Messen, steuern, regeln

mit dem

VC 20

und

commodore 64

Anders Andersson Arne Kullbjer

HALLER intelligente Arbeits- und Lernmittel

Messen, steuern, regeln

mit dem

und

commodore 64

Anders Andersson Arne Kullbjer

HALLER intelligente Arbeits- und Lernmittel

Wir bedanken uns bei der Firma Commodore Büromaschinen GmbH, Lyoner Straße 38, 6000 Frankfurt 71, die uns durch Einräumung von Nutzungsrechten an bestehenden Urheberrechten unterstützt hat.

ISBN 91-40-20683-1

Übersetzung aus dem Schwedischen: **Dipl.-Ing. Hugo Haller, Fachübersetzungen**, Saarbrücken
Titelgestaltung: Karlheinz Reinsch, Marketing, Saarbrücken
Reproreife Gesamtgestaltung: HALLER Verlag, Saarbrücken

Liber Tryck Stockholm 1984 304378

Inhalt

Kapitel 1: WIE DER COMPUTER ARBEITET 9

Einführung 10
Adressbus 12
Datenbus 12
Steuerbus 13
Basic-Interpreter 13
ASCII-Tabelle 14
Betriebssystem 15
Eingabe-/Ausgabe-Kanäle 15
Anwender-Schnittstelle 15
Kontaktbelegung der Anwender-Schnittstelle 17

Kapitel 2: VERWENDUNG DER ANWENDER-SCHNITTSTELLE 19

Stecker und Kabel 20
Binärcode 21
Grundsätzliches über die Eingabe-/Ausgabe-Kanäle
VIA (VC-20) und CIA (C-64) 23
Register der E/A-Prozessoren 23
Einschränkungen 25
PEEK und POKE 25
Einige Experimente 26
Eine Leuchtdiode zum Leuchten bringen 27
Eine Leuchtdiode blinken lassen 28
Schalter simulieren 28

Kapitel 3: REGELN MIT DEM MIKROCOMPUTER 31

Programmschaltwerk 32
Steuerung von Verkehrsampeln 33
Verstärker für größere Stromstärken 35
Relaisbetrieb 35
Halbleiterrelais 37
Automatische Telefonwahl 38
Erzeugung analoger Signale 40
Widerstandsnetzwerk 40
AD 7523 41
Veränderung der Pulsbreite 42
Steuerung von Gleichstrom-Motoren 43

Kapitel 4: MESSEN MIT DEM MIKROCOMPUTER 45

Mengenmessung mit einer Fotozelle 46
Basic-Programm 47
Rechner VC-20 48
Rechner C-64 50
Frequenzmessung 51
Analoge Signale für den Computer 52
Grundlagen der A/D-Wandler 53
V/f-Wandlung mit LM 331 54
Temperaturmessung 58

Kapitel 5: MASCHINENSPRACHE **61**

Was ist Maschinensprache? 62
Maschinensprache und Basic 63
Programmieren in Maschinensprache auf dem VC-20 und C-64 63
Der Akkumulator 64
Mnemonische Zeichen 64
Maschinen-Programme aufrufen 65
Adressierungsarten 65
Programme in den Speicher eingeben 69
Programmausführung 70
Impuls in Maschinensprache erzeugen 71
VCMON und 64MON 74

Kapitel 6: DATENÜBERTRAGUNG ÜBER DIE RS-232-SCHNITTSTELLE **75**

Was ist RS-232? 76
Signale und Steckverbinder 78
Wahl von Übertragungsgeschwindigkeit, Wortlänge und Parität 80
Signale senden und empfangen 81

Kapitel 7: DIE IEEE-SCHNITTSTELLE **87**

Die IEEE-488-Schnittstelle 88
Die IEEE-Schnittstelle des VC-20 und C-64 89
Signale und Stecker 89
Verwendung der IEEE-Schnittstelle 89

Kapitel 8: EINIGE PRAKTISCHE VERSUCHE **91**

Lichtstift 92
Datalogger 95
Temperaturregelung 101
Steuerung eines Schrittmotors 103

Anhang A: Register der Mikroprozessoren 6502 und 6510 ..... **107**

Anhang B: Befehlssatz der Mikroprozessoren 6502 und 6510 ..... **111**

Anhang C: Tabelle für Binärzahlen und deren Dezimaldarstellung **133**

Vorwort

Ein wichtiger Anwendungsbereich von Computern ist die Überwachung und Steuerung von Prozessen und das automatische Aufnehmen von Meßwerten.

Dieses Buch soll grundlegende Kenntnisse über das Messen und Regeln mit Mikrocomputern vermitteln und die Verwendung des VC-20 und C-64 bei einigen Prozessen zeigen.

Um den Inhalt des Buches anwenden zu können, sind Grundkenntnisse der Programmiersprache BASIC erforderlich und für einige Versuche auch Kenntnisse der Dateiverwaltung. Außerdem sind Grundkenntnisse der Digitaltechnik nützlich.

Das Buch wurde von Lennart Bergström, Computer-Presse-Verlag, bearbeitet.

Linköping, im Januar 1984

Die Autoren

KAPITEL 1

Wie der Computer arbeitet

VC-20 C-64

Einführung

Bisher haben Sie wahrscheinlich Ihren Computer zum Rechnen, Speichern und Verarbeiten von Daten benutzt und die Ergebnisse auf dem Bildschirm oder Drucker ausgegeben. Dies sind sehr häufig vorkommende Anwendungsbereiche für Computer. Schätzungsweise 90 % der Personal-Computer werden für Textverarbeitung und Berechnungen eingesetzt. Computer haben jedoch eine weitere wichtige Funktion: Sie stellen ein ausgezeichnetes Hilfsmittel zur Überwachung und Steuerung von Prozessen und zur automatischen Meßwerterfassung dar. Der VC-20 und der Commodore-64 (wir nennen ihn der Kürze wegen C-64) bilden hier keine Ausnahme, sie sind vielmehr für diesen Anwendungszweck hervorragend gut geeignet.

Im vorliegenden Buch werden wir die grundlegenden Begriffe über das Messen, Steuern und Regeln mit Computern erläutern und im weiteren den VC-20 und C-64 in einige Meß- und Regelprozesse einschalten, um wenigstens eine Andeutung von den großen Möglichkeiten des Gerätes zu zeigen, die in ihm stecken.

Zuerst müssen wir uns jedoch ansehen, wie ein Computer prinzipiell arbeitet. Wenn wir die Grundlagen dafür erarbeitet haben, können wir leichter verstehen, welche Grenzen vorhanden sind und welche Möglichkeiten es gibt, diese zu umgehen.

Arbeitsweise des Computers

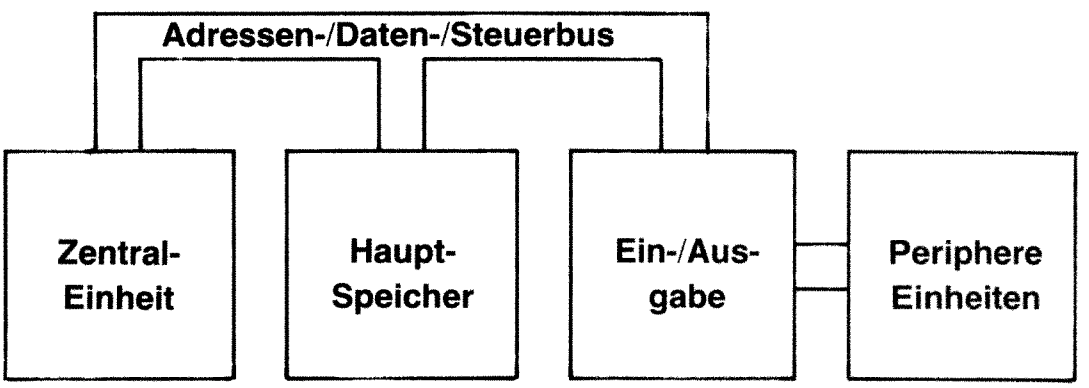


Abb. 1.1

Das Bild zeigt das Blockschaltbild eines Computers. Dieses Blockschema ist sehr allgemein gehalten . Im Prinzip sind alle Computer so aufgebaut, nicht nur der VC-20 und C-64. Wir wollen nun die Funktionsweisen näher betrachten.

Die Zentraleinheit führt Berechnungen und die Dateiverwaltung für den Speicher und angeschlossene Geräte durch.

Der Speicher enthält teils die Programme, die der Zentraleinheit mitteilen, was zu tun ist, teils Daten. Der VC-20 und C-64 enthalten zwei Speicherarten, den sogenannten Nur-Lese-Speicher »ROM« (Read Only Memory) und den Schreib-/Lese-Speicher »RAM« (Random Access Memory). Beide Arten sind Halbleiterspeicher, d. h. integrierte Schaltkreise, die aus Zehntausenden von Transistoren auf einem Siliziumchip bestehen, so groß etwa wie ein halber Fingernagel!

Die Nur-Lese-Speicher sind vom Hersteller fest programmiert und können nicht verändert werden. Im VC-20 und C-64 sind im wesentlichen zwei Programme im ROM enthalten: das Betriebssystem und der BASIC-Interpreter.

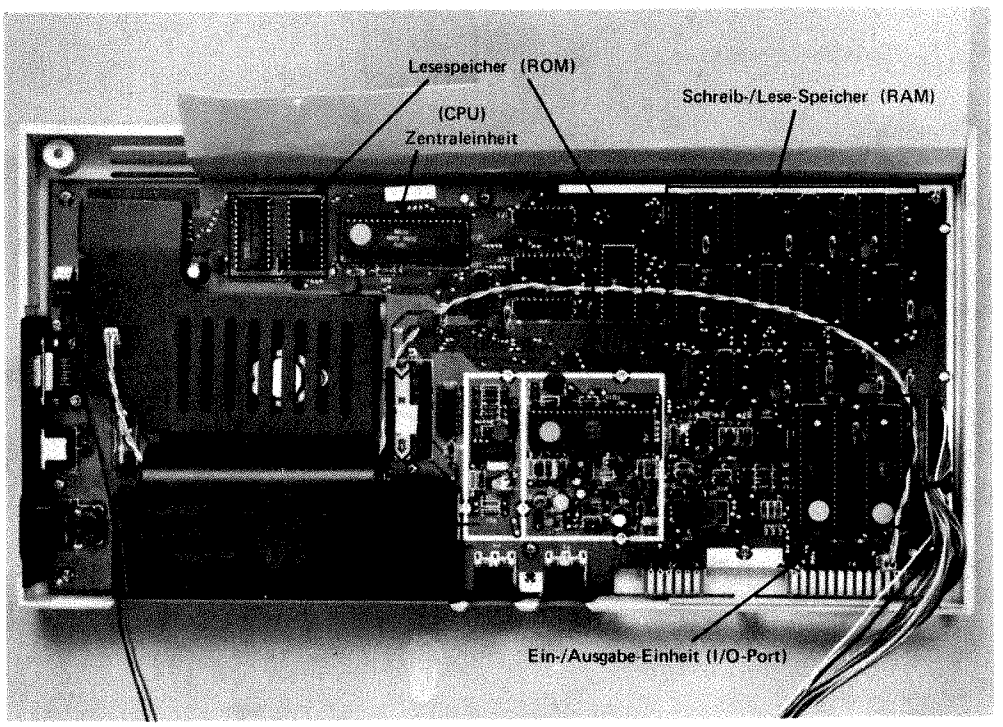


Abb. 1.2 VC-20 Mikroprozessoren

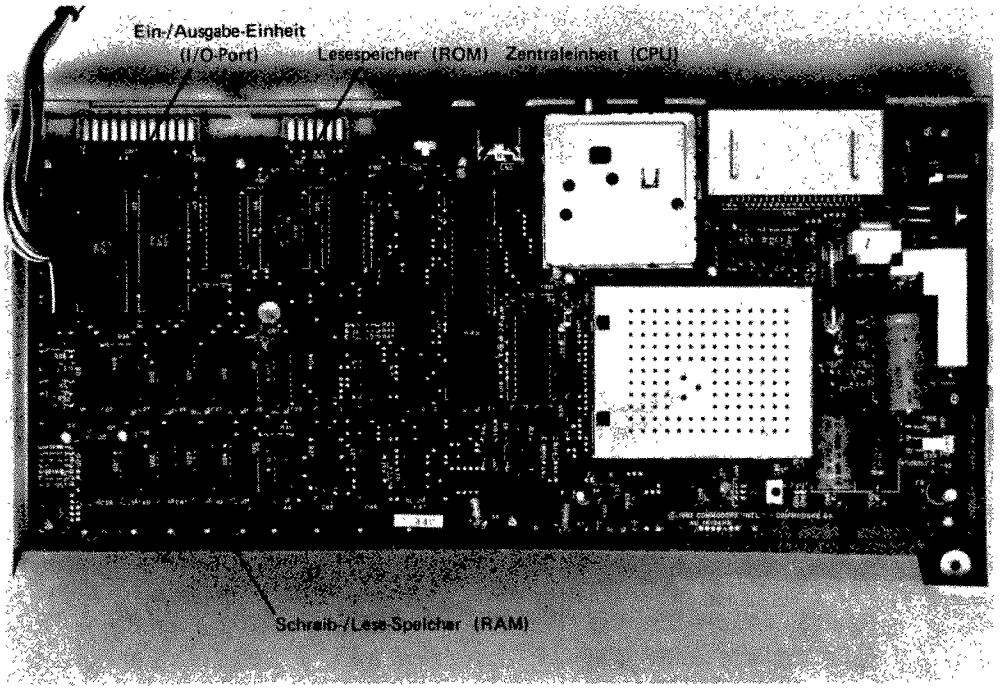


Abb. 1.3 C-64 Mikroprozessoren

Adressbus

Gehen wir für einen Augenblick zurück zu Abb. 1.1! Die Verbindung zwischen Zentraleinheit, Speicher und Eingabe-/Ausgabe-Kanälen wird als Bus bezeichnet und besteht aus einer Anzahl elektrischer Leitungen, die jede für sich eine 1 oder 0 übertragen können. Man unterscheidet zwischen Adreßbus, Datenbus und Steuerbus.

Der Adreßbus (der im VC-20 und C-64 aus 16 Leitungen besteht) läßt die Zentraleinheit bestimmen, welchen Platz, welche Speicheradresse und welchen Eingabe-/Ausgabe-Kanal (E/A-Kanal) sie zum Lesen oder Schreiben verwenden will. Man sagt, daß die Zentraleinheit den Speicher bzw. den E/A-Kanal (E/A-Adresse) adressiert. Die 16 Adreßbus-Leitungen gestatten den Zugriff zu $2^{16} = 65536$ Adressen.

Datenbus

Der Datenbus besteht beim VC-20 und C-64 aus 8 Leitungen. Er wird zum

Informationsaustausch zwischen Zentraleinheit einerseits und einem adressierten Speicherplatz oder einem E/A-Kanal andererseits benutzt. Die 8 Datenleitungen können jeweils eine achtziffrige Binärzahl übertragen. Informatiker nennen eine solche achtziffrige Binärzahl »8-Bits« oder ein Byte (wird Beit gesprochen). Das Wort Bit stammt eigentlich vom englischen »binary digit«, Binärzahl. »Byte« ist englischer Computerslang und heißt wörtlich »Mundvoll«. Ein Byte ist die Informationsmenge, die die Zentraleinheit (Mikroprozessor) im VC-20 und C-64 je Arbeitsschritt verarbeiten kann. Für die ASCII-Zeichen paßt das wunderbar: Die 128 verschiedenen Zeichen können in nur 7 Bits untergebracht werden ($2^7 = 128$).

Wie in aller Welt schafft es aber der Computer, Rechnungen mit so vielen Ziffern auszuführen? In 8 Bits ist ja nur für Zahlen zwischen 0 und 255 Platz. Kein Problem, der Computer kann große Zahlen verarbeiten, weil er nur jeweils einen kleinen Teil davon verwendet. Die Rechenzeit wird dadurch natürlich länger, das Ergebnis liegt aber trotzdem »blitzschnell« vor, wie Sie schon bemerkt haben werden. Die Übertragung von 8-Bit-Daten dauert nur wenige Mikrosekunden. Die Multiplikation von zwei 9-ziffrigen Zahlen dauert nur wenige Hundertstel Sekunden!

Steuerbus

Der Steuerbus beim VC-20 und C-64 besteht aus 10 bis 20 Leitungen, jede mit einer speziellen Steuerfunktion. Es gibt z. B. eine Leitung, die angibt, ob der Mikroprozessor aus dem Speicher lesen oder in ihn schreiben soll. Eine andere Leitung stellt den Computer beim Start auf definierte Startbedingungen. Für die Computerbauer ist der Steuerbus wichtig; uns nützen solche Details aber wenig, und wir gehen somit hier nicht näher auf sie ein.

Basic-Interpreter

Der Basic-Interpreter liegt im Nur-Lese-Speicher (ROM) des VC-20 und C-64, Abb. 1.2 und 1.3. Der Basic-Interpreter übersetzt Basic-Programme in Binärzeichen, die von der Zentraleinheit verstanden werden. Ein Binär-Code besteht aus Einsen und Nullen. Es entspricht eine Eins z. B. +5 V und eine Null 0V. Sobald Sie den Computer einschalten, beginnt er, Teilprogramme des Betriebssystems auszuführen. Über die Tastatur kann jetzt das Basic-Programm eingegeben werden, das der Computer ausführen soll. Jedes der eingegebenen Zeichen wird im Schreib-/Lese-Speicher (RAM) in kodierter Form als eine Folge von Einsen und Nullen gespeichert. Die verwendeten Zeichen werden ASCII-Zeichen genannt. Die ASCII-Zeichen sind Ihnen gewiß schon früher begegnet, wenigstens, wenn Sie das Buch »Programmieren in BASIC

auf dem VC-20 und Commodore-64« gelesen haben. Dort bestanden die ASCII-Zeichen jedoch aus gewöhnlichen Dezimalzahlen zwischen 0 und 127.

ASCII-Tabelle

ASCII-Code	Zeichen	ASCII-Code	Zeichen	ASCII-Code	Zeichen
000	NULL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	bkslash
007	BEL	050	2	093	]
008	BS	051	3	094	up arrow
009	HT	052	4	095	back arr
010	LF	053	5	096	space
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	;
038	&	081	Q	124	<
039	'	082	R	125	=
040	(	083	S	126	>
041	)	084	T	127	DEL
042	*	085	U		

Abb. 1.4 ASCII-Tabelle

Der ASCII-Code für den Buchstaben A ist 65, und für die Zahl Eins ist er 49. Der Computer behandelt die betreffenden Zeichen in binärer Form als achtziffrige Binärzahlen. Der Buchstabe A mit dem ASCII-Zeichen 65 wird als Binärzahl 01000001, die Ziffer 1 mit dem ASCII-Zeichen 49 als Binärzahl 00110001 abgelegt. Beim Ausschalten des Computers verschwinden durch Wegfall der Spannung alle Informationen im Schreib-/Lese-Speicher (RAM)!

Betriebssystem

Das Betriebssystem ist ein Programm, das u. a. den Computer die Tastatur abfragen und feststellen läßt, welche Taste gedrückt wurde und danach festlegt, was danach durchzuführen ist. Obwohl das Betriebssystem weit mehr Aufgaben hat, ist es hauptsächlich für den Datenfluß verantwortlich, der zwischen Tastatur, Bildschirm, Disketten, Kassettengerät, Drucker und Anwenderprogrammen stattfindet.

Eingabe-/Ausgabe-Kanäle

Aus Sicht der Zentraleinheit besteht kein Unterschied zwischen dem Schreib-/Lese-Speicher und den E/A-Kanälen. Für den Anwender bestehen jedoch erhebliche Unterschiede. Die Informationen, welche die Zentraleinheit in Form einer 8-Bit-Binärzahl an einen Ausgabe-Kanal weitergibt, kann an einem der Kontakte einer externen Steckerleiste abgenommen und zur Steuerung einer externen Einheit verwendet werden. Die Information, welche die Zentraleinheit an den Schreib-/Lese-Speicher weitergibt, bleibt dagegen im dunkeln, da nur die Zentraleinheit den Speicher wieder lesen kann! Entsprechend verhält es sich mit einem Eingabe-Kanal. Der Anwender kann von außen ein 8-Bit-Binär-Zeichen eingeben, das die Zentraleinheit durch Lesen des Eingabe-Kanals aufnehmen kann.

Anwender-Schnittstelle

Die Anwender-Schnittstelle des VC-20 besteht hauptsächlich aus einem integrierten Schaltkreis (Chip) mit der Bezeichnung 6522. Der Hersteller nennt sie oft VIA (engl.: Versatile Interface Adapter). Beim C-64 wird dieser Chip CIA (Complex Interface Adapter) genannt mit der Bezeichnung 6526. Im Zusammenhang mit Computern stellt eine Schnittstelle (engl. interface) eine Anordnung dar, die es dem Computer gestattet, an andere Einheiten zum Transfer von Daten und Steuerimpulsen angeschlossen zu werden. Im VC-20

und C-64 sind Schnittstellen für die Tastatur, den Anschluß von Joysticks (für Spielprogramme) und eines Modems (Anschluß an das Telefonnetz) vorhanden. Alle eingebauten Schnittstellen haben etwas gemeinsam: Sie benutzen die obengenannten integrierten Schaltkreise VIA oder CIA. Insgesamt enthalten der VC-20 und der C-64 2 dieser VIA- bzw. CIA-Bausteine, jeder mit 2 sogenannten E/A-Kanälen. Wenn die eingebauten Schnittstellen ihren Aufgaben entsprechend belegt sind, verbleibt eine Schnittstelle für den Anwender, die Anwender-Schnittstelle.

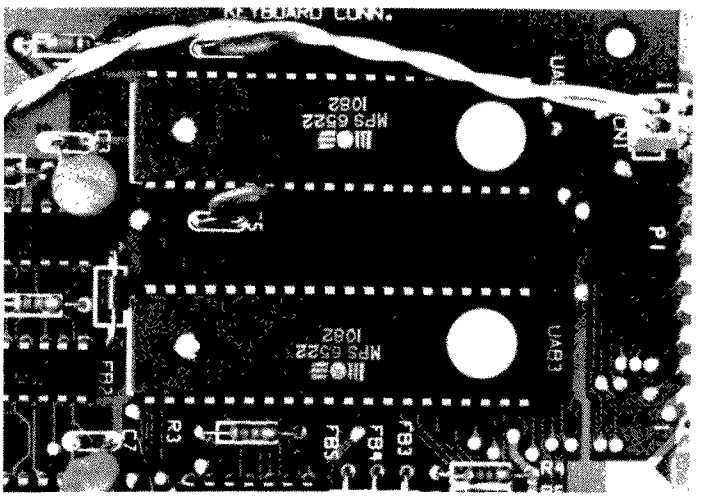


Abb. 1.5 6522 VIA (VC-20)

Die 8 Anschlüsse der Anwender-Schnittstelle (und ein paar weitere Leitungen) sind an einen Stecker an der Rückseite des Computers geführt. Der Hersteller bezeichnet diesen Stecker als Anwender-Schnittstelle (engl.: user port). Nun wissen wir, was unter einer Schnittstelle (oder interface) zu verstehen ist!

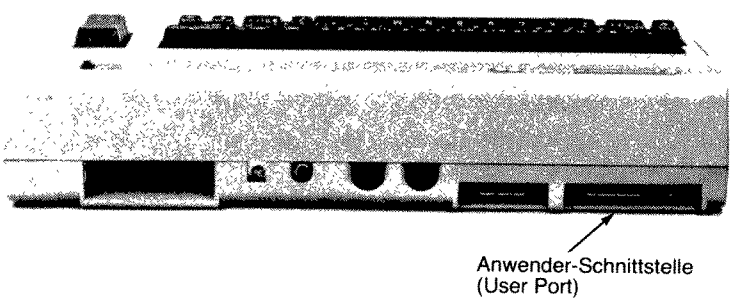
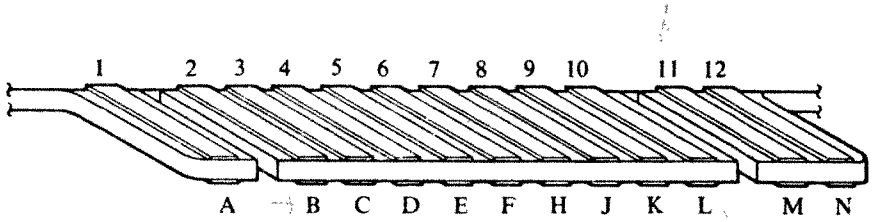


Abb. 1.6 Rückseite des C-64

Kontaktbelegung der Anwender-Schnittstelle

Die Signale werden an die Kontaktklemmen, wie in Abb. 1.7, geführt. Im nächsten Kapitel werden wir uns ansehen, was die verschiedenen Signale bedeuten und wozu sie verwendet werden.



Pin No.	VC-20	C-64
1	GND	GND
2	+5 V (100 mA max)	+5V (100 mA max)
3	RESET	RESET
4	JOY0	CNT1
5	JOY1	SP1
6	JOY2	CNT2
7	LIGHT PEN	SP2
8	CASSETTE SWITCH	PC2
9	SERIAL ATN IN	SERIAL ATN IN
10	+9 V (100 mA max)	9 V AC (50 mA max)
11	GND	9 V AC (50 mA max)
12	GND	GND
A	GND	GND
B	CBI	FLAG
C	PB0	BP0
D	BP1	PB1
E	BP2	PB2
F	PB3	PB3
H	PB4	PB4
J	PB5	PB5
K	PB6	PB6
L	PB7	PB7
M	CB2	PA2
N	GND	GND

Abb. 1.7 Kontaktleiste des VC-20/C-64 mit Pin-Belegung

Adresse der Anwender-Schnittstelle

Diese hat im VC-20 die Adresse 37136 und im C-64 die Adresse 56577. Vor dem Start ist es jedoch erforderlich, dem VIA- bzw. CIA-Prozessor mitzuteilen, ob er als Eingang oder Ausgang benutzt werden soll. Im nächsten Kapitel wird erläutert, wie das geschieht.

KAPITEL 2

Verwendung der Anwender-Schnittstelle

VC-20 C-64

Stecker und Kabel

Um mit dem User-Port (E/A-Kanal) Versuche machen zu können, brauchen Sie Kabel und einen geeigneten Stecker und auch eine Platine, auf der Sie schnell und einfach eine Schaltung verdrahten können. Es gibt mehrere handelsübliche Stecker, die auf die Anwender-Schnittstelle des VC-20 und C-64 passen. Der Stecker muß 2×12 Kontaktklemmen aufweisen, Teilung 0.156". Am besten löten Sie ein Flachbandkabel mit verschiedenfarbigen Adern an den Stecker. Dazu eignet sich ein 16-adriges Kabel, 0,5 bis 1 m lang. Auf der anderen Seite wird ein DIP-Stecker mit 16 Anschlüssen angebracht.

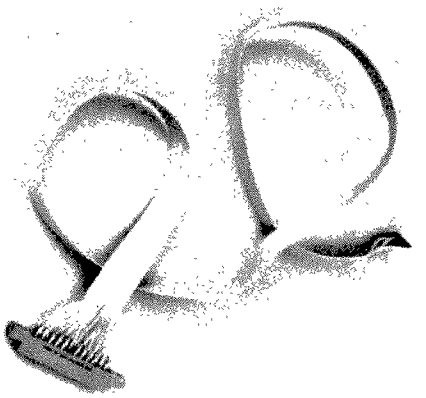


Abb. 2.1 Kabel mit Stecker

Bei den folgenden Versuchen gehen wir davon aus, daß Sie dieses Kabel benutzen. Die Verdrahtung muß folgendermaßen erfolgen:

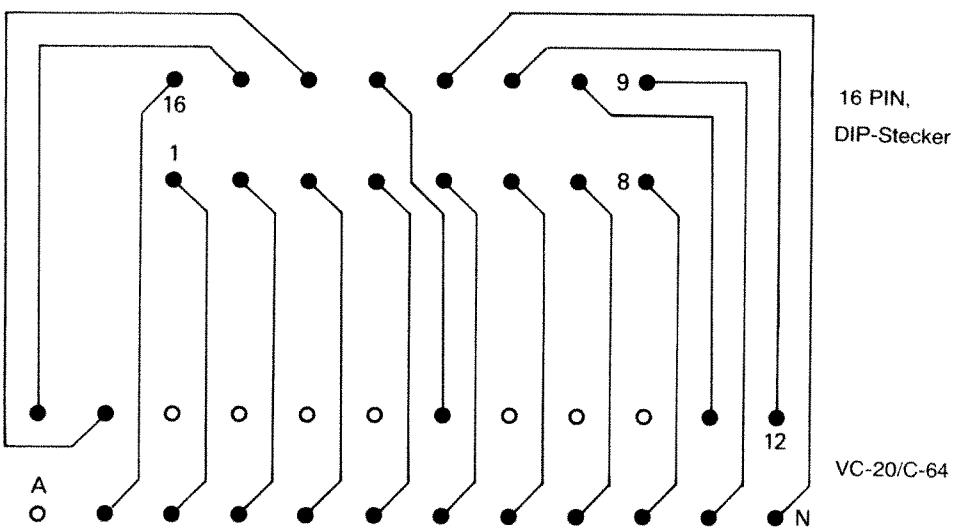


Abb. 2.2

Tabelle der PIN-Belegung

VC-20/C-64	DIP	Signal
B	16	CB1 (FLAG)
C	1	PBO
D	2	PB1
E	3	PB2
F	4	PB3
H	5	PBA
J	6	PB5
K	7	PB6
L	8	PB7
M	9	CB2 (PA2)
N	12	GND
1	15	GND
2	14	+5V
7	13	LJUSPENNA (SP2)
11	10	GND (9 VAC)
12	11	GND (GND)

Binärcode

Wir beginnen, uns dem Hauptziel dieses Kapitels zu nähern, nämlich den Computer die Elektronik außerhalb des Computers beeinflussen zu lassen, und zwar über die Anwender-Schnittstelle. Damit wir wissen, wie das funktioniert, müssen wir uns jedoch zuerst mit Binärzahlen beschäftigen.

Wir Menschen werden von Kind an darauf gedrillt, Dezimalzahlen zu benutzen, d. h. Zahlen mit den Ziffernsymbolen 0 bis 9. Es kann daher sein, daß Sie das binäre Zahlensystem mit seinen zwei Ziffernsymbolen (0 und 1) etwas befremdet. Das Dezimalsystem hat übrigens die Basis 10, das binäre System die Basis 2.

Wie kommt es eigentlich, daß wir größere und kleine Zahlen im Dezimalsystem ausdrücken können, obwohl es nur zehn Ziffernsymbole gibt? Wie können wir wissen, ob die Zahl 186,37 »Einhundertsechsundachtzig Komma siebenunddreißig« bedeutet? Nun, der Kniff bei der Sache ist, daß die Stellung einer Ziffer im Verhältnis zum Komma dieser Ziffer ein bestimmtes Gewicht oder einen bestimmten Wert darstellt. 186,37 ist eigentlich eine Abkürzung für

$1 \times 100 + 8 \times 10 + 6 + 1 + 3 \times {}^{1}/_{10} + 7 \times {}^{1}/_{100}$

Eine sehr brauchbare Abkürzung, nicht wahr? Mathematiker würden vielleicht lieber so schreiben:

$$1\times10^{2} + 8\times10^{1} + 6\times10^{0} + 3\times10^{-1} + 7\times10^{-2}$$

Sie verstehen jetzt sicherlich, warum wir die Basis des Dezimalsystems 10 nennen.

Das binäre System ist auch ein solches Positionssystem, d. h. die Stellung der Ziffern bestimmt deren Gewicht oder Wert. Im Binärsystem ist der Gewichtsfaktor natürlich 2 und nicht zehn.

Die binäre Zahl 101101.01 ist folglich die verkürzte Schreibweise für

$$1\times2^{5} + 0\times2^{4} + 1\times2^{3} + 1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 0\times2^{-1} + 1\times2^{-2}$$

oder, wenn Sie so wollen,

$$1\times32 + 0\times16 + 1\times8 + 1\times4 + 0\times2 + 1\times1 + 0\times\tfrac{1}{2} + 1\times\tfrac{1}{4}.$$

Die Summe der Einzelwerte ergibt den Dezimalwert der Binärzahl, mit der wir begannen. Die Summe beträgt 45,25. Sie kennen jetzt also eine Methode, mit der Sie Binärzahlen in die entsprechende Dezimalzahl umrechnen können. Abb. 2.3 ist nützlich, wenn Sie irgendeine 8-Bit-Binärzahl von der Basis 2 in die Basis 10 umwandeln wollen (oder umgekehrt).

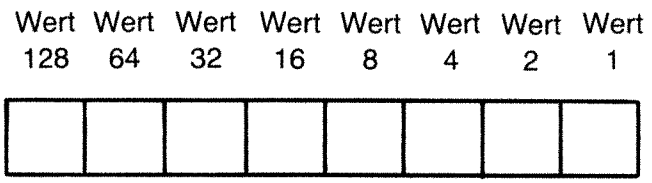


Abb. 2.3

Jedes Kästchen kann eine Eins oder eine Null enthalten. 10101100_2 ist also gleich 172_{10}.

