematikprogramme, Blackjack, Space Wars, Number Guess Game u. v. a.
Best.Nr. 1000 DM 35,--

INHALTSVERZEICHNIS DER BASIC PROGRAMMBIBLIOTHEK VOL I - VOL V

VOLUME ONE

Part 1 - Business + Personal Bookkeeping Programs

Name	Description
Bond	Computers price and interest for bond purchases.
Building	Analyzes the cost of building design proposals.
Compound	Computes effective compound interest rates.
Cyclic	Determines seasonal coefficients for two cycles.
Decision 1	Makes a Lease/Buy decision for you.
Decision 2	Makes a decision on whether to buy a component or make it.
Depreciation	Calculates depreciation by 4 different methods.
Efficient	Cal. the most efficient assignment of resources and / or personnel.
Flow	Predicts your yearly cash flow.
Installment	Performs monthly installment accounting.
Interest	Computes interest accruals, monthly.
Investments	Computes annual rates of return on investments.
Mortgage	Makes a comparison of mortgage terms.
Optimize	Optimizes the layout for a plant, shop, office, etc.
Order	Determines your economic order quantity for inventory items.
Pert Tree	Performs an analysis of a pert network.
Rate	Computes true annual interest rates.
Return 1	Computes lessor's rate of return for uncertain assets.
Return 2	Computes a lessor's rate of return after taxes.
Schedule 1	Schedules N jobs in a shop with M machines.

Part 2 - Games + Pictures

Name	Description
Animals Four	Teach the computer all about animals.
Astronaut	Land your spaceship on another planet.
Bagel	Advanced number game, numbers may be algebraic, few clues.
Bio Cycle	Calculate your Bio-Life Cycle and plan your days.
Cannons	An advanced war game with big guns.
Checkers	Plays a regulation game of checkers.
Craps	A dice game with hard way odds.
Dogfight	Air fight w/missiles; between a phantom and a mig.
Golf	Plays any number of holes; inc. obstacle course.
Judy	Have a rap session with Judy via your computer.
Line Up	Simple number game, all you have to do is unscramble them.
Pony	Authentic horse race, any number of players.
Roulette	Gamblers delight, plays Las Vegas rules.
Sky Diver	Sky dive on another planet
Tank	A war game between two tanks.
Teach Me	Teach the computer to learn new things.
A Newman	He's absolutely MAD! MAD! MAD!
J. F. K.	Our 35th. president.
Linus	Loveable "Peanuts" character, w/blanket.
Ms. Santa	A modern miss to put a twinkle in your eye.
Nixon	Former "United States" president.
Noel Noel	Christmas or anytime this is a beautiful creation.
Nude	A true work of art for anyone's gallery.
Peace	A message for all seasons.

Name	Description
Policeman	True and blue, he's the law.
Santa's Sleigh	In banner form, perfect for decorating the mantle.
Snoopy	That paragon of Dogdom even plays football.
Virgin	A picture you can read as well as see.

VOLUME TWO

Part 3 - Math + Engineering Programs

Name	Description
Beam	Evaluates and selects steel beam sizes.
Conv.	Calculates convolutions.
Filter	Calculates low pass filter components.
Fit	Performs interpolations by spline fits.
Integration 1	Uses Gaussion Quadrature to do integration.
Integration 2	Integrates a function by spline fits.
Intensity	Calc. and plots RF or Acoustic intensities.
Lola	Calc. Long. and Lat. from interstellar fix or distance.
Macro	Simulates a language compiler.
Max. Min.	Calc. the max. + min. values of funct. over a special interval.
Navaid	Calc. position from altitude and azimuth of celestial bodies.
Optical	Calculates Blackbody energies, w/filter look-up tables.
Planet	Calculates Sun and Moon positions, hourly.
PSD	Calculates Power Spectral Densities and FFT's.
Rand 1	Generates random numbers between 0 and 1.
Rand 2	Generates random integers between (X) and (Y).
Solve	Solves polynominals by "Bairstows Methods".
Sphere Trian	Solves any spherical triangle.
Stars	Locates 50 stars (celestial).
Track	Calc. course and distance and incremental vectors.
Triangle	Solves for all parts of any triangle.
Variable	Finds all variables in BASIC programs.
Vector	Calc. final position; given start and motion vectors

Part 4 - Plotting + Statistics Programs

Binomial	Calculates binomial probability distributions.
Chi-Sq.	Applies the Chi-Square test to samples.
Coeff	Calc. coefficients of fourier series to apprx. a function.
Confidence 1	Calculates confidence limits on linear regressions.
Confidence 2	Calculates confidence limits for a sample mean.
Correlations	Performs auto and cross correlations with plots.
Curve	Fits 6 different curves by the least squares method.
Differences	Calculates difference of means in non-equal variances.
Dual Plot	Plots two functions on the same sheet.
Exp-Distri	Calculates exponential distributions for a sample.
Least Squares	Performs least squares fit by linear, exp., or power function.
Paired	Compares 2 groups of data using the rank test.
Plot	Plots 6 equations on the same sheet.
Plotpts	Plots data points on standard teletypes.
Polynomial Fit	Performs least squares polynomial fit.
Regression	Performs multiple linear fit with or without transformations.
Stat 1	Finds the mean, variance and standard deviations.

Name	Description
Stat 2	Computes various stat. measures for a variable.
T-Distribution	Calculates normal and T-distributions.
Unpaired	Compares 2 groups of unpaired data.
Variance 1	Performs one way analysis of variances.
Variance 2	Analyzes a variance table of one way random design.
XY	Plots functions of X and Y.

APPENDIX A - BASIC STATEMENT DEFINITIONS

VOLUME THREE

Part 5 - Advanced Business Programs

Billing	Performs posting and billing of accounts.
Inventory	Maintains data for inventory records.
Payroll	Computes payrolls with full set of deductions.
Risk	Performs a risk analysis on capital investments.
Schedule 2	Performs the most effi. scheduling of men or resources to loca.
Shipping	Solves the problem of scheduling and assignments.
Stocks	Computes the value of stocks.
Switch	Calculates the effects of a bond switch.

VOLUME FOUR

General Purpose Programs

Bingo	An age old favorite. "B9, C23, D4, E13, F21, BINGO!
Bonds	Computes the yields for a bond for different periods.
Bull	If you ever dreamed of being a Matador, here's your chance.
Enterprise	Take charge of the Enterprise while Capt. Kirk is on leave.
Football	Authentic NFL version of this well known sport.
Funds 1	Calculates long-term predictions of funds.
Funds 2	Plots the results of Funds 1.
Go-Moku	Ancient Chinese game of chance.
Jack	Plays Blackjack, Las Vegas style.
Life	Life is truly a battle for survival, a real challenger!
Loans	Calculates annuities, loans and mortgages.
Mazes	Generates unique maze puzzles for you to solve.
Poker	Five card draw - for up to 5 players.
Popul	Performs population projections for defined areas.
Profits	Determines the profitability of a firms various depts.
Qublic	3-Dimensional Tic-Tac-Toe.
Rates	Calc. the effective annual interest rate for stated interest.
Retire	Calculates your Civil Service Retirement benefits.
Savings	Computes savings plan profiles.
SBA	Calculates repayment schedules for SBA loans.
Tic-Tac-Toe	An all time favorite for young and old alike.

