Sonderheft Nr. 70
Preis DM 12.–
öS 91.–, sfr 12.–

ELO

Vom Bit zum Beispiel

Zahlreiche Mikrocomputer-Einsatzbeispiele. Von automatischen Schaltwerken über ein Digitalvoltmeter bis zur sprechenden Computer-Uhr.

ELO – damit Sie die Welt der Elektronik kennenlernen

Hi-Fi-Anlagen, Kameras, Autos, Haushaltsgeräte, Computer und viele andere mehr arbeiten heute mit Elektronik. Sie ist zur zentralen Bedeutung unseres täglichen Lebens geworden. Und wie's funktioniert steht in der ELO, der Zeitschrift für die Welt der Elektronik.

Die ELO informiert alle, die wissen wollen, was Elektronik ist, alle, die Elektronik als Hobby betreiben und alle, die beruflich mit Elektronik zu tun haben – alles verständlich dargestellt und schnell erfaßbar gegliedert.

Einen großen Raum nehmen Grundlagen aus allen Bereichen der Elektronik ein. Es werden Bauelemente beschrieben, Meß-, Steuer- und Regeltechnik ausführlich behandelt, der Hi-Fi- und Video-Fan findet zahlreiche Anregungen und die Hobbycomputerei strahlt ihre Faszination aus. Auch der Hobby-Elektroniker findet in jedem Heft mehrere praxiserprobte Bauanleitungen mit unterschiedlichen Schwierigkeitsgraden.

Mehrseitige Reports behandeln in jedem Heft aktuelle Themen aus der Welt der Elektronik und zeigen die vielfältigen Einsatzmöglichkeiten.

Einen weiteren Schwerpunkt bilden in der ELO die Aus- und Weiterbildung. Regelmäßig werden die beruflichen Möglichkeiten im Bereich der Elektronik aufgezeigt, so daß Sie Ihre Chance schnell erkennen können.

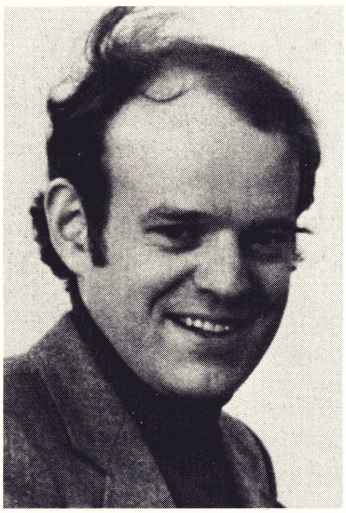
Berichte von Messen und Ausstellungen sowie Neuheitenvorstellungen und Tests informieren über Geräte und Produkte. Rundfunk- und Fernsehtips (alles zur Elektronik) und in jedem Heft ein Poster als Arbeitshilfe sind weitere nützliche ELO-Details.

Sie sehen, die ganze Welt der Elektronik steht in der ELO und es macht Freude, sie zu lesen. An jeder größeren Zeitschriften-Verkaufstelle gibt's die ELO für DM 3.50.

Franzis-Verlag

Karlstraße 37, 8000 München 2

Eine Abonnementbestellkarte finden Sie nebenstehend



Wunder dauern etwas länger

Mikrocomputer in der ELO – niemals, das wäre ja gelacht! Sie werden lachen, liebe Leser, aber diese Meinung ist lauthals in den verlagsinternen Gesprächen geäußert worden, als vor einigen Jahren die Konzeption der einzelnen FRANZIS-Zeitschriften diskutiert wurde.
Mittlerweile finden Sie in der ELO seit mehr als vier Jahren aktuelle Beiträge zum Thema Mikrocomputer, bei denen sich die Grundlagenbeiträge mit praktischen Anwendungsbeispielen größter Beliebtheit erfreuen, wie wir aus zahlreichen Leserbriefen und Gesprächen wissen.
Das vor Ihnen liegende Sonderheft faßt die in der ELO erschienenen Beiträge über das Mikrocomputer-System UMS-85 zusammen, das als preiswertes Komplettsystem speziell für die Belange des Hobbyisten konzipiert wurde, wobei geringe Kosten und Ausbaufähigkeit im Vordergrund standen.
Mit dieser Unterstützung hat insbesondere der Einsteiger die Möglichkeit, sich in das interessante Gebiet der Mikrocomputer-Technik einzuarbeiten, was ohne das Handwerkszeug eines realen Systems genauso unmöglich ist wie das rein auf die Theorie beschränkte Erlernen des Klavierspielens. Aber was nützt der schönste Mikrocomputer, wenn einem keiner sagt, was man damit anfangen kann?
Wir haben darum eine Reihe von Anwendungsbeispielen erarbeitet, die für Sie Information und Anregung zugleich sein sollen, um weitergehende Einsatzmöglichkeiten herausfinden und umsetzen zu können. Das beginnt mit einer Mikrocomputer-Musikbox, zeigt Ihnen ein Digitalvoltmeter auf Mikrocomputer-Basis und reicht bis hin zu einer sprechenden Uhr, die von Ihrem Mops angesteuert wird. Die Auswahl der Beispiele erhebt keinerlei Ansprüche auf Vollständigkeit, sie soll vielmehr einen Eindruck von der unglaublichen Vielseitigkeit dieses wundersamen Winzlings Mikroprozessor vermitteln.
Wenn es dabei gelingt, daß Sie mit Spaß bei der Sache sind und trotz der Erarbeitung des neuen Fachwissens Freude in Ihrer Freizeit finden, dann hat sich unser zum Teil doch recht erheblicher Aufwand mit Sicherheit gelohnt. Lassen Sie sich diese Freude in keinem Augenblick durch falschen Ehrgeiz oder übertriebenes Streben nach vollkommenen Lösungen verderben!

Wir wünschen Ihnen einen erbaulichen Weg durch Ihre Mikrocomputer-Praxis und allzeit recht viel Bit- und Byte-Salat.

Ihr Reinhard Gößler

Reinhard Gößler

1982

Franzis-Verlag GmbH, Karlstraße 37, 8000 München 2.
Produktion: ELVAG, Elektronik Verlag Luzern AG, CH-6002 Luzern.
Bearbeitet von Reinhard Gößler, Redaktion der Zeitschrift ELO. Für den Text verantwortlich: Henning Kriebel.

ISSN 0172-2786
Druck: Stämpfli + Cie. AG, CH-3001 Bern.
ZV-Artikel-Nr. 70031 · ST/ZV/282/579/10'

Inhalt

1. Mikrocomputer-Einsatz mit preiswertem Bausatz 4

Der Zusammenbau und die Inbetriebnahme des Mikrocomputer-Systems UMS-85 mit den detaillierten Schalt- und Bestükkungsplänen.

2. Der Mops macht Musik 9

Wie Sie Ihren Mikrocomputer dazu bringen, vorher eingegebene digitale „Noten" als Musikautomat abzuspielen; der Beginn der Wilhelm-Tell-Ouvertüre ist bereits fest im EPROM abgespeichert.

3. Schalten nach Schema 14

Wer sorgt dafür, daß während Ihrer Abwesenheit Licht ein- und ausgeschaltet wird, um mögliche Diebe abzuschrecken? Ganz einfach: Funktionieren Sie Ihren Mops doch zum Schaltautomaten um!

4. Aus Bits wird Spannung 19

Zu den faszinierendsten Anwendungsbeispielen gehören die Umwandlung digitaler Daten in Analogsignale bzw. die Digitalisierung analoger Größen; am Beispiel eines funktionierenden Digitalvoltmeters erfahren Sie, wie das vor sich geht.

5. Das tolle Telefon 24

Der Mops kann durchaus auch das Wählen einer Telefonnummer übernehmen; gleichzeitig wird die Bus- und Speichererweiterung beschrieben, die den Arbeitsspeicher ausbaut und Platz für weitere drei Interface-Karten bietet.

6. Bits vom laufenden Band 29

Das Eintippen von Programmen ist lästig, zeitaufwendig und fehlerintensiv; warum entscheiden Sie sich nicht für das Cassetten-Interface, mit dem Sie Daten und Programme auf normale Magnetband-Cassetten überspielen können?

7. Daten zu Papier gebracht 34

Mit dem Einsatz des Matrixdruckers bekommt Ihr Mikrocomputer-System schon fast einen professionellen Anstrich; der Nachbau ist unproblematisch, da das Druckwerk selbst komplett montiert geliefert wird.

8. Digitalwecker per Computer 40

Zu den Traumaufgaben eines Mikrocomputers gehört der Einsatz als Uhr. Um dies optimal zu gestalten, muß man programmtechnisch schon einige Klimmzüge machen, zu denen ganze Arithmetik-Programmteile gehören.

9. Die sprechende Computer-Uhr 44

Was noch vor kurzer Zeit niemand für möglich gehalten hätte, wird mit dieser Interface-Karte Wirklichkeit: Ihr Mops sagt laut und vernehmlich die Uhrzeit an! Nebenbei erfahren Sie die Verarbeitung von Interrupts.

10. Trickreiches Thermometer 50

Wollen Sie wissen, ob Ihr Wein richtig temperiert oder Ihr Wohnzimmer optimal beheizt ist? Dann bauen Sie Ihr Analog-Interface mit ganz geringem Aufwand zum Digitalthermometer um!

11. Dem Mops geht ein Licht auf 53

Der Mikrocomputer schaltet Lasten im 220-V-Netz und kann dort sogar als elektronischer Dimmer eingesetzt werden: Erweiterte Anwendungsbeispiele für das Analog-Interface aus dem 4. Teil.

Funktionen des 2-K-Monitors 56

Die erweiterte Monitor-Version verwaltet das Cassetten-Interface und hat zu den oben genannten Themen Programm- und Anwendungsbeispiele im EPROM gespeichert. Ein Ausbau der Leistungsfähigkeit Ihres Systems.

Mikrocomputer-Einsatz mit preiswertem Bausatz

Nach dem großen Erfolg, den unsere breit angelegte Einführungsreihe „Dem Mikrocomputer auf's Bit geschaut" bei unseren Lesern gefunden hat, ist immer wieder der eine Wunsch an uns herangetragen worden: Wir sollten ein Mikrocomputer-System vorstellen, das in allererster Linie so preiswert ist, daß es sich auch Hobbyisten mit schmalem Geldbeutel ohne weiteres leisten können. Nach umfangreichen Entwicklungsarbeiten präsentieren wir Ihnen hier ein solches System, das einschließlich Netzteil, Tastatur und Anzeige sowie Ein/Ausgabe-Kanälen nur 299.– DM kostet (Bausatzpreis inkl. Mehrwertsteuer und Versandkosten; Direktvertrieb über den Autor!) Damit wollen wir keineswegs unseren guten alten ELDO in die Ecke stellen, sondern ausschließlich eine preiswerte Parallele anbieten. Bei der Neuentwicklung der Interface-Karten (Matrixdrucker in ELO 9 und 10/80, Analog- und Digital-Interface folgen in Kürze) haben wir natürlich auf Kompatibilität mit beiden Systemen geachtet, und nach der Vorstellung des UMS-85 (Universelles Mikrocomputer-System auf 8085-Basis) sind unsere Einsatz- und Anwendungsbeispiele jeweils für beide Mikrocomputer passend. Das allgemeingültige Mikrocomputer-Grundwissen können Sie in allen Details dem oben zitierten FRANZIS-Sonderheft entnehmen, auf das wir uns auch im Text gelegentlich beziehen.

Dem Bausatz liegt übrigens eine Kurzanleitung bei, wie Sie Ihren Mikrocomputer nach dem Zusammenbau sofort zur Musikbox machen können; und das Programm für den Anfang der Wilhelm-Tell-Ouvertüre wird im PROM mitgeliefert, so daß flotte Klänge Ihre Mikrocomputer-Aktivitäten begleiten!

Computer komprimiert

Jeweils auf einer Europakarte (160 x 100 mm²) sind der Mikrocomputer und die Anzeige mit Tastatur untergebracht (**Bild 1**). Die Nabelschnur zwischen beiden bildet ein 16poliges Flachbandkabel, über das außer der Stromversorgung sämtliche Daten für die Anzeige und alle Informationen von der 24er-Tastatur laufen. An den Blockschaltbildern können Sie den funktionellen Aufbau des Systems verfolgen (**Bild 2** und **Bild 3**).

Am „Nordrand" der Computer-Platine finden Sie das Netzteil, das neben der Standard-5-V-Versorgung noch +12 V und –5 V liefert. Darunter prangt in voller Lebensgröße der Mikroprozessor vom Typ 8085. Er bezieht seinen Takt von einem quarzstabilisierten Oszillator, damit sämtliche Zeitvorgänge exakt bestimmbar sind. Das zweitgrößte IC nach der CPU ist der 24polige Programmspeicher, ein EPROM mit wahlweise 1-K- oder 2-KBytes Inhalt (in der Grundausbaustufe mit 1-K-EPROM bestückt). An den Platinenrändern befinden sich Schraubklemmen mit den Ein/Ausgabe-Kanälen: Links ist der 8 Bit breite parallele Eingangs- und rechts der parallele Ausgangsport angeordnet. Die Dreierklemme rechts in der Mitte bildet den seriellen Ein/Ausgabe-Kanal, und die kleine Leuchtdiode daneben signalisiert den Zustand der seriellen Ausgangsleitung. Erwartungsfroh harrt die 31polige Buchsenleiste am unteren Platinenrand darauf, eine der Erweiterungs-(Interface-) Karten aufzunehmen.

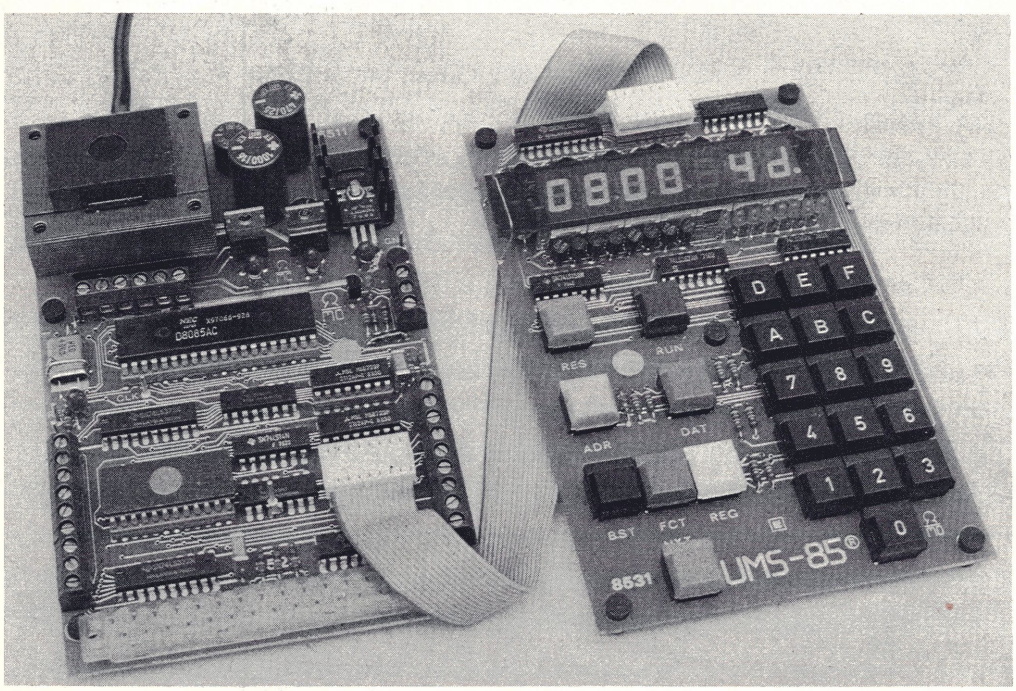


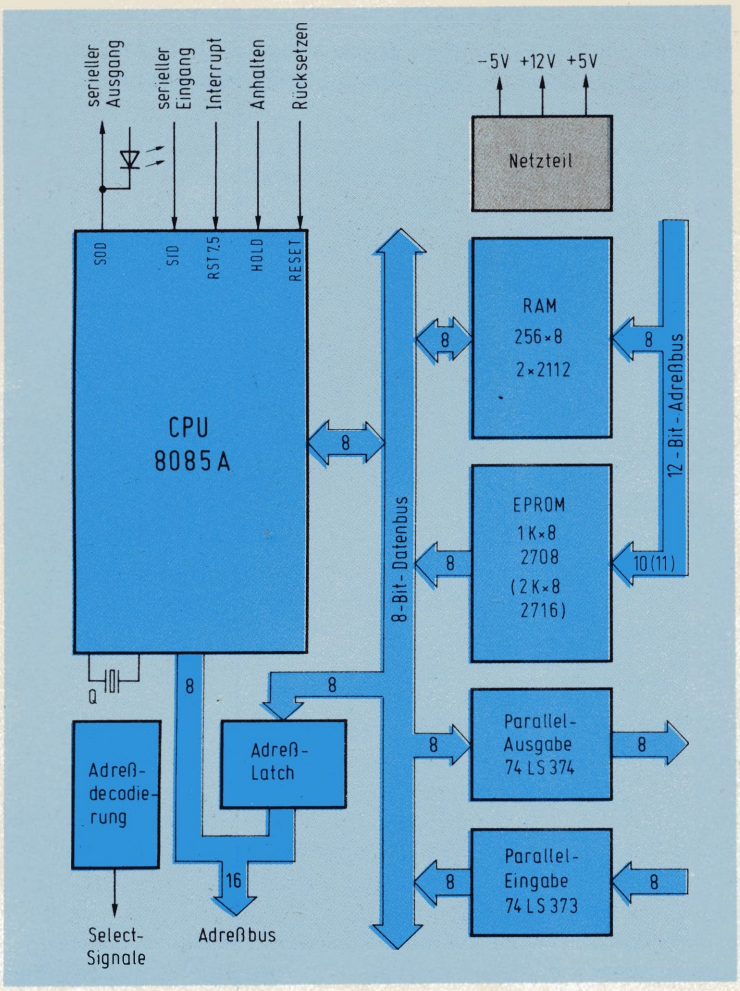
Bild 1: Auf Standard-Europakarten sind Mikrocomputer sowie Tastatur und Anzeige untergebracht.

Kommunikation per Anzeige und Tastatur

Mit lediglich fünf ICs kommt die Tastatur- und Anzeige-Platine aus, um die 24 Tasten und die 64 LEDs in den Anzeigen zu verwalten. Das gelingt nur mit einem ausgeknautschten Schaltungskonzept, das letztlich Garant für eine preiswerte Lösung ist.

In der Anzeige erscheint links die vierstellige Adresse einer Speicherstelle, und rechts daneben wird der Inhalt dieser Speicherstelle dargestellt. Auf Sonderfunktionen der Anzeige geht ein späterer Abschnitt im Zusammenhang mit der Monitor-Vorstellung ein. Hier sei noch ergänzt, daß die LED-Zeile unter den Anzeigen dazu dient, Bitmuster anschaulich sichtbar zu machen.

Die Bedeutung der (bunten) Befehlstasten und die der hexadezimalen Eingabe erfahren Sie ebenfalls bei der Vorstellung des



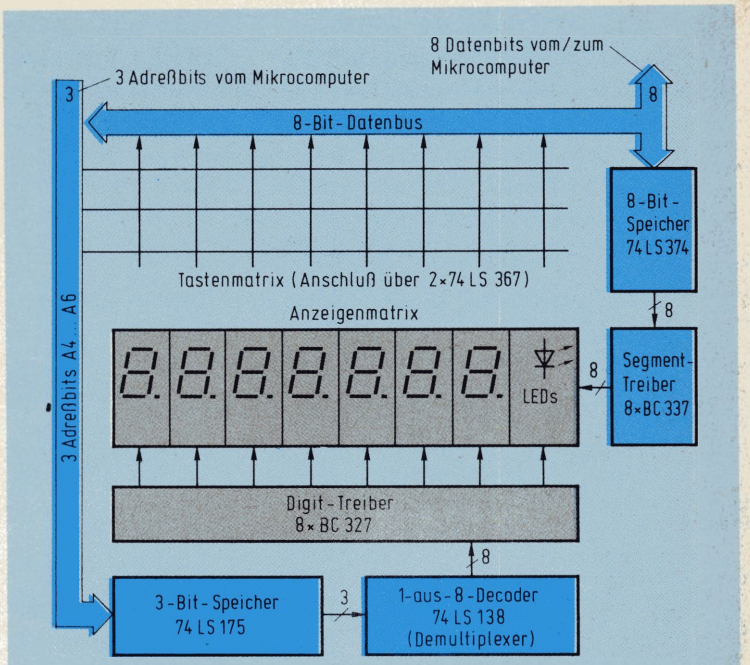


Bild 3: Blockschaltbild der Tastatur- und Anzeige-Platine.

◀ **Bild 2: Blockschaltbild der Mikrocomputer-Karte mit Netzteil.**

Monitor-Programms. Zusammenfassend stellt die **Tabelle 1** noch einmal die Eigenschaften des verwendeten Mikrocomputer-Systems UMS-85 zusammen.

- Preiswertes, modular aufgebautes System mit Busstruktur
- Durch Erweiterungs- und Interface-Karten universell ausbaufähig (kompatibel mit dem ELDO-Mikrocomputer)
  - Matrixdrucker mit Normalpapier (ELO 9 und 10/80)
  - Analog-Interface
  - Digital-Interface
  - Weitere Karten in Vorbereitung
- Zentraleinheit mit integriertem Netzteil sowie parallelen und seriellen Ein/Ausgabe-Kanälen
- Drei verschiedene EPROMs verwendbar (2708, 2758, 2716)
- Quarzstabilisierter Systemtakt
- Schraubklemmen für schnellen und lötfreien Peripherie-Anschluß
- Standard-Europakarten-Format 160 x 100 mm²

Tabelle 1. Eigenschaften des Mikrocomputer-Systems UMS-85

Und nun geht's ans Werk

Den kompletten Mikrocomputer mit Tastatur und Anzeige können Sie leicht an einem Samstagvormittag zusammenbauen (Sie müssen es aber nicht in dieser Zeit!). Denken Sie bitte daran, peinlichste Sorgfalt beim Bestücken und Löten walten zu lassen. Die Reparatur und Fehlersuche an einem solch' komplexen Gebilde ist nämlich derart aufwendig, daß dies weder dem ELO-Labor noch dem Autor zumutbar wäre!

Wenn Sie sich den Zusammenbau nicht zutrauen (oder Sie mit Recht feststellen, daß ein Dachpfannen-Löteisen für die zarten IC-Beinchen weniger geeignet ist), können Sie beide Platinen auch komplett montiert und getestet vom Autor beziehen (Komplettpreis 368.– DM).

Eine Selbstherstellung der Platine dürfte wegen der Durchkontaktierungen, der Zinnumschmelzung und Lötstoppmaske (beste Industriequalität!) übrigens kaum möglich sein.

Die Schaltung des Mikrocomputers finden Sie in **Bild 5**. Ohne uns hier im Detail zu verlieren, sollten Sie ein paar Einzelheiten registrieren. Der Prozessor 8085 schaltet an den Beinchen 12...19 abwechselnd Adreß- und Datenbits um; er besitzt also einen Multiplex-Adreß/Datenbus, und IS2, ein 8-Bit-Flipflop, entwirrt dieses Durcheinander wieder. Vorteil einer solchen Anordnung: man spart durch die Doppelbelegung acht Beinchen bzw. kann diese mit anderen Funktionen belegen; Nachteil: ein separates Adreß-Latch („Lätsch", zu

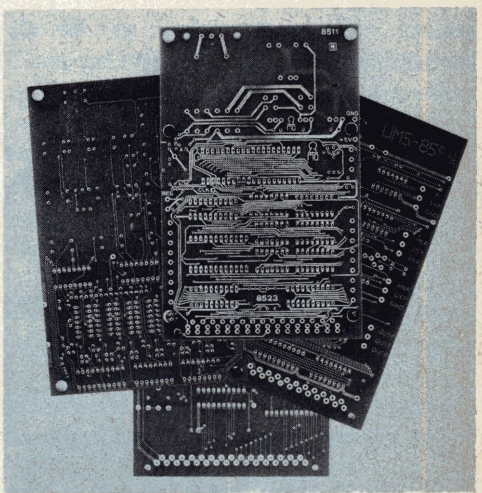


Bild 4: Mit Durchkontaktierung, Zinnumschmelzung und Lötstoppmaske besitzen sämtliche Platinen des UMS-85-Systems professionellen Charakter.

deutsch etwa „Auffangspeicher") muß her. Allein diese verschachtelten Abläufe machen deutlich, daß die Fehlersuche zu einem aufwendigen, fast hoffnungslosen Unterfangen wird. Daher die Bitte um Sorgfalt beim Löten!

An RAM stehen 256 Bytes zur Verfügung, von denen das Monitor-Programm je nach Funktion bis zu 40 Worte beansprucht. Als Festwertspeicher kann das EPROM 2708 mit 1 KBytes oder der Typ 2716 mit der doppelten Speicherkapazität eingesetzt

Speicher-adressen	Umfang in Bytes	Belegung
0000...03FF	1 K EPROM	Monitor-Programm
0400...07FF	1 K EPROM	Reserve für spätere Erweiterungen
0800...0BFF	1 K RAM	Arbeitsspeicher
0C00...0FFF	1 K	Externe Adressen für die Interface-Karten
Portadresse	Periphere Stelle (8-Bit-Parallel-Port)	
00	LED-Zeile der Anzeige	
04	CPU-Ausgabe-Kanal	
08	CPU-Eingabe-Kanal	
10....70	Siebensegmentanzeigen (Digit 0 = Port 10 = rechtsbündig)	
3C	Tastatur (Zeile Befehlstasten)	
5C	Tastatur (Zeile 8...F)	
6C	Tastatur (Zeile 0...7)	

Tabelle 2. Adreßraumbelegung des UMS-85

werden; die beiden Brücken A und B nehmen schaltungsmäßig die Anpassung vor und müssen entsprechend verdrahtet werden. Zur Adreßdecodierung dient IS8, und durch die Hardware-Struktur ergibt sich eine Adreßraumbelegung nach **Tabelle 2**. Von dem 16 Bit breiten Adreßbus gehen nur die unteren 12 Adreßbits in die Decodierung ein; die oberen vier Adreßbits können einen beliebigen Zustand haben, was bei der vierstelligen hexadezimalen Schreibweise statt „0800" teilweise durch ein „X800" gekennzeichnet wird. Sie sehen hier ferner, daß trotz unterschiedlicher Portadressen beim ELDO und UMS-85 dieselben Interface-Karten für Matrixdrucker, Analog- und Digital-Ein/Ausgabe angeschlossen werden können, weil dies bei der Konzeption berücksichtigt wurde.