Bezeichnet Basis 10 Bezeichnet Basis 2

Die Umwandlung in die andere Richtung ist schwieriger, aber es geht! Z. B. kann 135 in die Teilsummen 128 + 4 + 2 + 1 aufgeteilt werden, d. h. $135_{10} = 10000111_2$.

Für die Umrechnung gibt es Tabellen, die uns die mühselige Arbeit des Ausrechnens ersparen. Sie finden eine solche Tabelle in Anhang 3.

Grundsätzliches über die Eingabe-/Ausgabe-Kanäle VIA (VC-20) und CIA (C-64)

Die VIA- und CIA-Schnittstellen sind ganz allgemeine Eingabe-/Ausgabe-Kanäle (E/A-Kanäle) mit vielen möglichen Verwendungen. Für unsere Zwecke reicht glücklicherweise eine vereinfachte Betrachtung. In den folgenden Abschnitten wird nur so viel darüber gesagt, daß Sie den VC-20 und C-64 als Prozeßrechner verwenden können. Wollen Sie zu einem späteren Zeitpunkt mehr darüber wissen, sollten Sie die Datenblätter des Herstellers der Prozessoren 6522 (VIA) und 6526 (CIA) einsehen.

Register der E/A-Prozessoren

Die E/A-Bausteine enthalten nicht weniger als 16 verschiedene Bit-Register, die der Anwender alle erreichen kann. Aus Sicht des Computers ist ein Register eine Speicherstelle, und die Zentraleinheit kann jedes Register lesen und beschreiben. Jedes Register hat eine bestimmte Adresse. Die Register in dem E/A-Baustein, der die Anwender-Schnittstelle bildet, haben Adressen, die für den VC-20 im Bereich 37136-37151 und für den C-64 im Bereich 56576-56591 liegen. Der Anwender kann die Funktion der E/A-Bausteine bis in alle Einzelheiten festlegen, wenn er die passenden Daten in den Registern ablegt. VIA und CIA sind, wie gesagt, sehr komplexe Schnittstellen. Sie enthalten z. B. zwei Timer (Zeitgeber und Zeitmesser), einen seriellen Ausgabekanal und eine parallele Drucker-Schnittstelle. Die Register überwachen all das und noch einiges mehr.

Wir müssen uns zunächst nur mit 2 der 16 Register beschäftigen. Das eine ist die Anwender-Schnittstelle selbst. Über sie läuft der Informationsaustausch mit den angeschlossenen Geräten. Das andere ist das Datenrichtungsregister. Jeder E/A-Kanal besteht eigentlich aus 2 Hälften, genannt A und B. Es gibt also 2 Schnittstellen und 2 Register. Wir verwenden den B-Teil und haben daher Zugang zu Schnittstelle B und Register B.

Beim VC-20 hat die Schnittstelle B die Adresse 37136 und das Datenrichtungsregister 37138. Beim C-64 hat Schnittstelle B die Adresse 56577. Datenrichtungsregister B hat die Adresse 56579.

Die 8 Anschlüsse an Schnittstelle B haben die Bezeichnung PB0-PB7. Der Zugang erfolgt über den DIP-Stecker des Kabels:

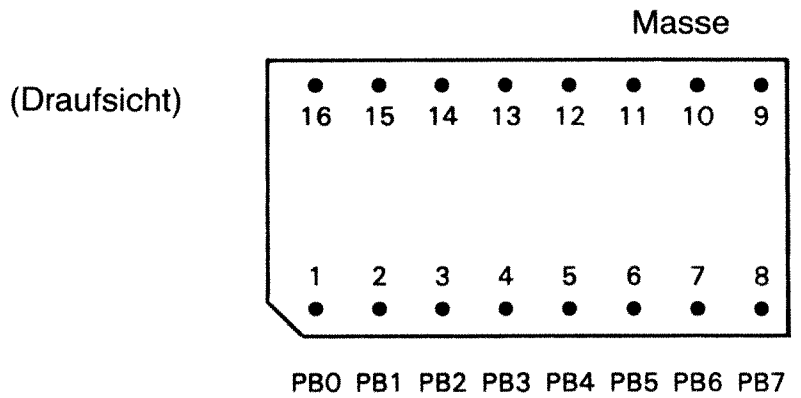


Abb. 2.4

Wenn Sie den Computer die Information über Schnittstelle B ausgeben lassen, dann sind die Signale am DIP-Stecker als »1« oder »0« (+5V oder 0V) zugänglich. PB0 ist das LSB (least significant bit), das Bit mit dem niedrigsten Stellenwert. Da nur 8 Bits angesprochen werden können, ist die größte Zahl, die ausgegeben werden kann, $255_{10} = 11111111_2$. Die Zahl $135_{10} = 10000111_2$ läßt folgende Signalkombination an Schnittstelle B entstehen:

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
+5V	0V	0V	0V	0V	+5V	+5V	+5V

Bevor Sie etwas ausprobieren, müssen Sie jedoch wissen, welchen Einfluß das Datenrichtungsregister hat. Datenrichtungsregister B enthält auch 8 Bits. Jedes Bit bestimmt die Richtung (d. h. Eingang oder Ausgang) eines der 8 Datenkanäle der Schnittstelle B. Wünschen Sie, daß alle 8 Datenkanäle PB0-PB7 Ausgänge sein sollen, so müssen zunächst 8 Einsen in das Datenrichtungsregister B geschrieben werden, d. h. $11111111_2 = 255_{10}$. Wollen Sie dagegen, daß alle Datenkanäle PB0-PB7 Eingänge sein sollen, dann müssen 8 Nullen ins Datenrichtungsregister B geschrieben werden, d. h. $00000000_2 = 0_{10}$. Ein- und Ausgänge können nach Gutdünken gemischt werden. Wollen Sie z. B. PB0 und PB1 zu Ausgängen und die übrigen zu Eingängen machen, verfahren Sie wie folgt:

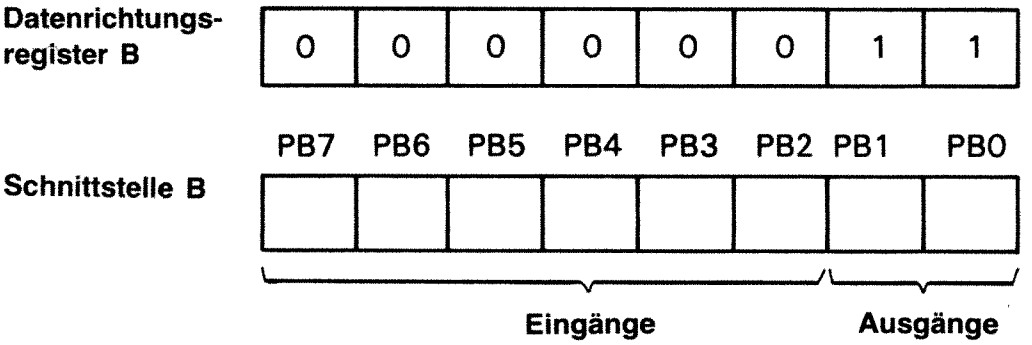


Fig. 2.5

Sie schreiben also die Zahl $00000011_2 = 3_{10}$ ins Datenrichtungsregister B, d. h. in Adresse 37138 (56579 im C-64). Im nächsten Abschnitt werden wir sehen, wie man Register liest und in sie schreibt. Vorher noch einige Worte zur Warnung.

Einschränkungen

E/A-Kanäle können beschädigt werden, wenn die Herstellerforderungen an Signale nicht eingehalten werden. Denken Sie deshalb daran, daß beim Verwenden eines E/A-Kanals als Eingang die Spannung des ankommenden Signals zwischen 0 und +5V liegen muß. Spannungen zwischen 0 und 0,8V werden als 0V betrachtet, Signale mit einer Spannung zwischen +2,4V und +5V als 1. Wenn ein E/A-Kanal als Ausgang benutzt wird, kann er mit maximal 3mA bei 1,5V belastet werden. Er kann deshalb nur zum Steuern eines Leistungstransistors oder einer Leuchtdiode dienen. Beachten Sie, daß die Anwender-Schnittstelle bei +5V nicht mit mehr als 100mA belastet werden darf!

PEEK und POKE

Der Befehl PEEK erlaubt es, den Inhalt einer bestimmten Adresse zu lesen, während POKE zum Schreiben in bestimmte Adressen verwendet wird. Wollen Sie z. B. einer Variablen den Wert zuordnen, der in Adresse 37136 (56577 beim C-64) steht, dann lautet der Befehl A=PEEK(37136) bzw. A=PEEK(56577). A erhält einen Dezimalwert zwischen 0 und 255. Wünschen Sie nur den Inhalt der Adresse 37136, so schreiben Sie natürlich PRINT PEEK(37136) und drücken die RETURN-Taste. Der Computer antwortet mit der Dezimalzahl, die dem Wert der Adresse entspricht. Es wird daher PEEK benutzt, wenn Daten von der Anwender-Schnittstelle abgefragt und an den Computer weitergeleitet werden sollen.

Wollen Sie eine Zahl (zwischen 1 und 255 dezimal) in eine bestimmte Speicher-Adresse schreiben, müssen Sie den POKE-Befehl benutzen. POKE 37138,255 schreibt die Zahl 255 (d. h. 11111111) in die Speicher-Adresse 37138, was im übrigen dem VIA mitteilt, daß der Kanal B des VC-20 acht Ausgänge haben soll. POKE 56579,255 führt Entsprechendes im C-64 aus.

Mit dem Befehl POKE 37136,1 wird die Zahl 1 (d. h. 00000001 binär) in die Adresse 36136 geschrieben. Es werden dadurch im VC-20 am Kontakt PB0 der Anwender-Schnittstelle +5V angelegt. Mit dem Befehl POKE 56577,1 geschieht das Entsprechende im C-64.

Wir wollen jetzt einige einfache Experimente ausführen.

Einige Experimente

Wir schlagen vor, daß Sie für die Versuche eine Experimentier-Platine benutzen, auf der Sie die Bauteile verdrahten. Es gibt im Handel eine Vielzahl geeigneter Experimental-Platinen.

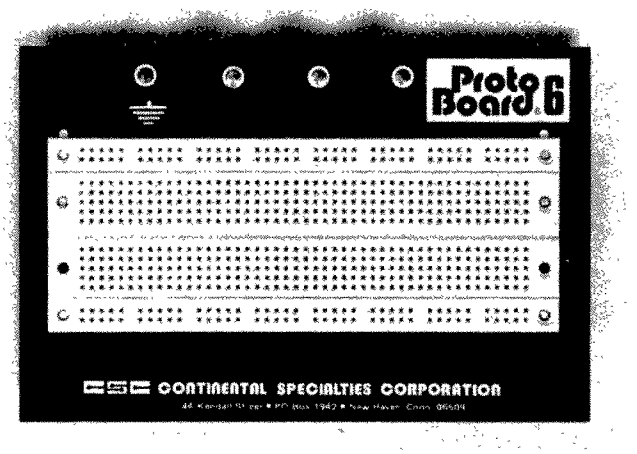


Abb. 2.6

Verbindungen werden durch handelsüblichen isolierten Draht von 0,6 mm Stärke hergestellt. Die übrigen Teile (Widerstände, Kondensatoren, Leuchtdioden, Digital-Bausteine mit Sockel) passen direkt auf die Platine. Das gilt auch für den DIP-Steckel der Anwender-Schnittstelle.

Eine Leuchtdiode zum Leuchten bringen

Schalten Sie eine Leuchtdiode zwischen PB0 und Masse. Die Kathode der Leuchtdiode wird an Masse angeschlossen. Der Kathodendraht ist etwas kürzer als der andere, kann jedoch auch etwas abgeschrägt sein.

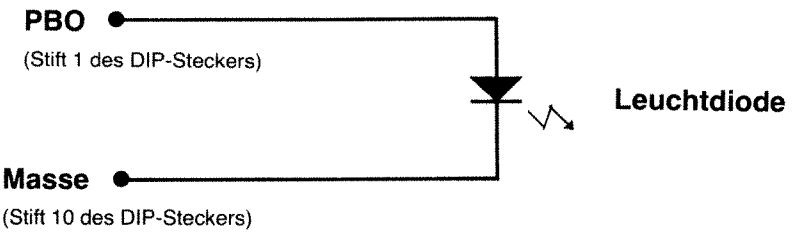


Abb. 2.7

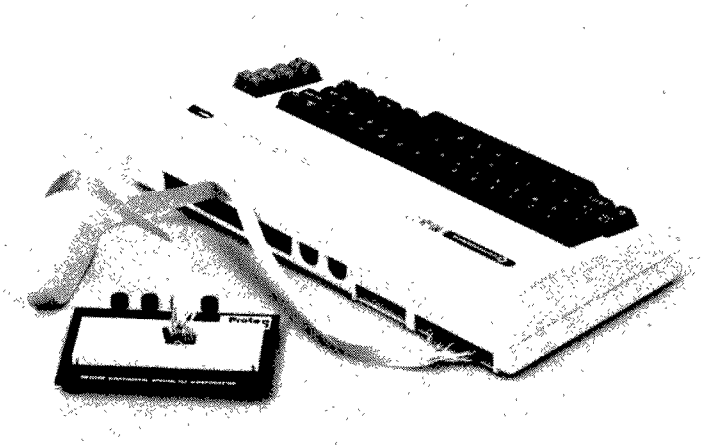


Abb. 2.8 Experimental-Aufbau

Folgendes Programm bringt die Leuchtdiode mit dem VC-20 zum Leuchten (für den C-64 sind die Adressen zu ändern):

```
10 POKE 37138,255
20 POKE 37136,1
```

Testen Sie jetzt das Programm. Zeile 10 schaltet Schnittstelle B 8 Mal auf Ausgang, Zeile 20 gibt eine »Eins« über PB0 aus. Können Sie die Diode ausschalten?

Eine Leuchtdiode blinken lassen

Ob die Diode erlischt, wenn man die Zahl 0 in Schnittstelle B schreibt, d. h. Adresse 37136? Wir wollen sehen, ob sie blinkt, wenn man das VC-20-Programm in eine Schleife einbindet. (Für den C-64 bitte die Adressen entsprechend ändern).

```
10 POKE 37138,255     (8 Ausgänge)
20 POKE 37136,1       (Diode ansteuern)
30 POKE 37136,0       (Diode ausschalten)
40 GOTO 20            (Rücksprung und wieder einschalten)
```

Lassen Sie das Programm laufen. Sie stellen kein Blinken fest! Der Grund: Das Ein- und Ausschalten geht viel zu schnell! Der Computer braucht nur wenige Tausendstel-Sekunden für jede Zeile, d. h. die Diode blinkt ca. 500 Mal je Sekunde. Das menschliche Auge kann nicht so rasch folgen (das Auge reagiert erst nach ungefähr 1/30 Sekunde). Es sieht daher so aus, als leuchte die Diode die ganze Zeit, wenn auch etwas schwächer.

Will man die Diode wirklich blinken sehen, muß man den Prozeß verlangsamen. Normalerweise verwendet man eine FOR...NEXT-Schleife, um eine Verlängerung zu erreichen. Jede Runde der Schleife benötigt ca. 1 ms. Soll die Diode 1/2 Sekunde an und 1/2 Sekunde aus sein, werden zwei FOR-...NEXT-Schleifen gebraucht, jede 500 Runden lang. Beim VC-20 kann das so aussehen:

```
10 POKE 37138,255     (8 Ausgänge)
20 POKE 37136,1       (Diode zum Leuchten bringen)
30 FOR I=1 TO 500     (1/2 Sekunde warten)
40 NEXT I
50 POKE 37136,0       (Diode löschen)
60 FOR I=1 TO 500     (1/2 Sekunde warten)
70 NEXT I
80 GOTO 20            (Zurückspringen und wieder
                       einschalten)
```

Jetzt blinkt die Diode einmal je Sekunde. Versuchen Sie auch andere Verzögerungen zu erreichen.

Schalter simulieren

Wenn Sie an die Anwender-Schnittstelle nichts anschließen und PB0-PB7 of-

fen lassen, deuten die E/A-Kanäle dies als »Einsen«. Das kommt daher, daß PB0-PB7 intern über hochohmige Widerstände an +5V angeschlossen sind. Probieren Sie folgendes VC-20-Programm aus:

```
10 POKE 37138,0          (8 Ausgänge)
20 A=PEEK(37136)         (A wird der Wert der 8 Eingänge
                          zugeordnet)
30 PRINT A               (Schreibt den Wert von A auf den
                          Bildschirm)
40 GOTO 20
```

Das Programm schreibt ständig den Dezimalwert der binären Zahl auf den Bildschirm, die in Schnittstelle B eingegeben wird. Sind alle Eingänge offen, wird 255 geschrieben, d. h. 11111111_2 binär. Wird stattdessen PB0 auf Masse gelegt, wird 254 angezeigt. Wird PB7 auf Masse gelegt, erscheint 127. Werden alle Eingänge auf Masse gelegt, so erscheint natürlich 0. Probieren Sie bitte einige Kombinationen aus!

Was macht man, wenn nur ein einziger Eingang interessant ist? Der VC-20 und C-64 haben einen Satz logischer Funktionen, die für diese Aufgabe wie maßgeschneidert sind. Die logische AND-Funktion kann zum Maskieren nicht erwünschter Informationen verwendet werden. Nehmen wir an, es seien drei Schalter angeschlossen:

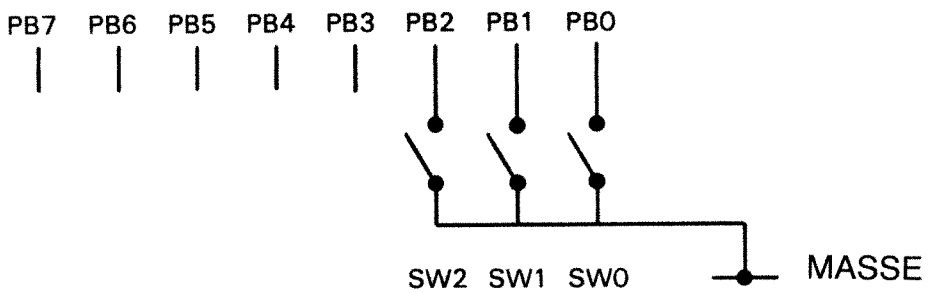


Abb. 2.9

Nehmen wir an, es interessiere nur die Stellung des Schalters SW2, angeschlossen an PB2. Der Computer soll alle Information der Schnittstelle B mit dem PEEK-Befehl abfragen (d. h. den Status aller acht Leitungen). Die sieben nicht gebrauchten Bits werden mit entsprechenden Maskenbits und einer AND-Funktion weggefiltert.

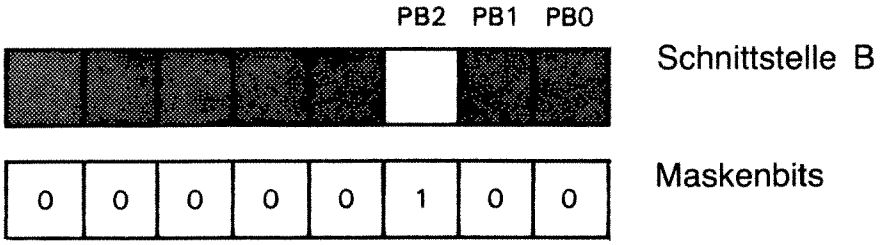


Abb. 2.10

Die AND-Funktion erzeugt Nullen in allen Positionen, in denen das Maskenbit Null war. In der Position, in der das Maskenbit Eins war, erscheint die ursprüngliche Information der Schnittstelle B, d. h. 0 oder 1. In obenstehendem Beispiel ist die Maske 00000100 (binär) = 4 (dezimal).

So sieht das Programm für den VC-20 aus (für den C-64 müssen die Adressen entsprechend geändert werden):

```
10 POKE 37138,0          (8 Eingänge)
20 A=PEEK(37136)
30 B=A AND 4
40 PRINT B
50 GOTO 20
```

Wenn SW2 offen ist, wird 4 auf den Bildschirm geschrieben, wenn SW2 geschlossen ist, wird 0 geschrieben. Die anderen Eingänge beeinflussen das Ergebnis nicht. Man kann sich die Maske als Pappscheibe mit einem »Guckloch« für die Einsen vorstellen. Nur die Bits, die man durch das Loch sehen kann, beeinflussen das Ergebnis mit ihrer entsprechenden Wertigkeit.

KAPITEL 3

Regeln mit dem Mikrocomputer

VC-20 C-64

Programmschaltwerk

Für die sequentielle Steuerung werden oft elektromechanische Programmschaltwerke gebraucht. Man findet sie z. B. in Haushaltsmaschinen (Geschirrspülautomaten und Waschmaschinen) und in industriellen Prozeßsteuersystemen. Eine Schaltuhr verwendet man häufig, um potentielle Einbrecher glauben zu machen, daß der Wohnungsinhaber zu Hause sei. Z. B. wird eine Stehlampe um 19.15 Uhr ein- und um 22.35 Uhr wieder ausgeschaltet, auch wenn sich der Wohnungsbesitzer im Süden in der Sonne aalt. Eine solche Schaltuhr ist ein einfaches elektromechanisches Programmschaltwerk.

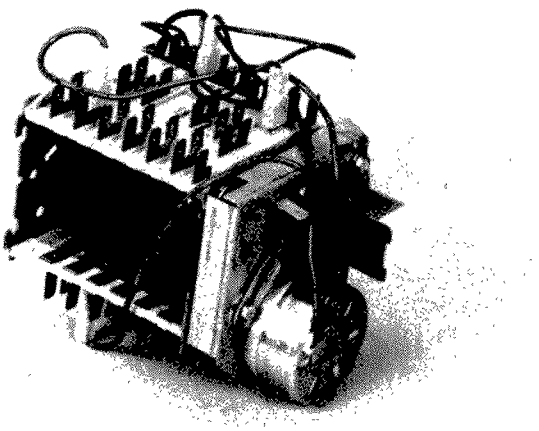


Abb. 3.1 Elektromechanisches Programmschaltwerk

Das Programmschaltwerk in z. B. einer Waschmaschine besteht aus einem Synchronmotor und einem Getriebe, das die Drehzahl der Motorwelle auf ca. 1 Umdrehung je Stunde herabsetzt. Auf der Welle sitzen eine Reihe von Nocken oder Kammscheiben, die während der langsamen Drehung der Welle Mikroschalter ein- und ausschalten. Die Mikroschalter steuern, nach einem durch die Kammscheiben festgelegten Programm, verschiedene Motoren, Pumpen, Heizelemente und Ventile in der Maschine. Normalerweise kann man auch die Drehung des Programmschaltwerks unterbrechen, bis eine bestimmte Bedingung erfüllt ist (z. B. bis ein Fühler bestätigt, daß die Maschine wirklich mit Wasser gefüllt ist).

Moderne Haushaltsgeräte enthalten Mikroprozessoren, die Relais und Motoren über Schnittstellen steuern. In der Prozeßindustrie werden Prozeßrechner für solche Steuerungsfunktionen eingesetzt. In diesem Kapitel wollen wir uns einige solcher einfacher Regelkreise ansehen, mit denen der VC-20 und C-64 nutzbringend verwendet werden kann. Wir werden uns auch eine Schnittstelle ansehen, die für die Regelung größerer Ströme und für die Erzeugung analoger Signale gebraucht wird.

Steuerung von Verkehrsampeln

Zur Regelung von Verkehrsampeln an Straßenkreuzungen wird eine Regelfolge verlangt, die eine Anzahl roter, gelber und grüner Lampen ein- und ausschaltet. In gewissen Fällen sollen Signale von Fühlern, die durch vorbeifahrende Fahrzeuge beeinflußt werden, die Regelfolge ändern. Zuerst wollen wir uns ansehen, wie der Computer eine periodische Steuerfolge erzeugen kann. Nehmen Sie hierzu möglichst rote, gelbe und grüne Dioden, die Sie an die Anwender-Schnittstelle anschließen. Auf folgende Weise wollen wir eine Verkehrsampel simulieren.

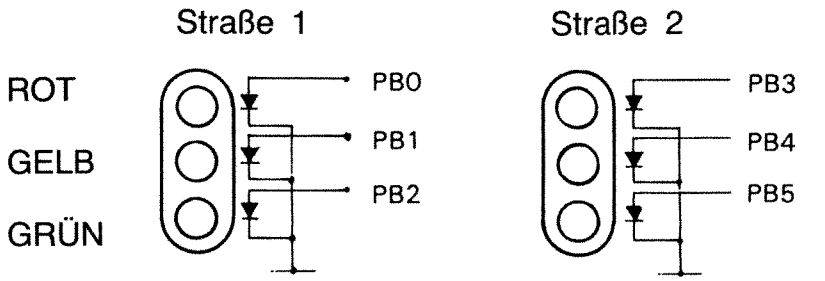


Abb. 3.2 Verkehrsampel

Die Verkehrsampel wird nach Abb. 3.3 periodisch mit einem Zeitintervall von 20 Sekunden gesteuert.

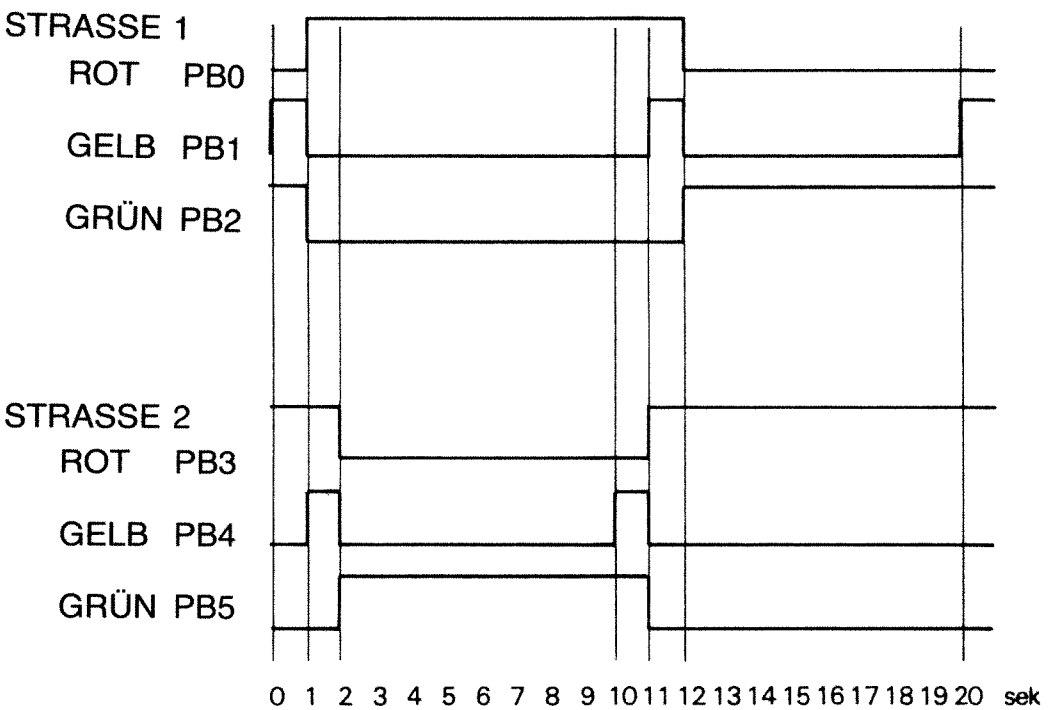


Abb. 3.3 Steuerfolge einer Verkehrsampel

Als erstes muß festgestellt werden, welche Signalkombinationen in jedem Zeitintervall erforderlich sind. Aus Abb. 3.3 ergibt sich folgende Tabelle:

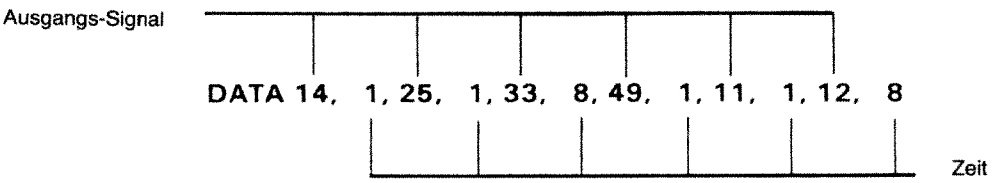
Zeitintervall (Sekunden)	Ausgangssignal an der Anwender-Schnittstelle							
	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0--1	0	0	0	0	1	1	1	0
1--2	0	0	0	1	1	0	0	1
2-10	0	0	1	0	0	0	0	1
10-11	0	0	1	1	0	0	0	1
11-12	0	0	0	0	1	0	1	1
12-20	0	0	0	0	1	1	0	0

Da wir Daten mit dem POKE-Befehl ausgeben, müssen die Dezimalwerte der Binärzeichen bestimmt werden (siehe Anhang C).

Zeitintervall (Sekunden)	Dezimales Ausgangssignal
0--1	14
0--2	25
2-10	33
10-11	49
11-12	11
12-20	12

Die nötige Zeitverzögerung kann einfach mit FOR...NEXT-Schleifen (vgl. Kapitel 2) erzielt werden.

Das Programm soll also den Computer die gewünschten Signale über Schnittstelle B ausgeben lassen, etwas warten, erneut Daten ausgeben, warten, usw. Am elegantesten macht man das, wenn die notwendige Information (Kombination der Ausgangssignale und Zeiten) in DATA-Zeilen enthalten ist. Mit einem READ-Befehl kann der Computer die Daten einlesen und die erforderlichen Maßnahmen ergreifen. Man kann z. B. die DATA-Zeile so schreiben, daß jede zweite Eingabe ein Aus-Signal und jede zweite Eingabe eine Zeiteingabe ist, z. B. so:



Vorschlag für ein Programm auf dem VC-20 (für den C-64 müssen die Adressen geändert werden):

```
10 POKE 37138,63        (PB0-PB5 Ausgänge)
20 READ A,T             (Ausgangs-Signal, Zeit lesen)
30 IF A=12 THEN         (DATA-Zeile nochmals lesen)
   RESTORE
40 POKE 37136,A         (Einlesen)
50 FOR I=1 TO T*800     (Zeitverzögerung T Sekunden)
60 NEXT I
70 GOTO 20
80 DATA 14,1,25,33,8,
   49,1,11,1,12,8
```

Die Zeitverzögerung der Zeilen 50 und 60 wurde experimentell mit der Stoppuhr bestimmt! Finden Sie bitte heraus, wie einem Fühler in Straße 1 die Möglichkeit gegeben werden kann, die Schaltfolge zu beeinflussen.

Verstärker für größere Stromstärken

Da an den E/A-Kanälen nur wenige mA abgenommen werden dürfen, ist oft eine Verstärkerstufe notwendig. Wir wollen prüfen, wie man an einem A-Kanal Relais anschließen kann.

Relaisbetrieb

Miniatur-Relais vom Typ »Reed-Schalter« können bei einer Schaltgleichspannung von 100V 0,5A schalten, dürfen aber nur bis 10W belastet werden. Zur Steuerung der Relais-Spule werden 20mA bei 5V gebraucht, was die Leistung eines A-Kanals überschreitet. Mit einer einfachen Transistor-Verstärkerschaltung funktioniert das Ganze jedoch!

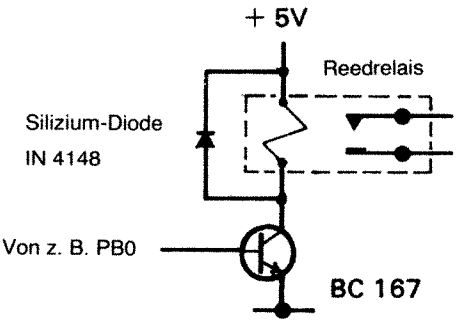


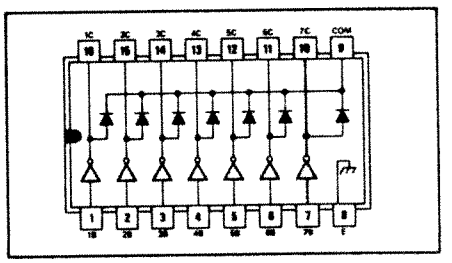
Abb. 3.4 Relaisverstärker

Die Wahl des Transistortyps ist unproblematisch, jeder npn-Transistor kleiner Leistung ist geeignet. Die Aufgabe der Diode ist es, den Transistor zu schützen, da beim Schalten der Relais-Spule Spannungsspitzen auftreten. Relais zum Schalten größerer Leistung, z. B. Netzspannung von 220V und 5A, erfordern Spulen mit größerer Leistung. Ein normales Relais kann bis zu 100mA bei 5V aufnehmen. Die Schaltung nach Abb. 3.4 funktioniert auch hier ganz ausgezeichnet. Wenn mehrere Relais gesteuert werden sollen, kann es vorteilhaft sein, einen speziellen Relaisverstärker zu verwenden. Der Baustein ULN2001A enthält 7 verschiedene Relaisverstärker (komplett mit Schutzdiode). Jede Verstärkerstufe läßt sich mit 0,5A bei 50V belasten.