VOLUME FIVE

Experimenter's Programs

Andy Cap	Draws this famous cartoon character.
Baseball	Plays a full 9 innings of baseball.
Compare	Compares two groups of data.
Confid 10	Determines the confidence limits for a normal population.
Descrip	Provides a description of uni-variant data.
Differ	Computes the diff. of the means for data of equal variance.
Engine	Calculates the otto cycle of engines.
Fourier	This program evaluates fourier series.
Horse	Draws a picture of a horse.
Integers	Computes integers as the sum of other integers.

Name	Description
Logic	Determines conclusions from logic statements.
Playboy	Draws the playboy symbol.
Primes	Factors numbers into their primes.
Probal	Calc. Chi-Sq. and probabilities from 2X2 data sets.
Quadrac	Solves quadratic equations
Red Baron	Draws a picture of the infamous Red Baron.
Regression 2	Calculates linear regressions.
Road Runner	"Beep! Beep!" Draws a picture of the Road Runner.
Roulette	Computerized "Wheel of Fortune", plays roulette.
Santa	Old Saint Nick appears as jolly as ever.
Stat 10	Calculates quantities for two groups of paired data.
Stat 11	Computes sample statistics.
Steel	Calculates steel beam capacities.
Top	Computes cost for surfacing a road or driveway, etc.
Vary	Performs an analysis of a vari. table; one-way random design.
Xmas	Generates a "SINGING" Christmas card.

APPENDIX B - STATEMENT CONVERSION ALGORITHMS

VOLUME SIX

A Complete Business System

Ledger	Maintains ALL Company accounts and generates ALL financial reports. Includes routines for: Pyrl, Inv. Depr. A/R, A/P, Balance Sheets and Profit + Loss statements, etc.
ACBS rev: 80	Users Manual - A Proprietary Package

VOLUME SEVEN

Professional Programs

Chess	Designed to challenge the average player, fairly comprehensive. Great fun for all, offers a unique opportunity for beginners in need of an opponent.
Medbil	For Doctors and Dentists alike, a complete patient billing system which also permits the maintaining of a patient history record.
Wdproc	Wordprocessing for lawyers, publishers, writers etc. Write, store and change from rough draft to final copy in a variety of formats.

VOLUME SEVEN

Professional Programs

Utility

Licensing Agreement

BESTELLSCHEIN

Heute noch an Ihren Buch- oder Fachhändler absenden! Bitte senden Sie mir folgende Bücher:

..... Stck. Vol. I , Best.Nr. 80/50, DM 99,--
..... Stck. Vol. II , Best.Nr. 80/51, DM 99,--
..... Stck. Vol. III, Best.Nr. 80/52, DM 149,--
..... Stck. Vol. IV, Best.Nr. 80/53, DM 39,--
..... Stck. Vol. V , Best.Nr. 80/54, DM 39,--
..... Stck. Vol. VI, Best.Nr. 80/48, DM 199,--
..... Stck. Vol. VII, Best.Nr. 80/49, DM 159,--
..... Stck. Programm Bibliothek BASIC, (Volume I bis Vol. V) Best.Nr. 80/21, DM 425,--

Hofacker Verlag **Postfach 437**
8000 München 75

GRUNDAUSSTATTUNG

Die Grundausstattung der BASIC Programmbibliothek besteht aus fünf Büchern mit insgesamt 1100 Seiten Programmlistings und genauen Beschreibungen. (Format DIN A 4)
Sie finden in diesem Werk 149 verschiedene BASIC Programme aus folgenden Bereichen:

Buchhaltung
Computerspiele
Bildprogramme
Mathematik und Ingenieurwissenschaften
Statistik, technische Mechanik etc.
Commerzielle Programme, Rechnungswesen, Fakturierung
Inventurprogramme, Lohn-und Gehaltsbuchhaltung
Experimentierprogramme u.v.a.

Jedes Programm ist genau beschrieben.
Die fünf Bände bilden eine komplette "Do It Yourself" Programmbibliothek, die sehr einfach zu handhaben ist. Es werden jedoch Grundkenntnisse in der Programmierung vorausgesetzt.

PREISSENSATION

Bei 149 Programmen in dieser umfangreichen Bibliothek bezahlen Sie nur
DM 2.85 pro BASIC Programm - Das ist einmalig!

BESCHREIBUNG DER PROGRAMMBIBLIOTHEK

Diese Bibliothek ist die umfangreichste und universalste ihrer Art auf dem Weltmarkt. Die Besonderheit dieser fünf Werke liegt darin, daß dem Anwender die Möglichkeit gegeben wird, in erster Linie sinnvolle und produktive Aufgaben wie Buchhaltung, Statistik, mathematische Probleme etc., zu lösen. Computerspiele sind jedoch auch enthalten.

Alle Programme wurden mehrmals auf verschiedenen Systemen ausgeführt und getestet. Jedes Programm ist mit einer genauen Beschreibung und einer Aufstellung der infrage kommenden Anwender versehen. Befehle und mögliche Grenzen der Programme sind auch aufgezeigt, wenn die Programme auf unterschiedlichen Systemen ausgeführt werden sollen. Auch der benötigte Speicherbereich ist für jedes Programm genau angegeben.
Jedes Programm ist in seiner vollständigen Größe aufgeführt und nicht reduziert. Es ist in sich selbst vollständig und kann im Ablauf genau verfolgt werden. Es gibt Aufschluß über alle wichtigen Daten. Bei den meisten Programmen folgt unmittelbar nach dem Programmlisting ein Probelauf. Die meisten Programme sind in kompatiblem BASIC geschrieben, welche auf den meisten 4 K BASIC-Versionen ausgeführt werden können.

Best.Nr. 80/21, 5 Bücher mit 1100 Seiten DIN A 4 (149 Programme) nur DM 425,--

Die Bände sind auch einzeln erhältlich:

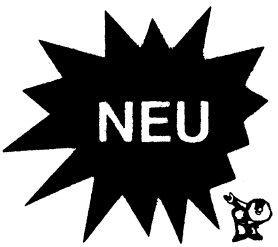
BASIC Volume I
Programme über Buchhaltung, Computerspiele und Bilder (Computerbilder per Programm)
Best.Nr. 80/50 DM 99,--

BASIC Volume II
Programme über mathematische Probleme, technische Programme, Zeichnen, Plotting und Statistik. Grundlegende Statements.
Best.Nr. 80/51 DM 99,--

BASIC Volume III
Erweiterte Geschäftsprogramme für den kommerziellen Bereich. Fakturierung, Inventuren, Gehaltsbuchhaltung.
Best.Nr. 80/52 DM 149,--

BASIC Volume IV
BASIC Programme aus allen Bereichen. Universelle Anwendung.
Best.Nr. 80/53 DM 39,--

BASIC Volume V
Experimentierprogramme.
Best.Nr. 80/54 DM 39,--



BASIC Volume VI
Ein komplettes, interaktives Programmpaket für Floppy Disk - Einsatz im kommerziellen Bereich. Gehaltsabrechnung, Inventur, Gewinn und Verlustrechnung u. v. a.
Best.Nr. 80/48 DM 199,--