Schließlich sei noch erwähnt, daß für die Ein- und Ausgangs-Ports keine teuren, großen und stromfressenden Peripheriebausteine verwendet werden, sondern daß hier preiswerte, kleine und leistungssparende 8-Bit-Flipflops (FFs) artfremd eingesetzt sind: Diese Typen 74LS373 bzw. 74LS374 haben nämlich außer ihrer Speicherfunktion den Vorteil von Tri-State-Ausgängen (im eingangs erwähnten Sonderheft erfahren Sie, was sich dahinter verbirgt), die beim Eingangskanal nur dann auf den Datenbus geschaltet werden, wenn die Adreßdecodierung dies veranlaßt. Beim Ausgangsport nehmen die acht FFs mit dem betreffenden Selektierungssignal (*Chip Select*) die auszugebende Information vom Datenbus auf und halten sie ausgangsseitig fest; hier sind die FF-Ausgänge immer im aktiven Zustand. Zur Versorgung des 2708-EPROMs (Grundausbaustufe) und für den Einsatz späterer Erweiterungskarten besitzt das Netzteil drei Versorgungsspannungen (**Bild 6**; das 2716-EPROM kommt mit nur einer Versorgungsspannung aus). Die Bestückung der Platine dürfte mit den Fotos

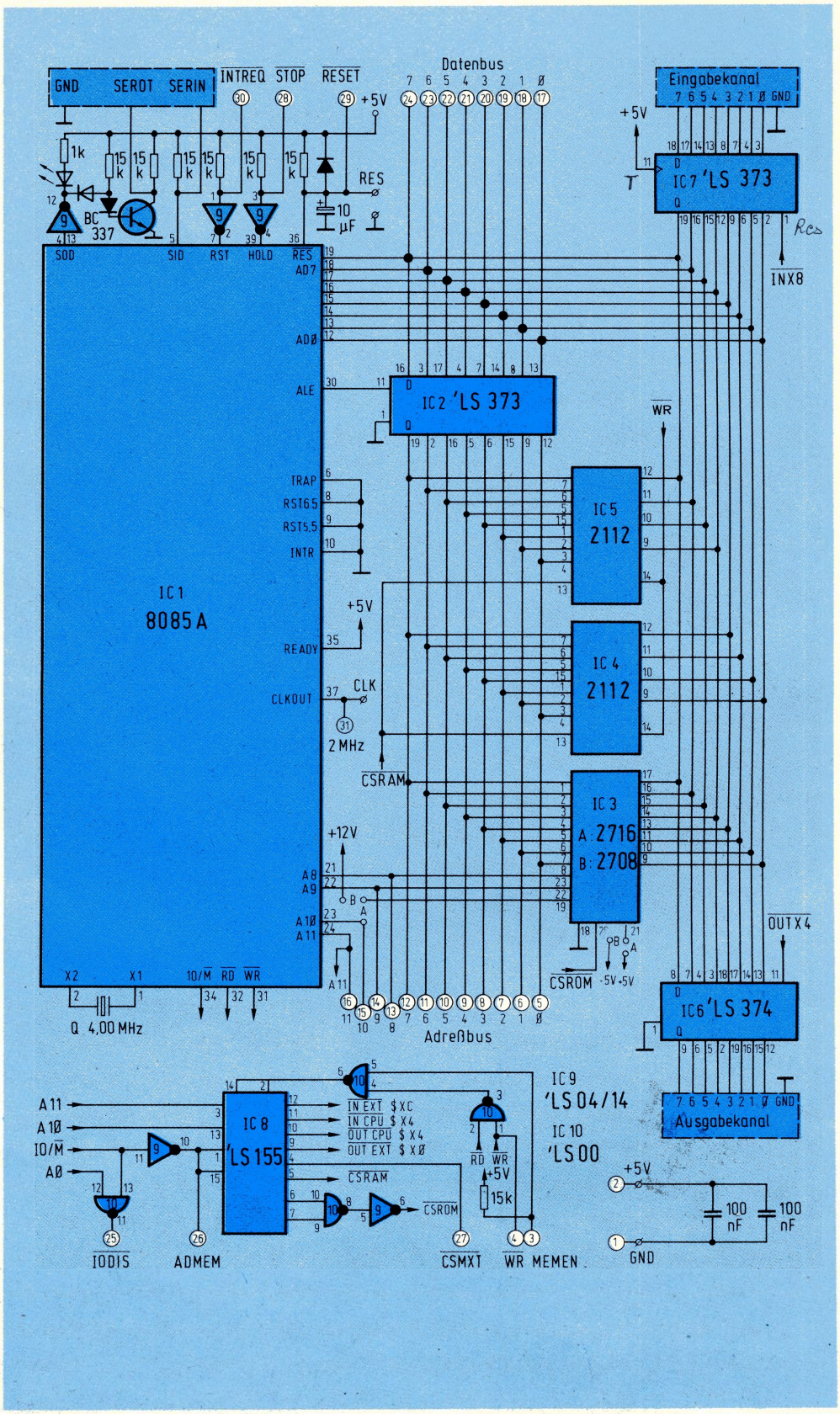


Bild 5: Schaltbild des Mikrocomputers mit Ein-/Ausgabe-Kanälen.

und dem detaillierten Bestückungsplan keine Probleme mehr aufwerfen, wobei Sie aber bitte die Polung der Elkos und Dioden beachten wollen (**Bild 7**). Alle ICs sind gleich ausgerichtet (Markierungskerbe links), und Mikroprozessor, EPROM und RAMs kommen auf Fassungen (im Augenblick noch nicht einsetzen). Eine weitere Fassung dient zur Aufnahme des Flachbandkabels für den Anschluß von Tastatur und Anzeige.
Vorsicht beim Umgang mit den MOS-ICs!

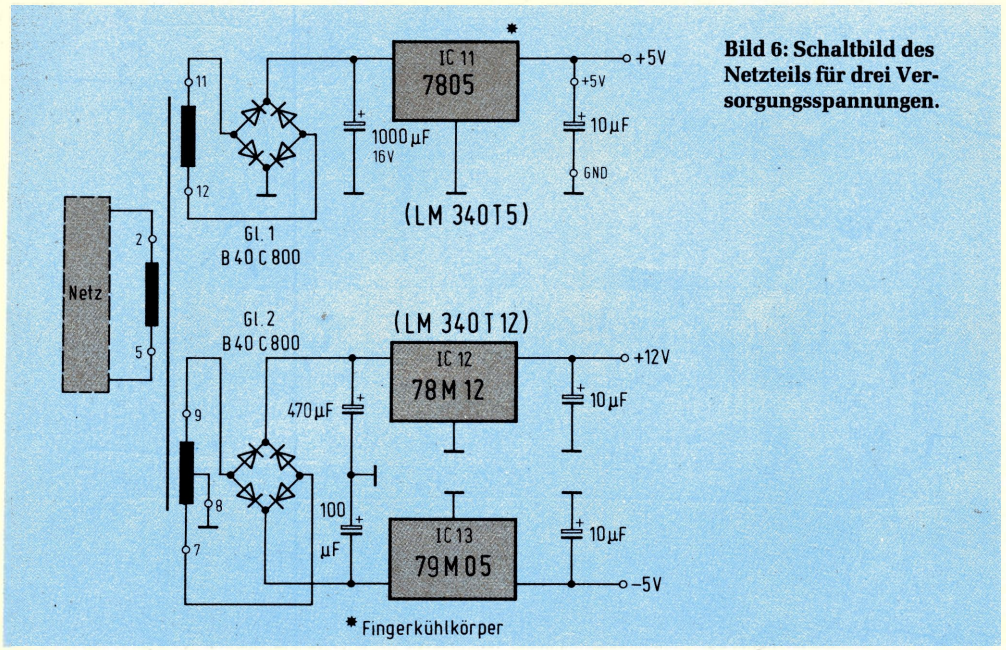


Bild 6: Schaltbild des Netzteils für drei Versorgungsspannungen.

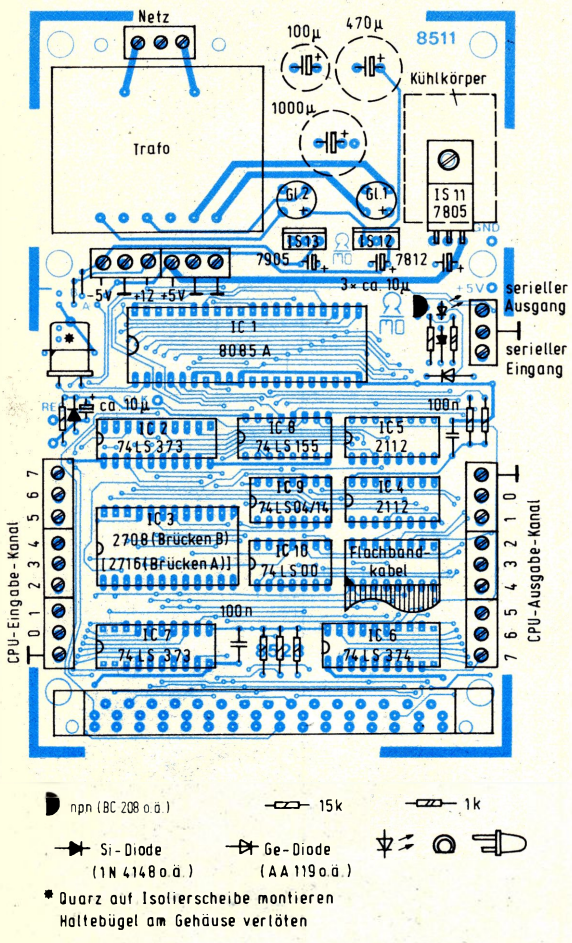


Bild 7: Bestückungsplan der Mikrocomputer-Platine mit Netzteil.

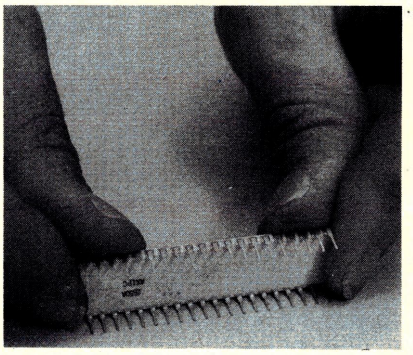


Bild 8: Mit sanftem Druck gegen eine ebene Fläche lassen sich die IC-Beinchen rechtwinklig biegen.

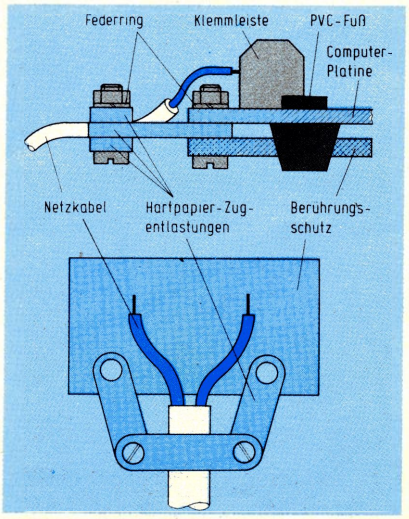


Bild 9: Die Zugentlastung für das Netzkabel wird zusammen mit dem Berührungsschutz montiert.

Sie sind bereits gegen statische Aufladungen des menschlichen Körpers äußerst empfindlich und sollten an ihren Anschlußbeinchen am besten gar nicht angefaßt werden. Vor dem Einsetzen in die Fassung biegen Sie bitte die schräg nach außen stehenden Beinchen so zurecht, daß sie sich parallel in die Fassung einführen lassen (**Bild 8**).
Das Netzkabel wird von einer Zugentlastung gehalten, die nach unten zum Berührungsschutz mit einer Pertinax-Scheibe abgedeckt ist (**Bild 10**). Zur Inbetriebnahme überprüfen Sie bitte erst die drei Versorgungsspannungen (noch ohne eingesetzten Mikroprozessor und Speicher); alle übrigen ICs sind dagegen eingelötet. Dann setzen Sie zunächst den Mikroprozessor ein (aber bei abgeschalteter Versorgungsspannung!) und können, soweit vorhanden, mit einem Zähler, Logikprüfstift oder Oszillografen am Testpunkt CLK (Clock-Ausgang) die Taktfrequenz von 2 MHz (halbe Oszillatorfrequenz) kontrollieren. Erst jetzt folgt (ebenfalls bei abgeschalteter Stromversorgung) das Einsetzen der RAMs und des EPROMs.

Gleich kommt Leben in die Schaltung

Die LED-Anzeige wird im Multiplex-Betrieb angesteuert, wobei das Monitor-Programm die MUX-Rate softwaremäßig erzeugt (**Bild 11**). Dies ist ein Schaltungskonzept mit extrem geringem Hardware-Aufwand, bei dem jedes einzelne Segment der Anzeige per Programm ein- und ausgeschaltet werden kann. Funktionell entspricht dies der ELDO-Anzeige, deren ausführliche Beschreibung Sie ebenfalls im ELO-Mikrocomputer-Sonderheft finden.
Auch die Tastatur ist in Form einer Matrix aufgebaut, um sie rationell nach Zeilen und Spalten abfragen zu können. Bei der Bestückung ist darauf zu achten, daß die Digit-Treiber-Transistoren (unterhalb der Anzeigen) pnp-Typen sind (BC327 o. ä.), und die oberhalb der Anzeigen liegenden Segmenttreiber sind npn-Transistoren (BC208 o. ä.; **Bild 12**). Bei den LEDs zeigt die Katode (das kurze Anschlußbein bzw. die abgeflachte Seite) immer zum unteren Platinenrand. Nach dem Einlöten der Anzeigen folgt die Montage der roten Filterscheibe, die beidseitig von einem Drahtbügel gehalten wird.
Wenn Sie nun beide Platinen über das Flachbandkabel miteinander verbinden (auch hier ist Vorsicht geboten, die zarten Stifte an den IC-Steckern können leicht verbiegen oder brechen!), muß sich der Monitor melden: In der Anzeige erscheint linksbündig die Adresse 0800 (die unter-

ste im Arbeitsspeicher, vgl. Tabelle 2), und rechts wird der Inhalt dieser Speicherstelle dargestellt. Ist dies der Fall dann können Sie sich recht glücklich schätzen, denn Ihr Computer funktioniert.

Beste Fehlererkennung: Ihr Auge

In der Fülle unserer Leserzuschriften finden wir immer wieder die kapitulierende Feststellung, daß ein Schaltungsnachbau „nicht geht". Die lapidare, manchmal sogar vorwurfsvolle Frage, woran das denn nun liegt, läßt sich zu 99 % so beantworten: An einem Bestückungsfehler, einer „kalten" Lötstelle oder einem versehentlich beim Löten entstandenen Kurzschluß. Das beste Mittel zum Aufspüren derartiger Fehler ist Ihr Auge, das sorgfältig alle Lötstellen auf falschen Kontakt zum Nachbarn untersuchen und natürlich auch die Bestückung überprüfen sollte. Vom ordnungsgemäßen Vorhandensein der Versorgungsspannung sollten Sie sich an jedem IC einzeln überzeugen, wenn etwas nicht funktioniert.

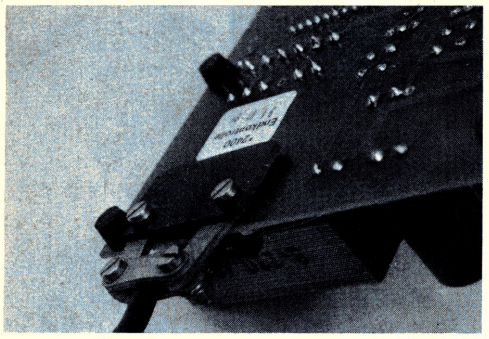


Bild 10: Das fertig montierte Netzkabel mit Zugentlastung und Berührungsschutz.

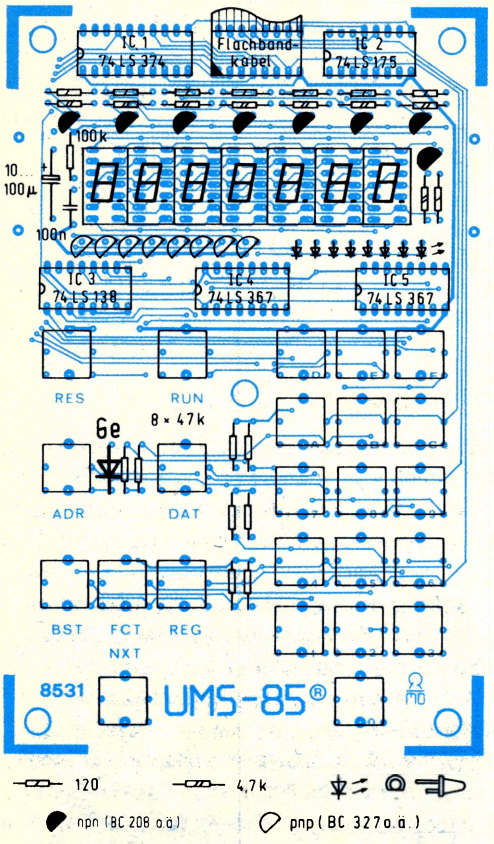


Bild 11: Schaltbild der Anzeige- und Tastenmatrix.

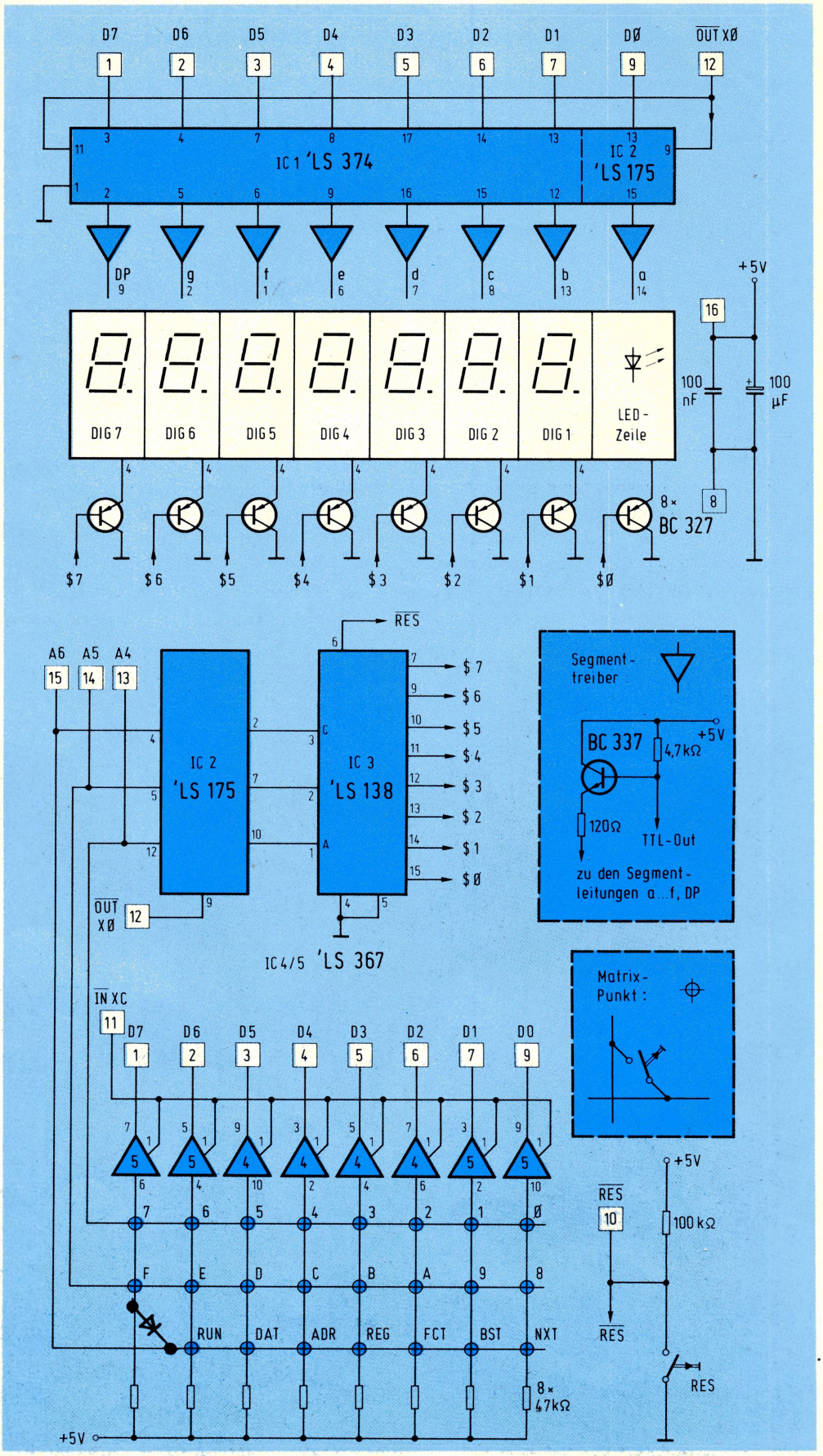


Bild 12: Bestückungsplan der Tastatur- und Anzeige-Platine.

Der Mops macht Musik

Im vorigen Beitrag haben wir Ihnen den Mikrocomputer UMS-85 vorgestellt, und Sie haben dort erfahren, wie Sie den entsprechenden Bausatz zum funktionierenden Gerät zusammenlöten.

Egal, ob Sie sich für den Kauf des Fertiggeräts oder den Selbstbau entschieden haben – hier beginnen wir mit den ersten praktischen Gehversuchen.

Dazu stellen wir zunächst kurz den UMS-Monitor vor, ehe das erste Programmbeispiel (gleichermaßen passend für den guten alten ELDO wie auch für das UMS-System) Leben in die Platinen bringt (**Bild 1**). Abschließend folgt ein Blick auf den Befehlssatz des 8085, auf den wir dann später immer wieder zurückgreifen; den brauchen Sie keineswegs auswendig zu lernen, sondern Sie sollten sich bei Bedarf nur daran erinnern, wo er wiederzufinden ist.

Monitors Auge wacht

Wenn Sie Ihren Mikrocomputer an die Stromversorgung anschließen und sich in der Anzeige definiert etwas regt, dann wird dies von einem fest abgespeicherten Programm, dem sogenannten Monitor, veranlaßt. Der ist in dem großen EPROM vom Typ 2708 abgelegt und fragt unentwegt die Tastatur ab, liest Programme und Daten ein (sofern Sie das wollen), zeigt Register- und Speicher-Inhalte an und sorgt zu guter Letzt auch noch für die Ausführung Ihrer (Anwender-)Programme. Und wenn Sie auf der Anzeige-Platine vergeblich einen Siebensegment-Decodierer oder eine Multiplex-Steuerung suchen, dann vermuten Sie richtig, wenn Sie auch diese Aufgaben dem Monitor-Programm zutrauen! Nach dem Einschalten (und nach jedem Rücksetzen über die Taste **RES**) bringt der Monitor links die Adresse 0800 zur Anzeige; rechts steht immer der Inhalt der adressierten Speicherstelle, und daß dies alles hexadezimale Zahlen sind, sei hier nur noch am Rande vermerkt (vgl. Sonderheft: „Dem Mikrocomputer auf's Bit geschaut"). Daß es ausgerechnet die Adresse 0800 ist, hat seinen guten Grund: Hier beginnt der Arbeitsspeicher des Systems, und es liegt nahe, daß Sie dort beginnend Ihre Programme ablegen.