TYPE ULN2001A, DARLINGTON TRANSISTOR ARRAY

- 500 mA Kollektor-Nennstrom
- Ausgangsspannung bis 50 V
- Klemmdioden am Ausgang
- Eingang kompatibel mit diversen Logiktypen
- Anwendungsgebiet: Relaisverstärker
- Austauschbar mit Bausteinen der Sprague-ULN2001A-Serien

Dual-in-line Gehäuse (Draufsicht)



Beschreibung

Die Bausteine der Reihen ULN2002A, ULN2003A und ULN2004A sind monolithische Darlington-Transistor-Arrays für hohe Spannungen und hohe Ströme. In jedem Baustein sind sieben npn-Darlington-Transistor-Paare enthalten. Alle Einheiten haben Hochspannungsausgänge mit Klemmdioden (gemeinsame Kathode) zum Schalten induktiver Lasten. Der Kollektorenstrom für jedes Darlington-Paar liegt bei 500 mA. Ein- und Ausgänge können für höhere Ströme jeweils parallel geschaltet werden. Anwendungsgebiete sind u. a. Relaisverstärker, Hammertreiber, Lampentreiber, Displaytreiber (LED und Gasentladung), Leistungstreiber und Logikbuffer. Bausteine für 100 V (ansonsten kompatibel) siehe bei SN75466 bis SN75469.

ULN2001A ist ein Universalbaustein, geeignet für DTL, TTL, P-MOS, CMOS usw. Der ULN2002A ist insbesondere ausgelegt für die Verwendung mit 14-25 V P-MOS Bausteinen. Hinter jedem Eingang liegen eine Zenerdiode und ein Widerstand, um den Eingangsstrom sicher zu begrenzen.
Im ULN2003A gibt es einen Basis-Vorwiderstand für jedes Darlington-Paar. Damit wird direkter Betrieb mit TTL oder 5 V CMOS möglich. ULN2004A hat einen geeigneten Eingangs-Vorwiderstand, um den direkten Betrieb mit CMOS oder P-MOS unter Nutzung der Speisespannung von 6-15 V zu ermöglichen. Der benötigte Eingangsstrom liegt niedriger als der für ULN2003A, während die benötigte Eingangsspannung kleiner ist als die für ULN2002A.

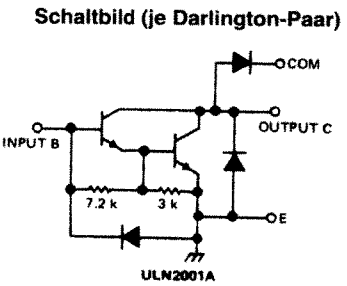


Abb. 3.5 ULN 2001 A Verstärker

Halbleiterrelais

Ein Halbleiterrelais besteht aus einem Optokoppler und einem Triac (oft kommt eine elektronische Form der Nulldurchgangsregelung hinzu). In bezug auf den Steuereingang stellt das Halbleiterrelais ganz einfach eine Leuchtdiode dar. Wenn die Diode leuchtet, wirkt das Licht auf einen Phototransistor ein, der über die Nulldurchgangssteuerung den Triac leitend macht. Beachten Sie, daß dieser Relais-Typ sich nur für Wechselstrom eignet! Das kommt daher, daß ein Triac nur geschaltet werden kann, wenn die Spannung über ihn Null wird, also zwei Mal je Periode. Triacs eignen sich besonders für das Schalten von Netzlasten. Sie verkraften zum einen große Stromstärken, zum anderen hat das Fehlen beweglicher Teile eine große Zuverlässigkeit zur Folge. Das Halbleiterrelais kann auf zwei Arten an den Ausgabe-Kanal angeschlossen werden, anhängig vom logischen Pegel des Ausgangssignals. Der Ausgangskanal kann selbst zwar nur mit wenigen mA belastet werden, kann aber die Steuerung von Strömen bei ca. 10 mA bewirken.

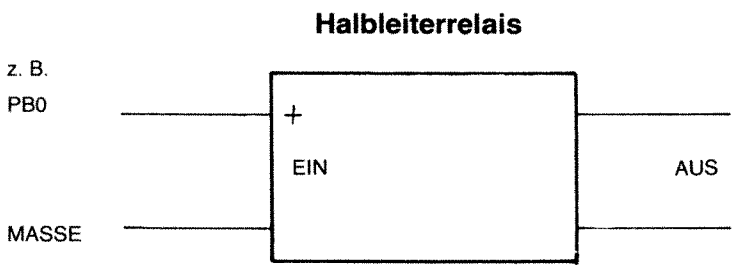


Abb. 3.6a Der Aus-Kanal schaltet das Halbleiter-Relais

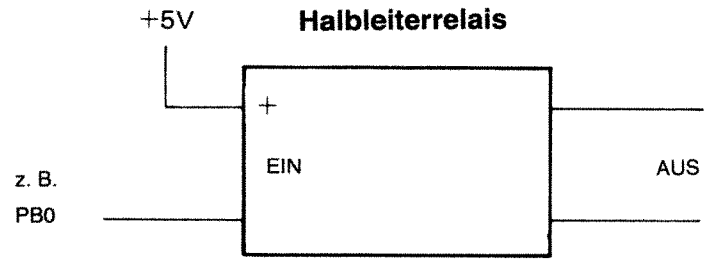


Abb. 3.6b Der Ausgabe-Kanal schaltet den Strom des Halbleiter-Relais ab

Automatische Telefonanwahl

Eine normale Wählscheibe eines Telefons unterbricht beim federgesteuerten Rücklauf einen Stromkreis N-mal, wenn die gewählte Ziffer N ist. Bei einer Neun entstehen also 9 Unterbrechungen, und bei einer Null sind es 10. Die Rücklaufgeschwindigkeit wird durch einen Zentrifugalregler bestimmt, der eine Periodenzeit von 100 ms einhält. Die Unterbrechungszeit beträgt hierbei 60 ms.

Die Zahl »Vier« ergibt folgende Taktfolge:

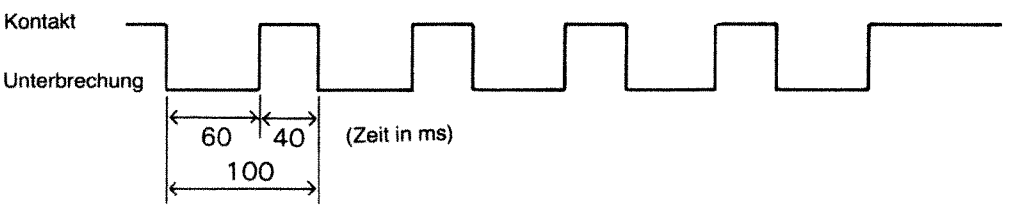


Abb. 3.7

Es sind weder Periodenzeit noch Unterbrechungszeit sonderlich wichtig, Abweichungen bis ±10% können die Vermittlungsstellen ausgleichen.

Man kann den Computer natürlich so programmieren, daß eine Taktfolge auf PB0 wie die obige entsteht. Die Zahlen können in eine Tabelle im Speicher geschrieben werden. Der Computer kann daher sowohl für die Ablage von Telefonnummern als auch für das Anwählen einer Nummer benutzt werden.

Sie wissen bereits, wie man Takte einer gewissen Zeitdauer erzeugt. Die Schwierigkeiten mit diesem Programm werden vermutlich mit dem Speichern und Lesen der Nummerninformation entstehen. Das mit Sicherheit einfachste Verfahren besteht darin, die Nummern Ziffer für Ziffer in DATA-Zeilen abzulegen. Z. B. wird die Nummer 1234567 DATA 1,2,3,4,5,6,7 geschrieben. Weiter braucht man einen READ-Befehl, um die Daten zu lesen und eine Subroutine, welche die Takte erzeugt.

Wir definieren »Kontakt« als »Null« und »Unterbrechung« als »Eins« im Ausgangs-Signal von PB0. Man könnte die Wählscheibe durch ein Relais am PB0 ersetzen, das verbietet jedoch die Post. Wir wollen daher das Ausgangssignal von PB0 mit einer Leuchtdiode überprüfen.

```
10 POKE 37138,1          (PB0 Ausgang)
20 READ N                (Ziffer lesen)
25 IF N=0 THEN N=10
30 FOR I=1 TO N
40 GOSUB 80              (Taktlänge erzeugen)
50 NEXT I
60 END
70 DATA 1,2,3,4,5,6,7    (Telefonnummer)
80 POKE 37136,1          (Eine »Eins« ausgeben)
90 FOR J=1 TO 60:        (60 ms warten)
   NEXT J
100 POKE 37136,0         (Eine »Null« ausgeben)
110 FOR J=1 TO 40:       (40 ms warten)
    NEXT J
120 RETURN               siehe Anhang D
```

Die Subroutine für die Ausgabe einer einzigen Taktlänge beginnt in Zeile 80. Die FOR...NEXT-Schleife in den Zeilen 30-50 erzeugt N Takte. Wiederholen Sie den Versuch mit einer anderen Telefonnummer in der DATA-Zeile.

Eine raffiniertere Variante des Programms wäre die Verwendung einer Telefonnummer, die als »String« abgespeichert ist. Mit Hilfe der Stringfunktionen (LEFT$, MID$, LEN usw.) können die einzelnen Ziffern der Telefonnummer gelesen und die entsprechenden Taktfolgen erzeugt werden. Dadurch entstehen ungeahnte Möglichkeiten, die automatische Nummernwahl über ein auf Diskette gespeichertes Namen-/Nummernverzeichnis zu steuern. Interessante Möglichkeiten bieten sich da für Ihre private Telefonanlage! Den

Anschluß an das öffentliche Telefonnetz muß natürlich die Post erst genehmigen! Verwenden Sie nun Ihre Kenntnisse im Programmieren und Einsetzen von Datenbanken und schreiben Sie ein Programm. Das elektronische Telefonbuch muß man natürlich auch ändern können.

Erzeugung analoger Signale

Es lassen sich leider nicht alle peripheren Einheiten mit einfachen Ein-/Aus-Signalen schalten. Will man z. B. die Drehzahl eines Gleichstrommotors steuern, muß ein analoges Signal zur Verfügung stehen, d. h. eine stetig veränderliche Spannung. Da der Computer an den E/A-Kanälen digitale Signale, also »Einsen« und »Nullen« erzeugt, ist eine Schnittstelle zur Umwandlung des Signals, ein D/A (digital/analog) -Wandler erforderlich.

Widerstandsnetzwerk

Die meisten D/A-Wandler verwenden das Prinzip der Summenbildung binär gewichteter Ströme, wo die Größe der Teilströme von der Wertigkeit der Binärzahl abhängt. Z. B. in einem 8-Bit D/A-Wandler läßt man das Bit mit dem niedrigsten Stellenwert die Stromquelle ansteuern, die z. B. 1 μA bei »Eins« abgibt (und 0 μA bei »Null«). Das nächste Bit gibt 2 μA, das nächste 4 μA usw. Das Bit mit dem höchsten Stellenwert ergibt 128 μA bei »Eins«. Die Ströme werden summiert und ergeben das Ausgangs-Signal des D/A-Wandlers. Die Schaltung nach Abb. 3.8 enthält Relais als Schalter, gesteuert von PB0-PB7.

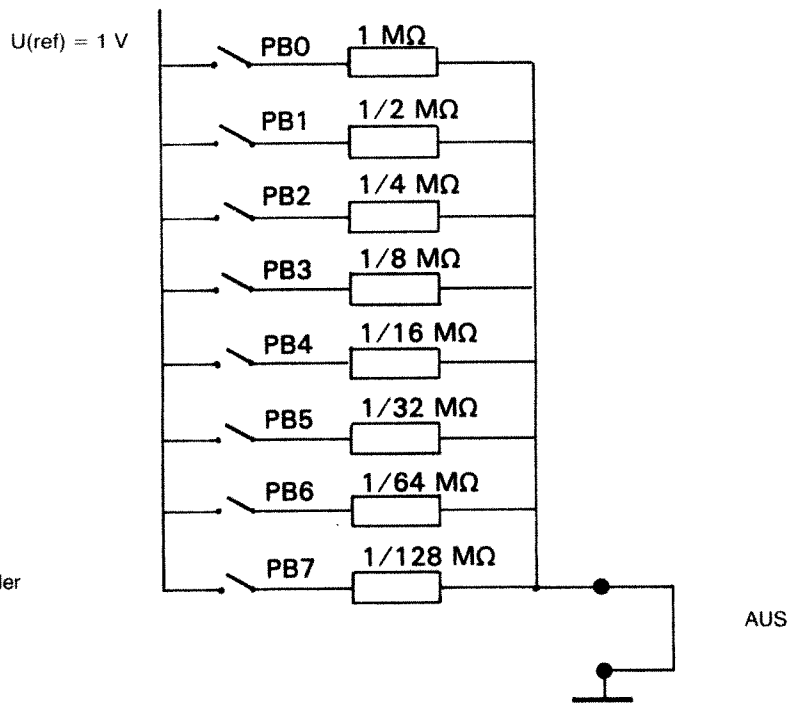


Abb. 3.8 D/A-Wandler

Die Stromquelle besteht aus geschalteten, binär gewichteten Widerständen und einer konstanten Referenz-Spannung. Die ausgehende Stromstärke kann alle Ganzzahlwerte von 0 bis 255 annehmen. Die Leerlaufspannung am Ausgang ist ebenfalls dem binären Eingangs-Signal proportional. Ein 8-Bit D/A-Wandler hat eine Auflösung von $1/256$, d. h. 0,4 %. Die verwendeten Widerstände müssen natürlich entsprechend genau sein. In der Praxis werden nicht Relais genommen, sondern Halbleiter-Schalter. D/A-Wandler kann man natürlich als IC kaufen und in den Computer einbauen. Ein 8-Bit Wandler kann für ein paar Mark gekauft werden. Es wird sich kaum lohnen, ihn selbst zu bauen. Im nächsten Abschnitt werden wir übungshalber einen D/A-Wandler aufbauen, vorher sehen wir uns jedoch einen handelsüblichen IC mit der Bezeichnung AD 7523 an.

AD 7523

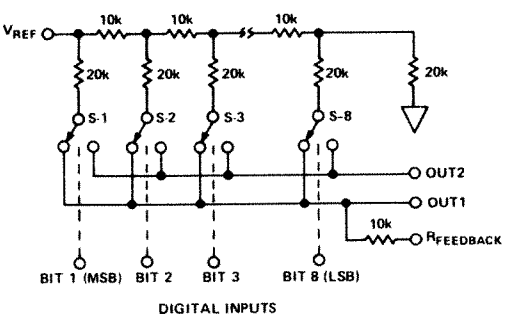
Der AD 7523 von Analog Devices ist ein typisches Beispiel für einen integrierten D/A-Wandler. Im Inneren wird ein Widerstandsnetzwerk zur Stromerzeugung benutzt, die Steuerung erfolgt über eingebaute Halbleiterschalter. Das Widerstandsnetzwerk sieht etwas anders aus. Es ist ein sogenannter R/2R-Leiter, bei dem der Strom in jedem Knotenpunkt geteilt wird. Dadurch ergeben sich binär gewichtete Teilströme, Abb. 3.9.

AD 7523 DIGITAL/ANALOG-WANDLER

EIGENSCHAFTEN
Niedrige Kosten
Kurze Setzzeit: 100 ns
Niedrige Verlustleistung
Kleiner Wandlungsfehler:
½ LSB bei 200 kHz
Volle vier-Quadranten Multiplikation

ANWENDUNGEN
Batteriegetriebene Geräte
Niedrige Verlustleistung, radiometrische A/D Wandler
Digital steuerbare Verstärker-Schaltkreise
Digital gesteuerte Dämpfungsglieder
CRT-Zeichenerzeugung (CRT = Kathodenstrahlröhre)
Rauscharme NF-Verstärkersteuerung

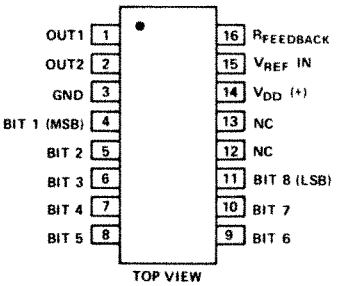
Funktionsschema

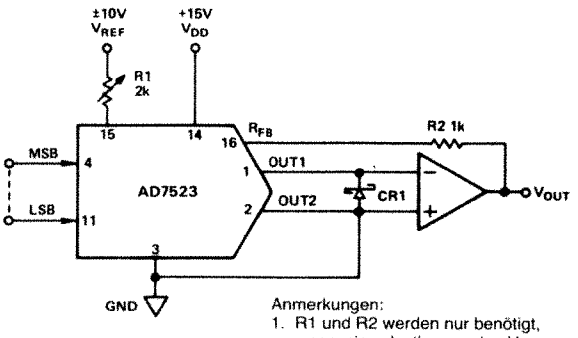


ALLGEMEINE BESCHREIBUNG

Der AD7523 ist ein preisgünstiger, monolithischer und verstärkender Digital/Analog-Wandler in einem 16-pin DIP (Dual-in-line-Gehäuse). Für den Baustein wird eine fortschrittliche Einkristall-, Dünnschicht-CMOS Technologie benutzt, die eine 8-Bit Auflösung mit einer Genauigkeit von 10 Bit bei sehr geringer Verlustleistung ermöglicht.

Die hervorragende Verstärkercharakteristik und die geringen Kosten des AD7523 erlauben seine Verwendung in unterschiedlichsten Einsatzbereichen, z. B. Steuerung rauscharmer NF-Verstärker, CRT-Zeichengenerierung, Steuerung von Motordrehzahl, digital gesteuerte Dämpfungsglieder usw.





Anmerkungen:
1. R1 und R2 werden nur benötigt, wenn eine Justierung der Verstärkerleistung notwendig ist.
2. CR1 schützt AD7523 vor negativen Spitzenspannungen.

DIGITAL INPUT MSB LSB	ANALOG OUTPUT
11111111	$-V_{REF}\left(\frac{255}{256}\right)$
10000001	$-V_{REF}\left(\frac{129}{256}\right)$
10000000	$-V_{REF}\left(\frac{128}{256}\right) = -\frac{V_{REF}}{2}$
01111111	$-V_{REF}\left(\frac{127}{256}\right)$
00000001	$-V_{REF}\left(\frac{1}{256}\right)$
00000000	$-V_{REF}\left(\frac{0}{256}\right) = 0$

Note: $1LSB = (2^{-8})(V_{REF}) = \frac{1}{256}\ (V_{REF})$

Abb. 3.9 AD 7523 Digital/Analog-Wandler

Veränderung der Pulsbreite

Ein ganz anderer Weg, ein analoges Signal zu erzeugen, besteht darin, von einer Pulsfolge mit konstanter Frequenz auszugehen. Die Pulsbreite muß proportional zu der umzuwandelnden Zahl sein. Nach Bilden des Mittelwertes in einem RC-Filter wird eine Gleichstrom-Komponente erhalten, die der Pulsbreite proportional ist und daher auch proportional zur ursprünglichen Zahl ist.

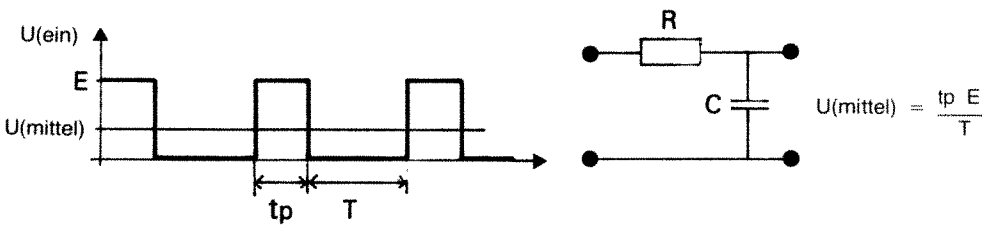


Abb. 3.10 D/A-Wandlung mit Veränderung der Pulsbreite

Mit einem Klein-Computer ist es einfach, eine Pulsfolge zu erzeugen. Die einzigen extern erforderlichen Komponenten sind ein Widerstand und ein Kondensator. Der Nachteil der Methode besteht darin, daß es verhältnismäßig lange dauert, das Ausgangssignal zu verändern. Mit einem Basic-Programm ist es schwierig, eine Frequenz zu erzeugen, die höher als einige Hertz ist, wenn die Pulsbreite in 256 Schritten (8 Bits) veränderlich sein soll. Die Zeitkonstante des RC-Filters sollte wenigstens 20 bis 30 Mal größer sein als die Periodenzeit. Das Ergebnis wird daher ein Wandler sein, der bis zu einer Minute braucht, um sein Ausgangssignal vom Mindest- zum Höchstwert zu verändern. Wir werden später sehen, daß ein Programm in Maschinensprache das Problem der langen Reaktionszeit lösen kann.

Steuerung von Gleichstrom-Motoren

In diesem Abschnitt wollen wir uns ansehen, wie ein einfacher 4-Bit D/A-Wandler gebaut werden kann, der mit einem Verstärker zur Steuerung eines kleinen Gleichstrom-Motors dient. Ein 4-Bit D/A-Wandler kann $2^4 = 16$ Ebenen ergeben und eine Auflösung von 1/16, d. h. 5 %. Diese verhältnismäßig geringe Anforderung an die Genauigkeit hat zur Folge, daß wir ein Widerstandsnetzwerk verwenden können, das direkt vom Ausgangs-Kanal angesteuert wird. Es werden also keine Schalter benötigt! Andererseits müssen die Widerstände ausreichend hochohmig sein, um eine übermäßige Belastung der Ausgangs-Kanäle zu vermeiden. Es eignen sich Widerstände, die über 100 Ohm liegen.

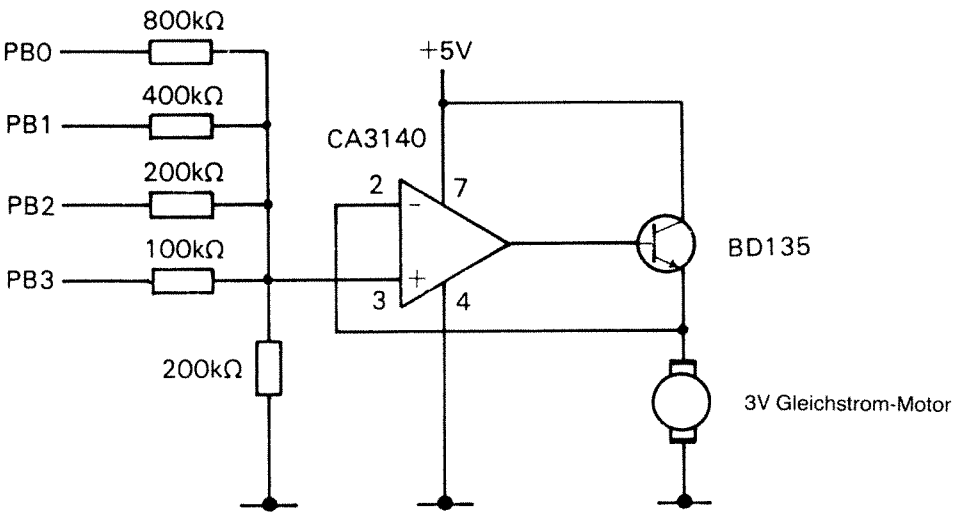


Abb. 3.11 Steuerung eines Gleichstrom-Motors

Die Spannung +E darf nicht dem Computer entnommen werden, da die Stromabnahme auf 100 mA begrenzt ist. Ein kleiner Gleichstrom-Motor (Spielzeugmotor) zieht jedoch mindestens 0,5 A. Der Motor sollte daher von einer 4,5V Taschenlampen-Batterie oder einem externen 5V Netzteil gespeist werden.

Operationsverstärker und Leistungstransitoren wirken als Leistungsverstärker mit der Spannungsverstärkung 1. Der 200 kOhm Widerstand nach Masse hin soll den D/A-Wandler belasten, so daß die am Operationsverstärker anliegende Spannung von 3V nicht überschritten wird. Um den Motor laufen zu lassen, ist für den VC-20 ein nur zweizeiliges Basic-Programm erforderlich. (Für den C-64 müssen dementsprechend die Adressen geändert werden.)

```
10 POKE 37138,31      (PB0-PB3 Ausgang)
20 POKE 37136,15      (Drehzahl)
```

Wenn Sie dieses Programm laufen lassen, sollte der Motor mit maximaler Drehzahl laufen. Wird die Zahl 15 auf 10 geändert, läuft der Motor bedeutend langsamer. Probieren Sie es aus!

Soll der Motor in einer bestimmten Folge gesteuert werden, kann das Ampel-Programm aus einem vorhergehenden Abschnitt dazu verwendet werden.

KAPITEL 4

Messen mit dem Mikrocomputer

VC-20 C-64

Mengenmessung mit einer Fotozelle

Eine der einfachsten Formen der Messung ist ohne Zweifel die Mengenmessung. Aus Sicht des Computers stellt sich Mengenmessung so dar wie das Festhalten der Anzahl Wechsel zwischen »0« und »1« am Eingangs-Kanal.

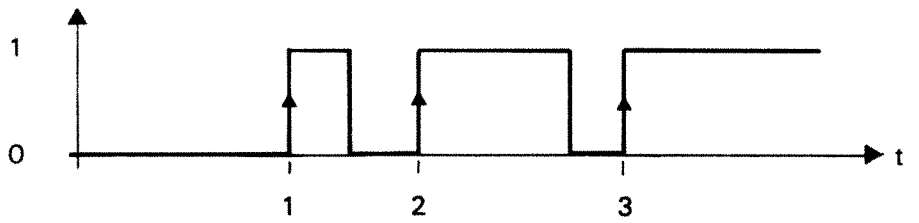


Abb. 4.1 Mengenmessung

In diesem Abschnitt wollen wir uns ansehen, wie eine Fotozelle an der Anwender-Schnittstelle angeschlossen wird. Sie wird dazu verwendet, festzustellen, wie oft ein Lichtstrahl unterbrochen wird. Das Prinzip wird oft verwendet, besonders in der Prozeßautomatisierung. Je nach Stärke der Lichtquelle kann die Schnittstelle verschiedenartig ausgelegt sein. Mit einer starken, gut auf die Fotozelle ausgerichteten Lichtquelle braucht man nur einen Geber, einen Fototransistor, der an PB6 angeschlossen wird.

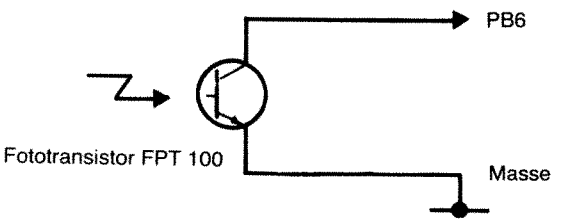


Abb. 4.2 Fototransistor

Wenn Licht auf das Lichtfenster des Fototransistors trifft, wird PB6 gleich »0«, und wenn der Lichtstrahl unterbrochen wird, wird PB6 gleich »1«. Bei niedrigerer Lichtstärke muß das Signal des Fototransistors verstärkt werden, bevor es an die Schnittstelle angeschlossen wird. Auch in diesem Fall wird das Ergebnis »0« bei Belichtung des Fototransistors und »1« bei Abdunklung, Abb. 4.3.

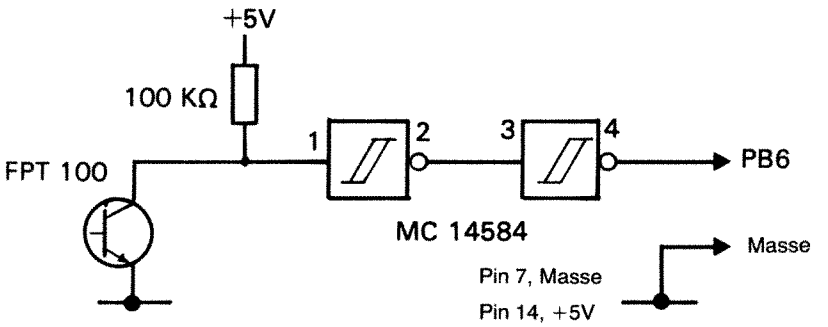


Abb. 4.3 Fototransistor mit Verstärker

Basic-Programm

Nun sehen wir uns ein Programm für den VC-20 an. (Für den C-64 müssen die Adressen geändert werden.)

```
10 POKE 37138,191
20 X=0
30 A=PEEK(37136) AND 64
40 IF A=0 THEN 30
50 X=X+1
60 PRINT X
70 A=PEEK(37136) AND 64
80 IF A=64 THEN 70
90 GOTO 30
```

Hier eine Erläuterung des Programms:

Zeile 10: Schaltet PB6 als Eingangs-Kanal

Zeile 20: X wird als Rechenvariable benutzt, d. h. X erhöht sich jedesmal, wenn PB6 »Eins« wird.

Zeile 30: PEEK(37136) mißt den Status an allen PB. AND 64 maskiert alle PB außer PB6. Wenn PB6=0, dann wird A=0, wenn PB6=1, dann wird A=64.

Zeile 40: Wenn A=0, bedeutet das PB6=0, d. h. der Lichtstrahl ist noch nicht unterbrochen worden. Man soll die Variable X nicht ausrechnen, sondern zurückgehen und erneut messen.

Zeile 50: A war nicht gleich 0 (d. h. 64), weswegen der Lichtstrahl unterbrochen worden sein muß. Rechnen Sie X aus.

Zeile 60: Wert von X auf den Bildschirm schreiben

Zeile 70: Wie Zeile 30!

Zeile 80: Wenn A=64 ist, bedeutet das, daß PB6=1 ist, d. h. der Lichtstrahl ist immer noch unterbrochen. Nach 70 zurückgehen und erneut messen, bis wieder Licht auf den Fototransistor trifft.

Zeile 90: Neuen Rechengang starten.

Da der Computer für jede Rechnung ca. 1 ms braucht, erfordert jeder Rechendurchgang ungefähr 10 ms. Die größte Rechengeschwindigkeit ist daher auf ca. 100 Takte pro Sekunde begrenzt. Sind die Anforderungen an die Rechengeschwindigkeit höher, muß ein schnellerer Rechner oder ein Maschinensprach-Programm benutzt werden. Wir kommen auf die Maschinensprache in Kapitel 5 zurück. Untersuchen wir zunächst die Unterschiede der beiden Mikrocomputer.

Rechner VC-20

Schon in einem vorherigen Kapitel hatten wir erwähnt, daß die VIA-Schnittstelle mehr als einen Kanal hat. U. a. enthält sie einen 16-Bit-Binär-Rechner, der zum Zählen der Pulse an PB6 herangezogen werden kann. Im Datenblatt des Herstellers wird der Rechner als »Timer 2« bezeichnet (es gibt aber noch einen!). Timer 2 ist in zwei 8-Bit-Hälften geteilt. Da wir mit einem 8-Bit Computer arbeiten, werden auf den Bussen 8-Bit-Daten transportiert. Sie können den Computer auf Null setzen oder den gewünschten Ausgangswert einstellen und natürlich auch ablesen. Wie gewohnt, werden hierzu POKE- und PEEK-Befehle benutzt.

Timer 2 kann auf 2 Arten verwendet werden – einmal als Zeituhr, einmal als Rechner. Sie können die Betriebsart wählen, indem Sie eine Zahl in das sogenannte Kontroll-Register schreiben.

Die acht Bits mit dem niedrigsten Stellenwert haben die Adresse 37144, die acht Bits mit dem höchsten Stellenwert haben die Adresse 37145. Das Kontrollregister hat die Adresse 37147. Bit 5 im Kontrollregister wird für die Wahl der Betriebsart verwendet. Bit 5 = »0« stellt die Betriebsart »Zeituhr«, Bit 5 = »1« die Betriebsart »Rechner« ein, Abb. 4.4.

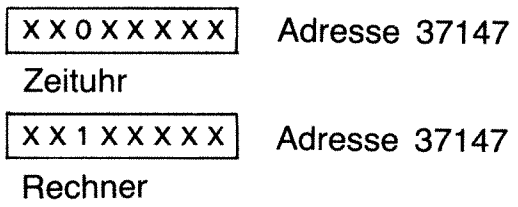


Abb. 4.4 Kontrollregister in VIA

Ein Haken bei der Sache ist, daß Timer 2 rückwärts zählt. Der Hersteller ging davon aus, daß der Baustein eine vorher festgelegte Anzahl messen sollte (die man vorher in den Rechner eingibt). Wenn der Rechner bis Null heruntergerechnet hat, erzeugt die VIA-Schnittstelle ein Signal, das an den Mikroprozessor weitergeleitet wird, der dann entsprechend reagiert. Wird der Baustein zunächst mit 65535 geladen, steht er nach dem ersten Rechentakt auf 65534, danach auf 65533 usw. bis auf Null. Die Zahl der erfolgten Takte ergibt sich aus der Differenz zwischen 65535 und der Rechnerstellung.