BASIC Volume VII
Das Schachprogramm, auf das Sie schon lange gewartet haben. Spielt zwei Farben. Leichter bis mittlerer Schwierigkeitsgrad. Kann einfach erweitert werden. Zugzeit ab 3 Minuten.
Best.Nr.
Medical Billing package (Patientengeschichte und Kostenübersicht für Ärzte)
Disk interaktives Textverarbeitungssystem.
Best.Nr. 80/49 DM 159,--

Genaue Beschreibung des BASIC Volume VII

Schachprogramm (Chess)
Diese Version eines Schachprogrammes in BASIC ist für die meisten 12 K BASIC-Versionen (mit oder ohne kleinen Änderungen) geeignet. Weiterhin sollten noch ca. 12 K im RAM-Bereich zur Verfügung stehen. Das Programm spielt Schach im Schwierigkeitsgrad für Anfänger, kann aber durch entsprechende Änderungen auch schwieriger gestaltet werden. Züge werden im Bereich von 3 Minuten oder mehr durchgeführt. Der erste Zug benötigt ca. 3 Minuten. Die Zugzeit steigt dann entsprechend dem Spielfortschritt entsprechend an. Das Programm kann wahlweise schwarz oder weiß spielen. Das Spielfeld wird auf dem Bildschirm angezeigt oder kann auf einem Drucker ausgegeben werden.

BASIC-Programm für die Arztpraxis (Medbil)
Dieses Programm wurde zur Erleichterung der so teuren und mühsamen Alltagsarbeit der Ärzte entwickelt. Dieses Programm ermöglicht die Speicherung und Durchsicht jeder Patientengeschichte auf dem Bildschirm, die vorher in einer Datenbank gespeichert wurde. Weiterhin besteht eine Möglichkeit der schnellen Überprüfung von aufgelaufenen Behandlungskosten pro Patient. Auf einer 250 K Byte Diskette können etwa 110 Patienten mit all ihren Daten aufgenommen werden. Mit einer Minifloppy (80 K Byte) reduziert sich die mögliche Patientenzahl auf ca. 25 pro Diskette.

Textverarbeitungsprogramm (Wrdpro)
Dieses Programm ermöglicht dem Besitzer von Microcomputern Texte zusammenzustellen und in beliebigen Formaten wieder auszudrucken. Das Programm läuft auf den meisten extended BASIC-Versionen mit mindestens 15 K freiem RAM-Bereich.

W. Hofacker Verlag

8000 München 75

Lieferung durch:

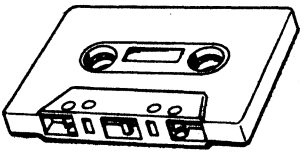
PET

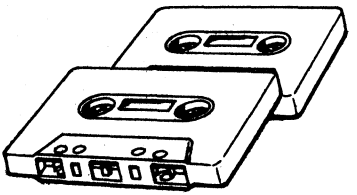
Programme und Bücher

NEU

Software

Ing. W. Hofacker GmbH Tegernseer Straße 18, 8150 Holzkirchen
Telefon 0 80 24/73 31





Cassettes

Programme für Ihren PET Heimcomputer auf Cassetten

U-Boot Jagd für PET, Zehn praktische Programme für Ihren PET auf Cassette. U-Boot Jagd 1, Uhr für den PET, Rechentest, Reaktionstest, Bounce, Faktors, Zahlenraten, Hauskauf, Basisumrechnung, U-Boot Jagd 2
Best.Nr. P1 **DM 49,--**

Einarmiger Bandit (Spielautomat) für PET, inclusive 4 weitere interessante Spielprogramme.
Best.-Nr. P 2 **DM 29,--**

Black Jack (17 und 4) für PET, inclusive 4 weitere interessante Spielprogramme.
Best.-Nr. P 3 **DM 49,--**

JOYSTICK mit Programm für PET, Mit diesem Joystick können Sie Bilder auf Ihrem PET selbst nach Belieben zeichnen. Komplett mit Anschlußkabel und Beschreibung.
Best.-Nr. P 4 **DM 149,--**

Biorythmus für PET, Hangman, Pferderennen. Der Biorythmus zeigt auf dem Bildschirm, abhängig von Ihrem Geburtstag Ihre physische, seelische und psychische Verfassung für ein Monat oder für ein ganzes Jahr. Das Pferderennen ist ein rießiger Spaß, da sich die Pferde auf dem Bildschirm richtig bewegen.
Best.-Nr. P 5 **DM 49,--**

Krieg der Sterne (Star Wars) für PET Mit diesem Programm werden Sie Kapitän der Enterprise und kämpfen gegen die „Klingons".
Best.-Nr. P 6 **DM 29,--**

Krieg der Sterne mit Abschießen für PET (Nur in Verbindung mit Best.-Nr. P 4)
Mit Hilfe des Joystickes können Sie in einem aktiven Computerspiel Flugobjekte abschießen. Wenn Ihnen die Munition ausgeht, sind Sie zur Niederlage verurteilt. Die Geschwindigkeit der Flugobjekte kann eingestellt werden. Getroffene Objekte explodieren.
Best.-Nr. P 7 **DM 29,--**

Las Vegas für PET, Blackjack (17 und 4), CRAPS, Pferderennen mit akkustischen Meldungen und Zeichen.
Best.-Nr. P 8 **DM 29,--**

Haushalts-Utilityprogramme für PET
Diese Cassette enthält 3 Programme: 1. Kaufen oder Mieten, ein Programm zur Analyse, ob ein Immobilienobjekt gekauft oder gemietet werden soll. Das Programm berücksichtigt folgende Begebenheiten: Inflation, Vermögenssteuer, Abschreibungen und anderes.
2. Loans, dieses Programm berechnet Ihnen, wie hoch Sie sich verschulden können, wieviel Sie pro Monat zu bezahlen haben und wie lange Sie zu bezahlen haben.
3. Calendar, Programm berechnet Wochentage und Tage zwischen 1900 und 2100.
Best.-Nr. P 9 **DM 29,--**

Haushalts-Utilityprogramme II für PET
Diese Cassette enthält 3 Programme: 1. Compound Interest, wie lange dauert es, bis ein bestimmter Betrag zusammengespart ist?
2. Loan Amortization Schedule, Programm berechnet, wieviel Sie Zins und Tilgung pro Monat für eine bestimmte Summe Darlehen bezahlen müssen.
3. Car Costs, dieses Programm berechnet Ihnen die Kosten für I Fahrzeug.
Best.-Nr. P 10 **DM 29,--**

Finanzprogramm I und II (Haushaltsfinanzen) für PET
Diese beiden Programme wurden speziell für die Aufnahme und Analyse sämtlicher Einnahmen und Ausgaben eines Haushaltes entwickelt. Programm I wird dazu benutzt, die Verwendungszwecke und Posten einzugeben und anzuzeigen. Sie können geändert, gelöscht und ausgewechselt werden. Programm II benutzt die Daten aus Programm I, analysiert diese und gibt Ihnen eine Aufschlüsselung nach Monatskosten, Jahreskosten oder Kosten pro Verwendungszweck. Graphische Darstellung für alle Posten.
Best.-Nr. P 11 / P 12 **DM 138,--**

Finanzprogramm III für PET Immobilienkauf, Hauskauf, Finanzierung.
Best.-Nr. P 13 **DM 69,--**