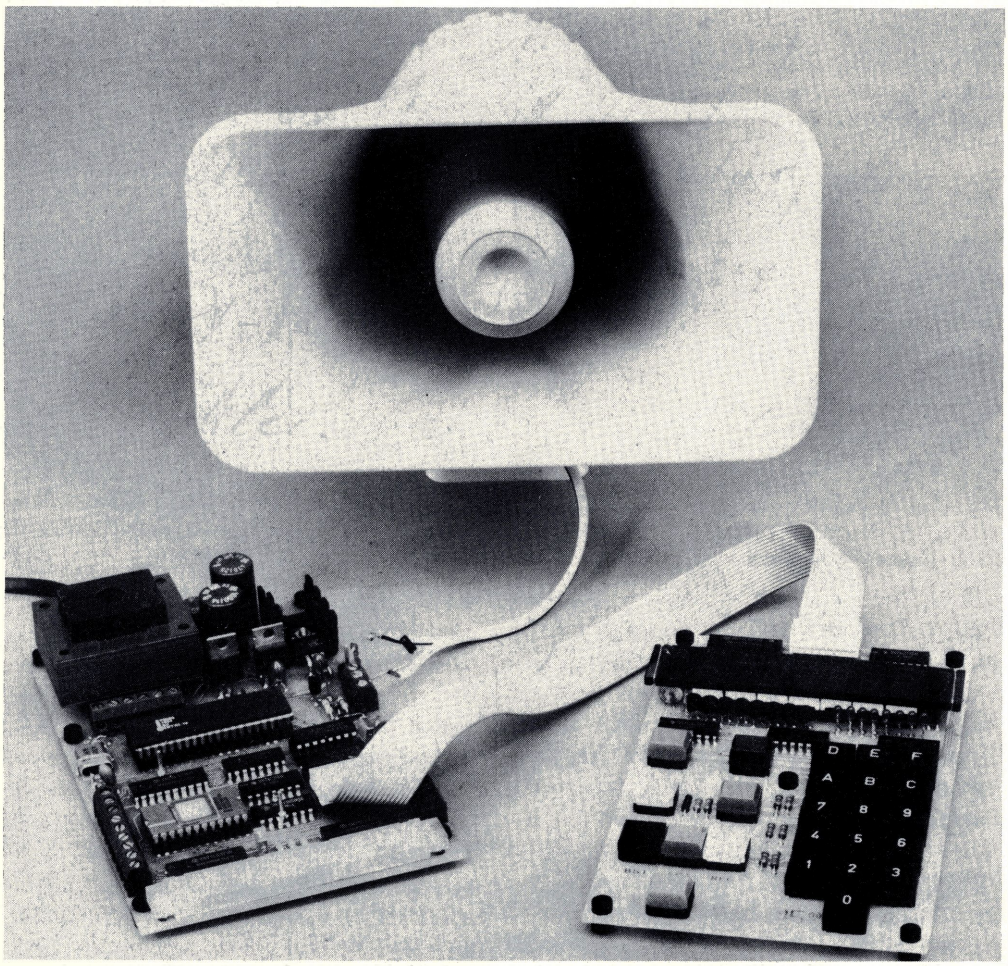


Bild 1: Ein im Monitor fest abgespeichertes Unterprogramm samt Notentabelle sorgt dafür, daß der Mikrocomputer richtige Musik erzeugen kann.

Wegweiser für die Eingaben

Wenn Sie schärfer hinsehen, entdecken Sie rechts im Datenfeld noch ein kleines Pünktchen. Es ist Kennzeichen dafür, daß die Eingaben von der (schwarzen) HEX-Tastatur im Datenfeld landen. Diese Betriebsart wählen Sie durch Druck auf die Taste **DAT**, was jedesmal erfolgen muß, bevor man Daten in einer Speicherstelle (oder einem Register) inspizieren oder ändern kann.
Der Druck auf **ADR** bewirkt das Aufleuchten des Dezimalpunkts im Adreßfeld; Eingaben von der HEX-Tastatur wandern nun dorthin und werden als Adresse verstanden. Die blonde Taste **NXT** (NEXT) be-

wirkt zweierlei: Erstens schaltet sie die Adresse um Eins weiter; und zweitens lädt sie die Information aus dem Datenfeld in diejenige Speicherzelle, deren Adresse (vor dem Tastendruck) im Adreßfeld steht. Um also beispielsweise die Daten „C3" unter der Adresse 0815 abzulegen, verfahren Sie wie folgt: Taste **ADR** (Adreßeingabe) drücken, gefolgt von den HEX-Tasten 0-8-1-5; dann Taste **DAT** (Umschalten auf Daten-Mode) drücken, gefolgt von den HEX-Tasten C-3. Das Einschreiben aber passiert erst dann, wenn **NXT** gedrückt wird (im Adreßfeld steht dann die nächste Adresse 0816). Wenn Sie das nicht glauben, gehen Sie mit **BST** (Back Step) einen Schritt zurück: Das Adreßfeld schaltet zurück auf 0815, und beglückt finden Sie dort Ihre „C3" wieder!

Bei „RUN" geht's los

Mit dem Druck auf die Taste **RUN** bringen Sie Ihren Computer auf Trab: Er beginnt dann mit der Programmausführung bei derjenigen Adresse, die links im Adreßfeld steht. Solange Sie noch kein Programm geladen haben, erfolgt natürlich auch noch keine definierte Reaktion! – Den restlichen beiden Tasten **REG** (Register-Anwahl) und **FCT** (Funktionserweiterungen) lassen wir im Augenblick noch ihre Ruhe.

Daß so ein Monitor eine Fülle von Unterprogrammen enthält, die der Anwender in seine eigenen Programme einbauen kann, wissen Sie schon von der Vorstellung des ELDO her. Wir wollen dies unmittelbar an einem Beispiel demonstrieren, damit das Ganze keine graue Theorie bleibt.

Dazu hat der Monitor das sonore Unterprogramm MUSIC (Startadresse # 02A4), das am seriellen Ausgang SEROT Musik erzeugt (tatsächlich!). Vor dem Aufruf dieses Unterprogramms (CALL-Befehl) muß MUSIC noch erfahren, wo die Noten stehen, die es zu Gehör bringen soll; dazu müssen im vorliegenden Fall die beiden CPU-Register H und L mit der Anfangsadresse der Notentabelle geladen werden (LXI-Befehl). Das wollen wir zunächst mit Adresse 033A versuchen. – Der Sprungbefehl JMP (JUMP) bewirkt zusammen mit der Zieladresse die Programmfortsetzung an der angegebenen Stelle.

Mops ist (fast) gleich Mops

Bevor wir aus diesen Elementen ein Programm formen, müssen Sie noch zwei Dinge wissen: Erstens kann der 8085 (wie auch sein greiser Vorgänger 8080) nur absolute Sprünge ausführen; man muß die Zieladresse also immer im Sprungbefehl (oder Unterprogramm-Aufruf) angeben. Und zweitens erfolgt die Adreßeingabe immer „verkehrt herum", d. h. nach dem Befehlscode für den Sprung „C3" folgt erst die untere Hälfte der Zieladresse und dann die obere. Soll also ein 8085 (oder einer seiner vielen Verwandten) zur Adresse 0800 springen, stehen im Programmspeicher nacheinander die drei Bytes C3-00-08. Grübeln und Fluchen hilft hier nichts, es ist so, und es bleibt auch so!

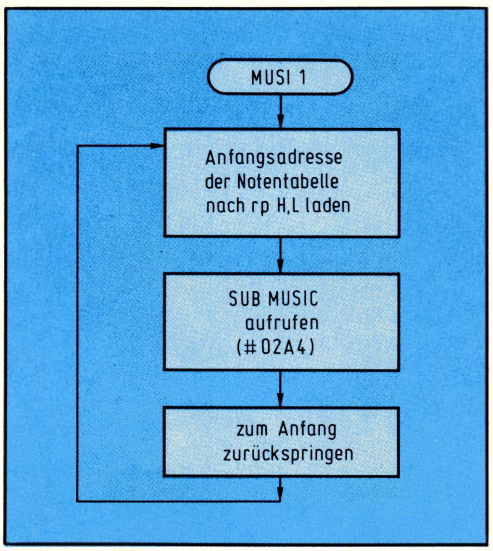


Bild 2: Der Aufruf des MUSIC-Unterprogramms erfordert nur drei Aktivitäten.

```
0800  21    MUSI1   LXI H,033A
0801  3A
0802  03
0803  CD            CALL MUSIC
0804  A4
0805  02
0806  C3            JMP MUSI1
0807  00
0808  08
0809  XX
```

Bild 3: Das Maschinenprogramm für die einfache Tonerzeugung verwendet den Tabellenanfang 033A.

```
0800  21    MUSI2   LXI H,02E0
0801  E0
0802  02
0803  CD            CALL MUSIC
0804  A4
0805  02
0806  C3            JMP MUSI2
0807  00
0808  08
0809  XX
```

Bild 4: Beim Aufruf der Notentabelle ab Adresse 02E0 spielt der Mikrocomputer klassische Musik.

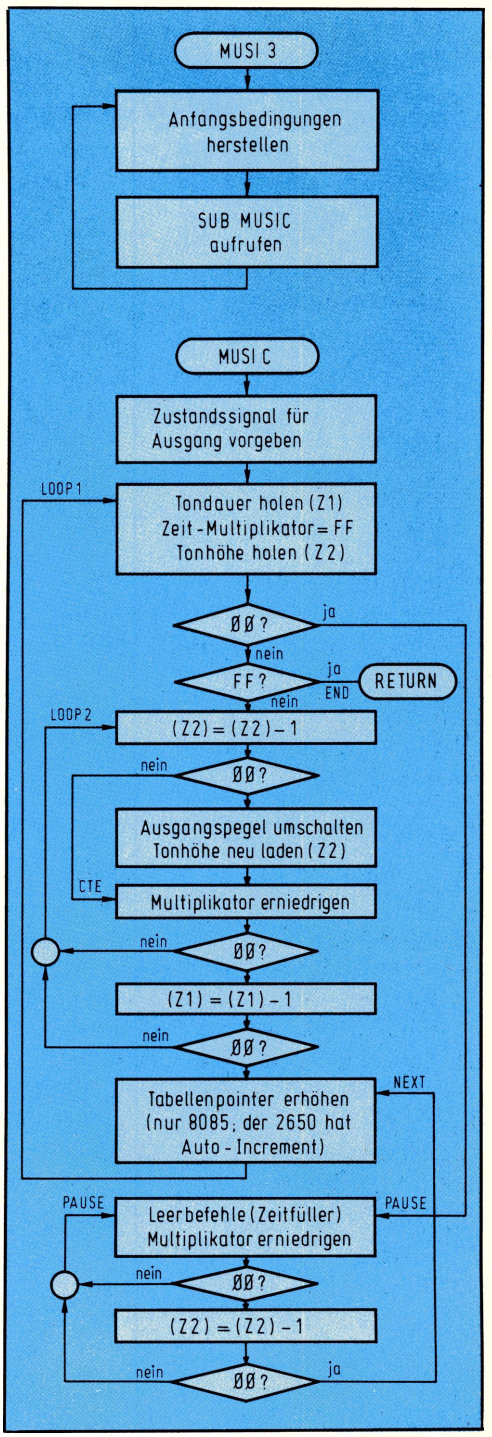


Bild 5: Das komplette Flußdiagramm für die Musikerzeugung.

Im übrigen nehmen sich die beiden Prozessoren 8085 und 2650 nicht viel, und das hatten wir ja seinerzeit auch versprochen, als es hieß, daß das Umsteigen von einem Prozessor auf den anderen ähnlich problemlos ist wie der Wechsel der Automarke. Auch der 8085 verfügt über sieben CPU-interne Datenregister, die hier die Bezeichnung A, B, C, D, E, H, und L tragen. Im Gegensatz zum 2650 kann man jederzeit auf alle zugreifen, ohne erst die eine oder andere Registerbank anzuwählen.

Ein ganz wesentlicher Unterschied zum

2650 besteht beim 8085 darin, daß die Zustandssignale (FLAGs; das sind die *Condition Codes* beim 2650) nur *nach* einer arithmetischen oder logischen Operation gesetzt werden, vom bloßen Datentransport also unberührt bleiben. Es gibt hier vier verschiedene FLAGs, die alle im FLAG-Register zusammengefaßt sind:
C = CARRY (Übertrag), Z = ZERO (Ergebnis ist Null), S = SIGN (Vorzeichenbit [MSB] ist HIGH, d. h. der Wert wird negativ [MINUS] aufgefaßt) und P = PARITY (Anzahl der HIGH-Bits im Ergebniswort ist gerade, d. h. *Parity Even*). Sowohl mit dem Auftreten als auch Ausbleiben dieser Zustandssignale lassen sich Sprungbefehle, Unterprogramm-Aufrufe und -Rücksprünge verknüpfen. Lassen Sie uns nun das Programm MUSI1 laden, das zunächst nur „tatütata" macht (**Bild 2**). Das passende Maschinenprogramm finden Sie in **Bild 3**, und Sie laden es, beginnend bei Adresse 0800, wie eingangs beschrieben. Achten Sie bitte darauf, daß auch die letzte Dateneingabe mit dem Druck auf **NXT** abgeschlossen wird.
Wenn Sie nun das Programm starten (**RES** bringt Sie zurück zur Startadresse 0800, und **RUN** bewirkt den Programmstart), können Sie an SEROT Ihre Musik abnehmen. Dazu schließen Sie einen kleinen dynamischen Lautsprecher mit einem Bein an SEROT an (oberer Anschluß der Dreierklemme am rechten Platinenrand) und das andere legen Sie an +5 V (unmittelbar darüber). Der guten Ordnung halber sollte parallel zum Lautsprecher eine kleine Diode liegen, und zwar mit der Katode (Querstrich) an +5 V. – Wenn Sie der Musik überdrüssig geworden sind, erlösen Sie sich und Ihre Umwelt von dem „Tatütata", indem Sie entschlossen **RES** drükken!

Musik alter Meister im neuen Gewand

Virtuoser wird die Vorführung (und dazu können Sie bereits erste Gäste laden), wenn Sie als Anfang der Notentabelle statt 033A jetzt 02EO laden. Dort stehen Auszüge aus Rossinis Ouverture zu „Wilhelm Tell", die sich ganz hervorragend für die Mikrocomputer-Interpretation eignen (**Bild 4**). Sollten Sie statt des angegebenen *kleinen* Lautsprechers versehentlich eine 100-W-Box angeschlossen haben, dröhnt diese martialisch laut; sagen Sie in diesem Fall bitte niemandem, daß so etwas in der ELO gestanden hat...
Um mit dem Mikrocomputer einen Ton zu erzeugen, braucht man nur eine Ausgangsleitung gezielt zwischen HIGH und LOW hin- und herzuschalten; passiert das 1000mal pro Sekunde, entsteht eine Frequenz von 1 kHz. Bei 440 Hz und angeschlossenem Lautsprecher vernehmen Sie den Kammerton „a" (musikalische Leute nennen es das „Eingestrichene a", was einen Strich am „a" kennzeichnet und nichts mit Anpinseln zu tun hat).
Die Dauer, während der die Ausgangsleitung auf HIGH bzw. LOW verharrt, wird vom Anfangswert eines Zählers bestimmt, der während dieser Zeit auf Null heruntergezählt wird. Je größer man diesen Zeitfaktor wählt, desto langsamer geht das Leerzählen vor sich und desto tiefer wird der Ton.
Um Musik zu machen, muß man pro Laut nicht nur die Tonhöhe, sondern auch noch die Tondauer angeben. Und um den einen Ton sauber vom nächsten zu trennen, muß man eine Pause einfügen, die ebenfalls unterschiedlich lang sein kann. Die Notentabelle enthält also für jede Note (und jede Pause) die Information über Zeitdauer und Tonhöhe (bzw. die Information „Pause"). Eine Pause erkennt das Programm daran, daß anstelle der Tonhöhe ein „00" eingegeben wurde. Und wenn das Programm statt der Tonhöhe das andere Extrem „FF" vorfindet, bricht es die Dudelei ab (Ende-Kennzeichen).

Tabelle 1: Zusammenhang zwischen Tabellenwert und Tonfrequenz

Ton	Frequenz [Hz]	HEX-Wert	Ton	Frequenz [Hz]	HEX-Wert
h3	1975	12	h2	988	25
a3	1760	14	a2	880	2A
g3	1568	17	g2	784	2F
f3	1397	1A	f2	698	34
e3	1318	1C	e2	659	38
d3	1174	1F	d2	587	3F
c3	1046	23	c2	523	47

Die Werte gelten für eine Taktfrequenz von 477 ns. Zeitfaktor 10 (HEX) entspricht etwa einer $^1/_{16}$-Note.

Ein hexadezimales Notenblatt

Die Notentabelle reserviert dementsprechend pro Ton vier Bytes: Das erste gibt die Ton*dauer* an, das zweite die Ton*höhe*; als drittes folgt die Pausen*dauer*, und daran schließt sich entweder ein „00" an (d. h. „Pause") oder ein „FF" (d. h. „Ende"). Die Zuordnung zwischen Tonhöhe und hexadezimalem Wert geht aus **Tabelle 1** hervor, die ebenfalls einen Anhaltspunkt für die Tondauer angibt. – Es bedarf sicher keiner Erwähnung, daß Sie hiermit weder eine elektronische Orgel noch ein „richtiges" Musikinstrument aufgebaut haben; aber der Effekt dieser Musikerzeugung ist so frappierend, daß man es der unscheinbaren Elektronik kaum zutraut!
Da die Taktfrequenz des ELDO mit ca. 1 MHz rund 50 % langsamer ist als die des UMS-Systems, müssen die Zahlenwerte der Tabelle 1 für den ELDO durch Zwei geteilt werden (oder aber Sie löten parallel zu dem 470-Ω-Widerstand, der neben dem Poti auf der CPU-Karte liegt, einen weiteren 470-Ω-Widerstand und nehmen die „Stimmung" dann am Poti vor).
Ein Flußdiagramm für diese programmierbare Musicbox finden Sie in **Bild 5**, und **Bild 6** stellt die Maschinenprogramme dar. Um beide Versionen direkt nebeneinanderstellen und kommentieren zu können, sind die Programme nicht optimiert worden! Hier und bei den folgenden Programmbeispielen aber soll der Schwerpunkt nicht auf Bit-Geizerei liegen, sondern er soll sich rühren, Ihr Mops, egal, ob er UMS oder ELDO heißt!
Sie erkennen eine ungewohnte Darstellung bei der Schreibweise der Programme; bei den Mehrwortbefehlen stehen die zwei (oder drei) Bytes eines Befehls *nebeneinander*, um einerseits Platz bei der Dokumentation zu sparen und andererseits die Übersichtlichkeit zu erhöhen. Das hat zur Folge, daß die Adressen in der linken Spalte bei Mehrwortbefehlen nicht fortlaufend weiterzählen, sondern sich um Zwei bzw. Drei erhöhen. Dieser scheinbare Nachteil wird durch die wesentlich bessere Lesbarkeit bei weitem wettgemacht.

Der Sprachschatz eines 8085

Abschließend wollen wir noch einen Blick auf den Befehlssatz des 8085 werfen, den Sie in Kurzform in den **Tabellen 2 und 3** zusammengestellt finden; nach und nach kommen wir dann später auf die einzelnen Befehlstypen zurück.

1. Datentransportbefehle

MOV r1,r2 *(Move* **to** *Register r1* **from** *Register r2)*
Datentransport **nach** r1 **aus** r2
MVI r,XX *(Move Immediate)*
Datenbyte XX ins Register r laden
LXI rp *(Load Immediate Register Pair)*
Registerpaar rp mit den beiden Datenbytes XX YY laden

LDA XX YY (*Load Accumulator Direct*)
Akkumulator mit (XX YY) laden
STA XX YY (*Store Accumulator Direct*)
Akkumulator nach XX YY laden
LDAX rp (*Load Accumulator Indirect*)
Akkumulator mit ((rp)) laden

STAX rp (*Store Accumulator Indirect*)
Akkumulator nach ((rp)) laden
LHLD XX YY (*Load H and L Direct*)
rp H&L (XX YY)&(XX YY + 1) laden
SHLD XX YY (*Store H and L Direct*)
rp H&L nach XX YY & XX YY + 1 laden

SPHL (*Exchange Stack Pointer and H&L*)
Inhalt von SP und rp H&L austauschen
XTHL (*Exchange Stack Top and H&L*)
Stack Top und rp H&L austauschen
XCHG (*Exchange H&L with D&E*)
rp H&L mit rp D&E austauschen

Tabelle 2: Tabellarische Zusammenstellung der 8085-Maschinenbefehle

UMS-85®

Register r	A	B	C	D	E	H	L	m
MOV A,r	7F	78	79	7A	7B	7C	7D	7E
MOV B,r	47	40	41	42	43	44	45	46
MOV C,r	4F	48	49	4A	4B	4C	4D	4E
MOV D,r	57	50	51	52	53	54	55	56
MOV E,r	5F	58	59	5A	5B	5C	5D	5E
MOV H,r	67	60	61	62	63	64	65	66
MOV L,r	6F	68	69	6A	6B	6C	6D	6E
MOV m,r	77	70	71	72	73	74	75	--
MVI r,XX	3E XX	06 XX	0E XX	16 XX	1E XX	26 XX	2E XX	36 XX

Registerpaar	B	D	H	SP
LXIrp,XXYY	01 YY XX	11 YY XX	21 YY XX	31 YY XX

Registerpaar	B	D	H	PSW
PUSH	C5	D5	E5	F5
POP	C1	D1	E1	F1

LDA Adr 3A YY XX	STA Adr 32 YY XX	XTHL E3
LHLD Adr 2A YY XX	SHLD Adr 22 YY XX	SPHL F9
		XCHG EB

Registerpaar rp	B	D	IN XX DB XX
LDAX rp	0A	1A	OUT XX D3 XX
STAX rp	02	12	

DATENTRANSPORTBEFEHLE

ADI XX C6 XX	ACI XX CE XX	SUI XX D6 XX	SBI XX DE XX	ANI XX E6 XX	XRI XX EE XX	ORI XX F6 XX	CPI XX FE XX

Register r	A	B	C	D	E	H	L	m
ADD r	87	80	81	82	83	84	85	86
ADC r	8F	88	89	8A	8B	8C	8D	8E
SUB r	97	90	91	92	93	94	95	96
SBB r	9F	98	99	9A	9B	9C	9D	9E
ANA r	A7	A0	A1	A2	A3	A4	A5	A6
XRA r	AF	A8	A9	AA	AB	AC	AD	AE
ORA r	B7	B0	B1	B2	B3	B4	B5	B6
CMP r	BF	B8	B9	BA	BB	BC	BD	BE
INR r	3C	04	0C	14	1C	24	2C	34
DCR r	3D	05	0D	15	1D	25	2D	35

Registerpaar	B	D	H	SP
INX rp	03	13	23	33
DCX rp	0B	1B	2B	3B
DAD rp	09	19	29	39

RLC 07	RRC 0F	DAA 27
RAL 17	RAR 1F	CMA 2F

ARITHMETISCH/LOGISCHE BEFEHLE

Bedingung	unc.	C	NC	Z	NZ	P	M	PE	PO
JMP XXYY	C3 YY XX	DA YY XX	D2 YY XX	CA YY XX	C2 YY XX	F2 YY XX	FA YY XX	EA YY XX	E2 YY XX
CALL XXYY	CD YY XX	DC YY XX	D4 YY XX	CC YY XX	C4 YY XX	F4 YY XX	FC YY XX	EC YY XX	E4 YY XX
RET	C9	D8	D0	C8	C0	F0	F8	E8	E0

PCHL E9

© 1980

n	0	1	2	3	4	5	6	7
RST n Sprungziel	C7 ØØØØ	CF ØØØ8	D7 ØØ1Ø	DF ØØ18	E7 ØØ2Ø	EF ØØ28	F7 ØØ3Ø	FF ØØ38

SPRUNGBEFEHLE

NOP 00	STC 37	EI FB	RIM 20
HLT 76	CMC 3F	DI F3	SIM 30

ÜBRIGE

8085-Maschinenbefehle

Tabelle 3: Erläuterung gebräuchlicher Abkürzungen

A	Register A
(A)	Inhalt von A
((H,L))	Inhalt derjenigen Speicherstelle, deren Adresse im rp H,L steht
m	Speicherstelle, deren Adresse im rp H,L steht
r	Register A, B, C, D, E, H oder L
rp	Registerpaar B&C, D&E oder H&L
SP	Stack Pointer (Register für die Unterprogramm-Verwaltung)
XX	Datenbyte XX
XX YY	16-Bit-Adresse XX YY
Sprungbedingungen:	
C (NC)	CARRY (NO CARRY)
Z (NZ)	ZERO (NON ZERO)
M (P)	MINUS (POSITIVE)
PE (PO)	PARITY EVEN (ODD)

PUSH rp (*Push*)
(rp) in den Stack überschreiben (retten)
POP rp (*Pop*)
(rp) aus dem Stack zurückholen

2. Arithmetische Befehle

ADD r (*ADD Register*)
(r) zu (A) addieren
ADC r (*ADD Register with Carry*)
(r) und CARRY zu (A) addieren
ADI XX (*ADD Immediate*)
Datenbyte XX zu (A) addieren
ACI XX (*ADD Immediate with CARRY*)
Datenbyte XX und CARRY zu (A) addieren
SUB r (*Subtract Register*)
(r) von (A) subtrahieren
SBB r (*Subtract Register with Borrow*)
(r) und CARRY von (A) subtrahieren
SUI XX (*Subtract Immediate*)
Datenbyte XX von (A) subtrahieren
SBI XX (*Subtract Immediate with Borrow*)
Datenbyte XX und CARRY von (A) subtrahieren
DAD rp (*Double Add Register Pair*)
(rp) zu (H&L) addieren (16-Bit-Addition)
DAA (*Decimal Adjust Accumulator*)
Dezimalkorrektur von (A)

3. Zählbefehle

INR r (*Increment Register*)
(r) um Eins erhöhen

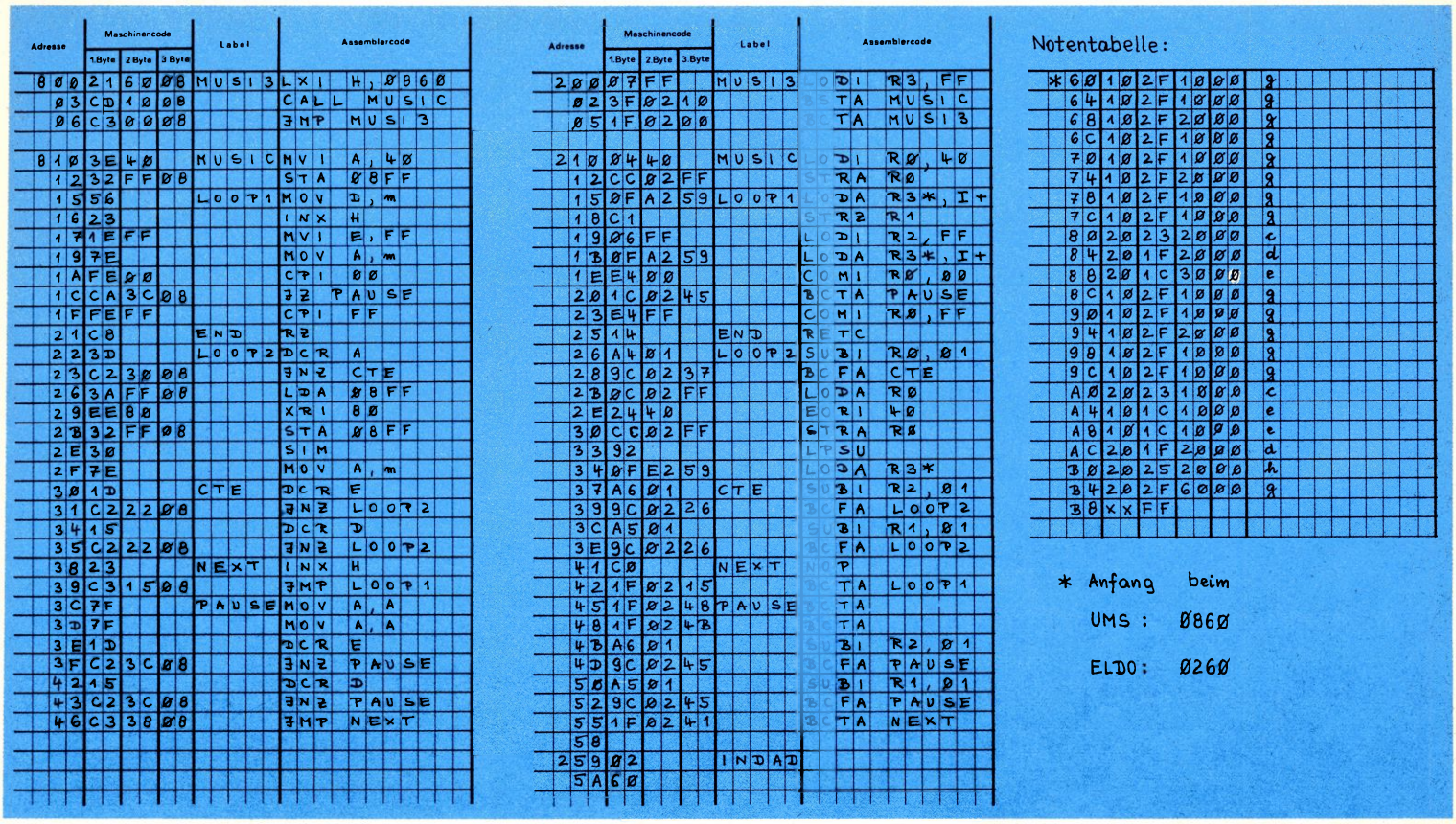


Bild 6: Die Maschinenprogramme nach Bild 5; für UMS und ELDO getrennt aufgeführt.