Um mit Timer 2 Takte zu berechnen, sind folgende Maßnahmen erforderlich:

1. Bit 5 des Kontrollregisters auf »1« stellen (Adresse 37147).
2. PB6 als Eingangs-Kanal einstellen. Dazu muß Bit 6 des Datenrichtungsregisters auf Null gesetzt werden (Adresse 37138).
3. Die 8 Bits mit dem höchsten Stellenwert voreinstellen, d. h. 255 in Adresse 37144 schreiben.
4. Die 8 Bits mit dem niedrigsten Stellenwert voreinstellen, d. h. 255 in Adresse 37145 schreiben.

Der Rechner startet, wenn Punkt 4 ausgeführt worden ist.

Wenn der Rechner bis Null heruntergerechnet hat, springt er zurück und beginnt von neuem, wie ein Kilometerzähler im Auto. Hierdurch können auch größere Zahlen als 65535 gemessen werden.

Programm:

```
10 POKE 37147, PEEK(37147) OR 32
20 POKE 37138,191
30 POKE 37144,255
40 POKE 37145,255
50 A=PEEK(37144)
60 B=PEEK(37145)
70 PRINT 65535-(256*B+A)
80 GOTO 50
```

Auch dieses Programm erfordert eine Erklärung:

Zeile 10: PEEK(37147) ergibt die Zahl, die im Augenblick im Kontrollregister steht. OR 32 hat zur Folge, daß alle Bits der Zahl unverändert bleiben mit Ausnahme von Bit 5 (Wertigkeit 32), das immer auf »1« gesetzt wird. Das Ergebnis wird im Kontrollregister abgelegt. Durch diese Maßnahme wird die Betriebsart »Rechner« eingestellt. Wir wollten zeigen, daß die übrigen Bits intern vom VC-20 gebraucht werden und eben deswegen nicht einfach eine Eins in Bit 5 geschrieben werden darf.

Zeile 20: Macht PB6 zum Eingang.

Zeile 30: Setzt die Rechnerhälfte mit dem niedrigsten Stellenwert auf binär 11111111_2 = dezimal 255_{10}.

Zeile 40: Setzt die Rechnerhälfte mit dem höchsten Stellenwert auf dezimal 255_{10} und startet den Rechner.

Zeile 50: Liest die Rechnerhälfte mit dem niedrigsten Stellenwert und legt das Ergebnis in A ab.

Zeile 60: Liest die Rechnerhälfte mit dem höchsten Stellenwert und legt das Ergebnis in B ab.

Zeile 70: (256*B+A) stellt die beiden 8-Bit-Zahlen zu einer 16-Bit-Zahl zusammen. Die Zahl mit den höchsten Stellenwert erhält die Wertigkeit 256. 65535, vermindert um diese Zahl, korrigiert für das Rückwärtsrechnen. Das Ergebnis wird auf dem Bildschirm ausgegeben.

Zeile 80: Zurückkehren und wieder den Rechner lesen.

Der Vorteil des Timer 2 ist, daß die Rechengeschwindigkeit bis 1 MHz ansteigen kann. Im Vergleich dazu stehen die 100 Hz, wenn man mit einem Programm rechnet. Eine Verzehntausendfachung der Rechengeschwindigkeit, das ist genau das, was hier gebraucht wird.

Rechner C-64

Auch der C-64 enthält einen 16-Bit-Timer-Baustein in der Schnittstelle CIA (Complex Interface Adapter). In ihm kann der Timer A gebraucht werden. Der Takt ist mit CNT 1 (Anschluß 4) zu verbinden.

Es gelten folgende Adressen:

8 Bits mit dem niedrigsten Stellenwert:	56580
8 Bits mit dem höchsten Stellenwert:	56581
Kontroll-Register:	56590

Ansonsten gelten die Regeln der VIA-Schnittstelle des letzten Abschnitts. Zeile 20 kann ausgelassen werden, weil PB6 nicht als Eingang beim C-64 benutzt wird.

Frequenzmessung

Zur Messung von Frequenzen brauchen wir eine Zeitbasis, welche die Berechnung von Takten pro Sekunde gestattet (ergibt ein Resultat in Hz). Wir haben früher schon Zeitintervalle mit FOR...NEXT-Schleifen erzeugt. Das ist jedoch eine ungenaue Methode, insbesondere bei kurzen Zeiten. Wir können nur schätzen, wie lange der Basic-Interpreter zur Übersetzung des Programms braucht, es scheint sich jedoch um einige ms je Zeile zu handeln. Es ist besser, einen Timer in der VIA-Schnittstelle zu benutzen oder die eingebaute Echtzeituhr im Computer. Letztere kann mit Basic-Befehlen abgelesen werden. Das folgende Programm benutzt PB6 als Eingang und Timer 2 zum Rechnen, genau wie im vorherigen Abschnitt. Das Programm ist für den VC-20 geschrieben (für den C-64 müssen die Adressen wie oben geändert werden, Zeile 20 kann entfallen):

```
10 POKE 37147, PEEK(37147) OR 32
20 POKE 37138,191
30 POKE 37144,255
40 T=TI
50 IF T=TI THEN 50
60 POKE 37145,255
70 IF TI<>T+60 THEN 70
80 PRINT 65535 - (256*PEEK(37145)+PEEK(37144))
90 GOTO 30
```

Kommentare zu obenstehendem Programm:

Zeile 10: Setzt Bit 5 im Kontrollregister auf »1«, um in der Betriebsart »Rechner« des Timers 2 zu starten,

Zeile 20: Macht PB6 zum Eingang.

Zeile 30: Stellt die Rechnerhälfte mit dem niedrigsten Stellenwert auf 255 ein.

Zeile 40: Liest die Echtzeituhr (in 1/60 Sekunden) und legt den Wert in T ab.

Zeile 50: Der Computer wartet hier, bis sich TI wieder ändert und macht dann weiter. Die Uhr wird ja »im Vorbeigehen« gelesen und hat eine Ableseungenauigkeit von 1/60 Sekunde, wenn nicht auf ein Zurückspringen gewartet wird.

Zeile 60: Stellt die Rechnerhälfte mit dem höchsten Stellenwert auf 255 ein.

Zeile 70: Der Computer wartet hier (zählt aber an PB6 die Takte weiter), bis insgesamt 1 Sekunde seit Beginn von Zeile 50 verstrichen ist. T+60 ergibt eine Sekunde mehr, da der Computer Sechzigstel-Sekunden mißt.

Zeile 80: Schreibt die Frequenz auf den Bildschirm.

Zeile 90: Springt zurück und führt eine neue Messung durch.

Die höchste Frequenz, die mit diesem Programm gemessen werden kann, ist 65 kHz, weil der Timer 2 nur bis 65535 zählen kann. Der Computer kann Zeiten, die unter einer Sekunde liegen, nicht genau messen. Es ergibt sich ein Fehler von einigen Millisekunden, weil der Basic-Interpreter einige Millisekunden zum Deuten der Zeilen 50, 60 und 70 braucht. Es ist nicht möglich, diesen Fehler zu vermeiden, solange die Programme in Basic geschrieben werden. Diese Sprache ist für derartige Verwendung eigentlich nicht geeignet. Wenn die Kenntnis genauer Zeiten wesentlich ist, programmiert man daher besser in Maschinensprache. Mehr darüber in Kapitel 5.

Wie auch immer – der Fehler von wenigen Promillen, die der Basic-Interpreter im obenstehenden Frequenzmeßprogramm verursacht, kann oft in Kauf genommen werden. Später werden wir Gelegenheit bekommen, das Programm anzuwenden.

Analoge Signale für den Computer

Eingangssignale für den Computer von verschiedenen Meßfühlern liegen oft in analoger Form vor. Im Normalfall muß eine Signalbehandlung durchgeführt werden, z. B. Verstärkung oder Filtern, bevor das analoge Signal verwendet werden kann. Mit der Elektronik, die heute zur Verfügung steht (z. B. Operationsverstärker) kann man davon ausgehen, daß die analogen Signale, die zur »Datenverarbeitung« anstehen, aus mehr oder weniger konstanten

Spannungen bestehen. Ehe der Computer die Daten weiterverarbeiten kann, müssen sie digitalisiert werden. Das wird mit einem A/D-Wandler (Analog-/Digital-Wandler) erreicht. Das Ausgangssignal des A/D-Wandlers besteht aus »Einsen« und »Nullen«, die der Eingangskanal des Computers verarbeiten kann.

Grundlagen der A/D-Wandler

Die A/D-Wandler, die zur Datensammlung für Computer benutzt werden, arbeiten hauptsächlich nach einem der folgenden Prinzipien: integrierend, nach dem Wägeverfahren, sukzessive Approximation und Spannungs-/Frequenz-Wandlung. Diese Arten der A/D-Wandler können als fertige Bausteine gekauft werden. Es müssen allerdings einige Widerstände und Kondensatoren angeschlossen werden, bevor der A/D-Wandler funktioniert.

Der integrierende A/D-Wandler

Ein integrierender A/D-Wandler ist eigentlich ein Spannungs-/Zeit-Wandler, in dem die Zeit mit einem Rechner und einer eingebauten Uhr gemessen wird. Die gemessene Zeit ist der »unbekannten« Spannung proportional. Dieser Typ eines Wandlers integriert das Eingangssignal mit Hilfe eines Operationsverstärkers und vergleicht es mit einer Referenzspannung. Es dauert verhältnismäßig lange, eine Messung durchzuführen, nämlich ca. 1/10 Sekunde. Der Vorteil dabei ist, daß eventuelle Störungen im Eingangssignal in den Mittelwert eingehen und der Einfluß daher abgeschwächt wird. Besonders wichtig ist dies, wenn man lange, nicht abgeschirmte Leitungen hat, die das 50-Hz-Brummen des Leitungsnetzes auffangen. Der Einfluß solcher Störungen kann ganz vermieden werden, wenn die Integrationszeit ein Vielfaches der Periodendauer der Netzfrequenz ist, z. B. 1/20 Sekunde, 1/10 Sekunde usw. Die meisten Digital-Voltmeter enthalten einen integrierenden A/D-Wandler.

A/D-Wandler (Prinzip der sukzessiven Approximation)

Ein A/D-Wandler, der nach dem Wägeverfahren arbeitet, benutzt einen D/A-Wandler, gesteuert durch digitale Elektronik und einen Komparator. Der Komparator vergleicht die »unbekannnten« Eingangssignale mit Spannungen des D/A-Wandlers, und die digitale Elektronik wird vom Ausgangssignal des Komparators gesteuert. Als Endergebnis ist das Ausgangssignal des D/A-Wandlers gleich der unbekannten Spannung. Das digitale Signal zum D/A-Wandler entspricht der unbekannten Spannung. Das digitale Signal zum D/A-Wandler ist gleich dem gesuchten digitalen Äquivalent der unbekannten Spannung.

Eine solche Umwandlung nimmt nur kurze Zeit in Anspruch. Normale Umwandlungszeiten für 8-Bit Wandler sind 1 ms bis 100 ms. Es ist wichtig, daran zu denken, daß Störungen großen Einfluß auf diesen Typ der Umwandlung haben, da die Mittelwertbildung entfällt. Lösungen bieten sich durch Filtern und Begrenzen der Bandbreite an.

A/D-Wandler vom Typ V/f (Spannungs-Frequenz-Wandler)

Der dritte Typ der A/D-Wandler, der V/f-Wandler oder Spannungs-Frequenz-Wandler, ist eigentlich ein spannungsgesteuerter Oszillator. Das Ausgangssignal ist eine Taktfolge, deren Frequenz der angelegten »unbekannten« Spannung proportional ist. Da der Mikroprozessor die Frequenz der Taktfolge mißt, erhält man ein Maß für die Spannung. V/f-Wandler haben verhältnismäßig lange Umwandlungszeiten, 1-10 Sekunden. Sie haben aber auch viele Vorteile, u. a. bilden sie Mittelwerte und vermindern dadurch den Einfluß von Störungen. Weiter ist nur ein einziger Draht für das Ausgangssignal erforderlich. Der Wandler kann daher sehr einfach mit einem Optokoppler galvanisch vom Computer getrennt werden. Die V/f-Wandler sind außerdem billig. Im nächsten Abschnitt wollen wir uns einen V/f-Wandler in einem IC, den LM 331, näher ansehen.

V/f-Wandlung mit LM 331

Mit dem LM 331 entsteht ein linearer und genauer V/f-Wandler. Ein Teil der Widerstände und Kondensatoren wird zusätzlich beim Aufbau auf der Experimentierplatine benötigt.

Schaltbild

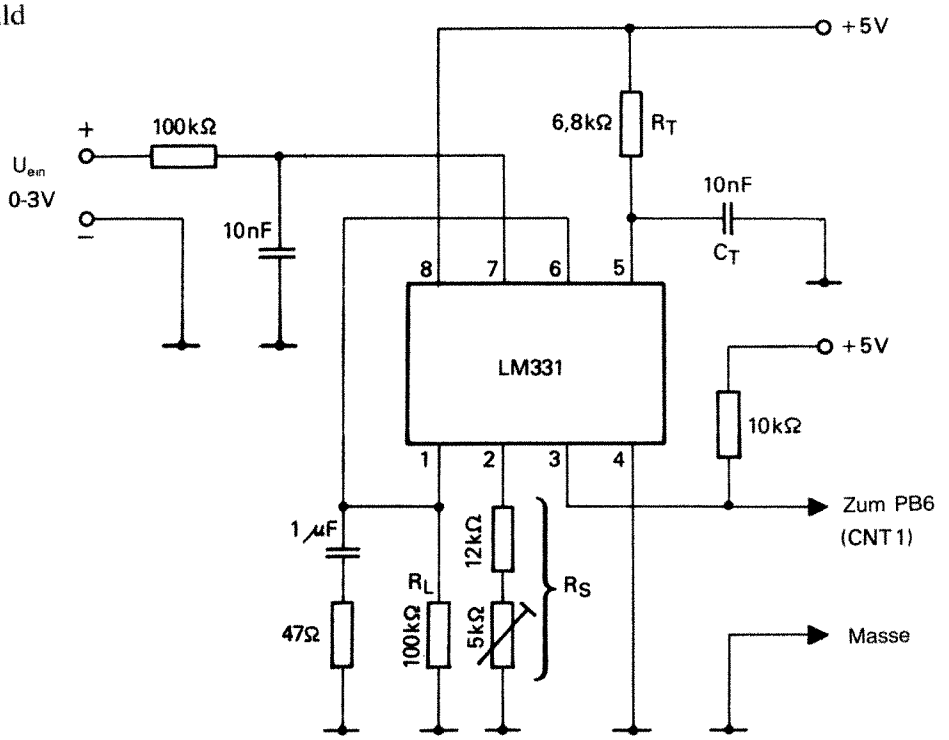


Abb. 4.5 V/f-Wandler mit LM 331

Für das obenstehende Schema gilt die Beziehung:

$$f = \frac{U_{ein} \times R_S}{2{,}09 \times R_L \times R_T \times C_T}$$

f ist die Frequenz des Ausgangssignals in Hz, U(ein) ist die Spannung des Eingangssignals in V. R_S, R_L und R_T werden in Ohm gemessen und C_T in F. Für R_S, R_L und $R_T$sollten nur Metallfilmwiderstände mit einer Toleranz von 1 % genommen werden. Für die Kondensatoren sollten nur Polyesterkondensatoren eingebaut werden. Besonders für C_T muß gute Qualität gewählt werden, weil er in den die Frequenz bestimmenden Teil der Formel eingeht. Das Abstimmpotentiometer wird für die Feinabstimmung bei der Eichung des Instruments gebraucht. Mit den angegebenen Werten für die Bauteile ist die Empfindlichkeit des Geräts 1 kHz/V. Mit C_T = 1 nF wird sie stattdessen 10 kHz/V. Die Linearität entspricht ungefähr der eines 3½-ziffrigen Digitalvoltmeters (0-1999). Mit diesem V/f-Wandler und dem Meßprogramm für Frequenzen des vorherigen Abschnitts haben Sie ein ausgezeichnetes Meßgerät für den Computer.

Möchten Sie sich Zugang zu mehreren Eingängen verschaffen, z. B. Daten von bis zu acht Gebern, dann schließen Sie am besten einen analogen Multiplexer an (ein Multiplexer ist ein Umschalter, der eines von mehreren Signalen auswählt).

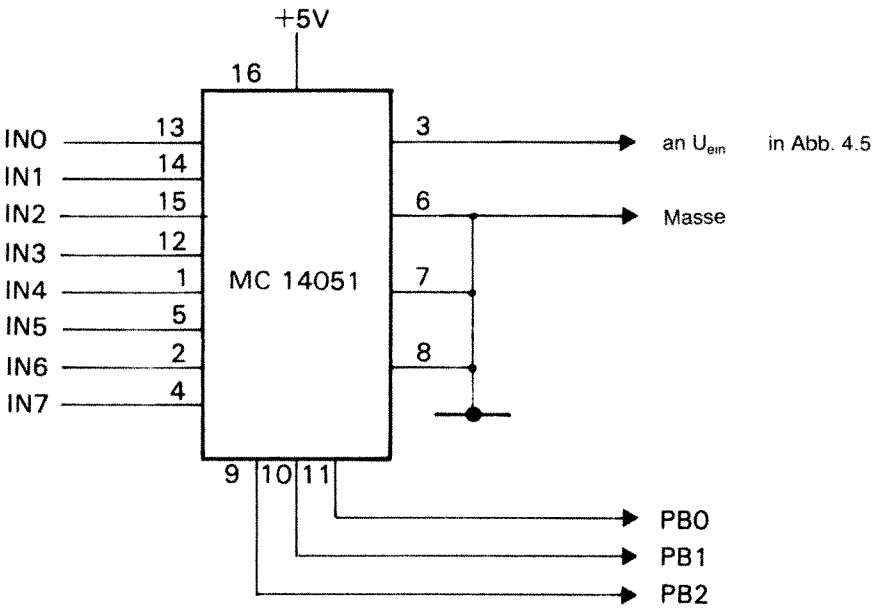


Abb. 4.6 Multiplexer

Der Multiplexer wird von PB0-PB2 aus angesteuert. Gibt der Computer die Zahl 0 an der Anwender-Schnittstelle aus, wird IN0 gewählt, gibt er 1 aus, wird IN1 gewählt usw. Das Programm auf Seite 51 muß um eine Zeile erweitert werden, die diese Auswahl trifft: 25 POKE 37136,N. N ist eine Zahl zwischen 0 und 7, die auf den gewünschten Eingang hindeutet. Beim C-64 müssen erst PB0-PB2 zu Ausgängen gemacht werden, erst dann kommt der POKE-Befehl nach Adresse 56577.

Potentiometer-Eingänge

Der VC-20 ist mit zwei Potentiometereingängen ausgerüstet, die eigentlich für die Joysticks der Spiele bestimmt sind. Die Potentiometereingänge sind an zwei A/D-Wandler angeschlossen, die den Widerstand des Potentiometers in digitale Form umwandeln. Die Potentiometer-Eingänge findet man leider nicht an der Anwender-Schnittstelle wieder. Sie befinden sich am Spielausgang an der rechten Seite des Computers. Die Eingänge werden POT X und POT Y bezeichnet.

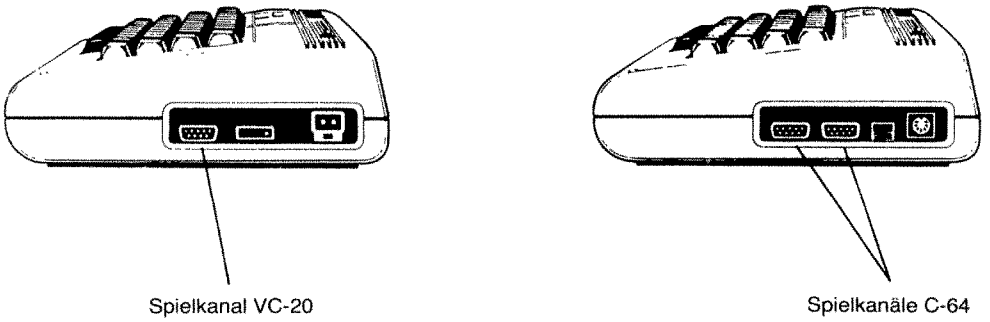


Abb. 4.7 Spielkanäle

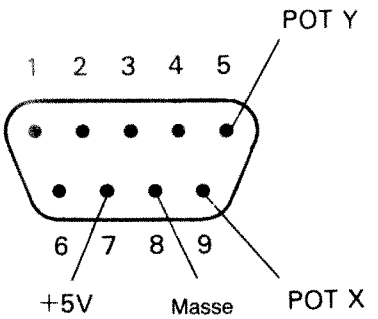


Abb. 4.8 VC-20 Eingang des Spielkanals

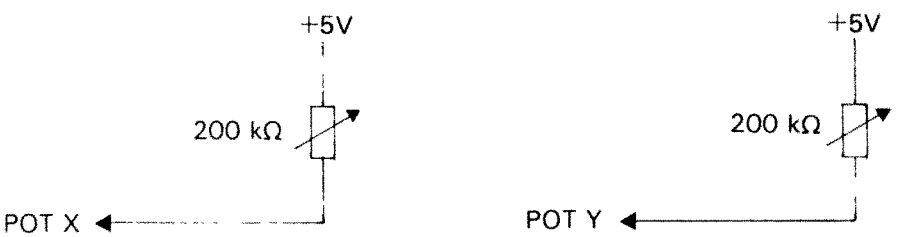


Abb. 4.9 Anschluß der Potentiometer

Den digitalisierten Wert von POT X findet man wieder in Adresse 36872, von POT Y in 36873. Die Werte können mit dem PEEK-Befehl abgefragt werden. Die Umwandlungszeit liegt bei einigen Millisekunden. Bitte ausprobieren!

Die entsprechenden Adressen für Spielkanal 1 sind beim C-64 54297 (POT X) und 54298 (POT Y). Ausprobieren!

Temperaturmessung

Viele physikalische Größen kann man mit billigen Gebern auf Widerstandsbasis messen.

Größe	Geber
Drehwinkel	Potentiometer
Lage	Schiebepotentiometer
Beleuchtung	Fotowiderstand
Temperatur	Thermistor

Sie sahen schon im vorherigen Abschnitt, wie die Eingänge POT X und POT Y zum Messen von Widerständen (und damit von Drehwinkeln) benutzt wurden. Wenn man die Potentiometer durch die anderen aufgeführten Geber ersetzt, lassen sich auch die anderen Größen messen. Wenn die Geber einen maximalen Widerstand haben, der 200 Ohm übersteigt, dann muß ein Kondensator eingefügt werden. Das Produkt R·C soll ca. 0,2 ms betragen. Verwenden Sie einen Geber mit 1 kOhm, dann muß der Kondensator eine Kapazität von 0,1 μF haben.

Wir wollen uns ein Beispiel ansehen, bei dem wir die Temperatur mit einem Thermistor messen, der an den Eingang POT X angeschlossen ist.

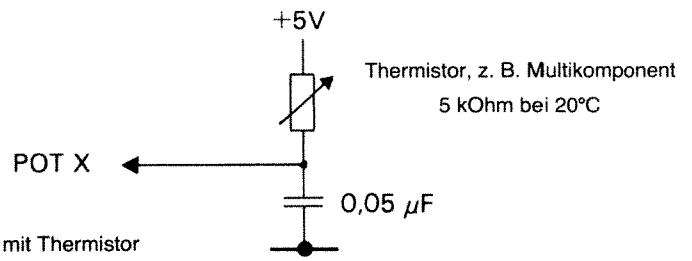


Abb. 4.10 Temperaturmessung mit Thermistor

Der Zusammenhang zwischen der Temperatur und der umgewandelten Zahl in Adresse 36872 (bzw. 54297 beim C-64) ist leider nicht linear. Der Widerstand ändert sich mit der Temperatur, aber die Abhängigkeit ist nicht linear. Durch Eichung bei verschiedenen Temperaturen ist es jedoch möglich, den Zusammenhang zu linearisieren. Mit einigen zusätzlichen Programmzeilen, die den wirklichen Zusammenhang zwischen Temperatur und dem abgelesenen Wert wiedergeben, erhält man ein brauchbares Thermometer.

Bei einem Versuch mit der Schaltung nach Abb. 4.10 ergab sich folgender

Zusammenhang zwischen Temperatur und der Zahl in Adresse 36872:

Temperatur	Werte
24	75
80	5
67	11
50	22

Mit Hilfe der Methode der kleinsten Fehlerquadrate kann aus diesen Werten eine mathematische Funktion abgeleitet werden, die den gewünschten Verlauf wiedergibt. Die Eichkurve sehen Sie in Abb. 4.11.

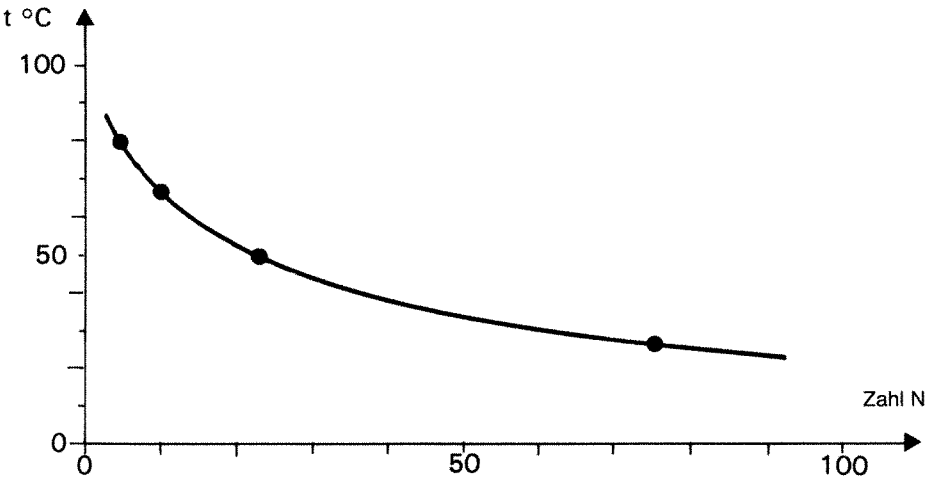


Abb. 4.11 Eichkurve

Die Methode der kleinsten Fehlerquadrate ist aufwendig, wenn man sie von Hand ausführt. Eine einfache grafische Annäherungslösung erfüllt mit ausreichender Genauigkeit diesen Zweck. Die Kurve kann näherungsweise mit zwei Geraden erstellt werden, Abb. 4.12.

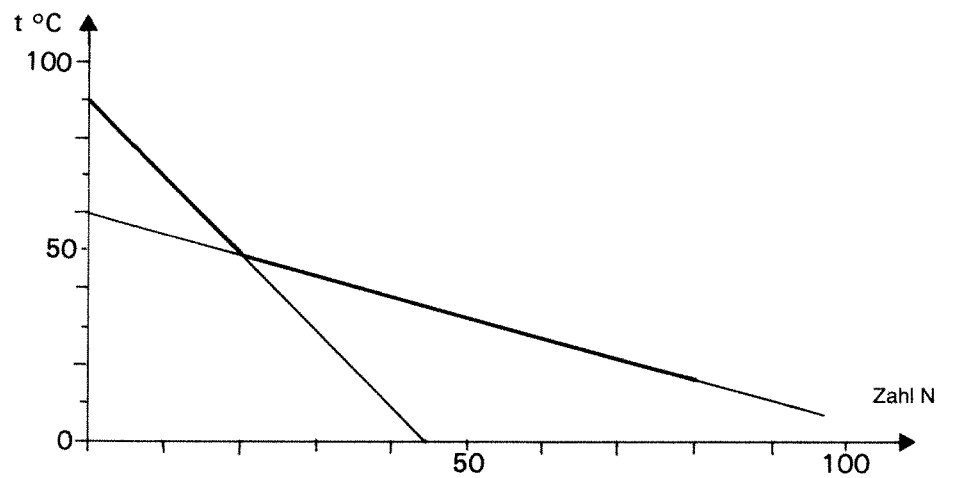


Abb. 4.12 Eichkurve mit zwei Geraden

Aus den beiden Geraden erhält man (Gleichung einer Geraden)

$t = 90 - 2{,}1 \times N$ für $0<N<20$
$t = 600 - 0{,}55 \times N$ für $N>20$

Das Programm für die Messung und Linearisierung (Adresse beim C-64 ist 54297):

```
10 N=PEEK(36872)
20 IF N<20 THEN PRINT 90-2.1*N
30 PRINT 60-0.55*N
40 GOTO 10
```

KAPITEL 5

Maschinensprache

VC-20 C-64

Was ist Maschinensprache?

Schon in Kapitel 1 wiesen wir darauf hin, daß das Betriebssystem und der Basic-Interpreter des VC-20 und C-64 im Lesespeicher in Form von 8-Bit binären Zahlen gespeichert sind. Beide Programme sind Maschinen-Programme, d. h. sie sind in Zeichen geschrieben, die der Mikroprozessor direkt lesen und ausführen kann. Wenn Sie also ein Basicprogramm auf dem Mikro-Computer laufen lassen, übersetzt der Basic-Interpreter das Basicprogramm in ein Maschinen-Programm.

Warum nun dieser Aufwand mit der Übersetzung? Wäre es nicht einfacher, wenn jeder Benutzer eines Computers von vornherein lernen würde, in Maschinensprache anstatt in Basic zu programmieren? Es gibt da einige Nachteile:

Erstens ist die Maschinensprache abhängig vom verwendeten Mikroprozessor. Die Maschinensprache für den Mikroprozessor 6502, der im VC-20 (6510 im C-64) sitzt, und z. B. der Mikroprozessor Z80, der im ABC 80 eingebaut ist, haben nichts miteinander gemeinsam. Dagegen ist das Basic im VC-20 und C-64 nahezu identisch mit dem Basic des ABC 80. Eine sogenannte gehobene Programmiersprache zu benutzen, hat aus Sicht der Vereinheitlichung einen großen Vorteil: Ein Basicprogramm kann ohne viele Änderungen auf nahezu jedem Computer laufen.

Zweitens ist der Umfang der Befehle in der Maschinensprache sehr begrenzt. Wenn man als Anfänger die Liste der Maschinenbefehle ansieht, stehen manchem die Haare zu Berge! Es zeigt sich, daß uns der Mikroprozessor, isoliert betrachtet, kaum weiterhilft. Er kann nämlich nur ganz einfache Aufgaben lösen, wie »addiere die Zahl mit einer 8-Bit Binärzahl«, »subtrahiere eins«, »addiere 2 Binärzahlen« (das ist schon die Höchstleistung!), »schiebe eine Binärzahl einen Schritt nach links«, »schiebe eine 8-Bit-Zahl vom Mikroprozessor zu einer bestimmten Adresse im Speicher« usw. Keine Multiplikation oder Division! Keine trigonometrischen Funktionen!

Bei näherem Hinsehen zeigt es sich jedoch, daß man durch Kombination von Befehlen alle gewünschten Operationen ausführen kann. Die Multiplikation kann z. B. durch eine Reihe von Additionen ersetzt werden (so wie man sie ausführt, wenn man mit Bleistift und Papier arbeitet). Das gilt auch für die Division. Der Arbeitseinsatz für ein Programm, mit dem man zwei 9-stellige Dezimalzahlen multiplizieren kann, ist vergleichsweise riesig. Ein geübter Programmierer in Maschinensprache brauchte etliche Stunden, um das Programm zum Laufen zu bringen. Zu Beginn der Computerwissenschaft, in den

vierziger Jahren, wurden grundsätzlich alle Programme in Maschinensprache geschrieben, von gelehrten Männern in weißen Kitteln. Allmählich entstanden die gehobenen Programmiersprachen (ALGOL, COBOL, FORTRAN, BASIC usw.). Das führte dazu, daß Computer schließlich auch von Nicht-Spezialisten programmiert werden können.

Zu diesem Zeitpunkt werden Sie sich sicherlich fragen, warum dieses Buch denn ein ganzes Kapitel über Programmierung in Maschinensprache enthält. Hier die Erklärung:

Maschinensprache und Basic

Unter gewissen Umständen kann es sinnvoll sein, kurze Maschinenprogramme zu schreiben anstatt eines (vermutlich noch kürzeren) Basic-Programms. Der wichtigste Grund ist die Forderung nach Geschwindigkeit, besonders im Umgang mit Daten von und zu peripheren Einheiten. Das ist der Fall, wenn mit dem Computer gesteuert und gemessen wird. Wie Sie schon wissen, benötigt der Basic-Interpreter ca. 1 ms, um eine Zeile eines Basic-Programms zu übersetzen und auszuführen. Wollen Sie z. B. an einem Ausgangskanal mit Hilfe des POKE-Befehls eine Impulsfolge erzeugen, wird die Frequenz dadurch auf einige Hundert Hz begrenzt. Ein einfaches Maschinenprogramm kann aber bei Bedarf eine Impulsfolge im Bereich von 50 kHz erzeugen!