Partyprogramm (Party-Time) für PET, Viele PET-Besitzer wollen die Leistungsfähigkeit Ihres neuen Computers gerne bei einer Party oder bei Besuch demonstrieren. Dieses Programm bringt Ihnen ein faszinierendes Frage- und Antwortspiel zwischen Computer und Gästen, indem er versucht, Ihre Gäste zu erkennen. Testprogramm für Ihre Reflexe um festzustellen, wie schnell Sie sind.
Best.-Nr. P 14 **DM 49,--**

Lernprogramme (Educator I) für PET für Kinder zwischen 4 und 6 Jahren.
Namenspiel: Sie geben Ihren Namen ein und befehlen dem Computer, Ihren Namen rückwärts zu buchstabieren, die Buchstaben zu zählen oder zu vertauschen. Eine gute Übung zur Erlernung des Alphabetes. Bildschirmspiel, Story Time, Ratespiel für Zahlen.
Best.-Nr. P 15 **DM 29,--**

Lernprogramme (Educator II) für PET, Für Kinder zwischen 6 und 12 Jahren.
Schreibtest: Testen Sie, ob Sie einen Satz genau nachschreiben können. Wenn Sie gut sind und sehr wenig Fehler machen, wird die Zeit, die der nachzuschreibende Satz auf dem Bildschirm verbleibt immer kürzer. Hangman, Teach, Mathematikprogramm mit einstellbarer Komplexität. Testen Sie Ihre Mathematikkenntnisse!
Best.-Nr. 15/1 **DM 39,--**

Musikprogramme für PET Mit einer kleinen Zusatzschaltung kann der PET Musikstücke spielen und komponieren. Maestro, Musical Mime. Spielen Sie Ihre Melodie auf klavierähnlichen Tasten und lassen Sie den Computer wiederholen. Markov Musik. Lernen Sie dem Computer die Fähigkeit zu Komponieren, dann spielt Ihnen, entsprechend dem Gelernten, der Computer eine Zufallsmelodie. Dieses Programm benutzt eine sehr hoch entwickelte Technik, bekannt als „Markov Musik".
Best.-Nr. P 16 **DM 49,--**

Hexmonitor für PET, Dieser Monitor erlaubt den Zugriff in die Speicherzellen des Pet. Programme können in Maschinensprache gestartet und abgearbeitet werden. Ändern von Adressen, Auslesen des Speichers in Hex. und ASCII, u. v. a.
Best.-Nr. P 17 **DM 19,80**

Supermonitor für den PET, Ein recht leistungsfähiger Monitor zur Erstellung von Programmen in Maschinensprache, die dann an das BASIC-Programm angehängt werden können. Mit Disassembler
Best.-Nr. P 18 **DM 19,80**

MUSIC MAESTRO mit 4 Tönen, Mit einer einfachen Zusatzplatine und der zugehörigen Software können Sie bis zu vier Töne gleichzeitig (Akkorde) mit dem PET erzeugen. Bausatz mit Platine und kompletter Software.
Best.-Nr. P 19 **DM 178,--**

Schachprogramm für PET, Spielt zwei Farben. 8 Schwierigkeitsgrade, Zugzeitmesser, Seitenwechsel in jeder Phase. Spielt auch gegen sich selbst. Phantastische Graphik.
Best.-Nr. P 20 **DM 79,--**

Linear Joystick (zwei Joysticks) Ein Bausatz für zwei Joysticks mit externer Logik für Ihren PET. Ideal für den Aufbau von Computerspielen. Bausatz mit Software.
Best.-Nr. P 21 **DM 199,--**

Programmiertricks für PET, Eine Cassette mit vielen kleinen praktischen Programmiertricks. Ideal für jeden, der die Möglichkeiten des PET voll ausschöpfen möchte.
Best.-Nr. P 22 **DM 39,--**

BASIC-Kurs für PET, Eine pädagogisch geschickt gemachte Einführung in die BASIC-Programmiersprache. Cassette mit Manual . (Wahlweise deutsch oder englisch, bitte bei Bestellung angeben)
Best.-Nr. P 23 **DM 99,--**

Graphik und Bewegung auf dem PET, Wie man graphische Darstellungen, Bilder und Bewegungsabläufe (Animation) auf dem PET programmieren kann, zeigt Ihnen diese Cassette.
Best.-Nr. P 24 **DM 29,--**

Assembler für PET, Sie können jetzt Ihre Programme in 6502 Assembler schreiben und in Maschinensprache übersetzen. 1 + 2 PASS-Assembler, EDITOR, EXECUTER.
Best.-Nr. P 25 **DM 96,--**

Wortverarbeitungsprogramm (Text) für PET, Zum Erstellen von Briefen, Artikeln etc. auf dem Bildschirm und Ausgabe über einen Drucker. Adressenschreiben, Briefe mit wechselnder persönlicher Anrede etc.
Best.-Nr. P 26 **DM 96,--**

REVERSI für PET, Das bekannte Familienspiel können Sie jetzt gegen Ihren PET spielen.
Best.-Nr. P 27 **DM 29,80**

Funktionsgenerator mit PET, Erzeugen Sie Ausgangswellenformen entsprechend mathematischer Ausdrücke: (Fourier) Ideal für Demonstrationen. Platine aus P 19 kann verwendet werden.
Best.-Nr. P 28 **DM 19,80**
Best.-Nr. P 28/1, Platinenbausatz dazu **DM 79,--**

GAMEPAC I, Spielesammlung für Ihren PET. JUMPCHECK, TIP THE DOMINOES, MATCHES, HAMURABI, SLOT MACHINE
Best.-Nr. P 29 **DM 29,80**

GAMEPAC II, Spielesammlung für Ihren PET. Weitere interessante Spiele wie: BUZZWORDS, NIMBLE, GOMOKO, BULLFIGHT, DARTS.
Best.-Nr. P 30 **DM 29,80**

Programmierexperimente für PET, Viele praktische Hilfsprogramme und nützliche Utilities für Ihre Arbeit mit dem PET.
Best.-Nr. P 31 **DM 39,80**

STIMULATING SIMULATIONS für PET, Eine große Programmsammlung mit Manual und Listings.
Best.-Nr. P 32 **DM 49,--**

ASTROLOGY PROGRAMM für PET
Best.-Nr. P 33 **DM 39,80**

BRIDGE-PROGRAMM für PET, Sie können jetzt alleine mit Ihrem PET Bridge spielen. PET simuliert die restlichen Mitspieler.
Best.-Nr. P 34 **DM 49,--**

Computermusik mit dem PET, Eine Cassette mit vielen Programmen und Ideen zur Musikerzeugung mit dem PET. Nützliche Unterprogramme u. v. a.
Best.-Nr. P 35 **DM 39,80**

MORSE-TRAINER, Der PET lehrt Ihnen den Morsecode. Drei verschiedene Betriebsarten. Zufallscode, Mitteilungen und direkte Übergabe.
Best.-Nr. P 36 **DM 29,80**

BÜCHER

BÜCHER MIT BASIC-PROGRAMMEN

25 Games in BASIC (Listing) Die Programme laufen auf dem PET. Inhalt: Number, Letter, Stars, Trap, Bagels, Mugwump, Hurkle, Snark, Reverse, Button, Chomp, Taxman u. v. a.
Best.-Nr. 80/57 **9,80**