INX rp *(Increment Register Pair)*
(rp) als 16-Bit-Zähler um Eins erhöhen
DCR r *(Decrement Register)*
(r) um Eins erniedrigen
DCX rp *(Decrement Register Pair)*
(rp) als 16-Bit-Zähler um Eins erniedrigen

4. Logische Befehle

ANA r *(AND Register)*
UND-Verknüpfung von (A) und (r)
ANI XX *(AND Immediate)*
UND-Verknüpfung von (A) mit dem Datenbyte XX
XRA r *(Exclusive OR Register)*
EXOR-Verknüpfung von (A) und (r)
XRI XX *(Exclusive OR Immediate)*
EXOR-Verknüpfung von (A) mit dem Datenbyte XX
ORA r *(OR Register)*
ODER-Verknüpfung von (A) und (r)
ORI XX *(OR Immediate)*
ODER-Verknüpfung von (A) mit dem Datenbyte XX
CMP r *(Compare Register)*
Vergleich von (A) und (r); setzt die FLAGs
CPI XX *(Compare Immediate)*
Vergleich von (A) mit dem Datenbyte XX
CMA *(Complement Accumulator)*
(A) invertieren
STC *(Set CARRY)*
CARRY-FLAG setzen
CMC *(Complement CARRY)*
CARRY-FLAG invertieren

5. Schiebebefehle

RLC *(Rotate Left)*
(A) zyklisch links verschieben
RRC *(Rotate Right)*
(A) zyklisch rechts verschieben
RAL *(Rotate Left through CARRY)*
CY als 9. Bit beim Links-Schieben einbeziehen
RAR *(Rotate Right through CARRY)*
CY als 9. Bit beim Rechts-Schieben einbeziehen

6. Ein/Ausgabe-Befehle

IN XX *(Input)*
Daten von Port # XX nach A laden
OUT XX *(Output)*
(A) nach Port # XX laden
RIM *(Read Interrupt Mask)*
u. a. SID-Pegel nach A, Bit 7 laden
SIM *(Set Interrupt Mask)*
u. a. (A), Bit 7 nach SOD laden

7. Sprungbefehle

JMP XX YY *(Jump Unconditional)*
Sprung zur Adresse XX YY
JC, JNC, JZ, JNZ, JP, JM, JPE, JPO XX YY
Sprung nach XX YY, wenn betr. FLAG gesetzt/nicht gesetzt ist
CALL XX YY *(Call)*
Sprung ins Unterprogramm bei XX YY
CC, CNC, CZ, CNZ, CP, CM, CPE, CPO XX YY
Sprung ins U.P. bei XX YY, wenn betr. FLAG gesetzt/nicht gesetzt ist
RET *(Return)*
Rücksprung aus dem Unterprogramm
RC, RNC, RZ, RNZ, RP, RM, RPE, RPO
Rücksprung, wenn das betreffende FLAG gesetzt/nicht gesetzt ist
PCHL *(Move H&L to PC)*
Programmzähler mit (H&L) laden (Unbedingter Sprung)
RST n *(Restart 0...7)*
Einwort-Sprungbefehl mit fester Zieladresse

8. Übrige Befehle

EI *(Enable Interrupt)*
Interrupts freigeben
DI *(Disable Interrupt)*
Interrupts sperren
NOP *(No Operation)*
Füllbefehl
HLT *(Halt)*
Prozessor anhalten

Schalten nach Schema

Wie können Sie an verschiedenen Stellen Ihres Hauses Lichter ein- und ausschalten, um beispielsweise während des Urlaubs Anwensenheit vorzutäuschen und Dieben den Anreiz zum Einsteigen zu nehmen? Oder wollen Sie an anderen Stellen nach einem frei wählbaren Plan Verbraucher zu- und abschalten, etwa in Form einer Ampelsteuerung oder bei einem PROM-Programmierer? Wenn Sie über einen Mikrocomputer mit Ausgabe-Kanal verfügen, gehören solche Aufgaben zu dessen einfachsten Taten (**Bild 1**). Wie Sie dazu beim ELDO-Mikrocomputer vorgehen, haben Sie prinzipiell bereits in unserer Einführungsserie „Dem Mikrocomputer auf's Bit geschaut" (bzw. im gleichnamigen Sonderheft) erfahren. Der folgende Beitrag geht zunächst auf die Ein-/Ausgabemöglichkeiten des UMS-85 ein (vgl. Teil 1) und stellt dann ein für beide Systeme passendes Programm vor, das Ihren Mikrocomputer zum vielseitigen Schaltwerk macht.

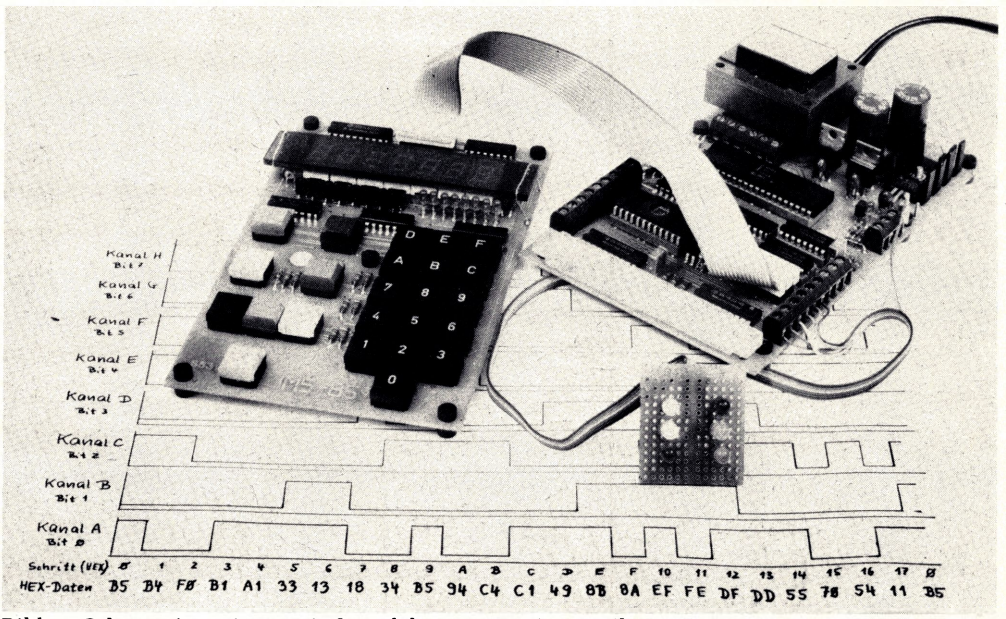


Bild 1: Schon mit geringem Aufwand kann man einen Mikrocomputer zum programmierbaren Schaltwerk machen.

Kanalisierter Datentransport

Für den Datenverkehr (Transfer) zwischen Mikrocomputer und dessen Umwelt gibt es sehr komplexe (und ebenso teure wie stromfressende) Interface-Bausteine, deren Handhabung recht kompliziert ist. Wie schon mehrfach betont, verzichten wir im Rahmen unserer einfachen Betrachtungen auf deren Einsatz, weil man auch mit bescheideneren Mitteln (fast) zum gleichen Ergebnis kommt.

Die Wege, auf denen sich der Datentransport zwischen Mikrocomputer und Peripherie abspielt (gleichgültig, in welcher Richtung), nennt man *Kanäle* oder *Ports*. Dient ein solcher Pfad nur zur Datenausgabe, heißt er „Ausgabe-Kanal" (*Output Port*); für den umgekehrten Weg gibt es entsprechend den Eingabe-Kanal (*Input Port*), und spezielle ICs (die oben erwähnten Interface-Bausteine) besitzen in der Regel sogar bidirektionale Ports, deren Übertragungsrichtung programmierbar ist (kombinierte Ein-/Ausgabe-Kanäle oder I/O-Ports).

Außer in der Übertragungsrichtung besteht eine weitere Unterscheidung im Format der zu übertragenden Daten. Diese können nämlich *wortparallel*, d. h. immer 8-Bit-weise, oder *bitseriell*, also kleckerweise ein Bit nach dem anderen übertragen werden. Wir wollen uns zunächst mit dem parallelen Datentransfer befassen, um Informationen aus dem Computer herauszubekommen bzw. dorthin zu übertragen. Zwei Dinge sind für den externen Datenverkehr wesentlich: Erstens spielt sich die Kommunikation immer zwischen einem festen CPU-Register (dem Akkumulator bzw. Register R0) und der Peripherie ab, und zweitens gelangen die Daten dabei immer über die dafür vorgesehene Sammelleitung, den Datenbus.

Zentraler Postverteiler: Die Decodier-Logik

Von der Vielzahl der verschiedenen E/A-Möglichkeiten wollen wir uns hier auf die E/A-Befehle IN XX (DB XX) zur Dateneingabe über Port XX bzw. OUT XX (D3 XX) zur Datenausgabe beschränken. Welche Zieladresse für den jeweiligen Kanal einzusetzen ist, hängt vom Aufbau des Mikrocomputer-Systems und der Organisation der Adreßdecodierung ab. Diese Decodier-Logik erzeugt in Abhängigkeit von der im E/A-Befehl spezifizierten Adresse einen Aktivierungs*impuls* (Select Signal; aktiv LOW), der nur diese eine Stelle im System erreicht. Für die Dauer dieses Impulses sind die Tri-State-Ausgänge der Eingangskanäle aktiv und schalten die eingangsseitig anliegende Information auf den Datenbus; von dort liest sie die CPU in den Akkumulator ein (**Bild 2**). Bei der Ausgabe übernehmen mit der positiven Flanke dieses Aktivierungsimpulses acht D-Flipflops diejenige Information vom Datenbus, die die CPU aus dem Akkumulator dorthin transportiert hat.

Bild 3 zeigt diesen Sachverhalt noch einmal anschaulich (vgl. auch Tabelle 2 im ersten Teil). Dort sind die insgesamt vier E/A-Kanäle des UMS-85 zu erkennen (je zwei für die Ein- und Ausgabe), wovon zwei vom System selbst belegt werden (Verwaltung von Tastatur und Anzeige) und zwei für den Anwender verfügbar sind. Beginnen wir bei den Anwender-Ports, die über lötfreie Schraubklemmen auf der CPU-Platine zugänglich sind. Bei beiden Klemmreihen finden Sie, wenn Sie frontal davor stehen, rechts außen Masse, gefolgt von Bit 0 (LSB)...Bit 7 (MSB) ganz links.

Raus und rein mit den Daten

Um nun Daten in den Ausgangskanal zu laden, geht man wie folgt vor: Gewünschtes Datenwort XY über MVI A,XY (3E XY) in den Akku laden und per OUT 04

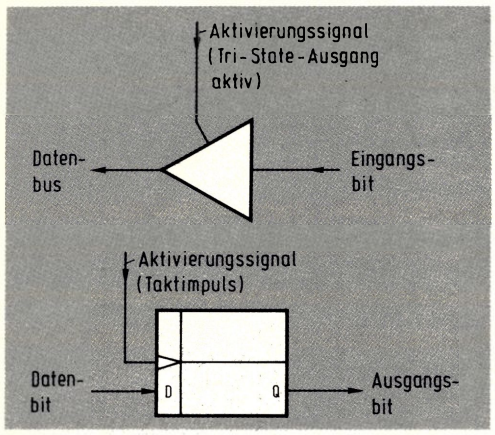


Bild 2: Tri-State-Gatter und D-Flipflop ersetzen in einfachen Anwendungen komplexe Peripheriebausteine.

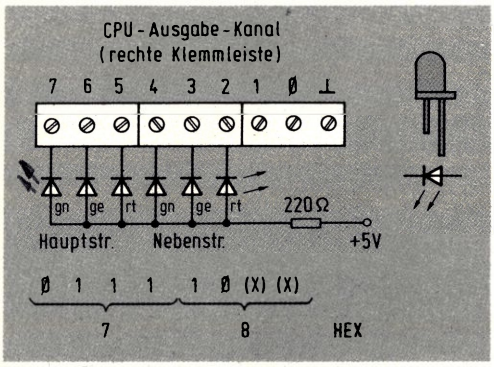


Bild 4: Verdrahtung des Ausgangskanals für eine Ampel-Demonstration.

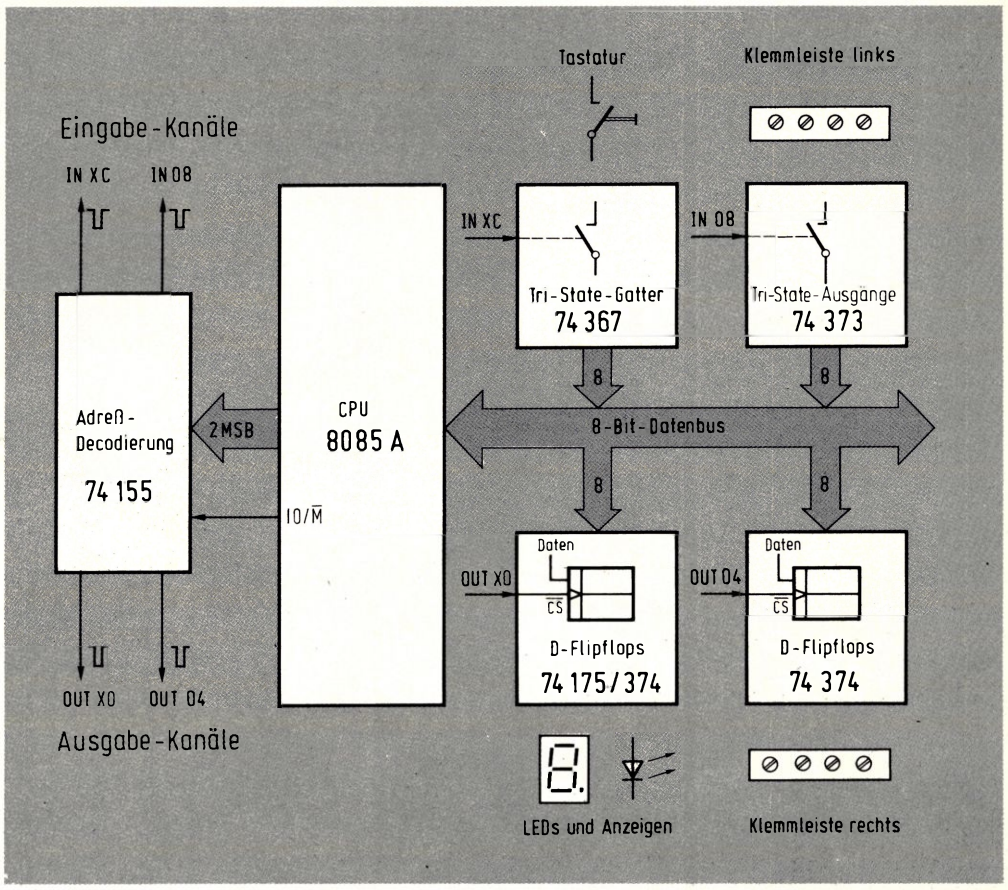


Bild 3: Blockschaltbild der vier Ein-/Ausgabe-Kanäle im UMS-85.

Klemme 2+7 bei „78" auf Low

(D3 04) ausgeben. Umgekehrt transportieren Sie den Pegel von den acht Bits des Eingangskanals in den Akku, wenn Sie den Befehl IN 08 (DB 08) verwenden. Das wollen wir am praktischen Beispiel anschaulich nachvollziehen.

Angenommen, sechs Bits des CPU-Ausgabe-Kanals sollen dazu dienen, eine Ampel anzusteuern; Sie können das recht plastisch demonstrieren, wenn Sie je zwei rote, gelbe und grüne Leuchtdioden verdrahten, wie es **Bild 4** zeigt (hundertprozentig, aber etwas aufwendiger wird die Sache, wenn jede LED ihren eigenen Vorwiderstand von 470 Ω erhält). Um eine der LEDs einzuschalten, muß das zugehörige Ausgangsbit auf LOW sein. Der Strom fließt dann von Plus über den Vorwiderstand, die LED und den leitenden Ausgangstransistor nach Masse; ist ein Ausgangsbit HIGH, bleibt die betreffende Leuchtdiode dunkel. Um, wie im Beispiel gezeichnet, die Bits 2 und 7 auf LOW zu setzen, müssen Sie die hexadezimale Information „78" in den Ausgabe-Kanal bringen (**Bild 5**). Neu in diesem Programm ist der HALT-Befehl HLT (76), der den Prozessor stoppt, bis ihn jemand über die RESET-Taste aus seinem Dornröschenschlaf erlöst. Hier soll er auch stoppen, nachdem er die Bits wie gewünscht gesetzt hat, denn bis zum nächsten Laden bleibt die Information im Ausgabe-Kanal konstant bestehen.

Wächter über Logikpegel: Die LED-Zeile

Wenn Sie das alles nicht glauben (oder Sie Ihre Mini-Ampel noch nicht angeschlossen haben), können Sie die Ausgangsinformation ja sowohl an die Klemmleiste (OUT 04) als auch an die LED-Zeile (OUT 00) laden; dort erstrahlen dann diejenigen LEDs, bei denen das korrespondierende Bit im Akkumulator auf HIGH liegt (**Bild 6**). Natürlich können Sie anstelle von „78" auch jede andere Information laden; dazu brauchen Sie nur die Speicherstelle 0801 zu modifizieren. Sie wissen doch noch, wie? Taste ADR drükken, gefolgt von 8–0–1; dann Taste DAT, gefolgt von 7–8, und zum Schluß NXT nicht vergessen!

Vervollkommnen Sie jetzt das Ganze, indem Sie in der LED-Zeile diejenige Information anzeigen, die Sie vom CPU-Eingangskanal einlesen (IN 08). Wenn Sie daraus eine Endlosschleife formen (Sprung zum Programmanfang: JMP INAUS), entsteht ein dynamisches Programm, dessen Auswirkungen zwar noch bescheiden sind, das aber Daten taufrisch in die CPU transportiert bzw. diese originalverpackt von dort herausbringt (**Bild 7**). Es sollte Sie nicht wundern, wenn nach dem Starten des Programms INAUS alle LEDs in der Zeile leuchten; die acht offe-

```
0800   3E    AUSG1   MVI A,78
0801   78
0802   D3            OUT 04
0803   04
0804   76            HLT
0805   XX
```

Bild 5: Sequenz zur Ausgabe eines Datenworts.

```
0800   3E    AUSG2   MVI A,78
0801   78
0802   D3            OUT 04
0803   04
0804   D3            OUT 00
0805   00
0806   76            HLT
0807   XX
```

Bild 6: Gleichzeitige Ausgabe des Datenworts an den CPU-Ausgabekanal und die LED-Zeile.

```
0800   DB    INAUS   IN 08
0801   08
0802   D3            OUT 00
0803   00
0804   C3            JMP INAUS
0805   00
0806   08
0807   XX
```

Bild 7. Darstellung der vom Eingangskanal eingelesenen Information an der LED-Zeile.

nen Eingänge des Eingabe-Kanals werden als HIGH gewertet, und das zeigen die Leuchtdioden auch artig an. Wenn Sie aber ein oder mehrere Bits des Eingangs-Kanals an Masse legen, verlöscht die korrespondierende Leuchtdiode. Besonders hinreißend ist das sicherlich noch nicht, aber Sie können sich ja damit trösten, daß reiche Leute jede LED ausschließlich auf diesem Umweg über einen Computer ein- und ausschalten!

Tabelle 1. Unterprogramme des UMS-Monitors und ihre Eigenschaften (Monitor-Version 3)

Anzeige	DISP2 #0372	stellt den Inhalt von 0BF8 in der Anzeige dar
	DISPL #011F	überschreibt DISP-Buffer 0BF0...F7 in die Anzeige; feste Laufzeit von 8 ms
	DSPSW#0275	zeigt (ACC) und Flag-Register an; Rücksprung erst nach Betätigen der NXT-Taste
Eingabe:	HHKEY#0061	fragt HEX-Tastatur ab; bei gedrückter Taste ist CY=1 und Tasten-Nr. im ACC
	NEXT #03A2	fragt NXT-Taste ab; bei gedrückter Taste ist CY=1, und (0BF8) wird nach 0BFE überschrieben
Eingabe/ Anzeige	NIBIN #0340	liest Nibble (Halbbyte) von der Tastatur ein und stellt es in der Anzeige dar; Rücksprung erst nach Betätigen der NXT-Taste
	INPUT #035A	liest Byte von der Tastatur ein und stellt es in der Anzeige dar; Rücksprung erst nach Betätigen der NXT-Taste
Laufzeit	DELY1 #03C7	feste Laufzeit von 1 ms
	DELAY#03BD	feste Laufzeit von 100 ms
	ONSEC#03B1	feste Laufzeit von 1 s
Akustik:	SOUND#02A1	Tonfrequenzerzeugung an SEROT (und +5 V)
	MUSIC#02A4	Abspielen einer Notentabelle (Tabellenanfang muß in rp H&L stehen); Tonfrequenz an SEROT (und +5 V)

Komfortabler wird's per Monitor

Einige Programme des Monitors können Sie in Ihre eigenen Programme einbauen, um auch die Ein- und Ausgabe von Daten eleganter zu gestalten (**Tabelle 1**). Wenn beispielsweise ein im Akkumulator stehendes Ergebnis in der Anzeige erscheinen soll, gehen Sie wie folgt vor: Sie transportieren den Akkumulator-Inhalt per Speicherbefehl STA 0BF8 (32 F8 0B) in die Speicherstelle 0BF8, und dann rufen Sie das Monitor-Unterprogramm DISP2 auf (CALL DISP2 = CD 72 03; vgl. Tabelle 1). Dieses Unterprogramm setzt nämlich den Inhalt von 0BF8 in den Siebensegmentcode um und überschreibt diesen anschließend in die Anzeige. Es zeigt also nicht unmittelbar den Akkumulator-Inhalt selbst an, aber auf dem gezeigten Umweg erreichen wir dieses Ziel doch (mit einem Mikrocomputer ist *alles* möglich, sofern man es nur klar definieren kann und genügend Zeit zur Verfügung steht!). Wichtig ist nur, daß dieses Unterprogramm in Form einer Endlosschleife durchlaufen wird, da es softwaremäßig die Multiplex-Frequenz für die Anzeige erzeugt.

Um beispielsweise den Akkumulator mit einem bestimmten Datenbyte zu laden, benutzen Sie die Sequenz nach **Bild 8**: Zuerst erfolgt das unmittelbare Laden der Information (z. B. „A7"), was mit dem MVI-Befehl vor sich geht (3E A7). Dann soll dieses Bitmuster in den Ausgabe-Kanal und die LED-Zeile gebracht werden, was die entsprechenden Ausgabe-Befehle OUT 04 (D3 04) und OUT 00 (D3 00) bewerkstelligen. Jetzt kommt das Ablegen dieser Information unter Adresse 0BF8 an die Reihe, gefolgt vom Aufruf des Unterprogramms DISP2. Dieses holt sich die Information aus 0BF8 und stellt sie in der Anzeige dar. Ein Rücksprung zum Programmanfang bei Adresse 0800 (C3 00 08) schließt die Endlosschleife. Der NOP-Befehl (00) in Adresse 0802 reserviert im Augenblick einen Speicherplatz, den wir gleich noch brauchen.