Bei Dateneingabe von einem schnellen A/D-Wandler kann alle 20 Millisekunden ein neuer Meßwert vorliegen. Ein Maschinen-Programm für den VC-20 und C-64 kann alle diese Daten aufnehmen , abspeichern und hat dann noch eine Sicherheitsspanne. Ein Basicprogramm könnte dagegen nur jeden fünfhundertsten Wert aufnehmen, der Rest der Meßwerte ginge verloren. Eine Kombination von Maschinen- und Basicprogramm ist nahezu unschlagbar! Daher nimmt man Maschinensprache für den zeitempfindlichen Teil und Basic für den mechanischen Teil.

Es dauert sicherlich einige Zeit bis man ein versierter Programmierer für Maschinensprache wird, die Grundbegriffe hingegen sind nicht schwer zu verstehen. In den folgenden Abschnitten wollen wir näher darauf eingehen.

Programmieren in Maschinensprache auf dem VC-20 und C-64

Der Befehlssatz des Mikroprozessors 6502 im VC-20 oder 6510 im C-64 um-

faßt 56 Grundbefehle. Viele von ihnen können verschiedene Varianten benutzen, sog. Adressierverfahren, die wir uns etwas später ansehen wollen. Insgesamt werden 151 verschiedene Codes verwendet, um die ausführende Operation zu beschreiben. Jeder dieser Codes wird Operationsbefehl oder OP-Code genannt. Er besteht aus einer 8-Bit-Binärzahl, d. h. einer Dezimalzahl zwischen 0 und 255. Den Befehlssatz für 6502 und 6510 finden Sie in Anhang B. Das Datenblatt des Herstellers enthält detaillierte Angaben zu den Befehlen.

Ein guter Maschinen-Programmierer muß alle Feinheiten der 56 Grundoperationen und deren Varianten kennen. Es dauert eine gewisse Zeit bis man alle Befehle meistert! Wenn man mit dem Computer nur steuern und messen möchte und außerdem den Basic-Interpreter für Berechnungen nutzen will, dann vermindert sich die Zahl der Codes auf eine Handvoll einfacher Befehle.

Der Akkumulator

Die beiden wichtigsten Befehle sind »Lade Akkumulator« und »Speichere Akkumulator«. Der Akkumulator ist ein 8-Bit-Register im Mikroprozessor, der an nahezu allen Operationen teilnimmt. Sollen z. B. zwei Zahlen addiert werden, muß die eine Zahl anfangs im Akkumulator stehen. Wollen Sie Information in einer Speicherzelle ablegen oder in einem Ausgangskanal, muß die Information vom Akkumulator kommen.

Mnemonische Zeichen

In der Befehlsliste finden Sie für jeden Befehl eine Abkürzung wieder, die aus drei Buchstaben und dem entsprechenden OP-Code besteht. Die Abkürzung wird mnemonisches Zeichen oder symbolisches Zeichen, gelegentlich Mnemonik, genannt. Sie soll das Gedächtnis des Programmierers unterstützen. Die Abkürungen sind sinnvoll ausgesucht, besonders wenn man Englisch spricht. Der mnemonische Code für »Lade Akkumulator« ist LDA (von LoaD Accumulator) und für »Speichere Akkumulator« STA (von STore Accumulator).

Maschinen-Programme aufrufen

Es gibt zwei Basic-Befehle, mit denen man Maschinen-Programme aufrufen kann, SYS(X) und USR(X). Am einfachsten läßt sich SYS(X) verwenden. X ist die Speicheradresse, an der das Maschinen-Programm beginnt. Das Maschinen-Programm muß mit dem Maschinenbefehl »Kehre vom Unterprogramm zurück« beendet werden, Mnemonik RTS, OP-Code 96 (dezimal). Wenn das Maschinen-Programm beendet ist, kehrt der Mikrocomputer zum Basic zurück.

USR(X) ist schon etwas komplizierter. Der Computer beginnt mit der Ausführung des Programms, das in den Speicheradressen 1 und 2 steht. X ist eine Fließkomma-Variable, die in den sog. Fließkomma-Akkumulator geladen wird. Fließzahlen sind nicht so einfach zu verarbeiten, wir werden daher den Befehl USR(x) nicht benutzen. Die Speicheradressen 4096-7679 beim VC-20 stehen für Benutzerprogramme zur Verfügung. Hiermit müssen Basic- und Maschinen-Programme auskommen. Basic-Programme werden dann automatisch von 4096 an aufwärts gespeichert, weshalb es am besten ist, die Maschinen-Programme »am anderen Ende« abzuspeichern. Es ist nun leider so, daß der Basic-Interpreter hier Strings (Text) abspeichert, nämlich von Adresse 7679 an abwärts. Wir haben ein Problem, das jedoch leicht zu beheben ist. Mit ein paar POKE-Befehlen können wir Basic glauben machen, daß der Speicherplatz etwas früher aufhört, z. B. in 7599. Der Basic-Interpreter speichert dann dic Strings ab Adresse 7599 abwärts ab, und wir haben 80 Adressen ab 7600 für Maschinensprache frei. Folgende Befehle müssen eingegeben werden:

```
POKE 51,175 : POKE 52,29
POKE 55,175 : POKE 56,29
```

Beim C-64, der bedeutend mehr Speicherplatz zur Verfügung hat, ist in den Adressen 49152-53427 Platz für Maschinen-Programme reserviert.

Adressierungsarten

Wir wollen uns den Befehl LDA »Lade Akkumulator« im Anhang B ansehen.

LDA *(LDA Load accumulator with memory)* LDA

Lade Akkumulator vom Speicher

Operation: M → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	LDA Oper	169	2	2
Seite 0	LDA Oper	165	2	3
Seite 0, X	LDA Oper, X	181	2	4
Absolut	LDA Oper	173	3	4
Absolut, X	LDA Oper, X	189	3	4*
Absolut, Y	LDA Oper, Y	185	3	4*
(Indirekt, X)	LDA (Oper, X)	161	2	6
(Indirekt), Y	LDA (Oper), Y	177	2	5*

* Addiere 1, wenn Seitengrenze überschritten wurde

Es gibt nicht weniger als acht verschiedene Adressierungsarten und folglich auch acht verschiedene OP-Codes (Operationsbefehle), zwischen denen man wählen kann. Alle acht laden den Akkumulator, aber auf etwas unterschiedliche Weise. Aus der Tabelle geht hervor, welcher OP-Code zu welcher Adressierungsart gehört. Es steht aber noch mehr Information darin. »Bytes« gibt an, wieviele 8-Bit-Gruppen insgesamt für einen Befehl benötigt werden. »Zeitperioden« gibt an, welche Zeit der Befehl zur Ausführung benötigt. Der Zykluszeittakt ist 0,902 Mikrosekunden lang und wird durch einen Quarzkristall sehr genau bestimmt. Nehmen wir an, wir wollten den Akkumulator aus irgendeinem Grund mit der Zahl 128 laden. (Erinnern Sie sich daran, daß der Akkumulator ein 8-Bit Register ist und nur Ziffern zwischen 0 und 255 verarbeiten kann.) Es stehen eine Reihe von Möglichkeiten zur Verfügung. Am einfachsten wäre, die »unmittelbare Adressierungsart« auszunutzen. Die Mnemonik schreibt sich dann LDA #128. Die Zahl 128 ist der »Operand«.

Der OP-Code ist nach der Tabelle 169, und der Befehl umfaßt zwei Bytes. Die beien Zahlen (OP-Code 169, Operand 128) werden in zwei aufeinanderfolgende Adressen im Speicher geschrieben. Weiter geht aus der Tabelle hervor, daß der Computer 2×0,902 Mikrosekunden, d. h. 1,804 Mikrosekunden braucht, um den Befehl auszuführen.

Was wäre geschehen, wenn wir den OP-Code 165 anstatt 169 genommen hätten? Die Adressierungsart hieße dann »Seite 0«, und nach der Tabelle muß der Mnemonik LDA 128 sein, aber ohne das #-Zeichen. In der Adressie-

rungsart »Seite 0« deutet der Mikroprozessor den Operanden als eine Adresse. Was dann geschieht, ist, daß der Akkumulator mit einer Zahl geladen wird, die in Adresse 128 steht. Nachdem wir beabsichtigten, den Akkumulator mit der Zahl 128 zu laden, ist offensichtlich die Adressierungsart »Seite 0« eine schlechte Wahl. (Es sei denn, die Zahl 128 ist in Adresse 128 gespeichert.)

Alle übrigen Adressierungsarten stellen verschiedene Möglichkeiten dar, die Speicherplätze anzugeben, auf denen die in den Akkumulator zu ladenden Zahlen wiedergefunden werden. Wir wollen jetzt darauf nicht näher eingehen, haben jedoch Grund, später die Adressierungsart »absolut« näher unter die Lupe zu nehmen. Zusammenfassend läßt sich sagen: Wollen Sie den Akkumulator mit einer bestimmten Zahl, einer Konstanten, laden, dann müssen Sie die Betriebsart »unmittelbar« benutzen.

Nehmen wir an, daß wir den Akkumulatorinhalt (die Zahl 128) in Richtung Anwender-Schnittstelle, Kanal B, ausgeben wollen, d. h. an Adresse 37136 (VC-20) bzw. 56577 (C-64). Wir setzen voraus, daß die Anwender-Schnittstelle, wie in Kapitel 2 erläutert, zum Ausgang gemacht worden ist. Der Befehl »Speichere Akkumulator im Speicher«, Mnemonik STA, tut genau das. Hier bestehen sieben Adressierungsarten. Welche soll man wählen? »Seite 0« deutet auf einen Speicherplatz hin, aber leider nur im Adressenbereich 0-255. Der Befehl umfaßt nur zwei Bytes, davon ist das eine der OP-Code und das andere die Adresse. In einem Byte kann keine Zahl gespeichert werden, die größer als 255 ist. Unseren Ausgabe-Kanal finden wir in Adresse 37136 (56577) wieder, wir müssen daher in der Tabelle weitersuchen. Die Adressierungsart »absolut« kann uns weiterbringen. Die Mnemonik ist STA 37136, OP-Code 141, Länge 3 Bytes (davon erstes Byte 141). Die verbleibenden 2 Bytes, d. h. 16 Bits, stellen die Adresse dar. Es ist leider etwas umständlich, die Adresse in zwei Hälften zu teilen. Gehen Sie folgendermaßen vor:

1. Adresse 37136 in die entsprechende Binärzahl übersetzen. Nach etwas Rechnen ergibt sich 1001000100010000_2.
2. Diese Zahl in zwei 8-Bit-Hälften aufteilen, d. h. 10010001_2 und 00100000_2.
3. Diese 8-Bit-Zahlen wieder in Dezimalzahlen umwandeln. 10010001_2 ergibt 145, 00010000_2 ergibt 16.
4. Ändern Sie die Reihenfolge der beiden Hälften (dies ist eine Eigenart des Mikroprozessors 6502). Die erste Zahl muß 16, die zweite 145 sein.

STA *(STA Store accumulator in memory)* **STA**

Speichere Akkumulator im Speicher

Operation: A → M

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	STA Oper	133	2	3
Seite 0, X	STA Oper, X	149	2	4
Absolut	STA Oper	141	3	4
Absolut, X	STA Oper, X	157	3	5
Absolut, Y	STA Oper, Y	153	3	5
(Indirekt, X)	STA (Oper, X)	129	2	6
(Indirekt), Y	STA (Oper), Y	145	2	6

STA

Der vollständige Befehl ~~LDA~~ 37136 umfaßt also drei Zahlen, gespeichert in drei aufeinander folgenden Speicheradressen: Erst der OP-Code 141, dann die Zahl 16 und zuletzt die Zahl 145. 16 und 145 stellen die Adresse dar, aufgeteilt in zwei Hälften, in der Reihenfolge vertauscht. Der Befehl hat einen Zeitbedarf von $4 \times 0{,}902 = 3{,}608$ Mikrosekunden.

Es gibt glücklicherweise einfachere Methoden, die Adressen aufzuteilen, z. B. die folgende:

1. Teile die Adresse durch 256. Der Ganzzahlanteil stellt die letzte der beiden Adressenhälften dar. In unserem Fall gilt: 37136 : 256 = 145,0625. Der Ganzzahlanteil 145 ist die letzte Adressenhälfte.

2. Ziehen Sie den Ganzzahlanteil vom Ergebnis der Division ab und multiplizieren Sie den Rest mit 256. Das Ergebnis ist eine Ganzzahl und ist der erste Teil der beiden Adressenhälften. Für unseren Fall gilt: $(145{,}0625 - 145) \times 256 = 16$. Wir wollen uns noch ein Beispiel ansehen. Möchten Sie den Inhalt des Akkumulators in Adresse 7600 speichern, dann sind die Codes in der richtigen Reihenfolge: 141 OP-Code, 176 (halbe Adresse), 29 (halbe Adresse), d. h. 141,176,29.

Wir können unser Programm jetzt schon fast laufen lassen, es fehlt jedoch noch etwas Wichtiges: Der letzte Befehl muß RTS sein, »Rücksprung von der Subroutine«. Dieser Befehl hat nur eine Adressierungsart, Mnemonik RTS, OP-Code 96, Zeit $6 \times 0{,}902 = 5{,}142$ Mikrosekunden.

RTS *(RTS Return from subroutine)* **RTS**

Kehre vom Unterprogramm zurück

Operation: PC ↑ , PC + 1 → PC

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	RTS	96	1	6

Unser in Mnemonik (auch Assembler-Programm genannt) geschriebenes Programm sieht folgendermaßen aus:

```
LDA #128
STA 37136          (56577 für C-64)
RTS
```

Das Assembler-Programm ist nur für den Programmierer und für die Dokumentation wichtig. Hier die in aufeinanderfolgende Adressen in den Speicher zu schreibenden Zahlen: 169,128,141,16,145,96. Oft schreibt man das Maschinen-Programm neben das Assembler-Programm, um die Kontrolle zu erleichtern.

```
169, 128                LDA #128
141, 16, 145            STA 37136   (56577 für C-64)
96                      RTS
```

Das Programm hat einen Zeitbedarf von (2 + 4 + 6) × 0,902 = 10,82 Mikrosekunden (μs).

Programme in den Speicher eingeben

Es ist am besten, den Beginn des Maschinen-Programms in Adresse 7600 beim VC-20 und in Adresse 49152 beim C-64 zu legen. Unser kleines Programm, das nur aus sechs Zeichen in Maschinensprache besteht, kann mit sechs POKE-Befehlen (VC-20) abgespeichert werden:

```
POKE 7600,169
POKE 7601,128
POKE 7602,141
POKE 7603,16
POKE 7604,145
POKE 7605,96
```

Das entsprechende Programm für den C-64:

```
POKE 49152,169
POKE 49153,128
POKE 49154,141
POKE 49155,176
POKE 49156,29
POKE 49157,96
```

Beachten Sie, daß die Zeichen in aufeinander folgende Adressen abgespeichert werden müssen! Es ist einfacher, eine Schleife zum Einlesen zu benutzen, besonders bei etwas längeren Programmen:

```
40 FOR I=7600 TO 7605
50 READ A
60 POKE I,A
70 NEXT I
80 DATA 169,128,141,16,145,96
```

Lassen Sie das Programm zur Probe laufen und kontrollieren Sie dann den Speicher mit PRINT PEEK(7600) usw., ob die gewünschten Zeichen wirklich in den beabsichtigten Adressen stehen.

Programmausführung

Es ist jetzt an der Zeit, alle Erkenntnisse zusammenzufügen. Um das Programm testen zu können, müssen Sie an die Anwender-Schnittstelle PB0-PB7 Leuchtdioden anschließen, so wie das früher besprochen wurde. Unser Basic-Programm muß mit POKE 37138,255 die Schnittstelle auf einen Ausgang mit 8 Kanälen umstellen. Das Programm muß weiter dem Basic-Interpreter vortäuschen, daß der Speicher bei der Adresse 7599 zu Ende sei. Hierzu werden die POKE-Befehle 51,175 : POKE 52,29 : POKE 55,175 und POKE 56,29 verwendet. Außerdem muß das Maschinen-Programm gespeichert werden, was mit einer Schleife, wie im vorigen Abschnitt beschrieben wurde, erfolgen kann. Schließlich muß das Maschinen-Programm mit SYS(7600) aufgerufen werden.

```
10 POKE 37138,255
20 POKE 51,175 : POKE 52,29
30 POKE 55,175 : POKE 56,29
40 FOR I=7600 TO 7605
50 READ A
60 POKE I,A
70 NEXT I
80 DATA 169,128,141,16,145,96
90 SYS(7600)
```

Hier ist das entsprechende Programm für den C-64:

```
10 POKE 56579,255
20 FOR I=49152 TO 49157
30 READ A
40 POKE I,A
50 NEXT I
60 DATA 169,128,141,16,145,96
70 SYS(49152)
```

Lassen Sie das Programm zur Probe laufen! Wenn alles in Ordnung ist, muß die Leuchtdiode, die an PB7 angeschlossen ist, leuchten. (Wir haben ja den Akkumulator mit 128 (dezimal) = 10000000 (binär) geladen.) Versuchen Sie den Akkumulator mit einer anderen Zahl zu laden, indem Sie die Zahl 128 in der DATA-Zeile 80 (60) gegen eine andere Zahl auswechseln.

Impuls in Maschinensprache erzeugen

Obiges Programm hat die Information an Schnittstelle B »blitzschnell« bereitgestellt. Leider sind wir nicht in der Lage, den Wahrheitsgehalt dieser Aussage zu überprüfen. Wenn Sie Zugang zu einem Oszilloskop haben, können Sie jedoch das Programm so ändern, daß eine Taktfolge erstellt wird. Hierbei zeigt sich die Geschwindigkeit. Wir nehmen im Assembler-Programm folgende Änderung vor:

```
LDA #128        eine »1« an PB7 ausgeben
LDA #0          eine »0« am PB7 ausgeben
STA 37136
JMP 7600        Rücksprung zum Anfang
```

Der JMP-Befehl hat einen Sprung zum Anfang des Programms zur Folge. Dadurch entsteht die Taktfolge an PB7. JMP ist ein neuer Befehl für uns:

JMP *(JMP Jump to new location)* **JMP**

Springe an neue Adresse

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

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Absolut	JMP Oper	76	3	3
Indirekt	JMP (Oper)	108	3	5

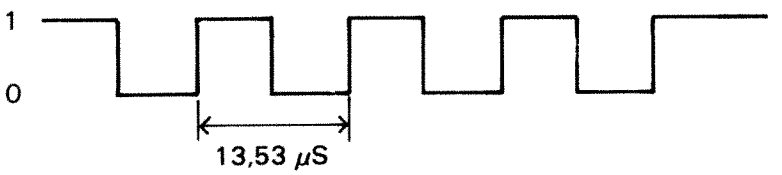


Abb. 5.1 Taktfolge an PB7, wie mit dem Oszilloskop festgestellt

Die Taktzeit ist 13,53 Mikrosekunden. In Basic mit POKE-Befehlen wäre sie ca. 17 Millisekunden gewesen, d. h. mehr als tausendmal langsamer. Bevor wir das Programm zum Laufen bringen, muß es in Maschinensprache übersetzt oder »assembliert« werden, wie man dies auch nennt.

```
169,128          LDA #128
141,16,145       STA 37136
169,0            LDA #0
141,16,145       STA 37136
76,176,29        JMP 7600
```

Die Adresse 7600 im JMP-Befehl wird 176,29 in der aufteilten Form, wie wir sie schon vorher berechnet haben. Das Basic-Programm muß in den Zeilen 40 und 80 etwas geändert werden:

```
40 FOR I=7600 TO 7612
80 DATA 169,128,141,16,145,169,0
        141,16,145,76,176,29
```

Das fertige Programm sieht dann folgendermaßen aus:

```
10 POKE 37138,255
20 POKE 51,175 : POKE 52,29
30 POKE 55,175 : POKE 56,29
40 FOR I=7600 TO 7612
50 READ A
60 POKE I,A
70 NEXT I
80 DATA 169,128,141,16,145,169,0
        141,16,145,76,176,29
90 SYS(7600)
```

Das entsprechende Programm für den C-64 kann so aussehen:

```
10 POKE 56579,255
20 FOR I=49152 TO 49164
30 READ A
40 POKE I,A
50 NEXT I
60 DATA 169,128,141,1,221,169,0,141,1,221,76,0,109
70 SYS(49152)
```

Aber Vorsicht! Unser Maschinen-Programm wird ja noch nicht mit RTS beendet, wie kurz vorher verlangt wurde! Nein, da unser Programm eine unendliche Schleife ist, kann es nicht beendet werden, und als Ergebnis kann der Mikro-Computer auch nicht zum Basic-Interpreter zurückkehren. Daraus ergibt sich, daß die Taktfolge auch nicht mit der STOP-Taste zum Stillstand gebracht werden kann. Die einzige Methode, die Maschine abzuschalten, ist, den Aus-Schalter zu bedienen, wobei das mühsam eingegebene Programm natürlch verloren ginge!

So können Sie das vermeiden:

1. Speichern Sie Ihr Programm auf Kassette oder Floppydisk ab, bevor Sie es laufen lassen.

2. Beenden Sie Maschinen-Programme mit RTS, wenn Sie zum Basic-Interpreter zurückkehren wollen.

Trotz aller Mühe werden Sie jetzt sicher von der Geschwindigkeit der Maschinen-Programme überzeugt sein.

VCMON und 64MON

Die im vorherigen Abschnitt beschriebene Methode, Maschinenprogramme zu schreiben, ist für kleinere, kurze Programme geeignet, die nicht mehr als 20 Maschinen-Befehle umfassen. Lange Programme werden jedoch nicht auf diese Art geschrieben. Der Grund hierfür: Der VC-20 und C-64 haben keinen Maschinensprachen-Monitor, mit dem man die Programme kontrollieren könnte. Wenn etwas falsch läuft, ist man hoffnungslos verloren, da keine Möglichkeit der Diagnostik besteht.

Es gibt jedoch für den VC-20 und C-64 einen Maschinensprachen-Monitor, der als Zubehör in Form einer Kassette erhältlich ist und an den Bus angeschlossen werden kann. Diese Kassette, VCMON bzw. 64MON, enthält alle nötigen Steuerprogramme, die das Programmieren in Maschinen-Sprache erleichtern können. U. a. ist darauf ein einfacher Assembler, der es erlaubt, die Programme in Mnemonik einzugeben. Weiter kann man die Register des Mikroprozessors einsehen und die Programme schrittweise ablaufen lassen.

Wenn Sie die Absicht haben, sich ernstlich mit dem Programmieren in Maschinensprache zu befassen, empfehlen wir ausdrücklich die Anschaffung des Monitors. Wollen Sie es dagegen bei kurzen Maschinenprogrammen bewenden lassen, wäre es eine unnötige Ausgabe.

KAPITEL 6

Datenübertragung über die RS-232-Schnittstelle

VC-20 C-64

Was ist RS-232?

RS-232 ist die Bezeichnung einer amerikanischen Industrienorm, die eine spezielle Art der seriellen Datenübertragung beschreibt. Sie wird oft für die Datenübertragung zwischen Computern und peripheren Einheiten benutzt. Die Daten werden als Taktfolge gesendet und empfangen.

RS-232 ist eine Art asynchroner Datenübertragung. Das bedeutet, daß mit den Daten kein spezielles Zeitsignal gesendet wird, was an die Synchronisierung zwischen Sender und Empfänger besondere Anforderungen stellen würde. Der Buchstabe »c«, ASCII-Zeichen 67, d. h. binär 1000011, kann folgendermaßen gesendet werden:

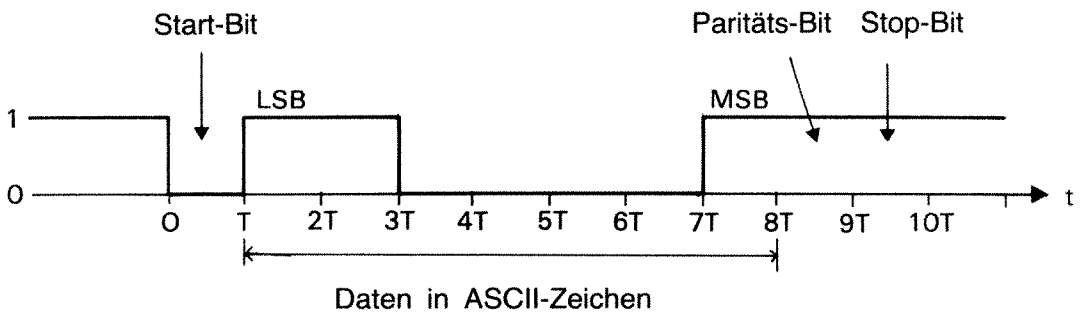
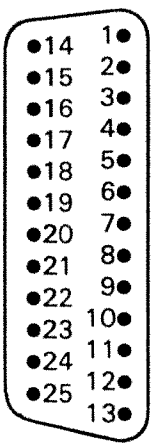


Abb. 6.1 Serielles Senden des Buchstaben »c«

»1« wird oft »MARK« und »0« wird »SPACE« genannt. Jedes Zeichen beginnt mit einem Start-Bit, das zum Synchronisieren von Sender und Empfänger gebraucht wird. Wenn das Start-Bit erfaßt wird, bewirkt dies im Empfänger das Starten eines Zeitmessers mit der Zeitperiode T, wodurch die Überprüfung des seriellen Datenstroms gesteuert wird. Überprüfung bedeutet, daß der Empfänger für jedes Zeitintervall feststellt, ob eine »1« oder eine »0« vorliegt und der Ziffer dann den richtigen Platz im Speicher zuweist. Nach jeweils sieben Überprüfungen werden die Daten im Register in parallele Form umgewandelt. Hier ist der Vorgang jedoch noch nicht abgeschlossen! Das nächste Bit ist ein Paritäts-Bit, mit dem Fehler in der Datenübertragung festgestellt werden können. Bei langen Leitungen, z. B. wenn Leitungen der Post verwendet werden, kann es leicht geschehen, daß auftretende Störungen eine »1« in eine »0« und umgekehrt verwandeln. Solche einfachen Bit-Fehler können mit Hilfe des Paritäts-Bits entdeckt werden. Das Prinzip ist einfach: Im Sender zählt man die Zahl der Einsen. Ist die Zahl der Einsen gerade, wird eine Null als Paritätsbit gesendet; ist die Zahl ungerade, dann wird eine Eins übermittelt. Man nennt dies »gerade Parität«, weil die Gesamtzahl der Einsen

immer eine gerade Zahl wird. Auf der Empfängerseite wird entsprechend verfahren, es wird die Zahl der Einsen gezählt, und wenn die Anzahl ungerade ist, dann weiß man, daß etwas nicht in Ordnung ist. Der Benutzer braucht sich um die Paritätskontrolle nicht zu kümmern, sie erfolgt automatisch. Das letzte Bit, am Ende, ist ein Stopbit (gelegentlich 2 Stopbits), die dem Empfänger mitteilen, daß ein Zeichen übertragen worden ist.

Jede serielle Datenübertragung muß ein Übertragungs-Protokoll haben, das sicherstellt, daß der Empfänger aufnahmebereit ist, bevor die Datenübertragung beginnt. Der Signalpegel der RS-232-Schnittstelle ist −12V für »1« und 12V für »0«. Es werden noch eine Reihe weiterer Eigenschaften des Signals festgelegt (u. a. muß eine RS-232-Schnittstelle kurzgeschlossen werden können, ohne daß Defekte auftreten), es würde jedoch zu weit führen, auf diese Einzelheiten einzugehen. Die Norm schreibt auch einen besonderen 25-poligen Stecker vor, Abb. 6.2.



PIN		
1	Gehäusemasse	(GND)
2	Daten Aus	(SOUT)
3	Daten Ein	(SIN)
4	Sende-Anforderung	(RTS)
5	Sendebereit	(CTS)
6	Datenübertragungs-Einrichtung klar	(DSR)
7	Signalmasse	(GND)
8	Daten-Empfangs-Quittung	(DCD)
9	(nicht Verwendet)	
10	"	
11	"	
12	"	
13	"	
14	"	
15	"	
16	"	
17	"	
18	"	
19	"	
20	Datenempfangsgerät bereit	(DTR)
21	(nicht verwendet)	
22	"	
23	"	
24	"	
25	"	

Abb. 6.2 Stecker und Signale beim VC-20/C-64

Signale und Steckverbinder

Trotz des genau spezifizierten Übertragungsprotokolls der RS-232-Schnittstelle ist es nicht üblich, alle zur Verfügung stehenden Steuersignale zu benutzen. Es werden nur die beiden Datenkanäle »Daten ein«, Signalbezeichnung SIN, und »Daten aus«, SOUT, verwendet. Es werden also drei Leitungen benötigt: Daten Ein, Daten Aus und die gemeinsame Rückleitung, Masse.

Der VC-20 und der C-64 sind in der Lage, mit dieser dreiadrigen Verbindung wie eine komplette RS-232-Schnittstelle mit allen Kontrollsignalen zu arbeiten. In einer Beziehung weichen der VC-20 und der C-64 von der Standard-Schnittstelle ab: Die Signalpegel sind 0 und +5V. Wir kommen hierauf etwas später zurück. Die Signale der RS-232-Schnittstelle finden wir an der Anwender-Schnittstelle gemäß der Tabelle in Abb. 6.3 wieder.

Signal	Ein/ Aus	Kontaktklemme der Anwender-Schnittstelle	Standard RS-232 Kontaktklemme
GND (Gehäusemasse)	-	A	1
SIN (Daten Ein)	Ein	B*	3
SIN (Daten Ein)	Ein	C*	3
RTC (Senderanforderung)	Aus	D	4
DTR (Datenempfangsgerät bereit)	Aus	E	20
RI (Ringindikator)	Ein	F	18
DCD (Daten-Empfangs-Quittung)	Ein	D	8
CTS (Sendebereit)	Ein	K	5
DSR (Datenübertragungseinrichtung klar)	Ein	L	6
SOUT (Daten Aus)	Aus	M	2
GND (Signalmasse)	-	N	7

(*) Pin B und C verbinden

Abb. 6.4 Schnittstelle

Bezeichnungen der Anwender-Schnittstelle:

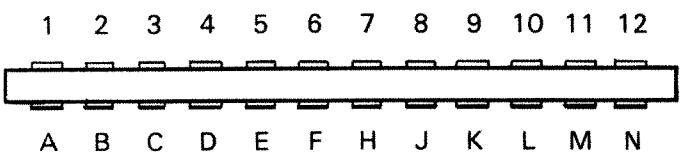
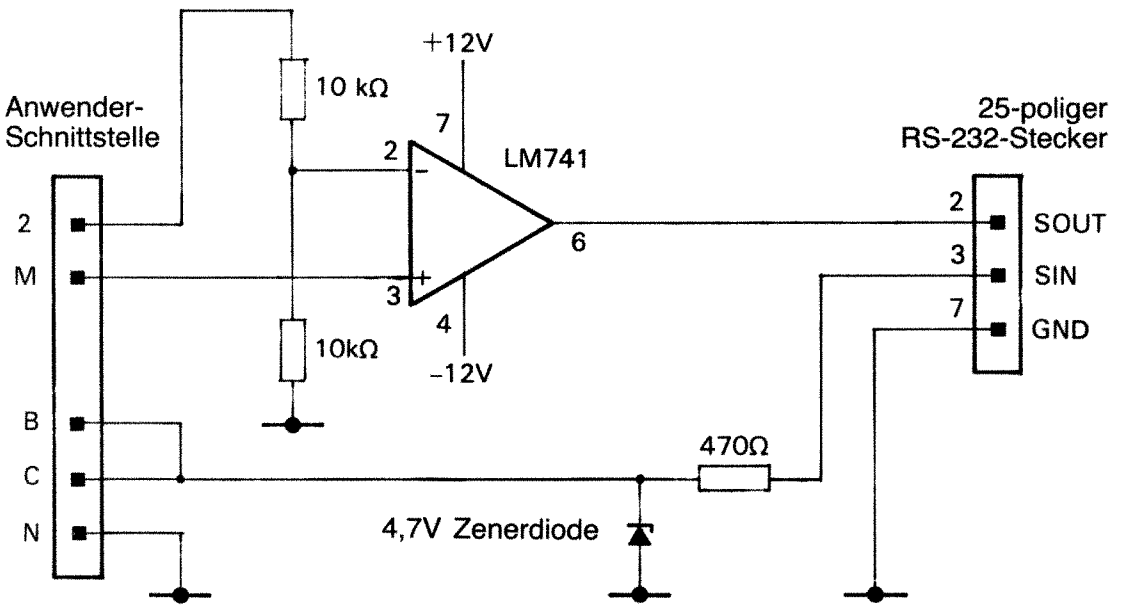


Abb. 6.3 RS-232 Signale der Anwender-Schnittstelle

Um den VC-20 und C-64 über die RS-232-Schnittstelle an eine andere Einheit anschließen zu können, müssen normalerweise die Signalpegel an die +12V/−12V des Standards angepaßt werden. Mit einer dreiadrigen Leitung reichen oft die 5V-Signale als Ausgangssignale vom Computer aus, besonders wenn die Leitung kurz ist. Schließen Sie nie ein +12V/−12V Signal als Eingangssignal an den Computer an, er gibt dann seinen Geist auf! Es gibt besondere Schnittstellen zu kaufen, die man zwischen die Anwender-Schnittstelle und den 25-poligen Steckverbinder der RS-232-Schnittstelle schalten kann. Für ein dreiadriges Kabel kann man diese Schnittstelle auch rasch selbst bauen, Abb. 6.4.