57 Practical Programs and Games in BASIC, ca. 200 Seiten Listings.
Best.-Nr. 1000 **DM 35,--**

BASIC Software Library Vol. I, Personal Bookkeeping, Games and Pictures, 49 BASIC-Programme.
Best.-Nr. 80/50 **DM 99,--**

BASIC Software Library Vol. IV, General Purpose und viele Spiele, 21 BASIC Programme.
Best.-Nr. 80/53 **DM 39,--**

Dr. DOBBs COMPUTERZEITSCHRIFT. Die komplett gebundene Sammlung der ersten, sehr interessanten Ausgaben. Sehr viele Programmlistings für den Hobbycomputerfan.
Best.-Nr. 80/20, ca. 260 Seiten DIN A 4 **DM 59,--**

Microprocessor Software Handbuch, C. Lorenz
Microcomputerprogrammierung für Anfänger und Fortgeschrittene. BASIC-Einführung in Deutsch.
Best.-Nr. 27, über 280 Seiten **DM 29,80**

PET 6502 Programmierhandbuch, das Handbuch für jeden PET-Besitzer und solche, die es werden wollen. Einführung, Programmbeispiele, PET-Erweiterungsmöglichkeiten.
Best.-Nr. 110 **DM 29,80**

The BASIC-Cookbook, Ken Tracton
Über 138 Seiten BASIC-Einführung mit vielen praktischen Programmierbeispielen. Alphabetisch geordnet, ideal auch als Lexikon.(Diagnose, Erhöhung der Rechengeschwindigkeit u. v. a.)
Best.-Nr. 1055 **DM 24,80**

A Beginner's Guide to Computers & Microprocessors – with projects
Eine Einführung von Anfang an in die Mikroprozessortechnik. Bauen Sie Ihren eigenen Computer, programmieren Sie ihn und bedienen Sie ihn. Viele praktische Schaltungen. Über 300 Seiten.
Best.-Nr. 1015 **DM 29,80**

ELCOMP (Zeitschrift für Microcomputertechnik)
Jahresabonnement incl. Porto und Versand **DM 42,--**

650X Software Manual (Programming Manual), sämtliche wichtige Daten und Programmierhinweise für die 6502 Programmierung. Ca. 200 Seiten.
Best.-Nr. 80/42 **DM 19,80**

650X Hardware Manual, 6502 Systemorganisation, Grundlagen, Applikationshilfen u. v. a. mehr.
Best.-Nr. 80/43 **DM 19,80**

6500 Seminarheft (Briefing Notebook), eine kleine Einführung in die 6500 Systemfamilie.
Best.-Nr. 80/45 **DM9,80**

6500 Datenblätter, eine Sammlung aller wichtigen Datenblätter aus der 6500-Familie.
Best.-Nr. 80/46 **DM 9,80**

6502 Microcomputer, Aufbau und Programmierung. Ein deutsches Anleitungsbuch zum Einstieg in die Microcomputertechnik mit Hilfe des KIM-1. Viele Programmierbeispiele von einem Pädagogen speziell für den Anfänger entwickelt. (PET, VIM und AIM)
Best.-Nr. 109 **DM 29,80**

Instant BASIC, eine Einführung für den Hobbyisten, Studenten oder Anfänger. Über 160 S. DIN A4.
Best.-Nr. 80/58 **DM 24,80**

My Computer Likes Me ... When I Speak in BASIC, eine leicht verständliche Einführung in die BASIC-Programmiersprache. Über 64 Seiten DIN A4 mit vielen Beispielen.
Best.-Nr. 80/56 **DM 9,80**

Your home computer, ein ideales Buch für jeden, der sich über den Einsatz von Heimcomputern informieren möchte
Best.-Nr. 80/55 **DM 24,80**

Spezial-Datencassetten für Microcomputer

Die Spezial-Microcomputer-Datencassetten der Fa. Ing. W. Hofacker GmbH haben eine Länge von ca. 15 Metern mit einer Laufzeit von 14 Minuten (7 Min. pro Seite).Dies bringt Ihnen einen Vorteil bei der Aufnahme und Wiedergabe auf Ihrem Computer, da Sie keine langen Vorlauf- und Rückspielzeiten haben. Für diese Spezial-Cassetten wird nur hochwertiges Bandmaterial verwendet (Agfa oder BASF). Das Gehäuse ist vierfach verschraubt und besonders widerstandsfähig und robust. Die Bänder wurden nach folgenden Eigenschaften ausgesucht:
1. Gleichmäßig hoher Ausgangspegel, 2. Fehlerfreier Betrieb bei allen Signaldichten der wichtigsten Microcomputersysteme. Incl. Schutzgehäuse und Aufkleber für persönliche Beschriftung.
Preis pro St. DM 2,90 – 10 St. DM 24,80 – 100 St. 198,--

Leercassetten

Produkt-Beschreibung

Die Microcomputer Daten-Cassetten der Firma Ing. W. Hofacker GmbH haben eine Länge von ca. 15 Metern mit einer Laufzeit von 14 Minuten (7 Minuten pro Seite). Es wurde für diese Cassette nur sehr hochwertiges Bandmaterial verwendet. (Agfa oder BASF). Das Gehäuse ist vierfach verschraubt. Es ist besonders widerstandsfähig und robust.

Das Band wurde besonders auf folgende Eigenschaften ausgesucht.

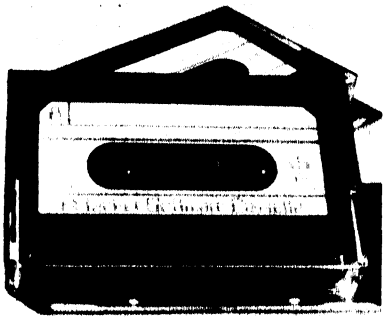
1. Gleichmäßig hoher Ausgangspegel
2. Fehlerfreier Betrieb bei allen Signaldichten der wichtigsten Microcomputersysteme.

Die Cassette bietet Ihnen ein Höchstmaß an Betriebssicherheit bezüglich fehlerfreier Aufnahme und Wiedergabe.
Jede Cassette wird in einem zweiteiligen Schutzgehäuse geliefert. Auf Vorder- und Rückseite sind zwei Aufkleber mit genügend Raum für Ihre persönlichen Beschriftungen.

Wieviel K Byte Programme kann man bei folgendem Computer auf einer Seite der Cassette speichern?

Computer	Speichermöglichkeit
Commodore PET	16 K Byte
IMSAI	43 K Byte
APPLE	36 K Byte
Heathkit	36 K Byte
Kansas City Std.	16 K Byte
Radio Shack Lev. II	16 K Byte
Tarbell Cassette	43 K Byte
Processor Technology	12 K Byte
KIM-1	12 K Byte
NASCOM	12 K Byte
EXIDY	12 K Byte
SIM-1	12 K Byte

Kein langes Suchen mehr!
HOFACKER-Software-Cassetten haben die richtige Länge. Praktisch, sicher, schnell!



Fragen Sie bei Ihrem Händler oder bestellen Sie direkt:

Lieferung per Nachnahme

BESTELLSCHEIN

Menge	Beschreibung	Preis/DM	Gesamt
	Muster	2,90	
	10er Packung	24,80	
	100er Packung	198,--	
	Gesamtbetrag		

Lieferanschrift:

Name.....Vorname.....