Auch zur Eingabe von der Tastatur steht Ihnen ein Monitor-Programm hilfreich zur Seite; es heißt „HHKEY", beginnt bei Adresse 0061 und fragt die HEX-Tastatur ab (die bunten Tasten bedient es *nicht*!). Beim Rücksprung aus HHKEY steht im Akkumulator die Nummer der aktivierten Taste; ist keine HEX-Taste gedrückt (oder nur die Null-Taste), bleibt der Akku leer. Das Programmbeispiel nach **Bild 9** unterscheidet sich von dem in Bild 8 nur in den ersten drei Bytes; jetzt erfolgt das Laden des Akkus nicht mehr mit einem festen, sondern mit dem von der Tastatur eingelesenen Wert. Und das ist doch schon um Klassen besser, verglichen mit dem simplen Ein- und Ausschalten der LEDs von eben!

Auch einzeln können Sie kommen

Für die Ein- und Ausgabe einzelner Bits besitzt der 8085 (ebenso wie der 2650) je

```
0800  3E      ANZEI  MVI A,A7
0801  A7
0802  00             NOP
0803  D3             OUT 04
0804  04
0805  D3             OUT 00
0806  00
0807  32             STA 0BF8
0808  F8
0809  0B
080A  CD             CALL DISP2
080B  72
080C  03
080D  C3             JMP ANZEI
080E  00
080F  08
0810  XX
```

Bild 8: Darstellung eines Speicherinhalts mit Hilfe des Unterprogramms DISP2.

```
0800  CD      TASTA  CALL HHKEY
0801  61
0802  00
0803  D3             OUT 04
0804  04
0805  D3             OUT 00
0806  00
0807  32             STA 0BF8
0808  F8
0809  0B
080A  CD             CALL DISP2
080B  72
080C  03
080D  C3             JMP TASTA
080E  00
080F  08
0810  XX
```

Bild 9: Eingabe von der Tastatur und Darstellung in der Anzeige.

Bild 10: Beeinflussung des SOD-Ausgangs am 8085.

```
0800  3E      BLINK  MVI A,C0
0801  C0
0802  30             SIM
0803  CD             CALL ONSEC
0804  B1
0805  03
0806  3E             MVI A,40
0807  40
0808  30             SIM
0809  CD             CALL ONSEC
080A  B1
080B  03
080C  C3             JMP BLINK
080D  00
080E  08
080F  XX
```

Bild 11: Die LED am SEROT-Ausgang blinkt mit diesem Programm im 1-Hz-Takt.

einen seriellen Ein- und Ausgang, der hier „SID“ bzw. „SOD“ heißt (zur Hebung Ihres Wissensstandes, aber nicht zum Blokkieren von Gehirnzellen: Abkürzung für *Serial Input* bzw. *Output Data*). Beide Anschlüsse sind an der dreipoligen Klemmleiste rechts auf der CPU-Platine verfügbar: Oben ist der serielle Ausgang SEROT, der von SOD angesteuert wird und einen vorgeschalteten Treibertransistor besitzt (parallel liegt eine LED zur Pegelanzeige, die bei HIGH an SOD leuchtet, während der Transistor dann *sperrt*); darunter ist eine Masseklemme, und ganz unten an diesem Dreierblock befindet sich der Eingang SERIN, der geradewegs an den CPU-Anschluß SID führt.

Vorsicht! An SERIN (und SID) darf nur TTL-Pegel angelegt werden (0...+ 5 V); Wechselspannung oder falsche Polarität können den armen Mikroprozessor zeitlebens verschandeln!

Um den CPU-Ausgang SOD zu beeinflussen, geben Sie den gewünschten Pegel (also 0 oder 1) ins höchstwertige Bit Nr. 7 des Akkumulators; außerdem muß das Bit 6 bei *jeder* Beeinflussung von SOD auf HIGH liegen (gleichgültig, ob Sie SOD auf 0 oder 1 setzen wollen; **Bild 10**). Wenn nach dieser Vorbereitung der SIM-Befehl folgt (30), gelangt der Pegel aus Bit 7 des Akkus nach SOD. Die Sequenz „MVI A,C0/SIM“ (3E C0/30) bringt SOD demzufolge auf HIGH, und „MVI A,40/SIM“ (3E 40/30) sorgt für LOW an SOD. Das Musikprogramm aus dem zweiten Teil dieser Reihe verwendet den SOD-Anschluß zur Erzeugung programmierbarer Rechtecksignale.

Um auch für die Einzelbit-Ausgabe ein Miniatur-Demonstrationsbeispiel anzuführen, können Sie wie folgt vorgehen: Sie erzeugen an SOD abwechselnd HIGH und LOW und warten dazwischen jedesmal eine Sekunde (bzw. Sie lassen das Programm eine Sekunde lang warten). Diese Zeitverzögerung kann das Unterprogramm ONSEC übernehmen, mit dessen Hilfe die Leuchtdiode an SEROT im Sekundentakt an- und ausgeht, sofern wir aus den genannten Elementen eine Endlosschleife formen (**Bild 11**).

Der Mops als Schaltautomat

In der Technik sind Schrittschaltwerke überhaupt nicht mehr wegzudenken. Das beginnt bei der Werkzeugmaschinensteuerung und führt über die Schaltung von Ampelanlagen bis hin zur Waschmaschine mit ihrer Vielzahl unterschiedlichster Verbraucher. Dank der Mikroelektronik kann man derartige Steuerungen heute „frei

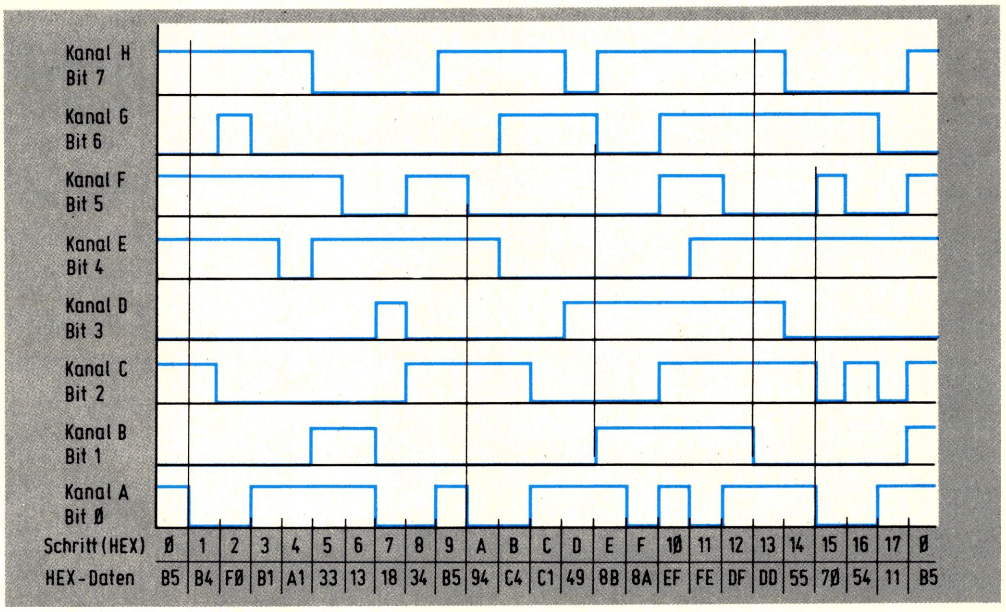


Bild 12: Beispiel einer achtkanaligen Ablaufsteuerung.

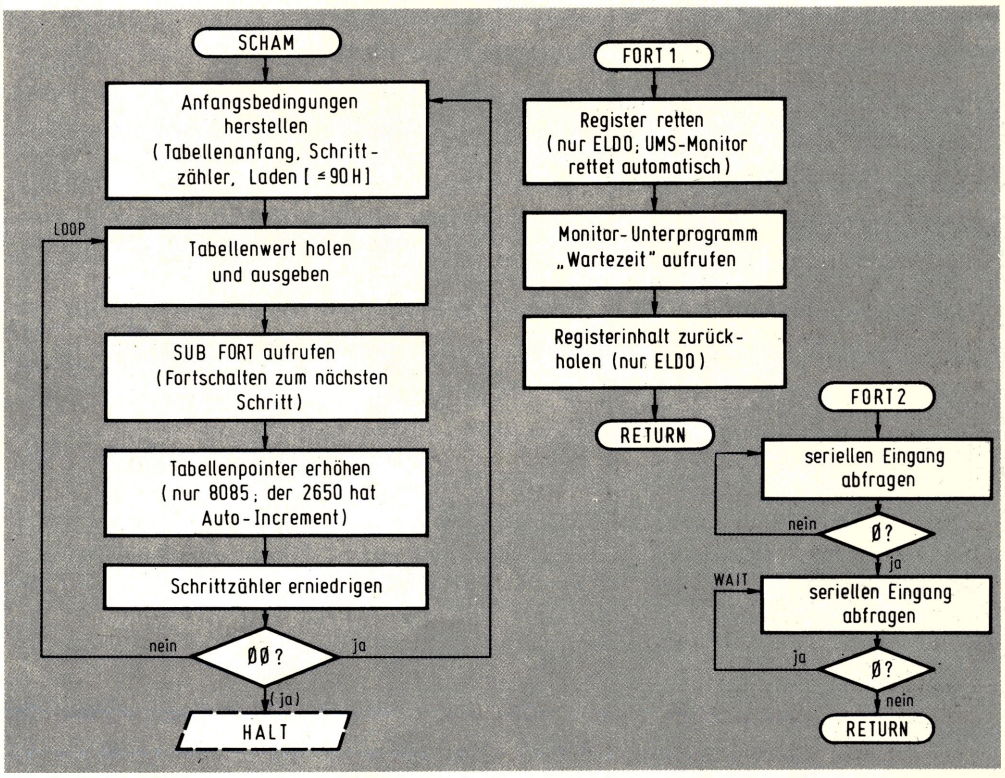


Bild 13: Flußdiagramm für das Programmbeispiel „Schaltautomat“.

programmierbar“ machen, d. h. die gewünschten Abläufe lassen sich ohne großen Aufwand per Tastatur vorgeben und auch problemlos ändern. Schauen wir uns an, mit welch' geringen Mitteln ein Mikrocomputer für diese Aufgabe gerüstet ist.

Stellen Sie sich einmal ein gewünschtes Impulsdiagramm nach **Bild 12** vor. Die einzelnen Kanäle können dabei beliebige Funktionen haben, z. B. Schalten der Hausbeleuchtung oder Ansteuern der Verbraucher in einer Waschmaschine (Ventil, Heizung, Schleuder usf.). Die acht Bits des Mikrocomputer-Ausgangskanal steuern diese einzelnen Schaltstellen an, und je nach Schritt-Nummer muß der Mikrocomputer die in der untersten Zeile von Bild 12 angegebene (hexadezimale) Information bereitstellen. Das fortlaufende Auslesen dieser Werte aus einer Tabelle übernimmt das Programm SCHAM (SCHALTAUTOMAT; **Bild 13**).

Nach dem Laden eines Tabellenpointers wird ein Tabellenwert nach dem anderen ausgelesen und an den parallelen Ausgangsport ausgegeben. Beim ELDO dient dazu der 8-Bit-Port auf dem Parallel-Interface; wer diese Platine nicht besitzt, kann

sich einen derartigen Ausgabe-Kanal für den ELDO (einschließlich LED-Anzeige) auch „selbst stricken" (**Bild 14**).
Zur Kontrolle soll der ausgegebene Tabellenwert auch noch in der Anzeige erscheinen; beim UMS-85 ist das die LED-Zeile, und beim ELDO das linke Digit. Dies blitzt nach dem Einschreiben allerdings nur kurz auf, weil die ELDO-Anzeige keinen eigenen Daten-Zwischenspeicher besitzt.
Für das Fortschalten von einem Schritt zum anderen ist das Unterprogramm FORT verantwortlich, für das es prinzipiell zwei Möglichkeiten gibt: Zwischen zwei Schaltstellungen kann jeweils eine feste Wartezeit vorgesehen sein, oder aber das Weiterschalten kann von einem externen Ereignis (Takt, Triggerimpuls) abhängig gemacht werden. Im ersten Fall handelt es sich um eine einfache Ablaufsteuerung, bei dem die Version FORT1 mit fester Wartezeit aufgerufen wird (1 s beim UMS und 0,5 s beim ELDO, resultierend aus den jeweiligen Monitor-Unterprogrammen); im zweiten Fall wird eine Synchronisation mit der Peripherie hergestellt, und die Unterprogramm-Version FORT2 muß aufgerufen werden. Dieser Programmteil wartet immer auf die *positive Flanke* am seriellen Eingang SERIN bzw. SENSE. *Aber bitte wieder nichts anderes als TTL-Pegel anlegen!* Natürlich können Sie provisorisch auch mit einem an Masse angeschlossenen Draht an SERIN antippen, um das Weiterschalten auszulösen. Damit legen Sie 0 V an und verstoßen nicht gegen das Gebot von oben.
Die Maschinenprogramme für das UMS-85 und den ELDO finden Sie in **Bild 15**. Wegen der Gegenüberstellung und gleichlautenden Dokumentation sind die Programme wiederum nicht optimiert; auch bezüglich der Listings verweisen wir auf den entsprechenden Hinweis im zweiten Teil dieser Reihe.
Die Tabellenwerte entsprechen dem Impulsplan nach Bild 12; bei anderen Abläufen müssen Sie ein neues Impulsdiagramm erstellen, für jeden Schritt die hexadezimale Information ermitteln und diese dann nacheinander in der Tabelle ablegen. Ob Ihnen dieser Schaltautomat dann als Stundenplan-Anzeige für die Schule dient (ein Taktimpuls täglich, und Sie wissen, welche Fächer/Freundin Sie erwartet), Sie einen Glücksspiel-Automaten aufbauen oder einfach eine Spielzeug-Ampel steuern wollen – den Einsatzmöglichkeiten sind wahrlich keine Grenzen gesetzt!

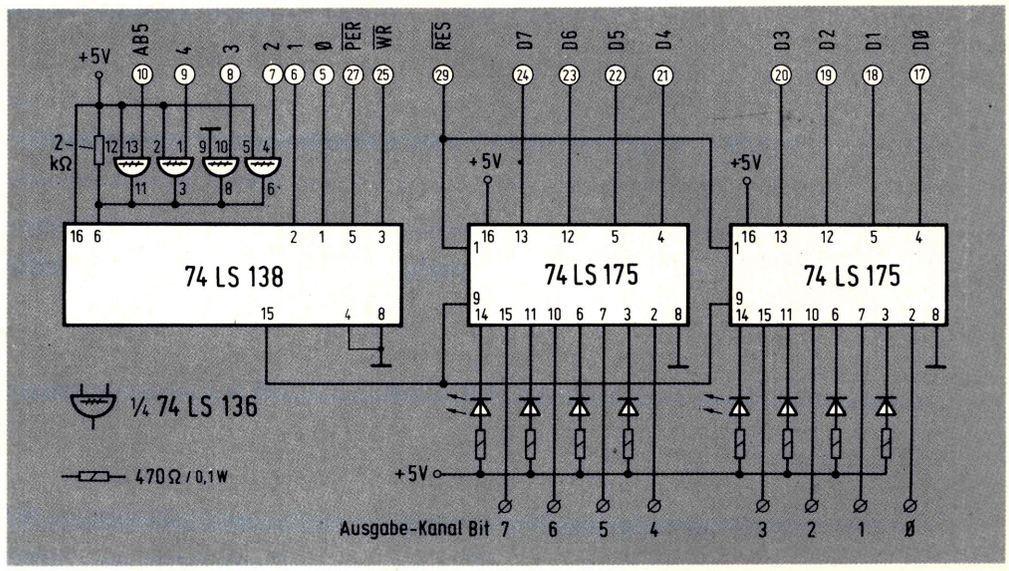


Bild 14: Ersatz des auf dem ELDO-Parallel-Interface enthaltenen Ausgabe-Kanals.

x = 18 = Schritt zahl

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode
		X			
8000	1E	18		SCHAM	MVI E, 18
0021	40	08			LXI H, 0840
0057E				LOOP	MOV A, M
0063	D3	04			OUT 04
0083	D3	00			OUT 00
00AC	D3	20	08	*)	CALL FORT1
00D2	3				INX H
00E1	D				DCR E
00FC	2	05	08		JNZ LOOP
0012	C3	00	08		JMP SCHAM
8020	00			FORT1	NOP
0021	00				NOP
0022	CD	B1	03		CALL ONSEC
0025	00				NOP
0026	00				NOP
0027	C9				RET
8028	20			FORT2	RIM
0029	E6	80			ANI 80
002B	C2	28	08		JNZ FORT2
002E	20			WAIT	RIM
002F	E6	80			ANI 80
0031	CA	2E	08		JZ WAIT
8034	C9				RET
	*) CD	28	08		bei FORT2

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode
2000	04	02		SCHAM	LODI R0, 02
0002	CC	02	21		STRA R0
0005	04	40			LODI R0, 40
0007	CC	02	22		STRA R0
000A	06	18			LODI R2, 18
000C	07	FF			LODI R3, FF
000E	0F	A2	21	LOOP	LODA R3*, I+
0111	D4	08			WRTE 08
0113	D4	00			WRTE 00
0115	3F	02	23	*)	BSTA FORT1
0118	C0				NOP
0119	A6	01			SUBI R2, 01
011B	9C	02	0E		BCFA LOOP
011E	1F	02	00		BCTA SCHAM
2121	(02)			INDAD	
2122	(40)				
2123	04	12		FORT1	LODI R0, 12
2125	93				LPSL
2126	3F	00	B6		BSTA DELAY
2129	04	02			LODI R0, 02
212B	93				LPSL
212C	17				RETC
212D	12			FORT2	SPSU
212E	44	80			ANDI R0, 80
2130	9C	02	2D		BCFA FORT2
3312				WAIT	SPSU
3444	80				ANDI R0, 80
3610	02	33			BCTA WAIT
2397					RETC
	*) 3F	02	2D		bei FORT2

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode
*4035				TABEL	
4134					
42F0					
4331					
44A1					
4533					
4613					
4718					
4834					
49B5					
4A94					
4BC4					
4C01					
4D49					
4E8B					
4F8A					
50EF					
51FE					
52DF					
53DD					
5455					
5570					
5754					
5711					

* Anfang beim
UMS : 0840
ELDO : 0240

Bild 15: Maschinenprogramm nach Bild 12, für UMS und ELDO getrennt aufgeführt.

Aus Bits wird Spannung

Ein ordentlicher Mikrocomputer ist dazu da, mit seiner Umwelt in ständigem Kontakt zu stehen (Datenaustausch) und auf Einflüsse von außen in bestimmter Form zu reagieren. So kann z. B. die Forderung bestehen, einen Temperatursensor abzufragen und unter Einbeziehung der Tageszeit eine Heizung ein- und auszuschalten. Da der Computer all seine Aktivitäten ausschließlich digital ausführt, besteht in solchen Fällen das Problem der Anpassung analoger Signale (z. B. Spannungen oder Widerstände) an die digitale Datenverarbeitungsanlage. Das speziell für diese Aufgaben entwickelte Analog-Interface erweitert unsere Mikrocomputer ELDO bzw. UMS-85 dahingehend, daß Sie damit vom Digitalvoltmeter bis zum computergesteuerten Dimmer die vielfältigsten Anwendungsbeispiele realisieren können. Die Karte ist als Bausatz (DM 153,–) bzw. fertige Baugruppe (DM 188,–) wiederum direkt vom Autor zu beziehen.

Ein- und Ausgänge in Hülle und Fülle

Aus der Blockschaltung (**Bild 1**) erkennen Sie bereits, daß sich auf dieser Interface-Karte einiges an verschiedenartigen Baugruppen tummelt. Kernstück ist ein integrierter Digital/Analog-Umsetzer (DAU), der primär dazu dient, digitale Eingangswerte in eine dazu proportionale analoge Größe umzusetzen. Die Ausgangsgröße (hier ein Strom) entspricht der Summe der Wertigkeiten der eingangsseitig auf HIGH liegenden Bits. Um dabei die volle Wortlänge unseres Mikrocomputers von 8 Bit auszunutzen, ist auch ein DAU mit 8-Bit-Auflösung eingesetzt worden. Die Wirkungsrichtung eines solchen DAU liegt demzufolge fest: Es handelt sich um eine **Ausgabe**-Schnittstelle des Systems, die allerdings mit geringfügigem Mehraufwand auch für die umgekehrte Übertragungsrichtung verwendbar ist. Es entsteht dann eine Einheit zur Analog/Digital-Umsetzung, mit der sich die eingangs erwähnte Funktion der Digitalisierung analoger Größen durchführen läßt. Wir kommen hierauf noch ausführlich zurück.

Dem DAU vorgeschaltet ist ein 8-Bit-Speicher (Latch), der von der CPU angesteuert wird. Die hierin festgehaltenen Daten landen im angeschlossenen DAU und werden dort schnurstracks in einen Strom entsprechender Größe umgesetzt. Der ausgangsseitig angeschlossene Operationsverstärker V1 setzt den DAU-Strom in eine proportionale Spannung um; die Verstärkung ist dabei über ein Poti einstellbar. Die hier entstehende analoge Ausgangsspannung ANA kann an einer Klemmleiste abgenommen werden; parallel geht sie an einen Komparator, der diesen Wert mit einer anderen, extern anliegenden Spannung UXT vergleicht. Dieser Komparator-Ausgang liefert eine Aussage darüber, welche der beiden Spannungen ANA und UXT größer ist. Und mit dieser Aussage ist das Digitalvoltmeter schon fast fertig, wie Sie nachher sehen werden.

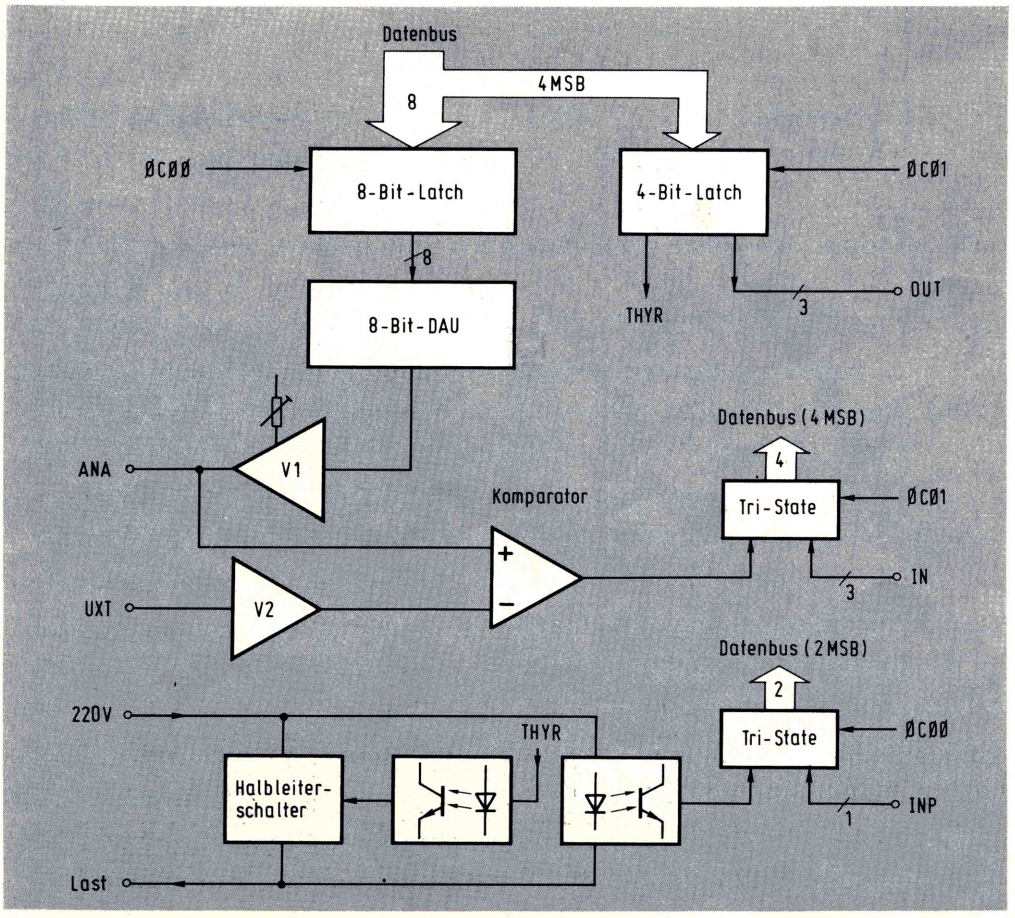


Bild 1: Aus der Blockschaltung sind die einzelnen Funktionseinheiten zu erkennen.

Informationen per Licht übertragen

Der zweite wesentliche Bestandteil dieser Karte ist ein kontaktloses Halbleiterrelais, das von einem Thyristor mit vorgeschaltetem Brückengleichrichter gebildet wird (**Bild 2**). Um damit auch Lasten im 220-V-Netz schalten zu können, erfolgt die Ansteuerung des Thyristors über einen Optokoppler, der für völlige galvanische Trennung zwischen Last- und Steuerkreis sorgt. Ein zweiter Optokoppler nimmt die Phasenlage der Netzfrequenz ab. Hierüber kann der Computer die Nulldurchgänge der Wechselspannung erkennen, um beispielsweise eine Phasenanschnittsteuerung (Dimmer) aufzubauen. Auf den Einsatz dieses Blocks kommen wir später (im 10. Teil) zurück, und Sie sollten solange noch keine wilden Experimente mit der Netzspannung unternehmen, damit Sie uns als Leser erhalten bleiben.