Falls Sie Schwierigkeiten haben, die +12V/−12V zu erzeugen, die der Operationsverstärker benötigt, kann folgender Spannungsumwandler verwendet werden. Die Ausgangsspannung beträgt zwar nur +9V/−9V, das reicht aber aus.

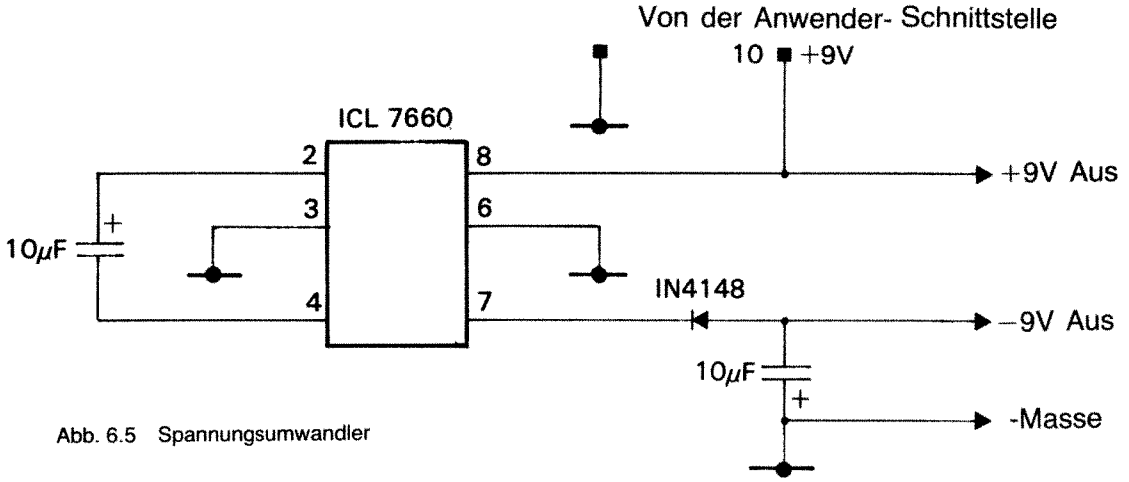


Abb. 6.5 Spannungsumwandler

Wahl von Übertragungsgeschwindigkeit, Wortlänge und Parität

Mit dem Inhalt von zwei Datenadressen kann der Anwender eine Menge wesentlicher Parameter der seriellen Datenübertragung festlegen. Die acht Bits in Adresse 659, dem Steuerregister, haben Aufgaben, die aus Abb. 6.6 hervorgehen.

Adresse 659

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Zahl der Stopbits
0 = 1 Stopbit
1 = 2 Stopbits

Wortlänge

Bit 6	Bit 5	Wortlänge
0	0	8 Bits
0	1	7 Bits
1	0	6 Bits
1	1	5 Bits

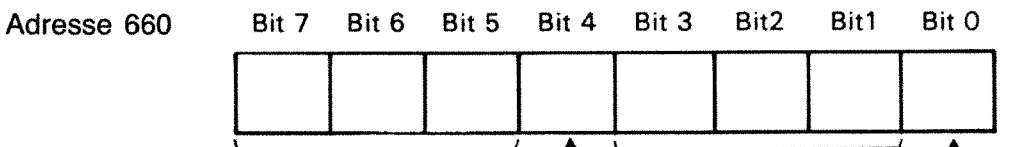
Datenübertragungsgeschwindigkeit in Baud

Bit 3	Bit 2	Bit 1	Bit 0	Baud
0	0	0	1	50
0	0	1	0	75
0	0	1	1	110
0	1	0	0	134,5
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	1800
1	0	1	0	2400

Abb. 6.6 Kontrollregister

Wird nicht verwendet!

Die acht Bits in Adresse 660, »Befehlsregister«, haben Aufgaben, die aus Abb. 6.7 hervorgehen.



Nicht gebraucht

Paritätsauswahl

Duplex
0 = Vollduplex
1 = Halbduplex

Hand shaking:
0 = 3-adriges Kabel
1 = echte RS-232

Bit 7	Bit 6	Bit 5	Parität
0	0	0	kein Paritätsbit
0	0	1	ungerade Parität
0	1	1	gerade Parität
1	0	1	immer »MARK«
1	1	1	immer »SPACE«

Abb. 6.7 Befehlsregister

Signale senden und empfangen

Bevor es möglich ist, Daten über die RS-232-Schnittstelle zu übertragen, muß sie als logische Datei eröffnet werden. In das Feld für Dateinamen können bis vier Zeichen eingegeben werden. Die beiden ersten werden benötigt, um dem Kontrollregister (Adresse 659) und dem Befehlregister (Adresse 660) geeignete Werte zuzuteilen. Die beiden restlichen sind für mögliche Erweiterungen des Computers vorgesehen.

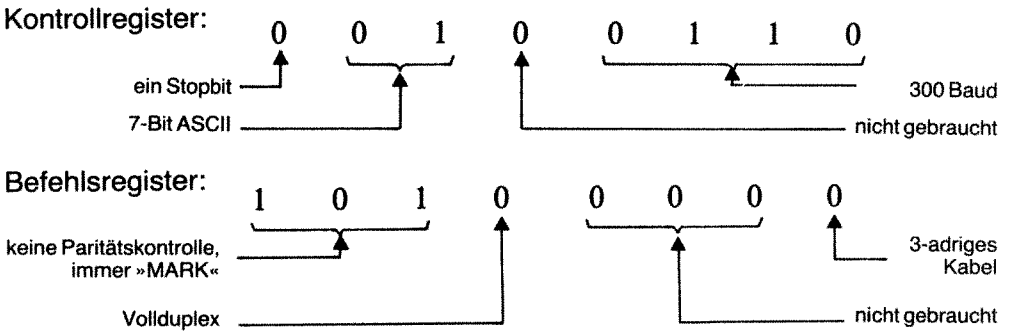
Um einen RS-232-Kanal zu eröffnen, wird der folgende Befehl gebraucht:

OPEN lf,2,0,(Kontrollregister)(Befehlsregister)

»lf« ist die Dateinummer, (1-255). Wenn lf>127, erfolgt ein automatischer Zeilenvorschub nach Wagenrücklauf. Ein Beispiel wird das Verständnis er-

leichtern. Gehen Sie davon aus, daß Ihr VC-20 oder C-64 Daten an einen Texas Silent 700-Terminal mit einem dreiadrigen Kabel übertragen soll. Das Silent-Terminal sei auf die Geschwindigkeit HIGH eingestellt, was 300 Baud entspricht. Verwendet wird 7-Bit-ASCII-Code, das Paritätsbit wird nicht benutzt, und wir wollen Vollduplex einsetzen. Ein Stoppbit wird benutzt. Wir teilen dem RS-232-Kanal die Dateinummer 2 zu.

Als erstes müssen wir feststellen, wie Kontroll- und Befehlsregister eingestellt werden sollen. Ein schneller Rückblick in den vorigen Abschnitt ergibt folgendes:



Das Kontrollregister muß also die Zahl 2 + 4 + 32 = 38 und das Befehlsregister die Zahl 32 + 128 = 160 aufweisen. Um den RS-Kanal zu eröffnen, müssen wir daher den folgenden Befehl verwenden:

```
OPEN 2,2,0,CHR$(38)+CHR$(160)
```

Beachten Sie, daß man den Kassettenrekorder oder die IEEE-Schnittstelle nicht gleichzeitig mit der RS-232-Schnittstelle benutzen kann. Wenn die Benutzung des RS-232-Kanals beendet ist, muß er geschlossen werden, was mit dem Befehl CLOSE lf geschieht, wobei lf die gleiche Dateinummer wie im OPEN-Befehl ist. In unserem Beispiel schließt man den RS-232-Kanal mit

```
CLOSE 2
```

Zeichen werden über die RS-232-Schnittstelle mit dem Befehl GET #lf,(Stringvariable) entgegengenommen. In unserem Beispiel wird vom Silent-Terminal der String A$ mit dem Befehl abgeholt:

```
GET #2,A$
```

Um Zeichen zu senden, wird der Befehl PRINT lf (Stringvariable) verwendet. In unserem Beispiel wird der Inhalt der Stringvariablen B$ mit folgendem Befehl ausgegeben:

```
PRINT # 2,B$
```

Abschließend noch ein paar Worte der Warnung. Öffnen Sie die RS-232-Schnittstelle immer, bevor Sie DIM-Zeilen oder Variablendefinitionen in Ihr Programm schreiben. Der Befehl OPEN hat automatisch CLR zur Folge, was bekanntlich die Variablen wieder auf Null setzt. Bedenken Sie auch, daß viele Drucker beim Wagenrücklauf keine Daten aufnehmen können. Bauen Sie deswegen in Ihr Programm eine Verzögerung ein, die es dem Drucker erlaubt, rechtzeitig zum Zeilenbeginn zurückzufahren, ehe er weiterschreiben soll.

KAPITEL 7

Die IEEE-Schnittstelle

VC-20 C-64

Die IEEE-488-Schnittstelle

IEEE-488 ist (wie der RS-232) die Bezeichnung einer amerikanischen Industrienorm, die eine Methode der Datenübertragung festlegt. Im praktischen Gebrauch spricht man auch vom IEEE-Bus oder dem HP-Bus, wenn man sich auf die Schnittstelle für IEEE-488 bezieht. Die Bezeichnung HP stammt von Hewlett-Packard, dem großen amerikanischen Hersteller von Meßinstrumenten, Rechnern und Computern. Hewlett-Packard war einer der Pioniere bei der Herstellung des IEEE-Busses.

Der IEEE-Bus ist hauptsächlich für die Datenübertragung zwischen Computern und Meßinstrumenten gedacht, z. B. zum Datensammeln oder Steuern von Prozessen. Heute gibt es auf dem Markt eine Reihe von Meßinstrumenten aller möglicher Hersteller, die an den IEEE-Bus angeschlossen werden können, wie z. B. Voltmeter, deren Meßbereich der Computer festlegt und die der Computer natürlich auch abliest. Es gibt Signalgeneratoren, für die der Computer über den IEEE-Bus Frequenz, Kurvenform und Amplitude vorgibt, und es gibt Gleichstromaggregate, für die der Computer Spannung und Polarität vorgeben kann, um nur einige Anwendungsgebiete zu nennen. Allen diesen Geräten ist gemeinsam, daß sie einen besonderen Steckverbinder und eine eingebaute Schnittstelle aufweisen, welche die Forderungen des IEEE-488 erfüllen.
Wir wollen einen kurzen Blick auf die IEEE-Busse werfen, ohne ins Einzelne zu gehen. Die Schnittstelle besteht aus 16 Leitungen und einer Rückführung, der Masse. Von diesen 16 werden 8 für den Quittungsbetrieb (»Hand shaking», »Protokoll«) gebraucht und 8 für Daten. Informationen, jeweils immer einzeln, werden als ASCII-Zeichen (parallel) gesendet. Der Vorteil des IEEE-Busses ist, daß man mehrere Instrumente gleichzeitig (10 bis 15 Stück) an den Bus anschließen kann. Jedes Instrument hat eine bestimmte 5-Bit-Adresse, die der Anwender normalerweise festlegen kann. Durch die Anwahl eines Instruments wird nur dieses aktiviert, die anderen bleiben inaktiv.

Es gibt drei verschiedene Instrumentenklassen, die man am Bus anschließen kann:

Steuereinheiten	(engl. Controller)
Empfänger	(engl. Listener
Sender	(engl. Talker)

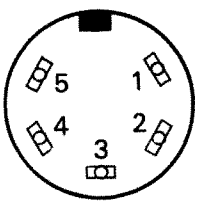
Die Steuereinheit (normalerweise ein Computer oder Rechner) steuert den Bus. Ein Empfänger empfängt Daten vom Bus, und ein Sender sendet Daten über den Bus.

Die IEEE-Schnittstelle des VC-20 und C-64

In den VC-20 und C-64 sind stark vereinfachte Formen des IEEE-Busses eingebaut worden. Anstatt des 16+1-adrigen Kabels wurde ein solches mit 4+1 Leitungen eingebaut. Die Hauptaufgabe des IEEE-Busses in diesen Computern ist die Datenausgabe an den speziellen VC-Drucker und die Datenübertragung mit der Floppydisk-Einheit. Beide sind an den VC-IEEE angepaßt. Es können jedoch keine gewöhnlichen IEEE-Instrumente an den VC-20 und C-64 angeschlossen werden. Hierfür ist ein Adapter erforderlich, der als Zubehör angeschafft werden kann und der die Computer an den normalen IEEE-488-Standard anpaßt.

Signale und Stecker

Am VC-20 und C-64 sitzt die 6-polige DIN-Steckbuchse zum IEEE-Bus auf der rechten Seite, Abb. 7.1.



Kontakt Nr.	Signal
1	Serielles SRQ ein
2	Masse
3	Serielles ATN ein/aus
4	Serielle Uhr ein/us
5	Serielle Daten ein/aus
6	nicht belegt

Abb. 7.1 Steckerbuchse des IEEE-Busses

SRQ bedeutet »Service Request« und ist das Signal einer peripheren Einheit, die an den Bus angeschlossen wird. ATN bedeutet »Attention« (Vorsicht) und wird verwendet, um Daten und Adressen zu trennen. Die Uhr wird zum Synchronisieren der Signale auf dem Bus verwendet. »Daten« ist schließlich der serielle Datenstrom, der einschließlich der Adressen mit dem Bus übertragen werden soll.

Verwendung der IEEE-Schnittstelle

Der VC-20 und C-64 können am IEEE-Bus nur als Steuereinheit, nicht aber als Hörer oder Sprecher gebraucht werden. Drei der IEEE-Adressen sind schon besetzt: Der Drucker VC-1515 hat die Adresse 4 oder 5 und die Floppydisk-Einheit die Adresse 8. Die Adressen 9-30 sind frei und können nach Belieben verwendet werden.

Bevor die IEEE-Schnittstelle benutzt werden kann, muß sie eröffnet werden. Dies geschieht wie immer mit dem OPEN-Befehl in folgender Form:

```
OPEN lf,d,sa,fn
```

»lf« bedeutet Dateinummer, »d« ist die Adresse der peripheren Einheit, mit der die Datenübertragung vorgenommen werden soll. »sa« ist die sekundäre Adresse (0-31), mit der die Betriebsart besonderer peripherer Einheiten bestimmt werden kann. »fn« schließlich ist eine Buchstabenfolge (String, 128 Zeichen) und wird hauptsächlich bei Betrieb der Floppydisk-Einheit benutzt.

Beim Drucker VC-1515 z. B. bedeutet die Sekundäradresse 7, daß damit Groß- und Kleinschrift zur Verfügung steht. Der Drucker hat die Adresse 4. Wenn man darüber Daten ausgeben will, muß man ihn erst als Datei eröffnen, z. B. mit der Dateinummer 1. Der Befehl ist

```
OPEN 1,4,7
```

Die Ausgabe der Information erfolgt mit PRINT lf,V$. lf ist die Dateinummer und V$ die Variable, welche die Daten enthält, die ausgegeben werden sollen. Im Falle des Druckers ist dies

```
PRINT #1,V$
```

Um Information von einer peripheren Einheit einzulesen, verwendet man

```
INPUT #lf,V$      oder
GET #lf,V$
```

Wie gehabt, bezieht sich »lf« auf die Dateinummer und V$ auf die Variable, welche die zu übertragende Information enthält. GET #lf,V$ holt nur ein Zeichen ab, während INPUT #lf,V$ einen String von Buchstaben aufbaut, bis ein Wagenrücklauf, d. h. CHR$(13) kommt. Ein solcher String darf maximal 88 Zeichen lang sein. Das IEEE-Protokoll, d. h. das Verfahren des »Hand shaking«, ist bei der vollausgebauten IEEE-Schnittstelle recht kompliziert. Als Benutzer merken Sie davon jedoch nichts, denn auch wenn der komplette IEEE-Adapter angeschlossen ist, werden die gleichen Befehle eingesetzt wie im Falle des vereinfachten Adapters.

KAPITEL 8

Einige praktische Versuche

VC-20 C-64

Einige praktische Versuche

In diesem Kapitel wollen wir uns mit einigen interessanten Anwendungen des Computers befassen. Auf dem bisher Gelernten können Sie weiter aufbauen, und Ihrer schöpferischen Phantasie ist keine Grenze gesetzt! Die Möglichkeiten umfassen sowohl Hardware als auch Software. Nach dem Durcharbeiten der jetzt folgenden Versuche dürfen Sie davon überzeugt sein, daß Sie Ihren Computer für das Messen, Steuern und Regeln optimal einsetzen können.

Lichtstift

Ein Lichtstift ist eine Vorrichtung, mit deren Hilfe man auf dem Bildschirm einen Punkt auswählen und vom Computer die Koordinaten anzeigen lassen kann. Der Lichtstift ist ein ausgezeichnetes Hilfsmittel zur Wahl von Menü-Alternativen und zum Zeichnen von Figuren auf dem Bildschirm. Um die Arbeitsweise zu verstehen, müssen wir uns erst ansehen, wie ein Bild auf dem Bildschirm entsteht. Es wird von einem Elektronenstrahl gezeichnet, der ein- und ausgeschaltet werden kann und den Bildschirm in einem bestimmten Raster überstreicht. Auf der Innenseite des Schirms befindet sich eine fluoreszierende Substanz, die an den vom Strahl getroffenen Stellen aufleuchtet. Hohe Strahlungsintensität erzeugt viel Licht und umgekehrt. Der Strahl wird durch Elektromagneten wie in Abb. 8.1 gesteuert. Das geht sehr schnell! Das Zeichnen einer Linie dauert 64 Mikrosekunden; für das ganze Bild benötigt der Elektronenstrahl 20 Millisekunden. Der Computer zeichnet 50 Bilder je Sekunde, so daß das Auge daher kein Flimmern feststellt.

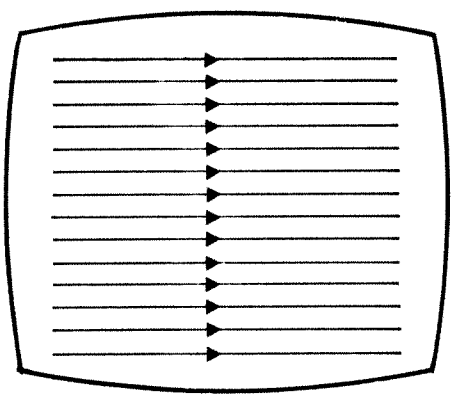


Abb. 8.1 Das Bild wird rasterförmig aufgebaut

Im Videoteil des Computers befinden sich Register, die festhalten, in welcher Position sich der Strahl jeweils befindet, d. h. seine X- und Y-Koordinaten. Soll nur ein Buchstabe, z. B. »E«, gezeichnet werden, geschieht dies wie in Abb. 8.2. Der Elektronenstrahl ist ausgeschaltet, außer wenn er auf die Punkte trifft, die das betreffende Zeichen darstellen.

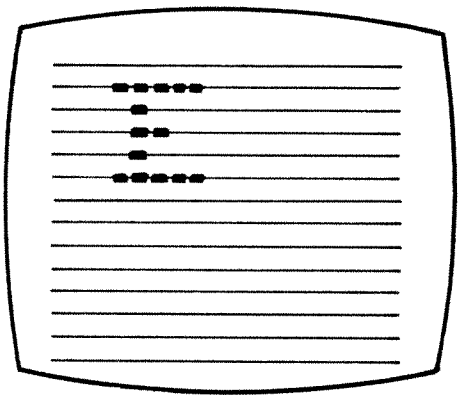


Abb. 8.2 Ein Zeichen wird durch Ein- und Ausschalten eines bewegten Elektronenstrahls aufgebaut

Ein Lichtstift ist ein einfaches Gerät. Er besteht aus einer Fotozelle, die auf das Licht eines Elektronenstrahls reagiert. Der genaue Zeitpunkt, zu dem das Licht festgestellt wird, läßt den Computer erkennen, auf welche Stelle des Schirms er jeweils auftrifft. Der VC-20 hat einen Eingang für den Lichtstift, besser gesagt, zwei parallelgeschaltete Eingänge. Der eine Eingang ist Pin 6 der Schnittstelle für Spiele, der andere ist Pin 7 der Anwender-Schnittstelle. Beim C-64 gibt es einen Eingang an der Schnittstelle 1 für Spiele, Pin 6. Sie können den Lichtstift als Zubehör kaufen, ihn aber auch einfach selbst basteln. Als Fotozelle nehmen Sie am besten einen Fototransistor vom Typ FPT 100, den Sie oben in den Schaft eines Kugelschreibers einsetzen, um den Fototransistor vor Streulicht zu schützen.

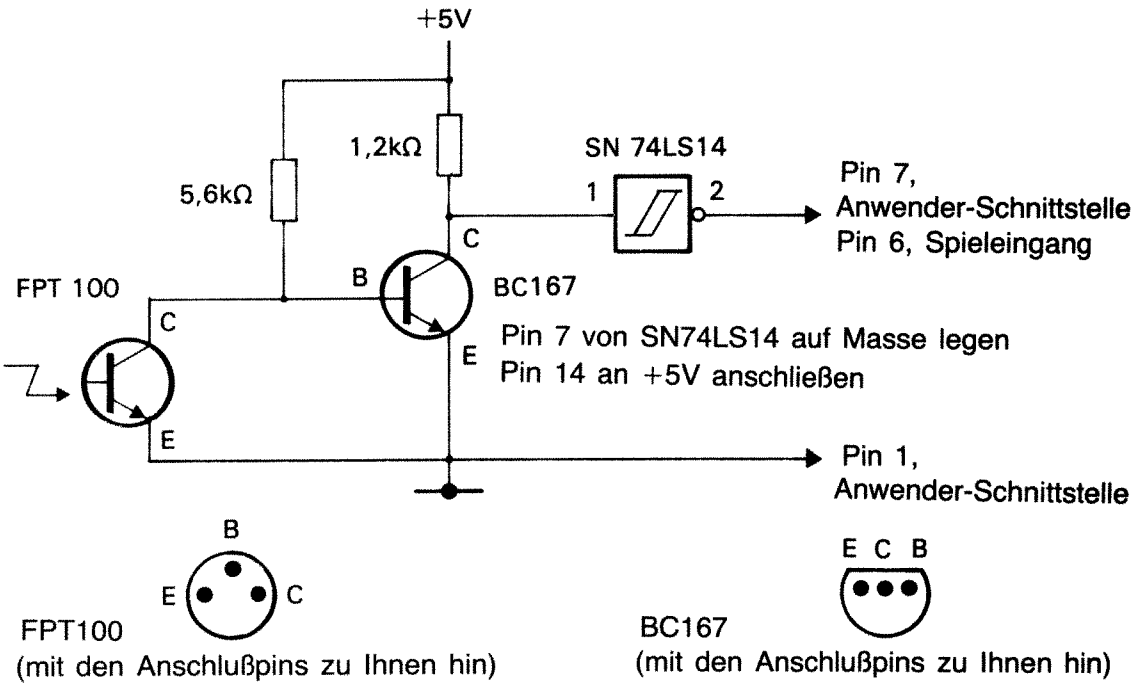


Abb. 8.3 Verdrahtung des Lichtstiftes

Im VC-20 und C-64 gibt es zwei Register, in denen die X-Koordinate und die Y-Koordinate der Lichtgriffelposition auf dem Bildschirm gespeichert werden. Im VC-20 hat das X-Register die Adresse 36870 und das Y-Register die Adresse 36871. Im C-64 hat das X-Register die Adresse 53267 und das Y-Register die Adresse 53268. Ein Programm, das die X- und Y-Koordinaten auf den Bildschirm schreibt, kann folgendermaßen aussehen (für den C-64 sind die Adressen zu ändern):

```
10 PRINT CHR$(147)
20 X=PEEK(36870)
30 Y=PEEK(36871)
40 PRINT"X=";X, "Y=";Y
50 GOTO 10
```

Zeile 10 löscht den Bildschirm, Zeilen 20 und 30 lesen das X- bzw. Y-Register, und Zeile 40 schreibt die Zahlen auf den Bildschirm. Eigentlich geht das Ganze etwas zu schnell für das Auge. Bauen Sie deshalb eine Verzögerung in das Programm ein:

```
45 FOR I=1 TO 500; NEXT I
```

Hier Vorschläge zum Ausbauen der Möglichkeiten mit dem Lichtstift:

1. Schreiben Sie ein Programm, mit dem Sie Alternativen aus einem Menü wählen.
2. Schreiben Sie ein Programm, das mit Hilfe des Lichtstifts Bilder auf den Bildschirm zeichnet.

Datalogger (Meßdatenaufnehmer)

Ein Datalogger ist ein Gerät, mit dessen Hilfe man von einem oder mehreren Gebern Meßwerte zu bestimmten Zeitpunkten erfassen und abspeichern kann. Die Speicherung kann z. B. dadurch erfolgen, daß die Meßwerte auf einem Drucker ausgedruckt werden. Man kann die Werte auch im Speicher des Computers aufbewahren, um sie später auf Kassette oder Floppydisk abzulegen. Es hat sich als sehr praktisch erwiesen, Meßdaten in einer Datei vorliegen zu haben, wenn später eine Weiterverarbeitung der Werte erfolgen soll, wie z. B. statistische Berechnungen, Trendanalysen usw.

In diesem Abschnitt sollen Sie einen Datalogger aufbauen, der an einen Multiplexer und einen Spannungs-/Frequenz-Wandler angeschlossen ist, wie wir das in einem der vorherigen Abschnitte schon besprochen haben. Der Datalogger soll zu Temperaturmessungen an bis zu acht Meßfühlern verwendet werden. Als Geber benutzen wir eine einfache Siliziumdiode, IN4148. Der Spannungsabfall über die Diode ändert sich nämlich im Temperaturbereich −40°C bis +150°C linear mit ca. 2 mv/°C. Der Spannungsabfall variiert natürlich von Diode zu Diode. Wenn man will, kann man Dioden auswählen, die den gleichen Spannungsabfall haben. Wenn man aber einen Computer hat, ist das nicht nötig. Abweichungen der Dioden untereinander kann man durch Eichwerte im Programm berücksichtigen.

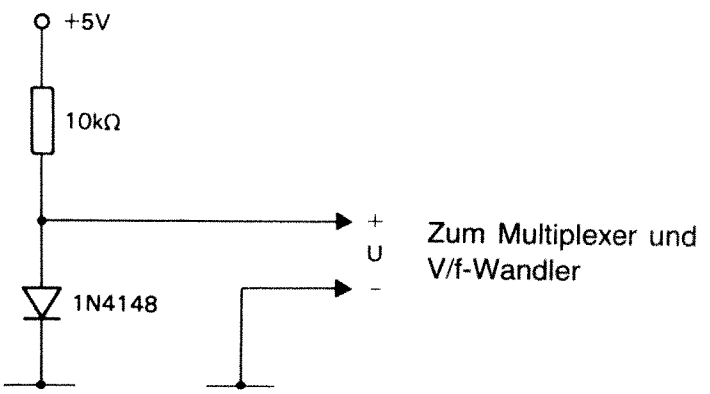


Abb. 8.4 Die Dioden können auf diese Weise angeschlossen werden

Die Stromstärke durch die Dioden beträgt ca. 450 μA und der Spannungsabfall bei 20°C 0,55 V. Wird die Diode auf 100°C erwärmt, beträgt der Spannungsabfall 0,39 V. Es tritt also nur eine geringfügige Änderung auf. Leider fällt die Spannung ab, wenn die Temperatur steigt, zweifellos wäre es angenehmer, wenn sie ansteigen würde. Mit einer Brückenschaltung und einem Differentialverstärker können wir zwei Mücken mit einer Klappe schlagen: Wir verstärken die Spannungsänderung fünfmal und erhalten darüber hinaus eine Spannung, die sich mit der Temperatur erhöht, Abb. 8.5.

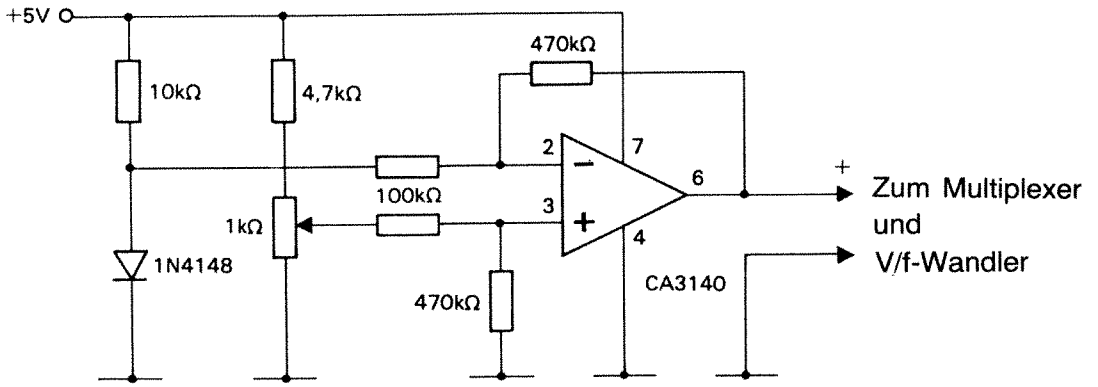


Abb. 8.5 Verstärker und Temperaturgeber

Vor dem Eichen der Dioden in einem Wasserbad bekannter Temperatur sollte man sie mit Schrumpfschlauch isolieren, denn Wasser kann u. U. ein guter Leiter sein! Bei der Eichung eines Geräts nach Abb. 8.5 wurde die Diode in Wasser getaucht, das schmelzendes Eis enthielt (0 Grad). Das 1 kOhm Potentiometer wurde so eingestellt, daß bei 0°C U = +1,00 V gemessen wurde. Bei +23°C ergab sich U = +1,23 V und bei +71°C U = +1,71 V. Es besteht also ein linearer Zusammenhang zwischen Spannung und Temperatur! Eine grafische Darstellung des Zusammenhangs zwischen Spannung und Temperatur finden Sie in Abb. 8.6. Es handelt sich hier um einen passenden Spannungsbereich für den Spannungs-/Frequenz-Wandler, den wir in einem vorhergehenden Abschnitt erläutert haben.

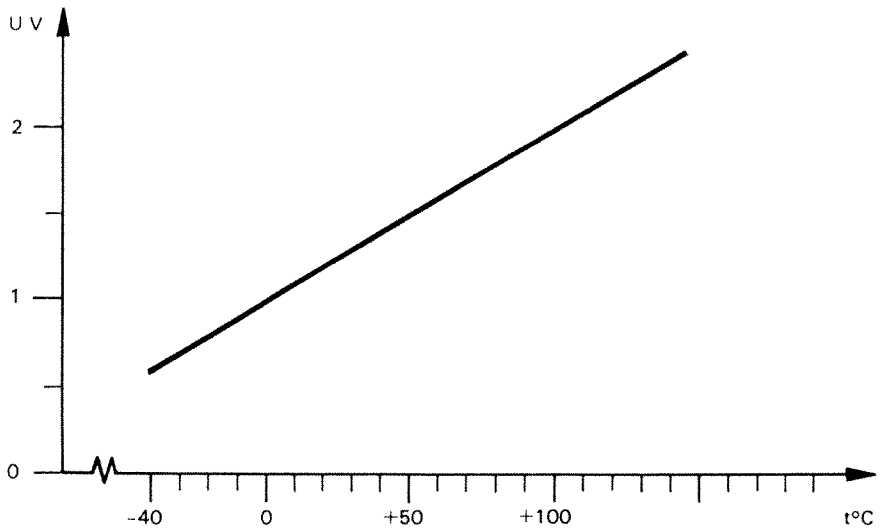


Abb. 8.6 Zusammenhang zwischen Spannung und Temperatur

Vollständig ausgebaut, mit allen 8 Gebern, Multiplexer und V/f-Wandler, finden Sie das Verdrahtungsschema in Abb. 8.7.