Straße:.....Ort:.....

Unterschrift:.....

Menge	Best.-Nr.	Titel/Bezeichnung	Einzelpreis DM	Gesamtpreis DM
.....	80/58	Instant BASIC (Einführung BASIC)	24.80	.....
.....	80/59	6502 Software Paket	89.--	.....
.....	80/60	6800 Software Paket	89.--	.....
.....	80/61	8080/8085/8048/ Software Paket	89.--	.....
.....	80/62	Z-80/Z-8/Z-8000 Software Paket	89.--	.....
		NEUANKÜNDIGUNGEN 1978/79		
.....	110	Programmierhandbuch für PET	29.80	.....
.....	111	TRS-80 Programmierhandbuch (Z 80)	29.80	.....
.....	112	PASCAL Programmierhandbuch	29.80	.....
.....	113	BASIC Programmierhandbuch	24.80	.....
.....	114	Der Microcomputer im Kleinbetrieb	39.80	.....
.....	115	FOCAL Programmierhandbuch	19.80	.....
.....	116	16 Bit Microcomputer	19.80	.....
.....	117	FORTRAN für Hobbycomputer	19.80	.....
.....	118	Microcomputertechnik für Anfänger	19.80	.....
		MICROCOMPUTER-DATENBÜCHER		
.....	80/63	M 6800 Programmierhandbuch	19.80	.....
.....	80/86	8080 Applikationsberichte	14.80	.....
.....	80/65	8080/8085 Assembly Lang.Manual	29.80	.....
.....	80/66	MCS 85 Product Description	9.80	.....
.....	80/67	MCS 48 Product Description	9.80	.....
.....	80/68	MCS 48 Appl. Seminar Notebook	19.80	.....
.....	80/69	MCS 85 Appl.Seminar Notebook	19.80	.....
.....	80/70	Peripheral Design Handbook	24.80	.....
.....	80/71	First Book of Kim	19.80	.....
.....	80/72	GAME Playing with BASIC	24.80	.....
.....	80/73	8080 Software-Bibliothek	699.--	.....
.....	80/75	8080 Applikationsberichte	14.80	.....
.....	80/85	8048 Assembly Lang. Manual	29.80	.....
		8080/6800 - SOFTWARE		
.....	80/76	6800 Text Editing Syst.,Listing m.Cassette	129.--	.....
.....	80/77	6800 Text Processing Syst., List.m.Cassette	160.--	.....
.....	80/81	6800 Micro BASIC PLUS List.m.Cassette	89.--	.....
.....	80/79	6800 Mnemonic Assembler, List.m.Cass.	129.--	.....
.....	80/80	6800 Disassembler, Listing m.Cassette	49.--	.....
.....	80/82	6800 Relocator, Listing mit Cassette	49.--	.....
.....	80/83	8080 Text Editing System, Listing	99.--	.....
.....	80/84	8080 Text ProcessingSystem, Listing	129.--	.....
.....	80/55	Your home computer	24.80	.....
.....	80/56	My Computer Likes Me (BASIC-Einf.)	9.80	.....
.....	80/57	Computer Games (PCC-Games)	9.80	.....
.....	1015	Beginners Guide to Microprocessors	29,80	.....
.....	1055	BASIC-Cookbook	24,80	.....
.....	N 1	TTL-Databook	19.80	.....
.....	N 2	Linear Datenbuch	24.80	.....
.....	N 13	Linear Applications, Volume 1	19.80	.....
.....	N 3	Linear Applications, Volume 2	14.80	.....
.....	N 4	Voltage Regulator Handbook	9.80	.....
.....	N 5	Audio Handbook	14.80	.....
.....	N 6	Special Function Data Book	9.80	.....
.....	N 7	c mos Integrated Circuits Data Book	9.80	.....
.....	N 7/1	MOS LSI Data Book	24.80	.....
.....	N 8	SC/MP Programmier-u.Ass.Handbuch	19.80	.....
.....	N 9	SC/MP Microprocessor Applications	19.80	.....
.....	N 11	SC/MP Technical Description	19.80	.....
.....	N 20	Memory Data Book	19.80	.....
.....	N 21	Interface Integrated Circuits Data Book	19.80	.....
.....	N 22	Data Acquisition Handbook	19.80	.....
.....	N 23	Power Transistor Databook	9.80	.....
.....	N 24	FET Databook	9.80	.....

BESTELLUNG

Heute noch an Ihren örtlichen Elektronik Fachhändler absenden.

Oder an Ing. W. Hofacker GmbH, Tegernseerstr. 18, 8150 Holzkirchen, Tel.: 08024/7331
Alle Preise sind empfohlene Verkaufspreise incl. MWSt. Es gelten unsere Lieferungs- und Zahlungsbedingungen. Lieferung erfolgt per NN oder Vorkasse auf Pschk.Mchn. 159 94-807.

Absender: Name ..... Vorname .....

Straße: ..... Ort: .....

BEST.NR.	STCK.	BEZEICHNUNG	PREIS / DM

BESTELLUNG für ein ELCOMP-Abonnement

○ Bitte senden Sie mir ab sofort alle künftigen Ausgaben der ELCOMP. Preis DM 42,-- pro Jahr, incl. Versand und Verpackung.

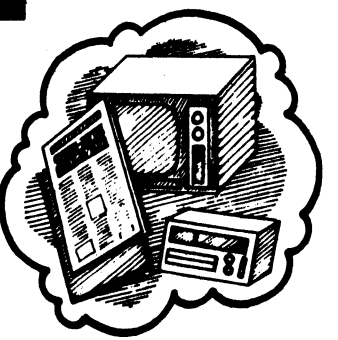
○ Ich bitte um Lieferung per Nachnahme. Mit der Lieferung des 1. Heftes wird der Betrag für ein Jahr von DM 42,-- + NN Gebühr erhoben. Kündigung 8 Wochen vor Jahresende.

Datum: ..... Unterschrift .....

Cassetten sind grundsätzlich vom Umtausch ausgeschlossen. Defekte Cassetten werden gegen eine neue des gleichen Types umgetauscht. Gerichtsstand für beide Teile ist München oder Hamburg.