Die übrigen Funktionsblöcke haben die Aufgabe, die eben beschriebenen Signale in den Computer zu leiten bzw. sie von dort kommend zur Ansteuerung der verschiedenen Ziele zu benutzen. So landen beispielsweise die vier oberen Bits des Datenbus' in einem separaten 4-Bit-Speicher, von dem

ein Ausgang den Thyristor ansteuert (über den Umweg des Optokopplers). Zwei Eingangsleitungen ermöglichen es der CPU, den Zustand des Komparator-Ausgangs und des Phasendetektors abzufragen. Weil in den hierfür vorgesehen ICs noch Platz frei ist, fallen quasi nebenbei noch ein paar Ein-/Ausgabe-Leitungen ab. Sie sind ebenfalls an Klemmleisten zugänglich (IN4...6 bzw. OUT4...6) und landen an den Bits 4...6 des Datenbus'. **Bild 3** zeigt den Bestückungsplan dieser Platine.

Speicher oder Peripherie – das ist hier die Frage

Wenn Sie noch etwas für Ihre Mikrocomputer-Bildung tun wollen, vergraben Sie sich in diesem Abschnitt und lesen ihn mit Verstand; wenn Sie nur wild aufs Programmieren sind, genügt ein Überfliegen. – Es ist Ihnen nicht neu, daß das Ansprechen der Vielzahl verschiedener Speicherstellen über den Adreßbus erfolgt; separat sorgt eine Decodier-Logik dafür, daß je nach Adresse nur einzelne Blöcke daraus aktiviert werden. Natürlich kann man ein (freies) Ausgangssignal dieser Decodier-Logik auch dazu verwenden, anstelle einer Speicherstelle eine periphere Baugruppe anzusprechen, etwa den DAU bzw. das vorgeschaltete 8-Bit-Latch. Dann wird diese Einheit programmtechnisch genauso behandelt wie eine Speicherstelle, d. h. der Datentransport dorthin vollzieht sich per Speicher- bzw. Ladebefehl und Kenner reden vom *Memory Mapped I/O* („Memmorie Meppd Ei-Oh"; Ein-/Ausgabe-Verwaltung wie beim Speicher). Die Alternative dazu ist das *I/O-Mapped I/O*, bei dem zum Ansprechen der Peripherie spezielle Ein-/Ausgabe-Befehle eingesetzt werden, bei deren Ausführung ein CPU-Ausgang (MEM/$\overline{IO}$) aktiv ist; dieser zeigt der Decodier-Logik an, daß gerade eine periphere Stelle angesprochen wird, und zwar definitiv mit einem speziellen E/A-Befehl (IN bzw. OUT).
Jetzt sagen Sie nicht, daß Ihnen solche Feinheiten egal sind. Auf dem Weg zum Vollprofi erkennen Sie nämlich den fundamentalen Unterschied zwischen beiden E/A-Verwaltungen: Die Ein-/Ausgabe-Befehle (I/O-Mapped I/O) sind äußerst plumpe Datentransportbefehle, die nichts weiter bewirken als den Informationsaustausch zwischen einem festen CPU-Register und einer ebenfalls festen Peripherieadresse; so müssen zur Ansteuerung der UMS-Anzeige z. B. acht einzelne Lade- und Ausgabe-Befehle aneinandergereiht werden, weil eine Programmschleife mit indirekter Adressierung bei E/A-Befehlen nicht möglich ist. Wohl

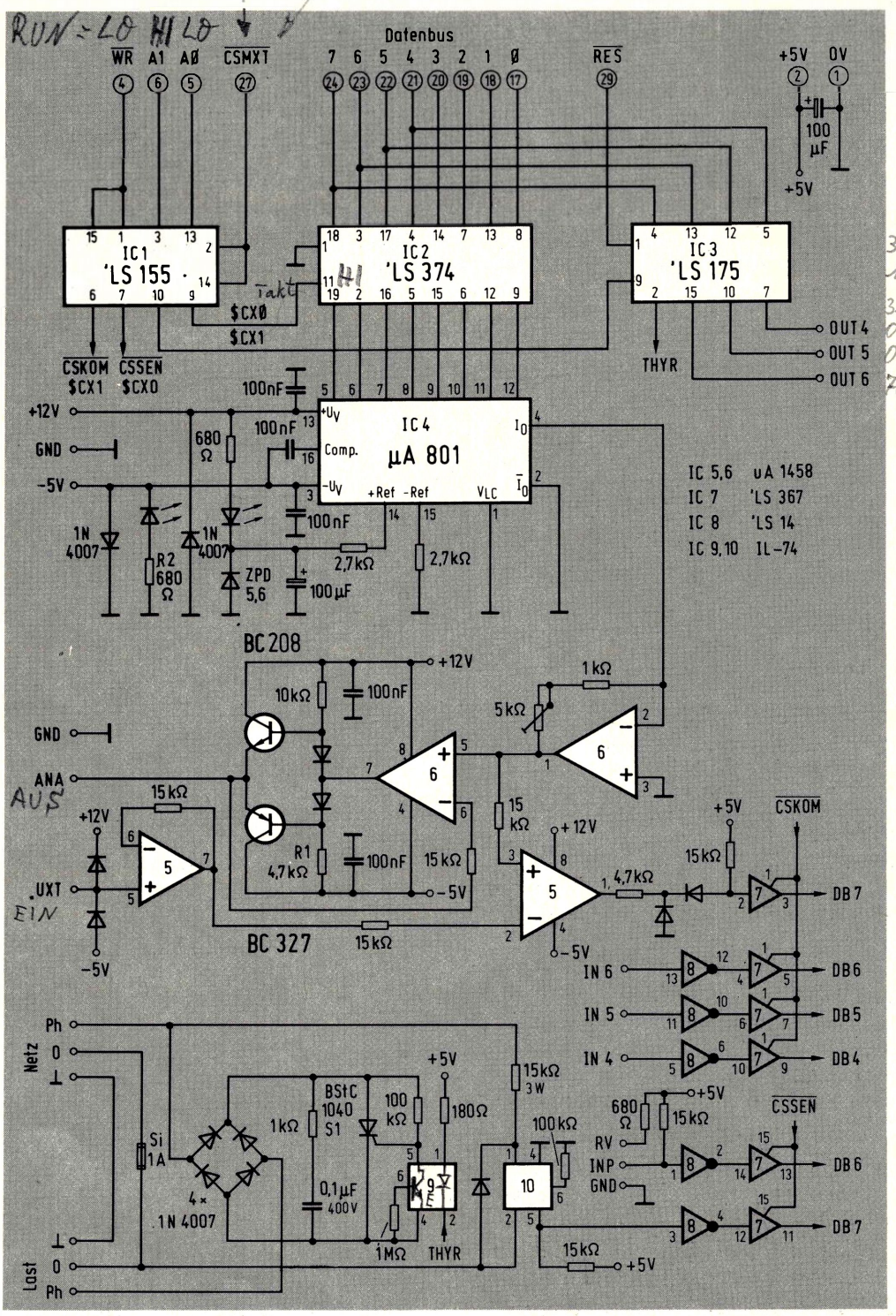


Bild 2: Das Detailschaltbild dieser Interface-Karte.

möglich ist dies beim Memory Mapped I/O, was demzufolge wesentlich eleganter zu handhaben ist. Warum man das dann nicht immer anwendet? Weil die derart belegten Adressen vom Speicherbereich abgehen und man den Mehraufwand für eine detaillierte Decodierung scheut.
Um diese elegante Ansteuerung auch beim Betrieb mit dem ELDO zu erreichen, sind auf der Rückseite der Interface-Platine zwei winzige Modifikationen erforderlich: Trennen Sie, wie in **Bild 4** gezeigt, die beiden Leiterbahnen auf und löten Sie dann nach Skizze die beiden Brücken ein; damit ergibt sich dann eine Adreßzuordnung gemäß **Tabelle 1**.

Mit dem Voltmeter an die Bits

Nach dieser Vorrede sollen Sie endlich Ihre Digital/Analog-Umsetzung haben, was ohne Zweifel recht eindrucksvoll ist, schlägt dies doch die Brücke zwischen analoger und digitaler Welt. Stecken Sie dazu die Interface-Karte in die Buchsenleiste der UMS-CPU bzw. in einen freien Steckplatz beim ELDO und schließen oben −5 V und +12 V an.

Diese Spannungen sind direkt am UMS-Netzteil abgreifbar (vgl. 1. Teil), und beim ELDO sind sie auf der großen Speicherkarte zugänglich. Sie können beim ELDO auch direkt die −12 V abnehmen und an die Klemme „−5 V" anschließen, sollten dann aber die Werte für R1 und R2 verdoppeln (für R1 = 10 kΩ und R2 = 1,5 kΩ einsetzen). Zur Kontrolle des richtigen Anschlusses leuchten die beiden grünen LEDs; bei Verpolung schützen die antiparallel geschalteten „Dummheitsdioden" die nachfolgende Schaltung, und die beiden LEDs geben kein grünes Licht.

Eichen Sie nun wie folgt Ihren D/A-Umsetzer: Voltmeter an ANA (Pluspol) und GND anschließen, „FF" an den DAU ausgeben (Adresse 0C00 bzw. 0300; vgl. Tabelle 1)

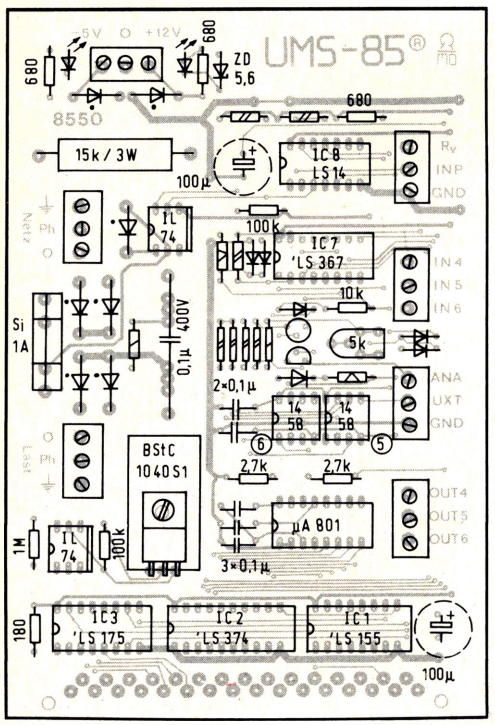


Bild 3: Der Bestückungsplan zeigt die übersichtliche Anordnung der Bauteile.

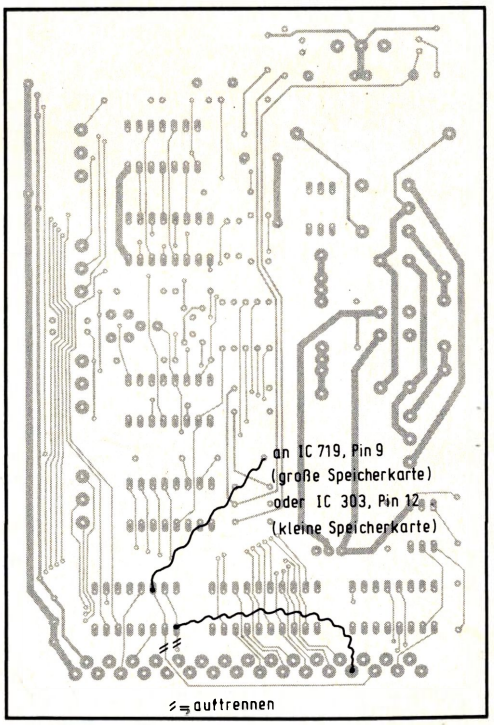


Bild 4: Für den Betrieb am ELDO müssen die gezeigten Modifikationen auf der Rückseite der Interface-Karte durchgeführt werden.

Tabelle 1. Adreßbelegung der Ein- und Ausgänge auf dem Analog-Interface.

Ausgänge:	Daten	UMS-85	ELDO*
D/A-Umsetzer	Bits 0...7	XC00	X300
Thyristor-Port	Bit 7	XC01	X301
OUT 4...6	Bits 4...6	XC01	X301
Eingänge:			
Netz-Phasenlage	Bit 7	XC00	X300
Sensor-Eingang			
INP	Bit 6	XC00	X300
IN 4...6	Bits 4...6	XC01	X301

* bei der Modifikation gemäß Bild 4

und das Poti so einstellen, daß am Ausgang 10 V anstehen; programmtechnisch gehen Sie so vor, wie es **Bild 5** zeigt. Wenn Sie danach statt „FF" andere Werte laden und ausgeben, ändert sich die Analogspannung dementsprechend; während die hexadezimale Information FFH (= 255D) dem Vollausschlag von 10 V entspricht, stellen sich bei 7FH (= 127D) nur noch etwa 5 V ein (genau 127/255 · 10 V; **Bild 6**).

Der UMS-Monitor besitzt ein Unterprogramm INPUT (#035A; vgl. Tabelle 1 im 3. Teil), das diese Ansteuerung noch eleganter gestaltet. Mit einem Programm nach **Bild 7** können Sie per Tastatur Daten eingeben und sich anzeigen lassen; nach Betätigen der NXT-Taste überschreibt dieses Programm die Daten in den DAU, und Sie können dann (ohne RESET) fortlaufend neue Werte eingeben, die mit jedem weiteren NXT die Ausgangsspannung ändern. **Bild 8** zeigt der Vollständigkeit halber das zugehörige Maschinenprogramm.

Wenn sich der Zeiger Ihres Voltmeters kontinuierlich weiterbewegen soll, können Sie ein Register (ACC bzw. R0) hochzählen und diesen stetig wachsenden Wert an den DAU ausgeben (**Bild 9**). Es entsteht dann ein sägezahnförmiger Spannungsverlauf (Rampenfunktion, **Bild 10**), bei dem Sie bei genauem Hinsehen (z. B. am Oszillografen)

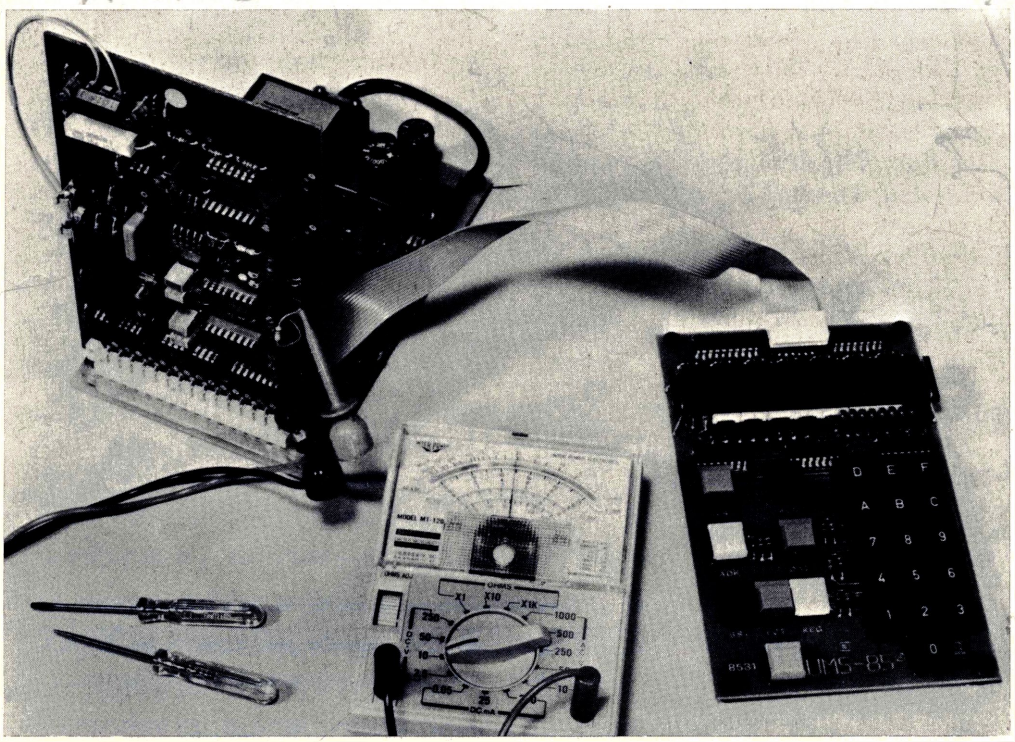


Bild 6: Wiederum über lötfreie Klemmleisten sind die einzelnen Anschlüsse am Analog-Interface zugänglich.

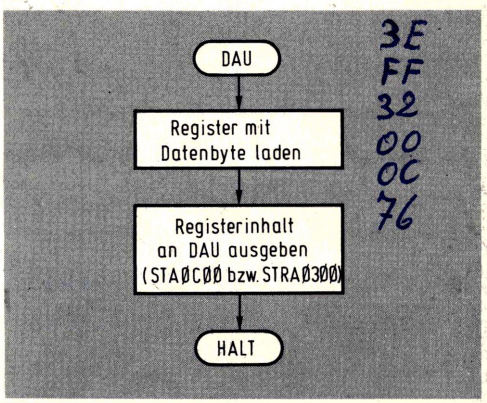


Bild 5: Zwei Befehle genügen, um einen bestimmten Wert in den Digital/Analog-Umsetzer zu laden.

die winzigen Spannungsstufen erkennen können (je 1/255 Vollausschlag), in denen sich die ausgangsseitige Änderung vollzieht. Mit der eingefügten Zeitverzögerung bremsen Sie das Hochzählen, um es mit dem trägen Auge bzw. Zeigerinstrument verfolgen zu können. Bei insgesamt 256 Zählschritten (danach geht es wieder bei Null los) und einer Zeitverzögerung von 100 ms dauert ein Zeigerdurchlauf 25,6 s. Bei längeren Zeitverzögerungen können Sie ganze Abende mit diesem Programm überbrücken, was eine abwechslungsreiche Alternative zum täglichen TV-Einerlei darstellt!

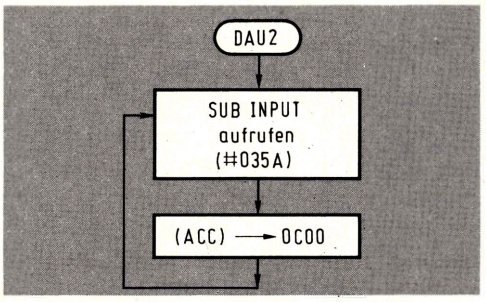


Bild 7: Mit einem UMS-85-Monitor-Programm gestaltet sich die dynamische Eingabe von Daten noch einfacher und eleganter.

```
0800  CD  DAU2  CALL  INPUT
0801  5A
0802  03
0803  32        STA   0C00
0804  00
0805  0C
0806  C3        JMP   DAU2
0807  00
0808  08
0809  XX
```

Bild 8. Das Maschinenprogramm zu Bild 7.

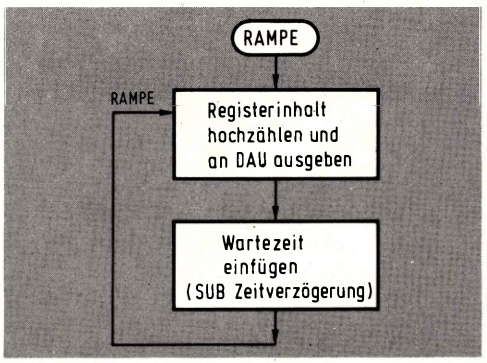


Bild 9: Ein besonders einfaches Beispiel für digital erzeugte Analogspannungen ist der Sägezahnverlauf.

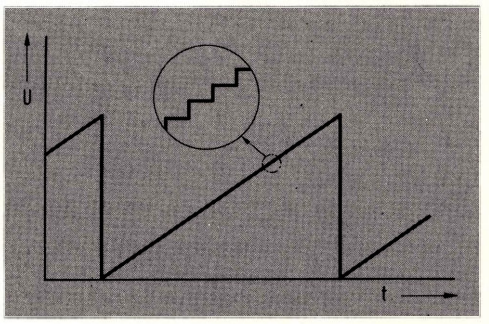


Bild 10: Bei genauem Hinsehen erkennt man, daß sich der Spannungsverlauf am DAU-Ausgang nicht stetig, sondern in winzig kleinen Stufen ändert.

Mit List und Tücke: Der DAU wird zum ADU

Wenn Sie in eine Kaffeemaschine oben Pulver und Wasser hineintun, kommt unten fertiger Kaffee heraus. Niemand käme auf die Idee, von einem solchen Apparat die umgekehrte Funktionsrichtung abzuverlangen (braune Brühe rein und klares Wasser mit Kaffeepulver raus). Unser Digital/Analog-Umsetzer aber kann in beiden Richtungen arbeiten (also auch als Analog/Digital-Umsetzer), wenn wir ihm nur eine winzige Hilfestellung in Form eines Komparators (Operationsverstärker) beistellen. Bevor wir damit ein funktionierendes Digitalvoltmeter (DVM) aufbauen, lesen Sie bitte mit feierlichem Ernst den folgenden Abschnitt: das bloße Eintippen des Programms mit gelangweilter Inbetriebnahme würde Ihren Mikrocomputer geradezu entweihen, weil es sich hierbei um einen der Höhepunkte Ihres gesamten Mikrocomputer-Daseins handelt!

Das folgende Programmbeispiel umfaßt ausgefeilte Interface-Techniken (einschließlich D/A- und A/D-Umsetzung), Abfragen externer Zustände (Komparator-Ausgang) mit bedingten Sprüngen, Schleifen, Unterprogramm-Verwaltung, gezieltes Setzen und Löschen einzelner Bits und die prinzipielle, äußerst geschickte Vorgehensweise bei der Mikrocomputer-Programmierung. Es bietet Ihnen so viel an Grundlagen, daß Sie darauf aufbauend Ihr Mikrocomputer-Wissen umfassend ausbauen können, wenn Sie sich nur ein wenig näher damit beschäftigen.

Zur Digitalisierung analoger Signale gibt es mehrere Möglichkeiten, von denen das Prinzip der schrittweisen Annäherung (Sie können auch sagen *Successive Approximation* [„Sacksessif Epproximaischen"]) besonders trickreich und Mikroprozessor-gerecht ist: Man setzt zunächst das höchstwertige Bit (MSB) des DAU auf HIGH (halber Vollausschlag) und vergleicht die DAU-Ausgangsspannung ANA mit der zu digitalisierenden Meßspannung UXT; diesen Vergleich übernimmt der Komparator. Sein Ausgang geht auf HIGH, wenn ANA>UXT ist; ist dies der Fall, dann ist die Wertigkeit des MSB zu groß. Es muß wieder auf Null gesetzt werden, und man probiert nun das nächsttiefere Bit. Bleibt der Komparator-Ausgang auf LOW, wird im digitalen Ergebniswort an dieser Bit-Position eine 1 gesetzt, im anderen Fall eine 0. Diese Entscheidung ist für den laufenden Umsetz-Zyklus endgültig, d. h. von der Stellung der nächstfolgenden Bits nicht abhängig; denn selbst alle folgenden Bits zusammen haben nicht das Gewicht wie das gerade aktivierte.

Das gezielte Setzen eines Bits geschieht per ODER-Verknüpfung mit einem Wort, bei dem das zu *setzende* Bit auf HIGH ist; das gezielte Löschen eines HIGH-Bits (bzw. das gezielte Invertieren eines Bits) vollzieht sich per EXOR-Verknüpfung (Exklusiv-ODER) mit einem Wort, bei dem das zu *löschende* (invertierende) Bit auf HIGH ist (vgl. „Dem Mikrocomputer auf's Bit geschaut").

Nach acht Setz- und Abfrage-Zyklen ist das Ergebnis komplett, und mit einer Unschärfe in der Größe des kleinstwertigen Bits hält das Ergebnisregister denjenigen Digitalwert, der der analogen Eingangsgröße entspricht. Er kann dann angezeigt werden, ehe ein neuer Umsetz-Zyklus beginnt (**Bild 11**).

Ein hexadezimales Voltmeter

Eichen Sie Ihren Analog-Ausgang nun auf 2,55 V bei Vollausschlag und schließen Sie eine externe Spannung von 0... ca. 3 V an Klemme „UXT" an. Das können Sie z. B. durch ein Poti simulieren, wie es in **Bild 12** gezeigt ist. Nach Eingabe und Starten des Programms (**Bild 13**) zeigte Ihre Anzeige hexadezimal an, welche Spannung Sie eingestellt haben. Drehen Sie das Poti einmal hin und her und verfolgen Sie, wie Ihr Digitalvoltmeter artig von 00 (UXT = 0 V) bis FF (UXT ≥ 2,55 V) nachzieht.

Die Eichung auf 2,55 V Vollausschlag hat den Vorteil, daß Sie mit Ihrem hexadezimalen Fachblick eine Anzeige „67" (HEX) sofort als dezimal 103 interpretieren, was in diesem Fall 1,03 V entspricht. Wäre es nicht eine lohnende Aufgabe für Sie, ein Unterprogramm zur Binär/Dezimal-Umwandlung einzufügen?

Bitte erwarten Sie jetzt nicht, daß Sie mit diesem Aufbau ein 15 000-DM-Laborvoltmeter ausstechen können. Verschiedene Einflüsse (Brummeinstreuungen, Rippel auf der Versorgungsspannung, preiswerte Bauteile usf.) führen zu Schwankungen in der Anzeige, aber die grundsätzliche Vorgehens- und Arbeitsweise können Sie recht eindrucksvoll kennenlernen und studieren!