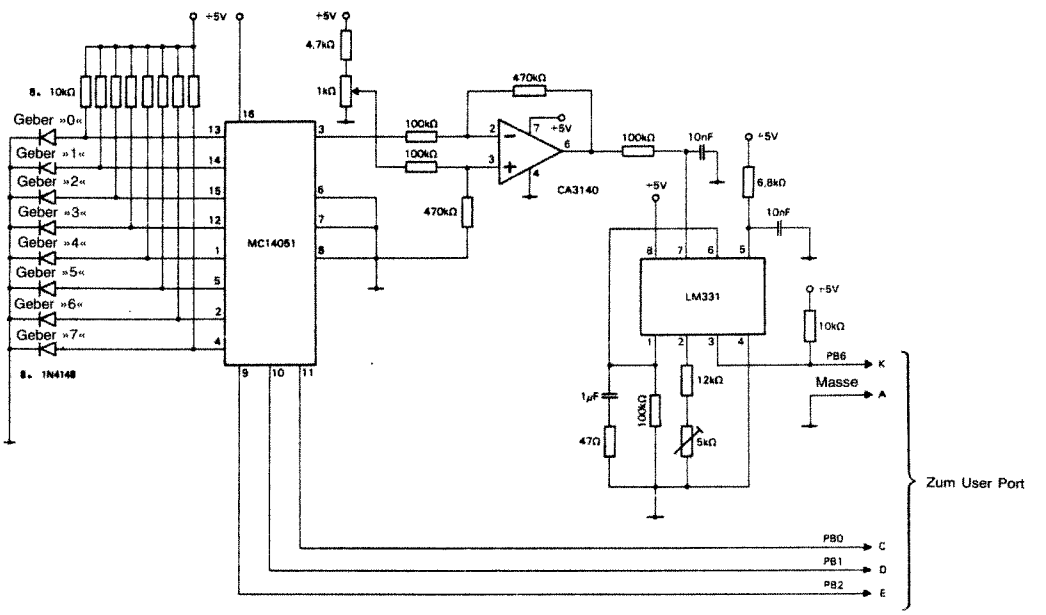


Abb. 8.7 8-Kanal Datalogger (Meßdatenaufnehmer)

Die Empfindlichkeit des V/f-Wandlers ist 1kHz/V. Nach Einstellung des 1 kOhm Potentiometers kann man davon ausgehen, daß 1 kHz 0°C entspricht, 0,6 kHz −40°C usw. Der gewünschte Geber wird mit PB0, PB1 und PB2 in Binärcode entsprechend der Tabelle gewählt:

PB2	PB1	PB0	Nummer des Gebers
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Wir sind mit unserem Programm jetzt beinahe fertig. Zunächst jedoch noch einen Hinweis. Wenn der Multiplexer auf einen neuen Geber umschaltet, braucht der V/f-Wandler ca. 0,2 Sekunden, um sich auf die neue Frequenz einzustellen. Das Programm muß also Verzögerungen enthalten, die dies berücksichtigen.

Ein Programm für den VC-20, das 10 Meßwerte von jedem der 8 Meßpunkte alle 10 Minuten abspeichert und um 13.00 Uhr zu messen beginnt, folgt unten. Wir benutzen die Uhr, die natürlich erst gestellt werden muß, zur Festlegung des Meßzeitpunktes und das Programm zur Frequenzmessung aus Kapitel 4 als Unterprogramm. Für den C-64 müssen die Adressen in Zeilen 10, 60, 200, 210, 240 und 260 geändert werden.

```
10 POKE 37138,191
20 K=130000
30 FOR S=0 TO 9
40 IF VAL(TI$)<K+1000*S THEN 40
50 FOR I=0 TO 7
60 POKE 37136,I
70 FOR N=0 TO 500 : NEXT N
80 GOSUB 200
90 M(I,S)=(F-1000)/10
100 NEXT I
110 NEXT S
120 END
```

```
200 POKE 37147, PEEK(37147) OR 32
210 POKE 37144,255
220 T=TI
230 IF T=TI THEN 230
240 POKE 37145,255
250 IF TI<>T+60 THEN 250
260 F=65535-(256*PEEK(37135)+PEEK(37144))
270 RETURN
```

Zeile 10: Macht PB6 zum Eingang und übrige PB zu Ausgängen der Anwender-Schnittstelle. (Beim C-64 muß man PB6 nicht zum Ausgang machen.)

Zeile 20: Legt die Zeit der ersten Messung in Stunden, Minuten und Sekunden fest.

Zeile 30: Bildet mit Zeile 110 eine FOR...NEXT-Schleife, die zehn Messungen ergibt. Die Variable S gibt an, welcher Meßwertgeber gerade mißt.

Zeile 40: Liest die Uhr ab und prüft, ob gemessen werden soll. Beachten Sie, daß TI$, die Zeitangabe, eine Stringvariable ist. VAL(TI#) ist dagegen eine numerische Variable. K+1000*S legt den Zeitpunkt der nächsten Messung fest. 1000 bedeutet 10 Minuten und 00 Sekunden. Der Computer wartet in Zeile 40 bis wieder eine Messung ausgeführt wird.

Zeile 50: Bildet mit Zeile 100 die Schleife, die jeweils eine der 8 Meßstellen auswählt. Die Variable I bezeichnet die Meßstelle.

Zeile 60: Legt die Meßstelle fest.

Zeile 70: Die Verzögerung von 0,5 Sekunden erlaubt dem V/f-Wandler, die neue Frequenz zu wählen.

Zeile 80: Springt zur Subroutine, welche die Frequenz mißt und gibt das Ergebnis an die Variable F weiter.

Zeile 90: Speichert die Temperatur in Matrix M.

Zeile 200: Subroutine zur Frequenzmessung (siehe Kapitel 4).

Bevor Sie das Programm laufen lassen, muß die Uhr eingestellt werden. Das macht man mit dem Kommando TI$="HHMMSS". HH sind Stunden, MM Minuten und SS Sekunden. Zum Test des Programms stellen Sie die Uhr am besten auf TI$="125900". Sie haben dann noch eine Minute Zeit, bis das Programm anläuft (die erste Messung wird um 130000 Uhr durchgeführt).

Das Ergebnis der Meßserie wird in der Matrix M(I,S) gespeichert. I ist die Nummer des Meßwertgebers, S ist die Nummer der Messung. Wollen Sie nach Abschluß der Meßserie das Ergebnis von Meßpunkt 5 um 132000 Uhr (dritte Messung) sehen, schreiben Sie PRINT M(5,3) und drücken auf RETURN. Sie erhalten dann das Resultat.

Vorschläge für weitere Versuche:

1. Ergebnisausgabe auf dem Bildschirm nach Ende jeder Meßserie.
2. Grafische Darstellung auf dem Bildschirm, wenn mehrere Messungen abgeschlossen sind.
3. Richten Sie es ein, daß Sie die Meßwertgeber beliebig anwählen können, z. B. 3, 4 und 7. Die Reihenfolge soll sich leicht verändern lassen.
4. Erweitern Sie das Programm so, daß die Meßwerte nach Beendigung der Messung auf Kassette oder Diskette abgespeichert werden können.
5. Schreiben Sie ein Programm, das es Ihnen gestattet, abgespeicherte Meßwerte weiterzubearbeiten. Es soll z. B. der Mittelwert der Messungen eines Meßwertgebers ermittelt werden.
6. Wie können weitere Geber vorgesehen werden?
7. Man kann mit dem Datalogger natürlich auch andere Werte als Temperaturen registrieren. Eine interessante Weiterentwicklung wäre eine Wetterstation. Windrichtung und Windgeschwindigkeit sollten einfach zu messen sein (Windflügel + Potentiometer oder optischer Geber / Propeller mit optischem Geber / Magnet mit Reedrelais). Luftfeuchtigkeit und Luftdruck sind schwieriger, aber keineswegs unmöglich zu messen.

Temperaturregelung

Ein Temperaturregler ist eine Anordnung, die z. B. in einem Prozeß oder in einem Zimmer eine gewünschte Temperatur einhält. In diesem Abschnitt soll der Mikrocomputer als ON/OFF- (Ein/Aus-) Regler eines Ofens dienen. Er soll dabei die Temperatur im Ofen einhalten, indem er die Wärmezufuhr ein- oder ausschaltet. Die Temperatur wird mit einem Thermistor gemessen. Die Meßwerte veranlassen den Computer, die Wärmezufuhr zu starten oder abzustellen. Es gibt sicherlich wesentlich bessere Regelmethoden, trotzdem wird diese häufig verwendet.

Für unseren Versuch wird der »Ofen« von einem Pappkarton dargestellt, der mit einer Glühbirne von 60W, 220V als Wärmequelle versehen ist. Der Computer schaltet die Lampe mit einem Halbleiterrelais des in Kapitel 3, Abb. 3.6 angegebenen Typs ein und aus.

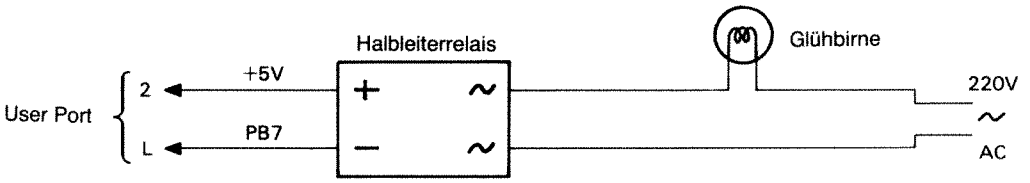


Abb. 8.8 Schaltschema der Temperaturregelung

Beachten Sie, daß PB7=»1« die Glühbirne ausschaltet, PB7=»0« sie einschaltet. Um die Temperatur im Ofen zu messen, benutzen wir einen Thermistor, angeschlossen an den POT-X-Eingang gemäß Kapitel 4.

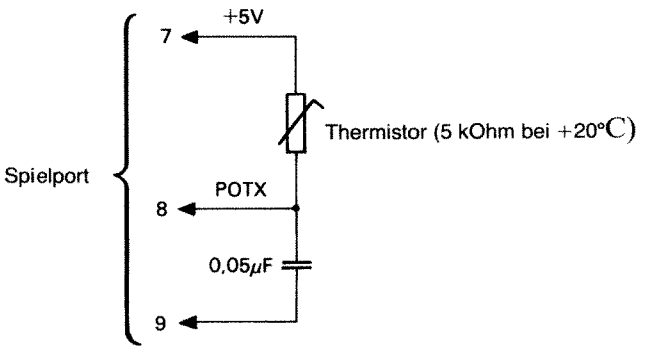


Abb. 8.9 Die Temperaturen werden mit einem Thermistor gemessen

Abb. 8.8 und 8.9 zeigen Ihnen alle Bauteile, die benötigt werden. Seien Sie mit den 220V-Anschlüssen vorsichtig! Diese müssen alle gut isoliert und richtig angeschlossen sein, bevor Sie den Strom einschalten. Weder Ihnen noch dem Computer bekommt die Berührung mit der Netzspannung!

Zur Messung der Temperatur benutzen Sie am besten das Programm in Kapitel 4, das auch den Thermistor linearisiert. Das Programm wird als Unterprogramm (Subroutine) aufgerufen, Zeile 100. Ein Programm, das als »Sollwert« die gewünschte Temperatur von 50°C einhält, kann folgendermaßen aussehen (für den C-64 ändern Sie bitte die Adressen in Zeilen 10, 20, 60 und 100).

```
10 POKE 37138,128
20 POKE 37136,128
30 SW=50
40 GOSUB 100
50 IF T>SW THEN 20
60 POKE 37136,0
70 GOTO 40
100 N=PEEK(36872)
110 IF N<20 THEN T=90-2.1*N
120 T=60-0.55*N
130 PRINT T
140 RETURN
```

Anmerkungen zu diesem Programm:

Zeile 10: Setzt PB7 der Anwender-Schnittstelle als Ausgang.

Zeile 20: Setzt PB7=»1«, d. h. stellt die Heizung an.

Zeile 30: Der Sollwert wird auf 50°C eingestellt.

Zeile 40: Springt zum Unterprogramm zur Temperaturmessung. Das Ergebnis wird der Variablen T zugeordnet.

Zeile 50: Der Computer prüft, ob die Temperatur zu hoch ist. Wenn ja, springt das Programm zurück nach Zeile 20, in der die Heizung abgestellt wird.

Zeile 60: Wenn die Temperatur zu niedrig ist, wird PB7=»0« gesetzt, d. h. die Heizung wird eingeschaltet

Zeile 70: Rücksprung nach Zeile 40, der Temperaturmessung.

Zeile 100: Den Wert N vom POT-X-Register holen.

Zeile 130: Die wirkliche Temperatur (Istwert) auf dem Bildschirm ausgeben.

Beachten Sie bitte, daß die Ausdrücke für die Linearisierung in Zeilen 110 und 120 vermutlich geändert werden müssen, um sie Ihrem Thermistor anzupassen (vgl. Kapitel 4). Machen Sie Probeläufe mit dem Programm, benutzen Sie verschiedene Sollwerte und stellen Sie sicher, daß Ihr Regelgerät wirklich funktioniert.

Vorschläge für den weiteren Ausbau der Temperaturreglung:

1. Der Vorteil des Computers im Vergleich zu einem einfachen Ein-/Ausreg-

ler, den man mit einem Operationsverstärker bauen kann, ist, daß man den Sollwert in Abhängigkeit von der Zeit, entsprechend einem im voraus festgelegten Temperaturprofil verändern kann. Ändern Sie das Programm so ab, daß 2 Minuten lang eine Temperatur von 40°C eingehalten wird, dann 2 Minuten 45°C und dann 3 Minuten lang 50°C. Tip: Benutzen Sie den DATA-Befehl, um Temperatur und Zeit festzulegen, und die eingebaute Uhr zur Zeitmessung.

2. Verändern Sie das Programm so, daß gleichzeitig zwei Öfen geregelt werden.

Steuerung eines Schrittmotors

Bis vor wenigen Jahren mußte man eine Menge Elektronik zur Steuerung eines Schrittmotors aufwenden. Heute gibt es IC's, die alle Signale zur Steuerung eines Schrittmotors erzeugen und obendrein den Verstärker zum Anschluß an die Motorwicklung enthalten.

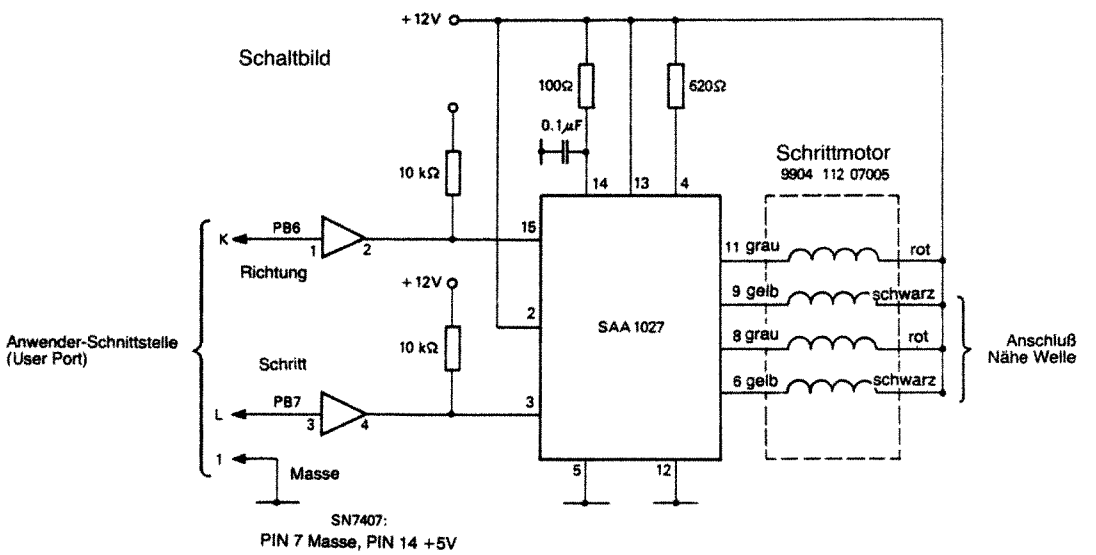


Abb. 8.10 Antriebselektronik eines Schrittmotors

In diesem Abschnitt wollen wir einen kleinen Schrittmotor von Philips, 9904 112 1705, mit dem Bauelement SAA1027, auch von Philips, steuern. Motor und Steuerelektronik erfordern eine Versorgung von +12V, für die ein gesondertes Netzteil erforderlich ist. Die Antriebselektronik hat zwei digitale Eingänge zur Steuerung des Motors. Der eine Eingang steuert die Drehrichtung, der andere verändert die Drehzahl Schritt für Schritt. Beide Steuersignale kann der Computer ohne Probleme erzeugen. Die einzige Schwierigkeit besteht darin, daß das Bauteil SAA1027 für »1« +12V und für »0« 0V verlangt. Dies ist kein großes Problem, ein TTL-Baustein SN7407 zwischen Computer und Antriebslektronik führt die Niveauveränderung aus, Abb. 8.10.

Ein Programm, das den Computer N Takte in Richtung R, mit konstanter Taktfrequenz, erzeugen läßt, sieht folgendermaßen aus (für den C-64 müssen die Adressen in Zeilen 10, 50 und 70 geändert werden):

```
10 POKE 37138,192
20 N=45
30 R=64
40 FOR X=1 TO N
50 POKE 37136,(128+R)
60 FOR T=0 TO 100 : NEXT T
70 POKE 37136,R
80 FOR T=0 TO 100 : NEXT T
90 NEXT X
```

Zeile 10: Setzt PB6 und PB7 der Anwender-Schnittstelle als Ausgang.

Zeile 20: N ist die Schrittzahl - hier 45 Schritte gewählt.

Zeile 30: R ist die Richtung. R=0 ergibt Drehung in der einen Richtung, R=64 in der anderen Richtung (PB6 hat die Gewichtung 64).

Zeile 40: Ergibt zusammen mit Zeile 90 die FOR...NEXT-Schleife, die N Takte erzeugt.

Zeile 50: Abhängig von R wird eine »1« über PB7 und eine »1« oder »0« über PB6 erzeugt.

Zeile 60: Zeitverzögerung von 0,1 Sekunden.

Zeile 70: Abhängig von R wird eine »0« über PB7 und eine »1« oder »0« über PB6 erzeugt.

Zeile 80: Zeitverzögerung von 0,1 Sekunden.

Die Pulsdauer wird hauptsächlich in Zeilen 60 und 80 bestimmt. Sie beträgt ca. 0,2 Sekunden. Sie kann verkürzt werden und beträgt ohne Zeilen 60 und 80 ca. 15 ms, das Schnellste, das in Basic einstellbar ist. Der Motor verträgt jedoch höhere Taktfolgen. Der Interessierte findet hier also ein typisches Beispiel für den Einsatz der Maschinensprache.

Weitere Ausbaumöglichkeiten:

1. Lassen Sie den Computer den Motor nach den Werten einer Tabelle steuern, d. h. Taktfolge, Drehrichtung usw.
2. Erweitern Sie die Tabelle, so daß auch die Taktfrequenz enthalten ist.
3. Erstellen Sie ein Programm, das die Drehzahl des Motors nach einer Stufenfunktion beschleunigt.

ANHANG A

Register der Mikroprozessoren 6502 und 6510

VC-20 C-64

Register der Mikroprozessoren 6502 und 6510

Die Mikroprozessoren 6502 und 6510 haben 6 Register, von denen eines 16 Bits und die übrigen 8 Bits enthalten.

Der Akkumulator

Der Akkumulator ist das wichtigste Register des Prozessors. Mit Hilfe von Befehlen in Maschinensprache kann der Inhalt von Speicherplätzen in den Akkumulator übertragen werden. Der Inhalt des Akkumulators kann auch in einen Speicherplatz übertragen und außerdem verändert werden. Der Akkumulator ist das einzige Register, in dem arithmetische Berechnungen ausgeführt werden können.

Indexregister X

Das Indexregister X kann wie der Akkumulator benutzt werden, es können jedoch keine arithmetischen Berechnungen ausgeführt werden. Einige Adressiermethoden benutzen es, um Adressen anzugeben.

Indexregister Y

Das Indexregister Y wird wie das Indexregister X verwendet.

Das Statusregister

Das Statusregister enthält 8 »Flags«, genannt N, Z, C, I und V. Jedes Flag zeigt an, ob ein bestimmtes Ereignis eingetreten ist oder nicht. Flag N wird auf »1« gesetzt, wenn ein Ereignis negativ ist. Flag Z wird auf »1« gesetzt, wenn das Ergebnis 0 ist. Flag C ist ein Erinnerungswert, der auf »1« gesetzt wird, wenn ein Überlauf der 8 Bits eintritt. Flag I steuert die Unterbrechung, und Flag D ist ein Indikator in Zusammenhang mit negativen Zahlen.

Der Programmschritt-Zähler

Der Programmschritt-Zähler oder Befehlszähler ist das einzige 16-Bit-Register. Es deutet auf die Adresse, in welcher der als nächstes auszuführende Befehl in Maschinensprache steht. Der Befehlszähler bearbeitet Adresse nach Adresse. Bestimmte Befehle, z. B. JMP (Sprung), können ihn auf einen neuen Wert setzen, und der Prozessor springt zu einem anderen Teil des Programms.

Der Stapelzeiger

Der Stapelzeiger deutet auf den ersten freien Platz im Stapel. Der Stapel ist ein Platz im Speicher, der für kurzzeitige Speicherung wichtiger Daten verwendet wird.

ANHANG B

Befehlssatz der Mikroprozessoren 6502 und 6510

VC-20 C-64

Befehlssatz der Mikroprozessoren 6502 und 6510

In der Befehlsliste werden folgende Bezeichnungen verwendet:

A	Akkumulator
X, Y	Index-Register
M	Speicher
P	Statusregister
S	Stapelzeiger
✓	geändert
-	keine Änderung
+	Addition
∧	Logisches UND
−	Subtraktion
∀	Logisches Exklusiv-ODER
↑	Bringe zum Stapelzeiger
↓	Bringe vom Stapelzeiger
← →	Daten werden in Richtung Pfeil transferiert
V	Logisches ODER
PC	Befehlszähler
PCH	Die höchstwertigen 8 Bits des Befehlszählers
PCL	Die niederwertigen 8 Bits des Befehlszählers
Oper	Operand
#	Adressierungsart "direkt"

ADC *(ADC Add memory to accumulator with carry)* ADC

Addiere Speicher zum Akkumulator mit Übertrag

Operation: A + B + C → A, C

N Z C I D V
✓ ✓ ✓ - - ✓

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	ADC Oper	105	2	2
Seite 0	ADC Oper	101	2	3
Seite 0, X	ADC Oper, X	117	2	4
Absolut	ADC Oper	109	3	4
Absolut, X	ADC Oper, X	125	3	4*
Absolut, Y	ADC Oper, Y	121	3	4*
(Indirekt, X)	ADC (Oper, X)	97	2	6
(Indirekt), Y	ADC (Oper), Y	113	2	5*

* Addiere 1, wenn Seitengrenze überschritten wird

AND *("AND" memory with accumulator)* AND

Logisches UND zwischen Speicher und Akkumulator)

Operation: A ∧ M → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	AND#Oper	41	2	2
Seite 0	AND Oper	37	2	3
Seite 0, X	AND Oper, X	53	2	4
Absolut	AND Oper	45	3	4
Absolut, X	AND Oper, X	61	3	4*
Absolut, Y	AND Oper, Y	57	3	4*
(Indirekt, X)	AND (Oper, X)	33	2	6
(Indirekt), Y	AND (Oper), Y	49	2	5

* Addiere 1, wenn Seitengrenze überschritten wird

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

Verschiebe nach links (Speicher oder Akkumulator)

Operation: C ← 7 6 5 4 3 2 1 0 ← 0

N Z C I D V
✓ ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Akkumulator	ASL A	10	1	2
Seite 0	ASL Oper	6	2	5
Seite 0, X	ASL Oper, X	22	2	6
Absolut	ASL Oper	14	3	6
Absolut, X	ASL Oper, X	30	3	7

BCC *(BCC Branch on Carry Clear)* BCC

Verzweige, wenn das Übertrags-Flag gelöscht ist

Operation: Verzweige, wenn C=0

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BCC Oper	144	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BCS *(BCS Branch on carry set)* BCS

Verzweige, wenn das Übertrags-Flag gesetzt ist

Operation: Verzweige, wenn C=1

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BCS Oper	176	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BEQ *(BEQ Branch on result zero)* BEQ

Verzweige, wenn das Ergebnis gleich Null ist

Operation: Verzweige, wenn Z=1

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BEQ Oper	240	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BIT *(BIT Test Bits in memory with accumulator)* BIT

Bits im Speicher mit dem Akkumulator testen

Operation: A ∧ M, M_7 → N, M_6 → V

Bits 6 und 7 werden in das Statusregister überführt. Wenn das Ergebnis von A ∧ M Null ist, wird Z=1 gesetzt, sonst Z=0.

N Z C I D V
M_7 ✓ - - - M_6

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	BIT Oper	36	2	3
Absolut	BIT Oper	44	3	4

BMI *(BMI Branch on result minus)* BMI

Verzweige, wenn das Ergebnis negativ ist

Operation: Verzweige, wenn N=1

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BMI Oper	30	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BNE *(BNE Branch on result not zero)* BNE

Verzweige, wenn das Ergebnis nicht gleich Null ist

Operation: Verzweige, wenn Z≠0

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BNE Oper	208	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BPL *(BPL Branch on result plus)* BPL

Verzweige, wenn das Ergebnis positiv ist

Operation: Verzweige, wenn N=0

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BPL Oper	16	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BRK *(BRK Force Break)* BRK

Programmierte Unterbrechung

Operation: Programmierte Unterbrechung PC +2 ↓ P ↓

N Z C I D V
- - - 1 - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	BRK	0	1	7

BVC *(BVC Branch on overflow clear)* BVC

Verzweige, wenn das Überlauf-Flag gelöscht ist

Operation: Verzweige, wenn V=0

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BVC Oper	80	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

BVS *(BVS Branch on overflow set)* BVS

Verzweige, wenn das Überlauf-Flag gesetzt ist

Operation: Verzweige, wenn V=1

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Bedingte Verzweigung	BVS Oper	112	2	2*

* Addiere 1 bei Verzweigung auf derselben Seite
* Addiere 2 bei Verzweigung zur anderen Seite

CLC *(CLC Clear Carry Flag)* CLC

Lösche Übertrags-Flag

Operation: O → C

N Z C I D V
- - 0 - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	CLC	24	1	2

CLD *(CLD Clear decimal mode)* CLD

Lösche Dezimal-Betrieb

Operation: O → D

N Z C I D V
- - - - 0 -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	CLD	216	1	2

CLI *(CLI Clear interrupt disable bit)* CLI

Lösche Sperrbits im Status-Register

Operation: O → I

N Z C I D V
- - - 0 - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	CLI	88	1	2

CLV *(CLV Clear overflow flag)* CLV

Lösche Überlauf-Flag

Operation: O → V

N Z C I D V
- - - - - 0

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	CLV	184	1	2

CMP *(CMP Compare memory and accumulator)* CMP

Vergleiche Speicher mit Akkumulator

Operation: A – M

N Z C I D V
✓ ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	CMP Oper	201	2	2
Seite 0	CMP Oper	197	2	3
Seite 0, X	CMP Oper, X	213	2	4
Absolut	CMP Oper	205	3	4
Absolut, X	CMP Oper, X	221	3	4*
Absolut, Y	CMP Oper, Y	217	3	4*
(Indirekt, X)	CMP (Oper, X)	193	2	6
(Indirekt), Y	CMP (Oper), Y	209	2	5*

* Addiere 1, wenn Seitengrenze überschritten wurde

CPX *(CPX Compare Memory and Index X)* CPX

Vergleiche Speicher mit Index-Register X

Operation: X – M

N Z C I D V
✓ ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	CPX Oper	224	2	2
Seite 0	CPX Oper	228	2	3
Absolut	CPX Oper,	236	3	4

CPY *(CPY Compare Memory and index Y)* CPY

Vergleiche Speicher mit Index-Register Y

Operation: Y – M

N Z C I D V
✓ ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	CPY Oper	192	2	2
Seite 0	CPY Oper	196	2	3
Absolut	CPY Oper,	203	3	4

DEC *(DEC Decrement memory by one)* DEC

Speicher um 1 reduzieren

Operation: M – 1 → M

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	DEC Oper	198	2	5
Seite 0, X	DEC Oper, X	214	2	6
Absolut	DEC Oper	206	3	6
Absolut, X	DEC Oper, X	222	3	7

DEX *(DEX Decrement index X by one)* DEX

Index-Register X um 1 reduzieren

Operation: X – 1 → X

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	DEX	202	1	2

DEY *(DEY Decrement index Y By one)* DEY

Index-Register Y um 1 reduzieren

Operation: Y – 1 → Y

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	DEY	136	1	2

EOR *(EOR "Exclusive-Or" memory with accumulator)* EOR

Logisches EXKLUSIV-ODER zwischen Speicher und Akkumulator

Operation: A⊻M → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	EOR # Oper	73	2	2
Seite 0	EOR Oper	69	2	3
Seite 0, X	EOR Oper, X	85	2	4
Absolut	EOR Oper	77	3	4
Absolut, X	EOR Oper, X	93	3	4*
Absolut, Y	EOR Oper, Y	89	3	4*
(Indirekt, X)	EOR (Oper, X)	65	2	6
(Indirekt), Y	EOR (Oper), Y	81	2	5*

* Addiere 1, wenn Seitengrenzen überschritten werden

INC *(INC Increment memory by one)* INC

Speicher um 1 erhöhen

Operation: M + 1 → M

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	INC Oper	230	2	5
Seite 0, X	INC Oper, X	246	2	6
Absolut	INC Oper	238	3	6
Absolut, X	INC Oper, X	254	3	7

INX *(INX Increment Index X by one)* INX

Index-Register X um 1 erhöhen

Operation: X + 1 → X

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	INX	232	1	2

INY *(INY Increment Index Y by one)* INY

Index-Register Y um 1 erhöhen

Operation: Y + 1 → Y

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	INY	200	1	2

JMP *(JMP Jump to new location)* JMP

Springe an neue Adresse

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

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Absolut	JMP Oper	76	3	3
Indirekt	JMP (Oper)	108	3	5

JSR *(JSR Jump to new location saving return address)* JSR

Springe zum Unterprogramm

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

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Absolut	JSR Oper	32	3	6

LDA *(LDA Load accumulator with memory)* LDA

Lade Akkumulator vom Speicher

Operation: M → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	LDA#Oper	169	2	2
Seite 0	LDA Oper	165	2	3
Seite 0, X	LDA Oper, X	181	2	4
Absolut	LDA Oper	173	3	4
Absolut, X	LDA Oper, X	189	3	4*
Absolut, Y	LDA Oper, Y	185	3	4*
(Indirekt, X)	LDA (Oper, X)	161	2	6
(Indirekt), Y	LDA (Oper), Y	177	2	5*

* Addiere 1, wenn Seitengrenze überschritten wurde

LDX *(LDX Load index X with memory)* LDX

Lade Index-Register X vom Speicher

Operation: M → X

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	LDX Oper	162	2	2
Seite 0	LDX Oper	166	2	3
Seite 0, Y	LDX Oper, Y	182	2	4
Absolut	LDX Oper	174	3	4
Absolut, Y	LDX Oper, Y	190	3	4*

* Addiere 1, wenn Seitengrenze überschritten wurde

LDY *(LDY Load index Y with memory)* LDY

Lade Index-Register Y vom Speicher

Operation: M ➡ Y

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	LDY # Oper	160	2	2
Seite 0	LDY Oper	164	2	3
Seite 0, X	LDY Oper, X	180	2	4
Absolut	LDY Oper	172	3	4
Absolut, X	LDY Oper, X	188	3	4*

* Addiere 1, wenn Seitengrenze überschritten wurde

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

Verschiebe Inhalt des Akkumulators logisch nach rechts

Operation: 0 → 7 6 5 4 3 2 1 0 → C

N Z C I D V
0 ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	LSR A	74	1	2
Seite 0	LSR Oper	70	2	5
Seite 0, X	LSR Oper, X	86	2	6
Absolut	LSR Oper	78	3	6
Absolut, X	LSR Oper, X	94	3	7

NOP *(NOP No Operation)* NOP

Keine Operation

Operation: keine Operation

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	NOP	234	1	2

ORA *(ORA "OR" memory with accumulator)* ORA

Logisches ODER zwischen Speicher und Akkumulator

Operation: A ∨ M → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	ORA # Oper	9	2	2
Seite 0	ORA Oper	5	2	3
Seite 0, X	ORA Oper, X	21	2	4
Absolut	ORA Oper	13	3	4
Absolut, X	ORA Oper, X	29	3	4*
Absolut, Y	ORA Oper, Y	25	3	4*
(Indirekt, X)	ORA (Oper, X)	1	2	6
(Indirekt), Y	ORA (Oper), Y	17	2	5