Microcomputer Software und Bücher

NEU



für
6502 KIM
SC/MP
8080
Z80
6800

Ing. W. Hofacker GmbH Tegernseer Straße 18, 8150 Holzkirchen
Telefon 0 80 24/73 31

6502 KIM-1

MASCHINENABHÄNGIGE PROGRAMME FÜR 650X

6502 KIM Software Paket, viele Listings und Dump-Listings für Ihren KIM. Zufallsgenerator, Hangman, Hurkle, Master Mind, Switch 6502, Acey-Ducey.
Best.-Nr. 80/59 DM 89,--

First Book of KIM, das ideale Einführungsbuch für KIM-Benützer mit vielen Programmierbeispielen. Additionsprogramm, Bandit, Black Jack, Clock, Craps, Horse Race, Nim, Lunar Lander, Music Box, Reverse, Wumpus, Memory Test u. v. a. mehr.
Best.-Nr. 80/71 DM 19,80

650X Software Manual (Programming Manual), sämtliche wichtige Daten und Programmierhinweise für die 6502 Programmierung. Ca. 200 Seiten.
Best.-Nr. 80/42 DM 19,80

650X Hardware Manual, 6502 Systemorganisation, Grundlagen, Applikationshilfen u. v. a.
Best.-Nr. 80/43 DM 19,80

6500 Seminarheft (Briefing Notebook), eine kleine Einführung in die 6500 Systemfamilie.
Best.-Nr. 80/45 DM 9,80

6500 Datenblätter, eine Sammlung aller wichtigen Datenblätter aus der 6500-Familie.
Best.-Nr. 80/46 DM 9,80

6502 Microcomputer, Aufbau und Programmierung. Ein deutsches Anleitungsbuch zum Einstieg in die Microcomputertechnik mit Hilfe des KIM-1. Viele Programmierbeispiele von einem Pädagogen speziell für den Anfänger entwickelt. (PET, VIM und AIM)
Best.-Nr. 109 DM 29,80

Weiterhin führen wir folgende Software für 6502: TINY-BASIC für KIM 6502, 6502 Assembler, Texteditor und BASIC, FOCAL für KIM sowie Programmierbeispiele in FOCAL.

SC/MP INS 8060

MASCHINENABHÄNGIGE PROGRAMME UND BÜCHER FÜR SC/MP (INS 8060)

SC/MP Programmier- und Assemblerhandbuch. Das ideale Nachschlagewerk für den Programmierer des SC/MP. Einführung, Verwendung Assemblersprache, Programmiertechniken, Beispiele u. v. a. mehr. Über 200 Seiten, DIN A 5.
Best.-Nr. N 8 DM 19,80

SC/MP Microprocessor Application Handbook. Eine Einführung in die Hardware des SC/MP. Viele Applikations- und Programmierbeispiele. Ca. 150 Seiten, DIN A 4.
Best.-Nr. N 9 DM 19,80

SC/MP-8060 Microcomputer Handbuch. Eine Anleitung zum Aufbau eines eigenen Microcomputersystems. Komplett mit allen Unterlagen (Schaltbilder, Printvorlagen, Anleitungen). Systemdaten: 4/8 K RAM, residentes BASIC, residenter Assembler/Editor/Debugger, TV-Interface, Floppy Disk Interface, Cassetten-Interface.
Best.-Nr. 108 DM 29,80

Compute-Hefte. SC/MP Clubzeitschrift. Eine Sammlung von 10 Heften mit vielen interessanten Schaltbeispielen und Programmierbeispielen. Eine echte Sensation!
Best.-Nr. 80/44 DM 9,80

SC/MP Technical Description. Technische Beschreibung des SC/MP (INS 8060) mit Systembeispielen und Systemvorschlägen. Dieses Buch braucht jeder SC/MP-Benutzer.
Best.-Nr. N 11 DM 19,80

Memory condensed Databook. Datenblätter vieler wichtiger Speicherbausteine wie MM 2114, MM 2101, MM 2102, ROMs, Schieberegister u. v. a. mehr.
Best.-Nr. N 26 DM 2,--

8080 8085 8048

MASCHINENABHÄNGIGE PROGRAMME UND BÜCHER FÜR 8080/8085 und 8048

8080 Software Paket. Spiele in Maschinensprache für 8080 Systeme. Komplette Listings mit Dump-Listing. Spiele: Hangman, Zufallsgenerator, Master Mind und Game Space Voyage.
Best.-Nr. 80/61 (Listings) DM 89,--

8080 Text Editing System. Dieser Editor ist zeilen- und inhaltorientiert. Dadurch wird die Programmierung in Assemblersprache wesentlich vereinfacht. Dieses System gehört zu den leistungsfähigsten Texteditoren für 8080 überhaupt. Speicherbedarf 6 K beginnend bei 1000 H.
Best.-Nr. 80/83 (Listings) DM 99,--

8080 Text Processing System. Es erlaubt den Gebrauch von über 50 Befehlen zur speziellen Textformatierung. (Multiple Spacing, Kontrolle des linken Zeilenanfangs, programmiertes Einrücken, Paging, rechter und linker Randausgleich, Zentrierung u. v. a. mehr. Es besteht die Möglichkeit der Macrodefinition zum Aufbau kundenspezifischer Befehle. Speicherbereich: 1000 Hex. bis 4184 Hex. Listing mit Einführung in die Textverarbeitung und Manual.
Best.-Nr. 80/84 DM 129,--

8080/8085 Software Bibliothek. Über 1000 Seiten Listings in Assemblersprache aus fast allen Bereichen der Technik. Das ideale Nachschlagewerk für alle 8080 Programmierer.
Best.-Nr. 80/73 (Listings) DM 699,--

8085 Seminar Notebook. Eine Seminarunterlage mit vielen Applikationen. Über 60 Seiten.
Best.-Nr. 80/69 DM 19,80

8085 Product Description. Eine Kurzbeschreibung des 8085 mit Daten und Applikationen. 47 S.
Best.-Nr. 80/66 DM 9,80

8080/8085 Assembly Language Manual. Ausführliche Programmierunterlagen mit vielen Beispielen. Jeder Befehl ist genauestens beschrieben. Ca. 200 Seiten, DIN A 4.
Best.-Nr. 80/65 DM 29,80

8080/8085 Applikationsberichte-Sammlung. Best.-Nr. 80/86 DM 14,80

8080/8085 Peripheral Design Handbook. Neueste Applikationsbeispiele, Datenblätter und Programme für die Interface-Familie der 8080/8085 und 8048 Prozessoren.
Best.-Nr. 80/70 DM 24,80

MCS 85 Users Manual. Das Handbuch für jeden 8085-Besitzer. Ca. 180 Seiten.
Best.-Nr. 80/6 DM 24,80

MCS 48 Users Manual. Das Handbuch für jeden 8048-Interessenten. Ca. 180 Seiten.
Best.-Nr. 80/7 DM 24,80

8048 Product Description, Kurzbeschreibung m. Datenblättern. Best.Nr.80/67 DM 9,80

8048 Application Seminar Notebook. Über 37 S. Best.-Nr. 80/68 DM 19,80

8080,8085,8048 Intel Datenkatalog 1978. Best.-Nr. 80/5 DM 29,80

8048 Assembly Language Manual. Programmierunterl., Beispiele. Best.Nr.80/85 DM 29,80

Z80

MASCHINENABHÄNGIGE PROGRAMME UND BÜCHER FÜR Z 80, Z 8, Z 8000

Z 80 Software Paket. Viele Spiele und nützliche Programmierbeispiele für Z 80 Microcomputer North Star Horizon auf Diskette.
Best.-Nr. 80/62 DM 89,--

Z 80 Assembler Handbuch (Benutzerhandbuch) in deutscher Sprache. Alles, was der Z 80 Programmierer für die Programmentwicklung benötigt. Ca. 200 Seiten DIN A 4
Best.-Nr. 80/29 DM 29,80

Z 80 Datenbuch, in deutscher Sprache. Datenblätter der gesamten Z 80-Familie. Z 80 CTC, Z 80 PIO, Z 80 DMA, Platinen und Entwicklungssysteme. Ca. 150 Seiten, DIN A 4.
Best.-Nr. 80/27 DM 29,80