Wenn Sie ein Oszilloskop an ANA und GND anschließen (der Widerstand dient nur als Klemmhilfe; er kann zwischen

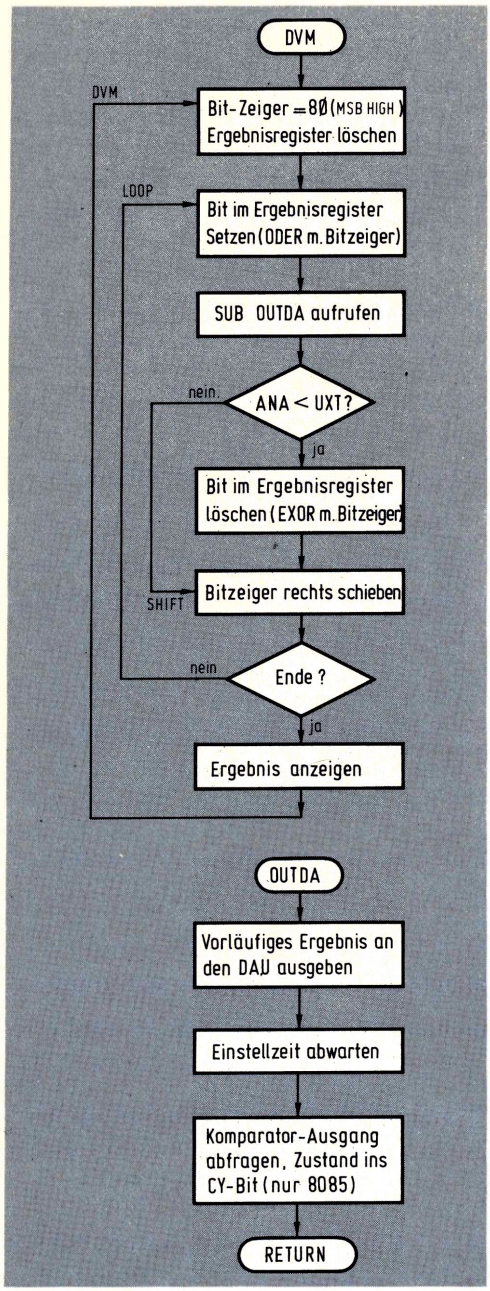


Bild 11: Dieses einfache Programmbeispiel macht aus dem Mikrocomputer samt Analog-Interface ein funktionierendes Digitalvoltmeter.

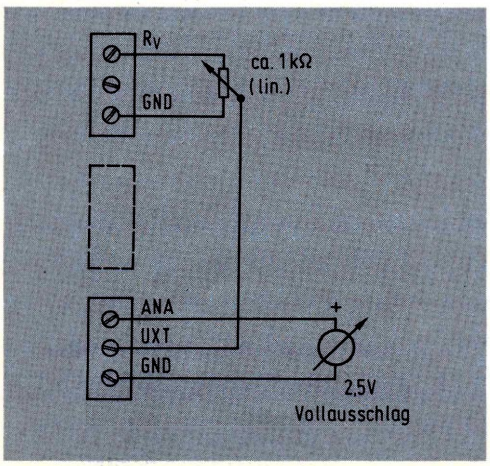


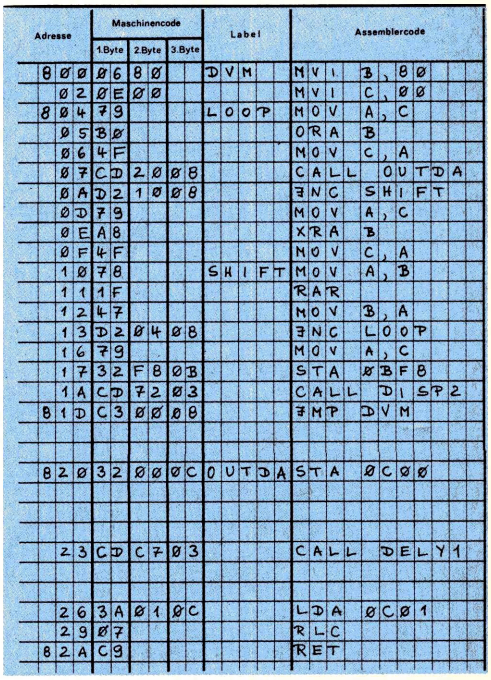
Bild 12: Der Poti-Abgriff simuliert eine zu messende Spannung; Sie können natürlich auch z. B. eine Batterie anschließen.

470 Ω und 1 MΩ groß sein), können Sie plastisch verfolgen, wie sich das Programm schrittweise an den Endwert annähert.

Bild 14 zeigt diesen Sachverhalt noch einmal anschaulich, und Sie können erkennen (mit dem Oszilloskop sehen Sie es live!), wie sich der Endwert aufbaut. Das Ergebnis ist immer in Skalenteilen vom Endwert zu verstehen und hängt demzufolge betragsmäßig von der anfangs durchgeführten Eichung ab.

Es sei der Vollständigkeit halber erwähnt, daß dieses Verfahren eine während der Umsetzzeit konstante Eingangsspannung voraussetzt. Die im Unterprogramm OUTDA eingebaute Wartezeit läßt dem D/A-Umsetzer so viel „Luft", daß er sich auf den neuen Eingangswert einstellen kann, um wohlgeordnet den passenden Ausgangsstrom bereitzustellen. Das brauchen eigentlich nur ca. 20 µs zu sein, aber der Einfachheit halber greifen wir auf eins der Monitor-Unterprogramme zurück.

Wenn Sie dieses Programm bis ins Detail nachvollzogen und verstanden haben, sind Sie um ein ganz fundamentales Stück Mikrocomputer-Erfahrung reicher!



Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode
8000	06	80		DVM	MVI B,80
0002	0E	00			MVI C,00
8004	79			LOOP	MOV A,C
0005	B0				ORA B
0006	4F				MOV C,A
0007	CD	20	08		CALL OUTDA
000A	D2	10	08		JNC SHIFT
000D	79				MOV A,C
000E	A8				XRA B
000F	4F				MOV C,A
1078				SHIFT	MOV A,B
1111	F				RAR
1247					MOV B,A
1300	D2	04	08		JNC LOOP
1679					MOV A,C
1732	F8	0B			STA 0BF8
1ACD	72	03			CALL DISP2
811D	C3	00	00		JMP DVM
8200	32	00	0C	OUTDA	STA 0C00
2300	CD	C7	03		CALL DELY1
2600	3A	01	0C		LDA 0C01
2907					RLC
820A	C9				RET

Bild 13: Das Maschinenprogramm nach Bild 11, für UMS-85.

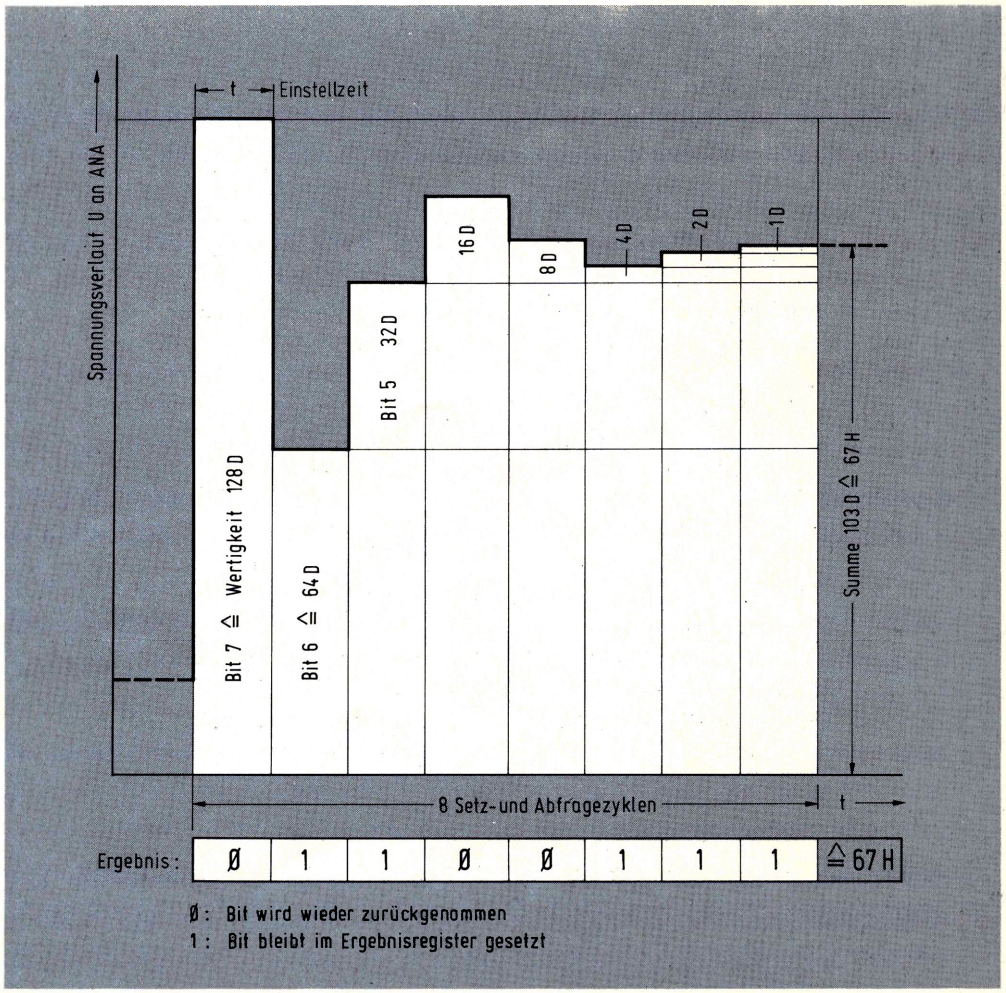


Bild 14: Diese Grafik verdeutlicht die Vorgehensweise bei der schrittweisen Annäherung an den Endwert.

Das tolle Telefon

Wir tragen heute digitale Datenverarbeitungsanlagen in der Westentasche mit uns herum, lassen den Fotoapparat vollautomatisch die Belichtung ausführen und rüsten unsere Autos mit Bordcomputern aus. Nur wenn es darum geht, eine der ältesten technischen Erfindungen, das Telefon, zu benutzen, leben wir größtenteils noch in der Steinzeit.
Stand der Technik wäre es sicherlich, die löchrige Wählscheibe durch eine elegante Tastatur zu ersetzen und die Telefonnummer für Rufwiederholungen nicht nur zu speichern, sondern einigen bevorzugten Teilnehmern eine Festtaste zuzuordnen (**Bild 1**). Der folgende Beitrag beschreibt, wie Sie Ihren Mikrocomputer UMS-85 bzw. ELDO für diesen Einsatz vorbereiten können.
Vorsorglich weisen wir darauf hin, daß ein Eingriff in Ihr (von der Post nur geliehenes) Telefon natürlich nicht statthaft ist; für private Nebenstellenanlagen (oder nach Erteilen einer Genehmigung für private Rufnummerngeber) ist dieser Telefonkomfort allerdings eine schicke Sache!

Bild 1: Der Einsatz als Rufnummerngeber bereitet einem Mikrocomputer keinerlei Schwierigkeiten.

Ein Bus mit noch mehr Plätzen

Bevor wir zu dem eigentlichen Programm- und Anwendungsbeispiel kommen, sollen Sie noch die Bus- und Speichererweiterungskarte 8501 kennenlernen, die das RAM des UMS-85 auf 1,25 KBytes erweitert und gleichzeitig drei Steckplätze für später vorgestellte Interface-Karten bietet (**Bild 2**). Die Karte wird mit ihrer Stiftleiste einfach in die dafür vorgesehene Buchsenleiste der CPU eingesteckt; ihre Stromversorgung erhält sie allerdings separat (s.u.). Mit dieser Erweiterungsplatine zieht das UMS-85 vom Arbeitsspeicher-Umfang her quasi gleich mit dem ELDO. Um diesem langgehegten Wunsch vieler UMS-Besitzer nachzukommen, folgen zunächst die kurze Vorstellung und Baubeschreibung.
Aus dem vereinfachten Blockschaltbild (**Bild 3**) erkennen Sie die Organisation der

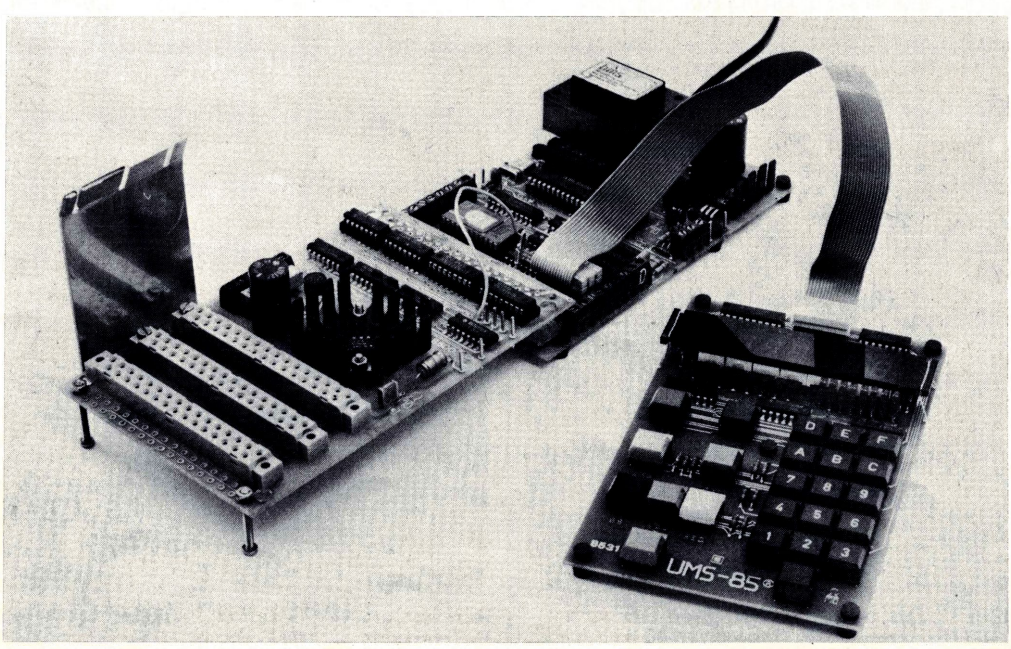


Bild 2: Die Bus-Erweiterungskarte wird in die Buchsenleiste der CPU-Platine eingesteckt.

Karte: Der Arbeitsspeicher umfaßt zwei IC-Paare und teilt sich funktionell auf in RAM 0 und RAM 1. Die beiden ICs 2112 (RAM 0) stammen von der CPU, von wo sie beim Betrieb der Erweiterungskarte entfernt werden müssen; sie landen in den beiden dafür vorgesehenen Steckplätzen auf der Erweiterungskarte und belegen dort den Adreßbereich XF00...XFFF (256 Bytes). Dieser Speicherbereich wird vom Monitor vollkommen in Ruhe gelassen. Es bleibt Ihnen freigestellt, was Sie damit machen, d. h. Sie können dorthin beliebig Daten oder Programme ablegen bzw. zurückholen oder ausführen. Für den Speicherausbau besitzt die Karte zwei ICs 2114 (RAM 1), die den Adreßbereich X800...XBFF (1024 Bytes = 1 KB) belegen. Die Anfangsadresse X800 ist Ihnen bereits von der Grundausbaustufe des Systems her bekannt; auch dort war dies schon die erste Adresse des Arbeitsspeichers, der allerdings nur bis X8FF reichte. Und von diesen 256 vorhandenen RAM-Worten hat das Monitor-Programm für den Eigenbedarf noch einmal bis zu 36 Bytes belegt (die obersten 36 Adressen). Bei den jetzt vorhandenen 1024 (+256) RAM-Zellen greift sich das Monitor-Programm ebenfalls 36 davon (wiederum die obersten), was aber weit weniger ins Gewicht fällt, als wenn diese von nur 256 vorhandenen abgehen.

Der Monitor denkt mit

Haben Sie eben bemerkt, daß sich das Monitor-Programm in jedem Fall die obersten

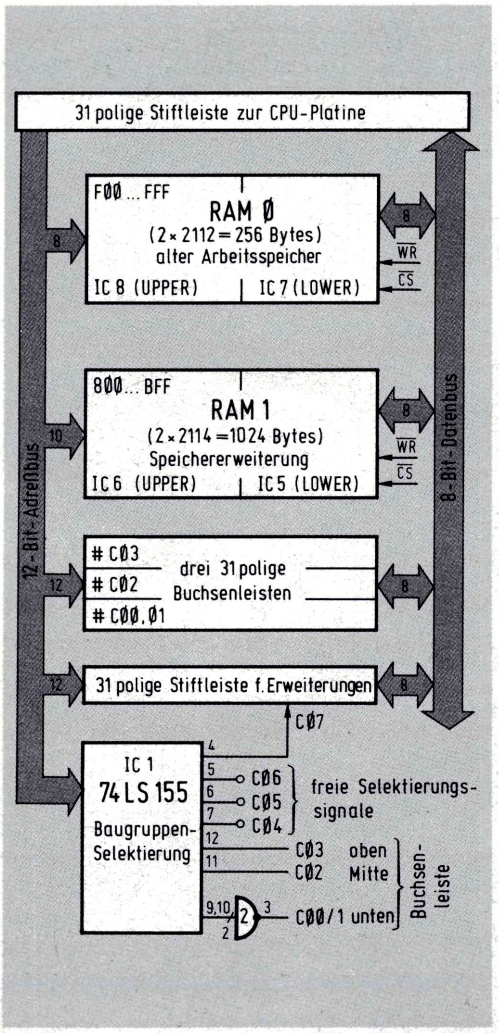


Bild 3: Das Blockschaltbild zeigt den Aufbau der Erweiterungskarte.

RAM-Zellen holt, egal, ob der Arbeitsspeicher nur 256 oder 1024 + 256 Bytes umfaßt? Das ist doch eigentlich ganz nett, denn wäre das nicht der Fall, wären beim 1-K-RAM irgendwo mittendrin ein paar Zellen für den Anwender tabu, und beim Programmieren müßte man umständlich darüber hinwegspringen. Um wieviel einfacher ist es zu wissen, daß man unabhängig von der Ausbaustufe immer ein paar Adressen vor dem Ende des Arbeitsspeichers aufhören muß, weil dann der Monitor-Bereich beginnt.

Damit Sie dieses Verhalten nicht für eine übernatürliche Fähigkeit des Monitors halten, sollten Sie sich die Zusammenhänge einmal klarmachen. Denjenigen Zellen, die das Monitor-Programm für sich beansprucht (um beispielsweise Informationen in den Siebensegmentcode umzusetzen und darzustellen), sind feste Adressen im Bereich XBDC...XBFF zugeordnet (die obersten 36 Worte). Greifen wir uns daraus als Beispiel einmal eine beliebige Adresse heraus (z. B. XB69) und sehen, was die auf ihrem Weg über den Adreßbus anrichtet (**Bild 4**).

Von den 16 Adreßbits A0...A15 des 8085 gehen beim UMS-85 nur die untersten 12 in die Adreßdecodierung mit ein; der Zustand von A12...A15 ist demzufolge beliebig, was vereinbarungsgemäß mit dem Pseudo-HEX-Digit „X" abgekürzt wird.

Die beiden Adreßbits A10 und A11 wählen vier 1-K-Speicherbaugruppen aus: Bei „00" oder „01" das EPROM (unteres bzw. oberes K), bei „10" das RAM und bei „11" externe

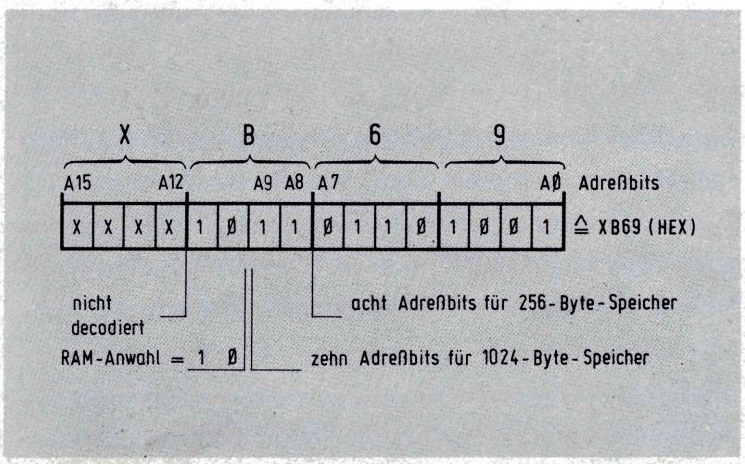


Bild 4: Zuordnung der einzelnen Adreßbits bei der UMS-RAM-Adressierung.

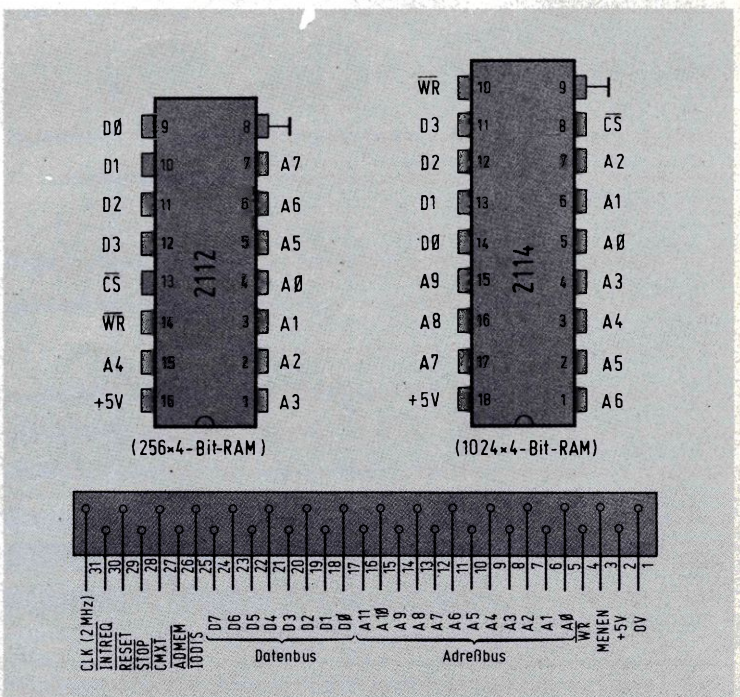


Bild 5: Anschlußbelegung der verwendeten RAMs und der Stift- bzw. Buchsenleiste.

Baugruppen (CSMXT). Bei Aktivierung des RAMs müssen bei dieser Struktur also A10=0 und A11=1 sein; bleiben die 10 restlichen Adreßbits übrig.
Um einen Speicher mit 1024 Worten zu adressieren, benötigt man genau diese 10 Adreßeingänge (vgl. Sonderheft „Dem Mikrocomputer auf's Bit geschaut"); hat der Speicher aber nur 256 Worte Umfang, besitzt er dementsprechend bloß acht Adreßeingänge, und A8 und A9 hängen bei der RAM-Adressierung „in der Luft", d. h. ihr Zustand ist beliebig. Daher kann der Monitor auch beim 0,25-K-RAM getrost die Zelle XB69 ansprechen; aus Bild 4 erkennen Sie, daß er damit zweifelsfrei X869 erreicht.
Wir halten fest, daß es dem Monitor egal ist, ob er mit dem kleinen oder dem ausgebauten RAM zusammenarbeitet; beim Betrieb mit der Erweiterungskarte müssen Sie nur die 2112-RAMs von der CPU-Platine herausnehmen und in die Sockel der Erweiterungskarte einsetzen.

Die Erweiterungskarte in der Totalen

Auf ein Detailschaltbild der Bus- und Speichererweiterungskarte wollen wir verzichten, weil das ziemlich unübersichtlich wäre. Wesentlich instruktiver ist das Blockschaltbild, das Sie zusammen mit den drei Einzelheiten aus **Bild 5** auch detailliert „lesen" können.

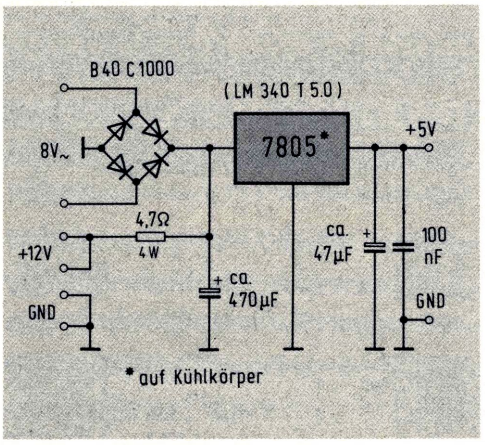


Bild 6: Auf der Erweiterungskarte ist ein eigener Festspannungsregler zur Stromversorgung vorgesehen.

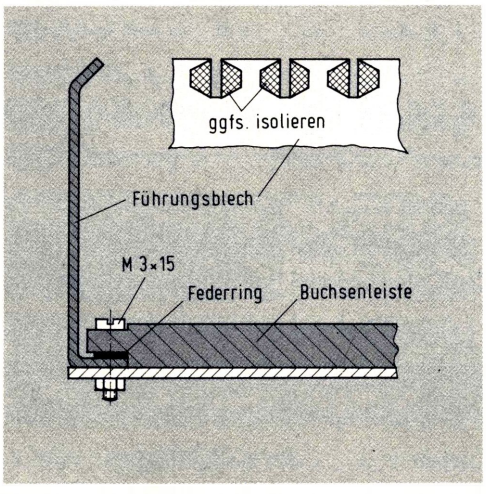


Bild 9: Wo Kurzschlußgefahr von Leiterbahnen besteht, muß das Führungsblech isoliert werden.