* Addiere 1, wenn Seitengrenze überschritten wurde

PHA *(PHA Push accumulator on stack)* PHA

Bringe Akkumulator auf Stapel

Operation: A ↓

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	PHA	72	1	3

PHP *(PHP Push processor status on stack)* PHP

Bringe Status-Register auf Stapel

Operation: P ↓

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	PHP	8	1	3

PLA *(PLA Pull accumulator from stack)* PLA

Lade Akkumulator von der Spitze des Stapels

Operation: A ↑

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	PLA	104	1	4

PLP *(PLP Pull processor status from stack)* PLP

Lade Status-Register von der Spitze des Stapels

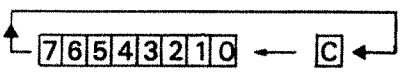
Operation: P ↑

N Z C I D V
vom Stapel

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	PLP	40	1	4

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

Rotiere ein Bit nach links (Speicher oder Akkumulator)

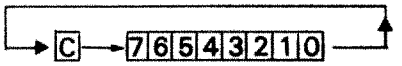
Operation: 

N Z C I D V
- - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Akkumulator	ROL A	42	1	2
Seite 0	ROL Oper	38	2	5
Seite 0, X	ROL Oper, X	54	2	6
Absolut	ROL Oper	46	3	6
Absolut, X	ROL Oper, X	62	3	7

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

Rotiere ein Bit nach rechts (Speicher oder Akkumulator)

Operation: 

N Z C I D V
✓ ✓ ✓ - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Akkumulator	ROR A	106	1	2
Seite 0	ROR Oper	102	2	5
Seite 0, X	ROR Oper, X	118	2	6
Absolut	ROR Oper	110	3	6
Absolut, X	ROR Oper, X	126	3	7

RTI *(RTI Return from interrupt)* RTI

Kehre von Unterbrechung zurück

Operation: P ↑ PC ↑

N Z C I D V
vom Stapel

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	RTI	64	1	6

RTS *(RTS Return from subroutine)* RTS

Kehre vom Unterprogramm zurück

Operation: PC ↑ , PC + 1 → PC

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	RTS	96	1	6

SBC *(SBC Subtract memory from accumulator with borrow)* SBC

Subtrahiere Speicher vom Akkumulator mit Übertrag

Operation: A − M − $\overline{C}$ → A
$\overline{C}$ = übertragene Zahl

N Z C I D V
✓ ✓ ✓ - - ✓

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Unmittelbar	SBC Oper	233	2	2
Seite 0	SBC Oper	229	2	3
Seite 0, X	SBC Oper, X	245	2	4
Absolut	SBC Oper	237	3	4
Absolut, X	SBC Oper, X	253	3	4*
Absolut, Y	SBC Oper, Y	249	3	4*
(Indirekt, X)	SBC (Oper, X)	225	2	6
(Indirekt), Y	SBC (Oper), Y	241	2	5*

* Addiere 1, wenn Seitengrenze überschritten wurde

SEC *(SEC Set carry flag)* SEC

Setze Übertrags-Flag

Operation: 1 → C

N Z C I D V
- - 1 - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	SEC	56	1	2

SED *(SED Set decimal mode)* SED

Setze Dezimal-Betrieb

Operation: 1 → D

N Z C I D V
- - - - 1 -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	SED	248	1	2

SEI *(SEI Set interrupt disable status)* SEI

Sperre Unterbrechnungen

Operation: 1 → I

N Z C I D V
- - - 1 - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	SEI	120	1	2

STA *(STA Store accumulator in memory)* STA

Speichere Akkumulator im Speicher

Operation: A → M

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	STA Oper	133	2	3
Seite 0, X	STA Oper, X	149	2	4
Absolut	STA Oper	141	3	4
Absolut, X	STA Oper, X	157	3	5
Absolut, Y	STA Oper, Y	153	3	5
(Indirekt, X)	STA (Oper, X)	129	2	6
(Indirekt), Y	STA (Oper), Y	145	2	6

STX *(STX Store index X in memory)* STX

Speichere Index-Register X im Speicher

Operation: X → M

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	STX Oper	134	2	3
Seite 0, Y	STY Oper, Y	150	2	4
Absolut	STX Oper	142	3	4

STY *(STY Store index Y in memory)* STY

Speichere Index-Register Y im Speicher

Operation: Y → M

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Seite 0	STY Oper	132	2	3
Seite 0, X	STY Oper, X	148	2	4
Absolut	STY Oper	140	3	4

TAX *(TAX Transfer accumulator to index X)* TAX

Bringe Akkumulator zum Index-Register X

Operation: A → X

N Z C I D V
✓✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TAX	170	1	2

TAY (TAY Transfer accumulator to index Y) TAY

Bringe Akkumulator zum Index-Register Y

Operation: A → Y

N Z C I D V
✓✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TAY	168	1	2

TSX *(TSX Transfer stack pointer to index X)* TSX

Bringe Stapelzeiger zum Index-Register X

Operation: S → X

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TSX	186	1	2

TXA *(TXA Transfer index X to accumulator)* TXA

Bringe Index-Register X zum Akkumulator

Operation: X → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TXA	138	1	2

TXS *(TXS Transfer index X to stack pointer)* TXS

Bringe Index-Register X zum Stapelzeiger

Operation: X → S

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

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TXS	154	1	2

TYA *(TYA Transfer index Y to accumulator)* TYA

Bringe Index-Register Y zum Akkumulator

Operation: Y → A

N Z C I D V
✓ ✓ - - - -

Adressierungsart	Mnemonik	OP-Code	Bytes	Zeit-perioden
Selbstverst.	TYA	152	1	2

ANHANG C

Tabelle für Binär-zahlen und deren Dezimaldarstellung

VC-20 C-64

Tabelle für Binärzahlen und deren Dezimaldarstellung

Dezimal	Binär	Dezimal	Binär
0	0	41	10 1001
1	1	42	10 1010
2	10	43	10 1011
3	11	44	10 1100
4	100	45	10 1101
5	101	46	10 1110
6	110	47	10 1111
7	111	48	11 0000
8	1000	49	11 0001
9	1001	50	11 0010
10	1010	51	11 0011
11	1011	52	11 0100
12	1100	53	11 0101
13	1101	54	11 0110
14	1110	55	11 0111
15	1111	56	11 1000
16	1 0000	57	11 1001
17	1 0001	58	11 1010
18	1 0010	59	11 1011
19	1 0011	60	11 1100
20	1 0100	61	11 1101
21	1 0101	62	11 1110
22	1 0110	63	11 1111
23	1 0111	64	100 0000
24	1 1000	65	100 0001
25	1 1001	66	100 0010
26	1 1010	67	100 0011
27	1 1011	68	100 0100
28	1 1100	69	100 0101
29	1 1101	70	100 0110
30	1 1110	71	100 0111
31	1 1111	72	100 1000
32	10 0000	73	100 1001
33	10 0001	74	100 1010
34	10 0010	75	100 1011
35	10 0011	76	100 1100
36	10 0100	77	100 1101
37	10 0101	78	100 1110
38	10 0110	79	100 1111
39	10 0111	80	101 0000
40	10 1000	81	101 0001

Dezimal	Binär	Dezimal	Binär
82	101 0010	126	111 1110
83	101 0011	127	111 1111
84	101 0100	128	1000 0000
85	101 0101	129	1000 0001
86	101 0110	130	1000 0010
87	101 0111	131	1000 0011
88	101 1000	132	1000 0100
89	101 1001	133	1000 0101
90	101 1010	134	1000 0110
91	101 1011	135	1000 0111
92	101 1100	136	1000 1000
93	101 1101	137	1000 1001
94	101 1110	138	1000 1010
95	101 1111	139	1000 1011
96	110 0000	140	1000 1100
97	110 0001	141	1000 1101
98	110 0010	142	1000 1110
99	110 0011	143	1000 1111
100	110 0100	144	1001 0000
101	110 0101	145	1001 0001
102	110 0110	146	1001 0010
103	110 0111	147	1001 0011
104	110 1000	148	1001 0100
105	110 1001	149	1001 0101
106	110 1010	150	1001 0110
107	110 1011	151	1001 0111
108	110 1100	152	1001 1000
109	110 1101	153	1001 1001
110	110 1110	154	1001 1010
111	110 1111	155	1001 1011
112	111 0000	156	1001 1100
113	111 0001	157	1001 1101
114	111 0010	158	1001 1110
115	111 0011	159	1001 1111
116	111 0100	160	1010 0000
117	111 0101	161	1010 0001
118	111 0110	162	1010 0010
119	111 0111	163	1010 0011
120	111 1000	164	1010 0100
121	111 1001	165	1010 0101
122	111 1010	166	1010 0110
123	111 1011	167	1010 0111
124	111 1100	168	1010 1000
125	111 1101	169	1010 1001

Dezimal	Binär	Dezimal	Binär
170	1010 1010	213	1101 0101
171	1010 1011	214	1101 0110
172	1010 1100	215	1101 0111
173	1010 1101	216	1101 1000
174	1010 1110	217	1101 1001
175	1010 1111	218	1101 1010
176	1011 0000	219	1101 1011
177	1011 0001	220	1101 1100
178	1011 0010	221	1101 1101
179	1011 0011	222	1101 1110
180	1011 0100	223	1101 1111
181	1011 0101	224	1110 0000
182	1011 0110	225	1110 0001
183	1011 0111	226	1110 0010
184	1011 1000	227	1110 0011
185	1011 1001	228	1110 0100
186	1011 1010	229	1110 0101
187	1011 1011	230	1110 0110
188	1011 1100	231	1110 0111
189	1011 1101	232	1110 1000
190	1011 1110	233	1110 1001
191	1011 1111	234	1110 1010
192	1100 0000	235	1110 1011
193	1100 0001	236	1110 1100
194	1100 0010	237	1110 1101
195	1100 0011	238	1110 1110
196	1100 0100	239	1110 1111
197	1100 0101	240	1111 0000
198	1100 0110	241	1111 0001
199	1100 0111	242	1111 0010
200	1100 1000	243	1111 0011
201	1100 1001	244	1111 0100
202	1100 1010	245	1111 0101
203	1100 1011	246	1111 0110
204	1100 1100	247	1111 0111
205	1100 1101	248	1111 1000
206	1100 1110	249	1111 1001
207	1100 1111	250	1111 1010
208	1101 0000	251	1111 1011
209	1101 0001	252	1111 1100
210	1101 0010	253	1111 1101
211	1101 0011	254	1111 1110
212	1101 0100	255	1111 1111

Liber

ISBN 91-40-20683-1

HALLER VERLAG
Bahnhofstraße 80
6600 Saarbrücken
Tel. 06 81 / 3 69 81
Telex 4 421 446
Bestell-Nr.: 3-92 40 28-03-6

ANHANG D

Messen, steuern, regeln

mit dem

VC 20

und

C64 commodore 64

>Bauteilempfehlungen
>Änderungshinweise
>Ergänzungen
>Liste der Fachbegriffe

ISBN 91-40-20683-1 Anhang D

VC-20 C-64

BAUTEILEMPFEHLUNGEN, ÄNDERUNGSHINWEISE UND ERGÄNZUNGEN

Seite 17 / Abb. 1.7:

Für VC-20: PIN 10 und 11 9 V AC (max 50 mA)
Für C-64: PIN B "FLAG 2"

Seite 21 / Abb. Tabelle der PIN-Belegung:

	VC-20/C-64	DIP	Signal
Ändern	H	5	PB4
"	7	13	Lichtgriffel (SP2)
"	11	10	9 V AC
Hinzufügen	10	--	9 V AC

Seite 24 / Abb. 2.4:

Masse liegt an Anschluß 11.

Seite 27 / Abb. 2.7:

Stift 10 in 11 des DIP-Steckers ändern.

Seite 38:

Vor dem letzten Absatz ist einzuschieben:
"Es muß eine Verzögerung von 300 ms zwischen den Ziffern vorgesehen werden, damit die Vermittlungsstelle feststellen kann, wann eine Ziffer gewählt wurde und daß der nächste Takt von einer neuen Ziffer herstammt.

Seite 39 - Geändertes Programm:

```
10 POKE 37138,1                    PB0 Ausgang
20 READ N                          Ziffern lesen
30 IF N<0 THEN 130
40 FOR I=0 TO N
50 POKE 37136,1
60 FOR J=1 TO 60 : NEXT J          Taktlänge erzeugen
70 POKE 37136,0
80 FOR J=1 TO 40 : NEXT J
90 NEXT I
100 FOR J=1 TO 300 : NEXT J        300 ms warten
110 GOTO 20                        zum Programmbeginn springen
120 DATA 1,2,3,4,5,6,7,-1          und erneut beginnen
130 END
```

Seite 43 / Abb. 3.11:

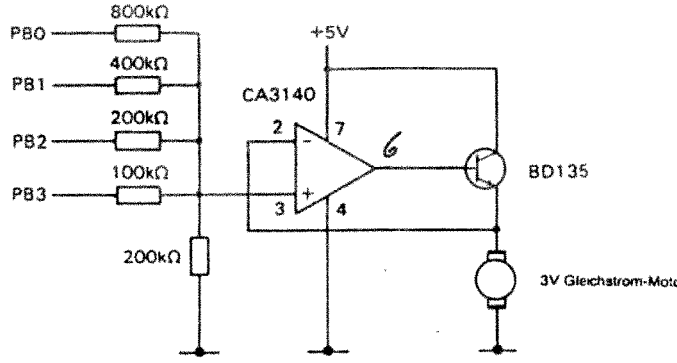


Abb. 3.11 Steuerung eines Gleichstrom-Motors

Seite 44 - Änderungen:

```
10 POKE 37138,15
20 POKE 37136,15
```

"Wenn Sie dieses Programm laufen lassen, sollte der Motor mit maximaler Drehzahl laufen. Wird in Zeile 20 die Zahl 15 auf 10 geändert, läuft der Motor bedeutend langsamer. Probieren Sie es aus!"

Seite 46 / Abb. 4.2:
Fototransistor FPT 100 kann ersetzt werden durch BPY 62 II von Siemens.

Seite 47 / Abb. 4.3:
Dito wie Seite 46.

Seite 50:
Vorletzter Satz muß lauten:
"Der Takt ist mit CNT 2 (Anschluß 6) zu verbinden.

Seite 51:
Zweiter Absatz entfällt und wird ersetzt durch:
"Im C-64 reicht es nicht, Bits im Kontrollregister 56590 auf "eins" zu setzen. Bit Ø wird als Start- und Stopbit benutzt. Eine "Eins" startet den Rechner und eine "Null" hält ihn an. Für den C-64 sieht das Programm folgendermaßen aus:"

```
10 POKE 56580,255
20 POKE 56581,255
30 A=PEEK(56580)
40 B=PEEK(56581)
50 POKE 56590,PEEK(56590) OR 33        Bit 5 und Bit Ø werden auf "eins" gesetzt
60 PRINT 65535-(256xB+A)
70 GOTO 30
```

Die Taktfrequenz wird mit CNT 2 gerechnet.

Absatz "Frequenzmessung":

Letzter Satz muß heißen:

"Das Programm ist für den VC-20 geschrieben."

Seite 52:

Anzuschließen an Kapitel "Frequenzmessung":

"In Übereinstimmung mit dem vorher Gesagten, muß des Programm geändert werden, um auf dem C-64 zu laufen. Es reicht nicht aus, den Zähler des C-64 auf Null zu stellen. Er muß vorher angehalten werden. Wir müssen wiederum Bit Ø ändern.

Vorschlag für ein C-64-Programm:

```
10 POKE 56580,255
20 POKE 56581,255
30 T=TI
40 IF T=TI THEN 40
50 POKE 56590,PEEK(56590) OR 33
60 IF TI <>T+60 THEN 60
70 POKE 56590,PEEK(56590) AND 254
80 PRINT 65535-(256*PEEK(56581)+PEEK(56580))
90 GOTO 10
```

Zeile 70 erfordert vermutlich eine Erklärung:

$(254)_{10} = (11111110)_2$.

Die AND-Funktion führt dazu, daß nur Bit Ø verändert wird. Die übrigen sieben Bits werden nicht beeinflußt.

Seite 59 / Hinweis zu Abb. 4.11:

Sie müssen natürlich für den von Ihnen benutzten Thermistor selbst eine Eichkurve aufstellen.

Seite 60:

3. Zeile von oben ändern wie folgt:

t = 60 - 0,55 * N für N > 20

Programmzeile 20 muß lauten:

```
20 IF N<20 THEN PRINT 90-2.1*N : GOTO 10
```

Seite 70:

POKE 49155,176	soll lauten:	POKE 49155,1
POKE 49156,29	soll lauten:	POKE 49156,221

Seite 71:

60 DATA 169,128,141,176,29,96 soll lauten: 60 DATA 169,128,141,1,221,96

Seite 73:

Änderung der Programmzeile in:

60 DATA 169,128,141,1,221,169,0,141,1,221,76,0,192

Seite 78 / Abb. 6.4:

In der Tabelle unten auf Seite 78 soll das D in der fünften Zeile von unten in G geändert werden.

Seite 79: Das Schaltschema Abbildung 6.4 auf Seite 79 wie folgt abändern:

Eingang 2 und 3 an LM 741 sind zu vertauschen. Ein Transistor BC 337 oder Ähnliches sowie zwei Widerstände von 10 KΩ sind einzufügen.

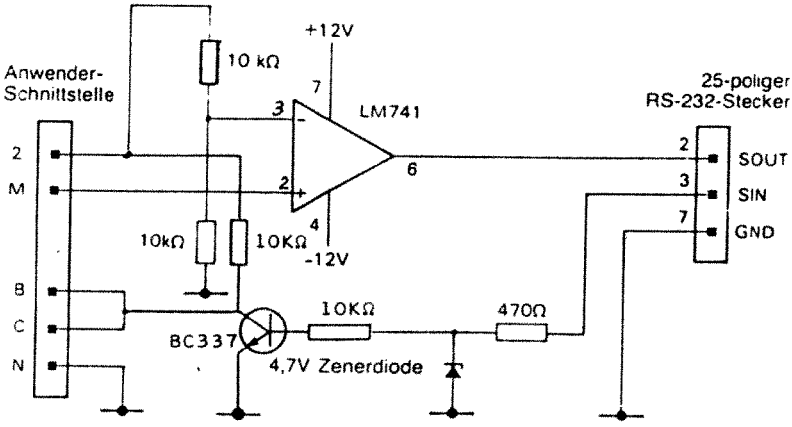


Abb. 6.4.

Seite 80: Für VC-20: Das Schaltschema Abb. 6.5 auf Seite 80 muß wie folgt abgeändert werden:
Der Ausgang mit Bezeichnung 7 muß mit 5 bezeichnet werden.

Für Commodore 64: Im C-64 muß erst die Wechselspannung von 9V zwischen 10 und 11 gleichgerichtet werden, z. B. mit einer Brückenschaltung.
Die fertige RS-232-Schnittstelle kann als Zubehör gekauft werden.

Seite 83: Nachtrag: Vor dem letzten Absatz nach Print # 2, B $ ist folgendes Programm einzufügen; es verwandelt den VC-20 oder C-64 in ein Terminal.

```
10  PRINT CHR $ (147)
20  OPEN 2,2,0, CHR $ (38) + CHR $ (160)
30  GET #2, A $
40  IF A $ = " " THEN 100
50  A = ASC (A $)
60  A = A AND 127
70  IF A >96 THEN A = A - 32
80  PRINT CHR $ (A);
90  GOTO 30
100 GET B $
110 IF B $ = " " THEN 30
120 PRINT #2, B $;
130 GOTO 30
```

Seite 89 / Abb. 7.1:

Für Kontakt 6 muß das Signal lauten: "Serielle Uhr ein/aus".

Abb. 7.1 Steckerbuchse des IEEE-Busses

Seite 94 / Abb. 8.3:

Statt FPT 100 kann ein BPY 62 II von Siemens benutzt werden. Es muß dann der Widerstand von 5,6 kΩ gegen einen Widerstand von 33 kΩ ausgetauscht werden. Wird ein Schwarz-weiß-Fernseher oder Monitor verwendet, ist ein möglichst heller Hintergrund zu empfehlen. Der VC-20 hat von Anfang an einen guten Farbton. Für den C-64 schreibt man POKE 53281,1 oder POKE 53281,7, wodurch ein weißer oder gelber Farbton auf dem Bildschirm erhalten wird.

Seite 98:

Letzter Satz im letzten Absatz ersetzen durch:
"Für den C-64 ändern Sie das Programm, wie im Kapital 4 angegeben."

Seite 99:

Zusatz zur Zeilenerläuterung von Zeile 10:
"Benutzen Sie CNT2 als Eingang."

Seite 102 / Programm:

Programmzeile 110 ersetzen durch:

```
110 IF N<20 THEN T=90-2.1*N : GOTO 130
```

Seite 103 / Abb. 8.10:

Kontaktbezeichnungen 15 und 3 sind miteinander zu vertauschen.

Seite 104:

Zum Ende des ersten Absatzes:
"Man kann den SN7407 auch durch eine einfache Transistorstufe ersetzen.
Das Signal wird dann invertiert."

Seite 115:

In Tabelle BMI ist in Spalte OP-Code 30 durch 48 zu ersetzen.

Seite 119:

In Tabelle CPY ist in Spalte CP-Code 203 durch 204 zu ersetzen.

Erläuterung der Fachbegriffe

VC-20 C-64

Im folgenden finden Sie eine Aufstellung mit Erläuterung einiger im Buch vorkommender Fachwörter und Abkürzungen.

ADRESSE

In der Datenverarbeitung versteht man unter Adresse einen Platz im Speicher des Computers. In diesem befindet sich eine Anzahl von Steuerzeichen, Texten oder anderen Zeichen. Damit der Computer sich in allen Zeichen zurechtfindet, hat jedes Zeichen eine Ordnungsnummer. Für die gängigen Heim-Computer benutzt man die Ordnungsnummern von 0-65535. Diese 65536 Adressen (0 eingeschlossen) enthalten Informationen über Ihr Programm, die eingegebenen Daten und außerdem all die Programme, die schon von Beginn an im Rechner sind, z. B. BASIC-Interpreter usw.

ASCII

ASCII (sprich: Aski) ist die Abkürzung für den bekanntesten Code, der in der Datenverarbeitung für Texte benutzt wird. ASCII kommt aus den USA und ist die Abkürzung für »American Standard Code for Information Interchange« und heißt: Code zur Übertragung von Informationen.

AUFLÖSUNG

Auflösung ist eigentlich der kleinstmögliche Abstand zwischen zwei auf dem Bildschirm sichtbaren Punkten. Es kann also sein, daß ein auf dem Bildschirm erscheinender Punkt in Wirklichkeit aus zwei verschiedenen Punkten besteht. Das Wort »Auflösung« wird häufig falsch benutzt, um die Anzahl der Punkte auf dem Bildschirm anzugeben, wie z. B. 200 × 320.

BAM

BAM ist eine spezielle Abkürzung für die Floppy-Disk-Einheiten VC 1540 und VC 154. Die Abkürzung steht für Block Availability Map und bezeichnet einen Teil der Diskette. BAM überwacht, welche Teile der Diskette von Programmen und Daten belegt sind und welche frei zur Verfügung stehen.

BILDSPEICHER

Dieser Teil des Speichers enthält die Codes, die über die jetzt auf dem Bildschirm zu zeigenden Zeichen Auskunft geben. Diese zeigen die Plätze (Adressen) im Zeichengenerator an. Der Zeichengenerator legt fest, wie das Zeichen aussehen soll.

BITMAP

Wenn der Computer für Grafiken benutzt werden soll, bedient er sich einer Methode, bei der jeder Punkt auf dem Bildschirm einem Bit (Binär-Ziffer) entspricht. Ein Bitmap ist also der Teil des Speichers, der Informationen über das Aussehen der grafischen Darstellung enthält.

BYTE

Byte (gesprochen: Beit) ist eine in der Datenverarbeitung häufig benutzte Bezeichnung. Ein Byte stellt eine achtziffrige Binärzahl dar (1 oder 0, die sogenannten Bits).

COMPUTER-GRAFIK

Computer-Grafik ist ein Sammelbegriff für die verschiedenen Techniken, Bilder, Texte oder andere Informationen auf Papier oder Bildschirm u. dgl. darzustellen.

DATEI

Mit Datei (engl.: file) bezeichnet man einen Kanal oder Platz auf einer Diskette oder einem anderen externen Speichergerät, in dem man Informationen und Programme abspeichern kann.

DATEINUMMER

Der VC-20 und der C-64 können mit mehreren Dateien gleichzeitig arbeiten. Um die Dateien unterscheiden zu können, nummeriert man sie mit Dateinummern (Ordnungsnummern) zwischen 0 und 255, über die man mit den betreffenden Dateien korrespondieren kann.

DATEI MIT DIREKTEM ZUGRIFF

Eine Datei mit direktem Zugriff (Random Access File) ist eine Datei, die so geöffnet wird, daß alle Datenblöcke in der Datei einzeln zugänglich sind. Diese Datei benötigt Informationen darüber, welche Spuren und Sektoren sie verwenden soll. Diese Informationen sind von BAM erhältlich.

DATENSTRUKTUR

Eine Datenstruktur ist eine geordnete Sammlung von Daten. Eine Datensammlung hat spezielle Regeln für die Ein- und Ausgabe zu und von der Datenstruktur. Beispiele für Datenstrukturen sind Vektoren, DATA-Anweisungen etc.

DIN-STECKER

Ein DIN-Stecker ist ein bezüglich Form und Funktion genormter Stecker. Dieser ist wichtig, damit zwei zusammengeschaltete Geräte zusammen funktionieren. DIN ist die Abkürzung für Deutsches Institut für Normung.

DIRECTORY (s. INHALTSVERZEICHNIS)

DISKETTE

Eine Diskette ist der Datenträger, worauf das Diskettenlaufwerk (Floppy-Disk) In-

formationen abspeichert. Der Name kommt vom englischen »flexible disk« (biegsame, flexible Scheibe).

DOS

DOS, Abkürzung für Disk Operating System, ist zuständig für die Kontrolle aller Kommandos, die zur Floppy-Disk-Einheit geschickt werden. Außerdem überwacht das DOS, daß Fehler, die entstanden sind, nicht verheerende Folgen haben. Fehlermeldungen werden vom DOS ausgegeben, wenn die Fehler von DOS nicht behoben werden können.

EINHEITSNUMMER

Einheitsnummer ist eine Nummer, welche die an VC-20 oder C-64 angeschlossenen Einheiten identifiziert. Einheitsnummer darf jede Zahl zwischen 0 und 31 sein. Wenn mehrere Floppy-Disk-Einheiten an den gleichen Computer angeschlossen werden, dürfen sie z. B. nicht die gleiche Einheitsnummer haben. Auf der Diskette VC 1541 TEST/DEMO gibt es ein Programm, DISK ADDR CHANGE genannt, das für diesen Zweck verwendet werden kann.

FARBSPEICHER

Ein Farbspeicher ist der Speicher, der Informationen über die Farbe der augenblicklich auf dem Bildschirm zu zeigenden Zeichen enthält. Bei Benutzung eines Schwarz-Weiß-Fernsehers erhält man stattdessen eine Grautonskala. Beim VC-20 befindet sich der Farbspeicher normalerweise unter den Adressen 38400-38911 (Grundausführung) oder 37888-38399 bei einem auf mehr als 3 KB erweiterten VC-20. Beim C-64 befindet sich der Farbspeicher unter den festen Adressen 55296-56295.

FLUSSDIAGRAMM

Unter einem Flußdiagramm versteht man einen Übersichtsplan, der veranschaulicht, wie das Programm Anweisungen ausführen soll. Das Flußdiagramm gibt einen guten Überblick darüber, wie der Computer die Programmaufgabe löst. Das Flußdiagramm ist nicht die einzige Methode, um ein Programm darzustellen, aber die am leichtesten zu erlernende.

INDEX

Ein Index ist eine Ordnungszahl, die eine Variable in einem Variablenfeld (Matrix) anzeigt. Eine oder mehrere Zahlen können Index sein, z. B. A$(1) oder A$(1,1), wobei A$ ein bzw. zwei Indizes hat. Beim VC-20 und C-64 sind bis zu 255 Indizes erlaubt.

INHALTSVERZEICHNIS

Ein Inhaltsverzeichnis (Directory) ist eine Liste über die Programme und Datendateien, die auf einer Diskette gespeichert sind. Das Inhaltsverzeichnis ist meistens auf einer bestimmten Stelle auf der Diskette gespeichert, damit es nicht im Wege ist, wenn Programme gespeichert werden sollen. Dies muß besonders bei Dateien mit direktem Zugriff beachtet werden, da sie sonst die Inhaltsverzeichnisspur zerstören können. Das Inhaltsverzeichnis kann mit dem Kommando LOAD "$",Einheitsnummer gelesen werden (VC-20 und C-64). Das Inhaltsverzeichnis kann auch wie eine Datei mit OPEN 1,Einheit,2,"$" eröffnet werden, um Programme o. ä. das Inhaltsverzeichnis lesen zu lassen. Das Inhaltsverzeichnis kann auch so geändert werden, daß »gelöschte« (DEL) Daten wieder zugänglich sind. Wenn eine Datei gelöscht wird, wird sie vorerst nur aus dem Inhaltsverzeichnis entfernt.

MASCHINEN-CODE

Unter Maschinen-Code versteht man das Zeichensystem, das der Zentral-Mikroprozessor (CPU) des Computers akzeptiert. Der BASIC-Interpreter ist ein in Maschinencode geschriebenes Programm, das ein in BASIC geschriebenes Programm für den Zentral-Mikroprozessor (CPU) »übersetzt«.

MASSENSPEICHER

In einem Massenspeicher werden große Mengen an Informationen abgespeichert, die auch dann noch abrufbar sein sollen, wenn z. B. der Strom ausfällt. Massenspeicher sind z. B.: Kassettenrekorder, Diskettenlaufwerk u. a.

MENÜ

Ein Menü ist in der Computersprache eine Liste über die Befehle und Funktionen, mit denen man ein Programm ausführen kann. Die Funktionsauswahl erfolgt einfach mit einer Ziffer, einem Buchstaben oder Ähnlichem.

PARALLELBUS

Ein Parallelbus ist ein Bündel von Leitungen (gewöhnlich 8 oder 16), das zur schnellen Informationsübertragung zum Computer verwendet wird.

PIXEL

Pixel ist ein anderes Wort für einen grafischen Punkt in einer grafischen Darstellung.

POSTEN

Ein Posten ist eine Sammlung von Daten, die zusammengehören. Mehrere Posten bilden ein Register oder eine Liste. Ein Posten kann z. B. ein Name oder eine Telefonnummer sein.

PROGRAMMSTRUKTUR

Eine Programmstruktur ist eine Beschreibung, wie ein gewisser Programmteil gelöst werden kann. Standardisierte Programmstrukturen sind Verzweigungen, Programmschritt-Folgen, Eingaben, Ausgaben etc.

RECORD

Ein Record ist die englische Bezeichnung für einen Datenblock. Normalerweise hat ein Record eine bestimmte Länge. Bei relativen Dateien kann der Programmierer aber festlegen, wie lang ein Record sein soll.

RELATIVE DATEI

Eine relative Datei ist eine Datei, in der die Struktur widerspiegelt, wie ein Register aussehen soll. Man kann festlegen, wie groß jedes Record (Datenblock) sein soll. Ungenützte Teile eines Datenblocks sind Verschwendung von Speicherplatz. Wenn man eine passende Größe für jedes Record wählt, kann man mehr Informationen auf einer Diskette abspeichern.

RGB-EINGANG

Ein RGB-Eingang ist ein Signal an einem Fernseh-Bildschirm, der eine Leitung für jede der drei Farben Rot, Grün und Blau hat. Den RGB-Eingang gibt es nur an teuren Spezial-Ferbfernsehern, den sogenannten Farbfernseh-Monitoren.

SCHLEIFE

Schleife (engl. loop) bezeichnet eine Programmfolge, die endlos abläuft oder bis eine bestimmte Bedingung erfüllt ist.

SEQUENTIELLE DATEI

Eine sequentielle Datei ist eine Sammlung von Datei, die nacheinander gelesen oder geschrieben werden.

SERIELLER BUS

Zwei Leitungen bilden gewöhnlich einen seriellen Bus, wenn sequentiell, d. h. nacheinander, Informationen zwischen zwei Geräten übermittelt werden.

SID

SID ist eine Abkürzung für Sound Interface Device, d. h. der Baustein oder Mikroprozessor des Tongenerators beim C-64.

UHF-EINGANG

Ein normales Fernsehgerät hat nur einen gewöhnlichen Antenneneingang. Dieser wird UHF-Eingang genannt.

VIC und VIC-II

VIC ist eine Abkürzung für Video Interface Chip. VIC ist die Bezeichnung für den Chip, der das Bildschirmbild beim VC-20 und beim C-64 erzeugt.

ZEICHENGENERATOR

Ein Zeichengenerator ist ein Speicher, der zeigt, wie jedes Zeichen auf dem Bildschirm aussehen soll. Die Standardzeichen sind in einem ROM (Festwertspeicher, Nur-Lese-Speicher) gespeichert. Programmierbare Zeichen werden im RAM-Speicher gespeichert, wo sie einfach geändert werden können.