Z 80 Microcomputer Applicationen, in deutscher Sprache. Ca. 20 Seiten, DIN A 4.
Best.-Nr. 80/28 DM 14,80

Z 80 Anwenderhandbuch Z 80 KIT. Bauanleitung für den Z 80 KIT. Ca. 45 Seiten, DIN A 4.
Best.-Nr. 80/2 DM 19,80

Microcomputer Technik von Blomeyer. Eine umfassende, einführende und weiterführende Hilfe zum Einstieg in die Microcomputer Technik mit Programmierbeispielen. Als praktische Grundlage dient der supermoderne Microcomputer Z 80A von Zilog.
Best.-Nr. 24 DM 29,80

6800

MASCHINENABHÄNGIGE PROGRAMME UND BÜCHER FÜR 6800 MICROCOMPUTER

6800 Software Paket. Spiele mit dem Microcomputer 6800. Floating Point Package u.v.a.
Best.-Nr. 80/60 (Listings) DM 89,--

6800 Micro BASIC Plus. Ein leistungsfähiges BASIC für 4 K/ 8 K Systeme. Alle wichtigen Statements incl. NEXT, DIM (einfache u. mehrdimensionale Felder bis zu 98 x 98) u. v. a.
Best.-Nr. 80/81 (Listing mit Cassette) Kansas City Format DM 89,--

6800 Mnemonic Assembler/Disassembler und Relocator, 3 Listings, 3 Cassetten.
Best.-Nr. 80/79/80/82 DM 227,--

6800 Text Editing System. Eigenschaften wie 8080 Text Editing System, jedoch Speicherbedarf 5 K RAM beginnend bei 0 Hex.
Best.-Nr. 80/76 (Listing mit Cassette) Kansas City Format DM 129,--

6800 Text Processing System, Eigenschaften wie 8080 Text Processing System.
Best.-Nr. 80/77 (Listing mit Cassette) Kansas City Format DM 160,--

6800 Programmierhandbuch, in deutscher Sprache. Das Handbuch für jeden 6800 Programmierer.
Best.-Nr. 80/63 DM 19,80

Bestellen Sie noch heute bei Ihrem Buch- oder Fachhändler oder direkt beim Hofacker Verlag, Tegernseerstr. 18, D-8150 Holzkirchen / Obb. - Tel.: 08024/7331.

Hofacker Verlag

Ing. W. Hofacker GmbH

Tegernseer Straße 18, 8150 Holzkirchen
Telefon 0 80 24/73 31

Fachbücher u. Datenbücher

BOOK STORE

ELEKTRONIK
FACHBÜCHER
vom
HOFACKER VERLAG

Fachbücher ELEKTRONIK
Mikroprozessoren
Mikrocomputer

Lieferung durch Ihren Fachhändler

ELCOMP
Ein kompetentes Microcomputer-Magazin

10 Hefte pro Jahr. Einzelpreis DM 4,50 (Doppelheft DM 9,--) Jahresbezugspreis DM 59,-- incl. Porto, Verp. u. Mwst. Heute noch abonnieren. Erhältlich auch bei Ihrem Fachhändler!

HOFACKER-VERLAG

Verlagsprogramm

Übersicht lieferbarer Bücher

Bestell Nr.	ISBN	Verfasser	Titel	Preis DM
1	3-921682-01-0	Hofacker	Transistor Berechnungs- und Bauanleitungshandbuch, Band 1	19,80
2	3-921682-02-9	Hofacker	Transistor Berechnungs- und Bauanleitungshandbuch, Band 2	19,80
3	3-921682-03-7	Gebauer	Elektronik im Auto	9,80
4	3-921682-04-5	Lorenz	IC-Handbuch, TTL, CMOS, Linear	19,80
5	3-921682-05-3	Steinbach	IC-Datenbuch, TTL, CMOS, Linear	9,80
6	3-921682-06-1	Steinbach	IC-Schaltungen, TTL, CMOS, Linear	9,80
7	3-921682-33-9	Hofacker	Elektronik Schaltungen	5,–
8	3-921682-08-8	Lorenz	IC-Bauanleitungs-Handbuch	19,80
9	3-921682-09-6	Lorenz	Feldeffekttransistoren	5,–
10	3-921682-34-7	Lorenz	Elektronik und Radio, 4. Auflage	19,80
11	3-921682-11-7	Lorenz	IC-NF Verstärker	9,80
12	3-921682-12-6	Bernstein	Beispiele Integrierter Schaltungen (BIS)	19,80
13	3-921682-13-4	Lorenz	HEH, Hobby Elektronik Handbuch	9,80
14	3-921682-14-2	Lorenz	IC-Vergleichsliste	29,80
15	3-921682-15-0	Lorenz	Optoelektronik Handbuch	19,80
16	3-921682-16-9	Bernstein	CMOS Teil 1 Einführung Entwurf, Schaltbeispiele	19,80
17	3-921682-17-7	Bernstein	CMOS Teil 2 Entwurf und Schaltbeispiele	19,80
18	3-921682-18-5	Bernstein	CMOS Teil 3 Entwurf und Schaltbeispiele	19,80
19	3-921682-19-3	Lorenz	IC-Experimentier Handbuch	19,80
20	3-921682-20-7	Lorenz	Operationsverstärker	19,80
21	3-921682-21-5	Lorenz	Digitaltechnik Grundkurs	19,80
22	3-921682-22-3	Bernstein	Mikroprozessoren, Eigenschaften und Aufbau 2. Aufl.	19,80
23	3-921682-23-1	Lorenz	Elektronik Grundkrus Kurzlehrgang Elektronik	9,80
24	3-921682-35-5	Hans Peter Blomeyer-Bartenstein	Mikrocomputer Technik	29,80
25	3-921682-25-8	C. Lorenz	Hobby Computer Handbuch	29,80
26	3-921682-26-8	H. Bernstein	Mikroprozessor Teil 2	19,80
27	3-921682-27-4	C. Lorenz	Mikrocomputer Software Handbuch	29,80
28	3-921682-28-2	C. Lorenz	Lexikon + Wörterbuch für Elektronik und Mikroprozessortechnik LEM	29,80
29	3-921682-29-0	C. Lorenz	Mikrocomputer Datenbuch	49,80
30	3-921682-30-4	C. Lorenz	Aktivtraining-Mikrocomputer (Ordner)	49,80
31	3-921682-31-2	C. Lorenz	57 Programme in BASIC (Cames)	39,–
32	3-921682-32-0	C. Lorenz	Circuits Digital Et Pratique	19,80
33	3-921682-36-6	Dr. Hatzenbichler	Microcomputer Programmierbeispiele	19,80
41	–	–	Experimentierplatine für 14, 16, 24, 28 und 40 polige DIL IC's	79,–
105/1	–	–	TTL-Experimentierbuch	5,–
106/1	–	–	CMOS-Experimentierbuch	5,–
109	3-921682-43-6	P. Heuer	6502 Microcomputer Aufbau und Programmierung	29,80
110	3-921682-49-5	C. Lorenz	Programmierhandbuch für PET	29,80
113	3-921682-48-7	C. Lorenz	BASIC Programmierhandbuch	19,80

Ing. W. Hofacker GmbH Verlag
8 München 75