Wegen der Datenbreite von nur vier Bits pro Speicherstelle müssen die verwendeten RAMs in einem 8-Bit-System immer paarweise betrieben werden (untere = *lower* und obere = *upper* Hälfte).
Aus Bild 3 erkennen Sie ferner die Adreßbereiche, die sich aus der Hardware-Struktur ergeben. Die Aktivierung der für die drei Buchsenleisten vorgesehenen Karten erfolgt im Memory-Mapped-Verfahren (vgl. 4. Teil) über die Adressen XC00...XC07. Wenn Sie also beispielsweise bei eingesteckter Erweiterungskarte die Adresse 0C00 ausgeben, spricht diese die in der vordersten Buchsenleiste steckende Interface-Karte an. Und wenn Sie sich jetzt noch einmal den vierten Teil dieser Folge vornehmen, erkennen Sie, daß in diesen Steckplatz das Analog-Interface mit seinen zwei Portadressen paßt. Das System beginnt zu wachsen, aber groß ist es im Augenblick noch nicht!

Strom aus verschiedenen Quellen

Zur Stromversorgung der Erweiterungskarte gibt es drei verschiedene Möglichkeiten (**Bild 6**). Wenn Sie keine zusätzliche Interface-Karte verwenden, können Sie die +5 V direkt von der CPU-Platine abnehmen; die Masse-Verbindung erfolgt bereits über die Buchsenleiste (**Bild 7**). In diesem Fall bleibt der 5-V-Spannungsregler auf der Erweiterungskarte schlaff und ohne Funktion.

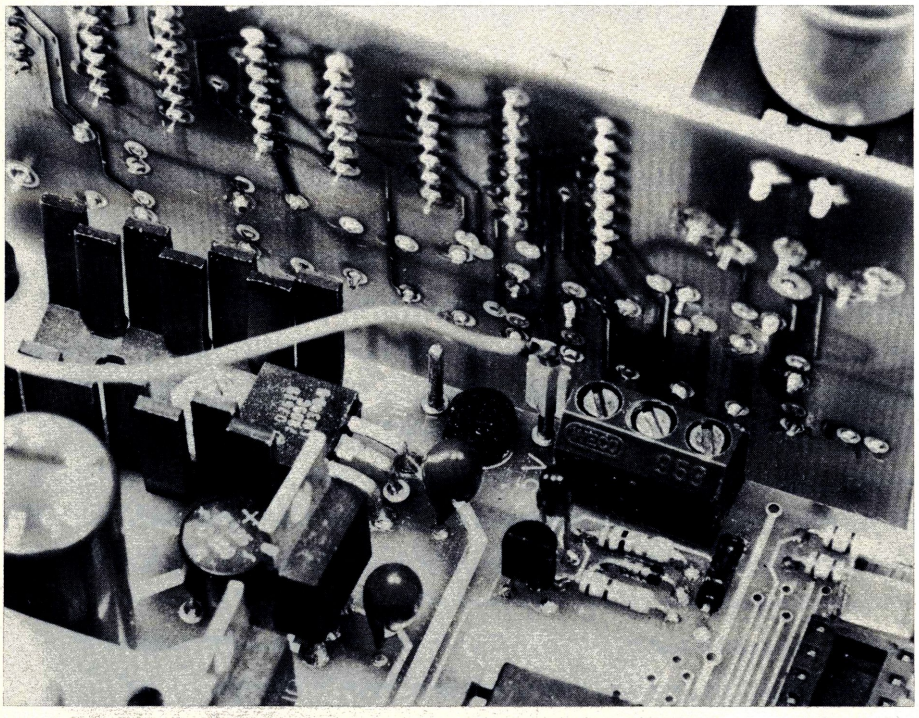


Bild 7: Über eine Steckverbindung kann man die +5 V von der CPU-Platine abnehmen.

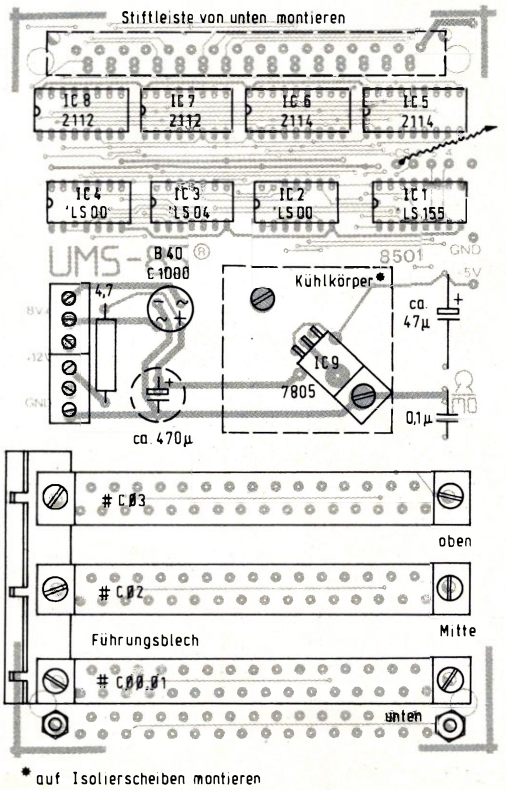


Bild 8: Bestückungsplan der Erweiterungskarte.

Betreiben Sie aber eine oder mehrere Interface-Karten auf der Bus-Erweiterung, schafft dies das CPU-Netzteil nicht mehr. In diesem Fall können Sie entweder eine Wechselspannung von ca. 8 V/1 A an die Klemmleiste links außen anschließen, oder Sie versorgen die Karte von einem externen 12-V-Netzteil, das Sie beim Betrieb des Druckers ohnehin benötigen. Im letztgenannten Fall muß der 4,7-Ω-Lastwiderstand eingelötet sein; er nimmt dem Stabi etwas von der entstehenden Verlustwärme ab. Wenn Sie hier einen Hochlastwiderstand verwenden (der die Wärme besser los

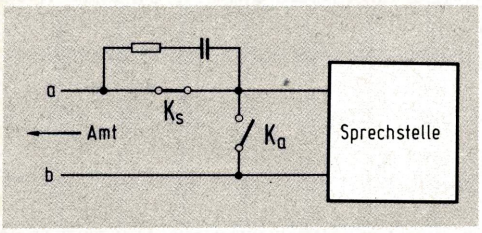


Bild 11: Zwei Kontakte, ein Öffner und ein Schließer, übernehmen das „Wählen" der gewünschten Nummer.

wird), können Sie den wegen seines Formats auch hängend von unten einlöten. Vor dem Bestücken lesen Sie bitte die folgenden Hinweise erst einmal durch; manche Fehler lassen sich nämlich nur schwer rückgängig machen. Beginnen Sie die Bestückung mit den vier Fassungen, den TTL-ICs und den Kondensatoren (**Bild 8**). Die Stiftleiste am oberen Platinenrand wird dann von unten eingesteckt und von oben verlötet. Beim Montieren des Kühlkörpers müssen zwischen diesem und der Platine zwei Isolierscheiben eingefügt werden. Um das Führungsblech für die senkrecht stehenden Interface-Platinen fest zu montieren, fügen Sie *vor dem Einlöten* zwischen Buchsenleiste und Blech einen Federring ein (**Bild 9**). Um einen Kontakt zwischen Leiterbahnen und Führungsblech zu verhindern, sind die Außenkanten der eingefrästen Schlitze gegebenenfalls mit Isolierband abzukleben. Zwei Schrauben M3 x 25 werden dann schließlich mit jeweils einem Mutternpaar plus Federring am vorderen Platinenrand als Standfüße montiert.
Für den Betrieb benötigt die Karte außer der 5-V-Versorgung (s.o.) noch das CS-Signal von der CPU-Platine. Löten Sie dazu einen Lötstift an die Beine 3 und 4 des IC10 auf der CPU-Platine (74LS00; vgl. Bild 5 im 1. Teil) und führen Sie dieses Signal ebenfalls über eine Steckverbindung zum Punkt „CS" auf der Erweiterungskarte (**Bild 10**).
Stecken Sie nun die 2112-RAMs um (von der CPU-Platine in die Sockel der Erweiterungskarte), bringen Sie die Bus-Erweiterung an ihren Platz, und schalten Sie die Stromversorgung ein. Rein äußerlich ist in der Anzeige keine Veränderung festzustellen, denn bei richtiger Verdrahtung (und keinem Lötklecks) erstrahlt wieder „0800 XX" auf der Anzeige-Platine; jetzt aber können Sie unter Adresse X900, XA00 und XB00 etwas anderes ablegen als unter X800. Und außer diesem 1-K-RAM-Bereich (abzüglich 36 Bytes für den Monitor) haben Sie noch ein Viertel-K im Bereich XF00...XFFF – welch' eine Wonne, sich darin auszutoben!

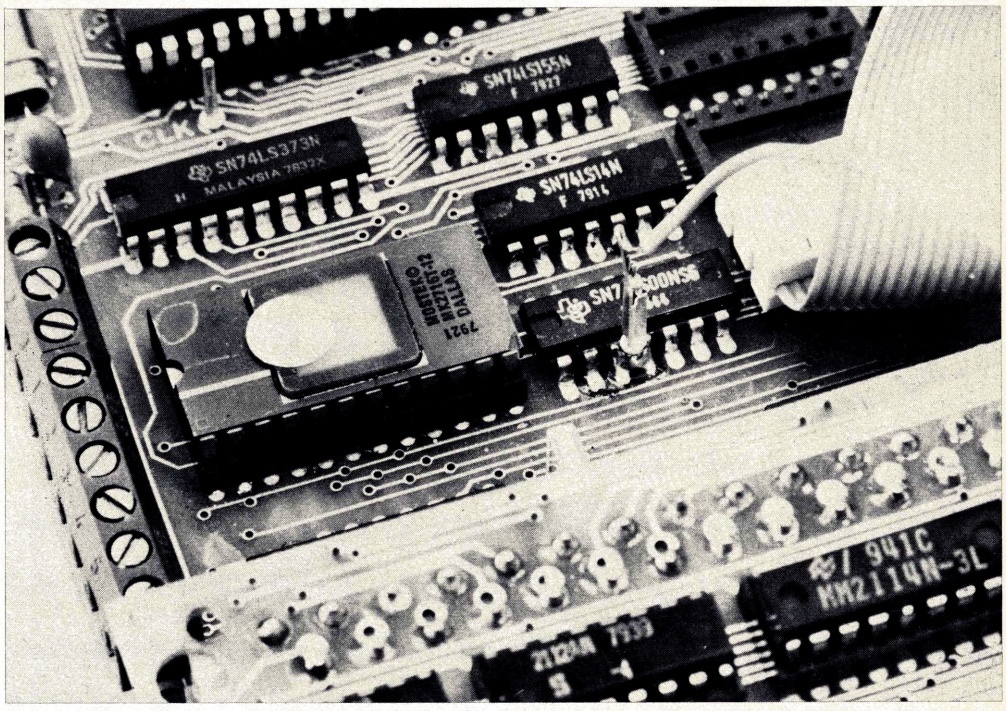


Bild 10: Eine separate Steckverbindung führt der Erweiterungskarte das CS-Signal zu.

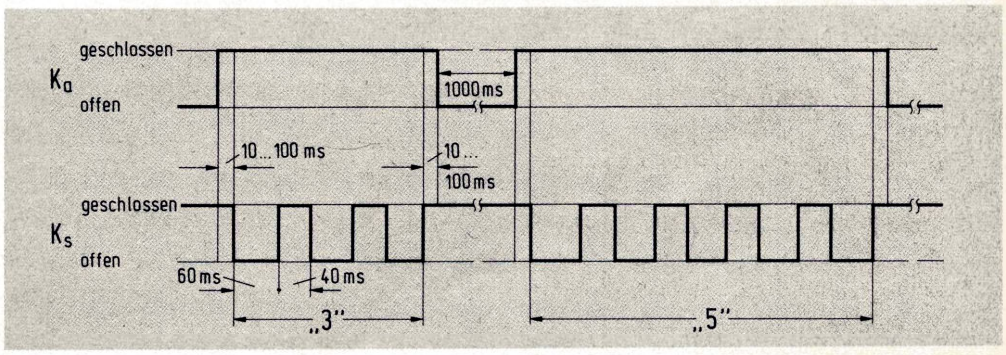


Bild 12: Ablaufdiagramm beim Wählen einer Telefon-Nummer.

Der Weg zum elektronischen Telefon

Sie verstehen die folgenden Ausführungen bitte richtig: Wir wollen Sie in keiner Weise dazu verleiten, in Ihrem Telefon herumzulöten. Hintergrund des folgenden Programmbeispiels ist die Erzeugung definierter Impulsfolgen, das Einlesen der benötigten Parameter von der Tastatur und die Verwaltung einer Datenbank. Daß dies am realen Beispiel Telefon passiert, nimmt dem Ganzen nur den sterilen Beigeschmack der Praxisferne.
Beginnen wir mit der Aufgabenstellung, die einen Blitzkurs Fernmeldetechnik einschließt (**Bild 11**): Beim Wählen einer Rufnummer sind zwei Kontakte K_a (Arbeitskontakt) und K_s (Stromstoßkontakt) aktiv; K_a schließt zu Beginn des Wählvorgangs den Eingang der Sprechstelle kurz, und K_s öffnet dann periodisch so oft, wie es die zu wählende Ziffer vorgibt. Danach öffnet K_a wieder, und Sie können bis zur nächsten Ziffer lauschen, ob vielleicht mittendrin schon besetzt ist.
Sehen Sie sich dazu einmal den entsprechenden Zeitablauf an (**Bild 12**), den wir mit zwei Relais (einem Öffner und einem Schließer) ohne weiteres realisieren können.

Das passende Programmbeispiel WAEHL (**Bild 13**) geht davon aus, daß in einem reservierten RAM-Bereich, dem sogenannten Nummern-Buffer, die zu wählenden Ziffern abgelegt sind; pro Ziffer (= BCD-Digit) ist ein Byte vorgesehen.
Das Programm muß für jede Null eine Zehn einsetzen (bei der Null erzeugt Ihr Telefon nämlich zehn Impulse), und außerdem muß es Leerzeichen erkennen, die die nicht mit der Telefon-Nummer belegten Buffer-Zellen auffüllen. Der Rest des Programms ist simpelste Ablaufsteuerung, die Sie auch ohne angeschlossene Relais verfolgen können: Jede Ziffer, die gerade „gewählt" wird,

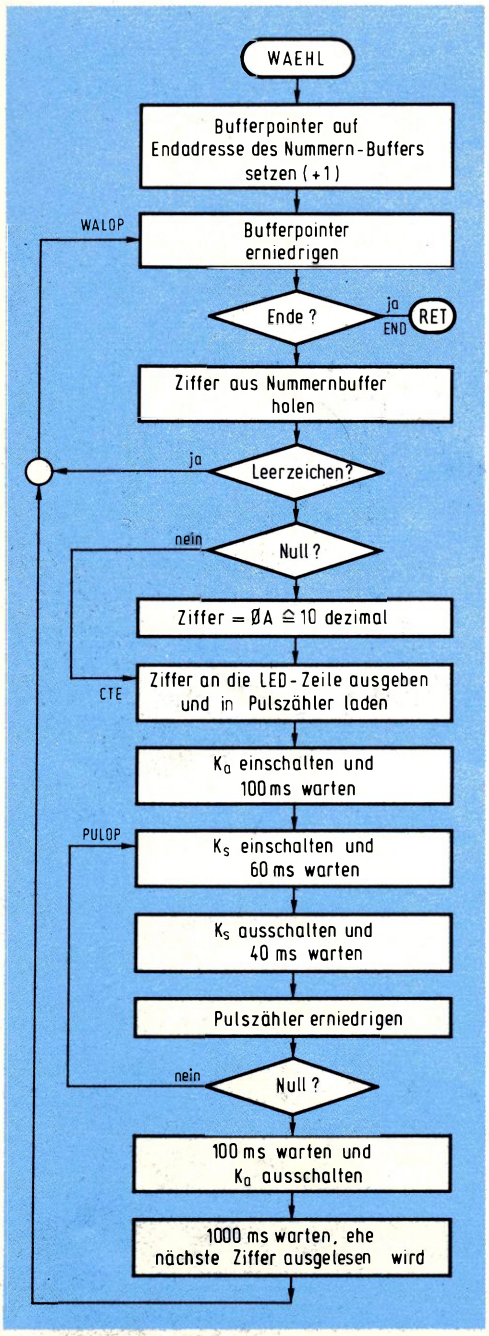
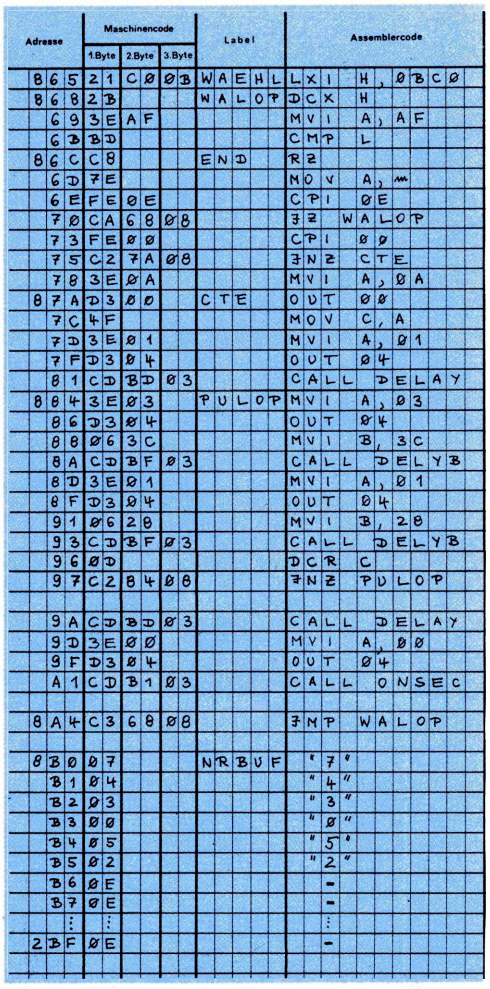


Bild 13: Flußdiagramm für ein Unterprogramm, das Telefon-Nummern „wählen" kann.



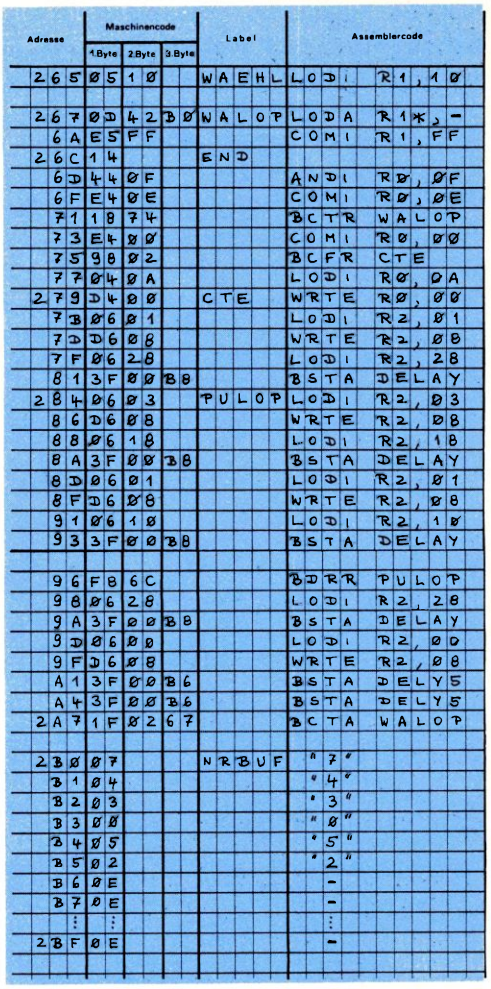


Bild 14: Das Maschinenprogramm zu Bild 12. Beispiel im Nummern-Buffer: Tel.-Nr. 25 03 47.

erscheint auch in der Anzeige (LED-Zeile beim UMS und Aufblinken im Digit 3 beim ELDO), was gleichzeitig Kontrollmöglichkeit und kurzweilige Unterhaltung bietet. Das zugehörige Maschinenprogramm zeigt **Bild 14,** eine geeignete Relais-Ansteuerung finden Sie in **Bild 15.**

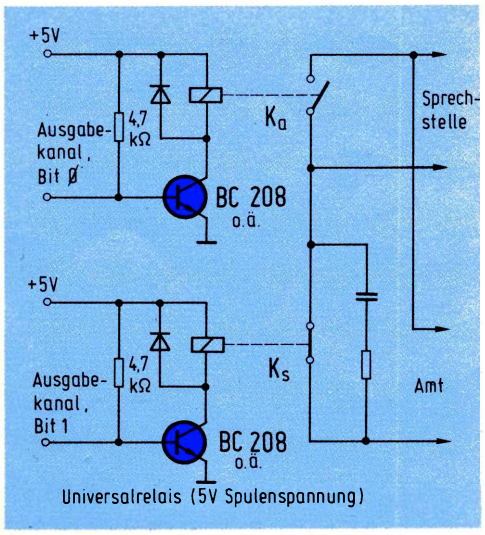


Bild 15: Die beiden Relais werden von je einem Bit des CPU-Ausgabe-Kanals angesteuert.

Laden Sie nun die gewünschte Telefon-Nummer nacheinander in den Nummern-Buffer (XBB0...XBBF beim UMS und X2B0...X2BF beim ELDO), wobei Sie unter jeder Adresse nur ein Digit ablegen (z. B. „05" für die Fünf). Die letzte Ziffer der Nummer muß in XBB0 (bzw. X2B0) stehen, und die bis zum Buffer-Ende XBBF (bzw. X2BF) nicht benötigten Zellen müssen mit dem Leerzeichen „0E" (für *Empty* = leer) aufgefüllt werden. Wenn Sie danach WAEHL als Unterprogramm aufrufen, klappern die angeschlossenen Relais munter vor sich hin, was Sie u. a. mit provisorisch angelöteten LEDs verfolgen können.

Beim UMS-85 dient der CPU-Ausgabe-Kanal zur Relais-Ansteuerung, und beim ELDO können Sie sich – soweit Sie nicht die große Interface-Karte besitzen – mit einer Schaltung nach Bild 14 aus dem dritten Teil dieser Reihe behelfen. Wie Sie diese Grundkonfiguration zum richtigen Tastentelefon einschließlich Festnummern und Löschtaste ausbauen können, erfahren Sie am Ende des folgenden Beitrags.

Bits vom laufenden Band

Im Rahmen unserer Mikrocomputer-Anwendungsbeispiele haben Sie schon eine ganze Reihe von Einsatzmöglichkeiten kennengelernt; das reicht inzwischen von der Musikbox über das programmierbare Schaltwerk bis hin zum Digitalvoltmeter, und im letzten Beitrag haben wir Ihnen sogar gezeigt, wie Sie Ihrem Mops die Grundfunktion des Telefonierens beibringen können. Ehe wir dieses tolle Telefon komplettieren, wollen wir Ihnen das Cassetten-Interface für das UMS-85 vorstellen (für den ELDO haben wir eine solche Schaltung bereits veröffentlicht; vgl. „Dem Mikrocomputer auf's Bit geschaut", Abschnitt 4.17). Es ermöglicht Ihnen, Programme und Daten auf normalen Magnetbandcassetten abzulegen und von dort wieder einzulesen, um bestehende Programme nicht jedesmal neu eintippen zu müssen (**Bild 1**). Auch diese Baugruppe ist als Bausatz bzw. fertig montiert direkt vom Autor beziehbar.

Die serielle Datenübertragung

Die hier betrachteten Mikroprozessoren 8085 bzw. 2650 haben eine feste Wortlänge von 8 Bit (= 1 Byte), d. h. sämtliche Befehle oder Daten besitzen immer dieses 8-Bit-Format. Wenn das nicht ausreicht, werden mehrere 8-Bit-Worte aneinandergereiht, um beispielsweise in einem Befehl auch noch eine Adresse unterzubringen.
Um eine Folge solcher Bytes, die z. B. ein Programm bilden, extern zu speichern, bräuchte man eigentlich ein achtkanaliges Speichermedium; denn auch die CPU-internen Speicher, wie etwa RAM oder PROM, besitzen ja diese 8-Bit-Struktur zur parallelen Aufnahme ganzer Worte.
Daher bieten sich Magnetbandmaschinen mit acht Spuren förmlich für diese Aufgabe des externen Massenspeichers an. Sie haben nur den (nicht zu vernachlässigenden) Nachteil, abartig teuer zu sein. Was halten Sie deshalb davon, Ihr preiswertes Standard-Cassetten-Gerät für dieselbe Aufgabe einzusetzen, ohne dazu die geringste Modifikation vornehmen zu müssen? Die Daten dürfen dann nur nicht paketweise im 8-Bit-Format ankommen (parallel), sondern gesittet eins nach dem anderen (seriell). Die Umsetzung vom parallelen ins serielle Format und umgekehrt übertragen wir dem Mikrocomputer; daß dies etwas Zeit in Anspruch nimmt (einige Sekunden), ertragen Sie sicherlich gern, zumal Sie dadurch etliche tausend Mark für die Acht-Kanal-Bandmaschine sparen!
Dieser Zeitaufwand ist natürlich auch der Grund, warum man überhaupt Halbleiterspeicher verwendet und nicht die serielle Speicherung auf Band vorzieht: Auf ein Byte im RAM oder ROM können Sie in Bruchteilen einer Mikrosekunde zugreifen, während der bloße Datentransport eines vom Band eingelesenen Bytes rund die hunderttausendfache Zeit in Anspruch nimmt, das Aufsuchen der richtigen Stelle auf dem Band noch nicht gerechnet....

Wohlklingende Bits

Da ein Tonbandgerät keine 0- oder 1-Gleichspannungspegel aufzeichnen kann, macht das Cassetten-Interface aus den HIGH- und LOW-Zuständen Tonfrequenzsignale. Einem HIGH-Bit wird dabei eine Frequenz von ca. 3 kHz zugeordnet und einem LOW-Bit ein halb so hoher Ton von ca. 1,5 kHz. Um bei der Aufzeichnung und

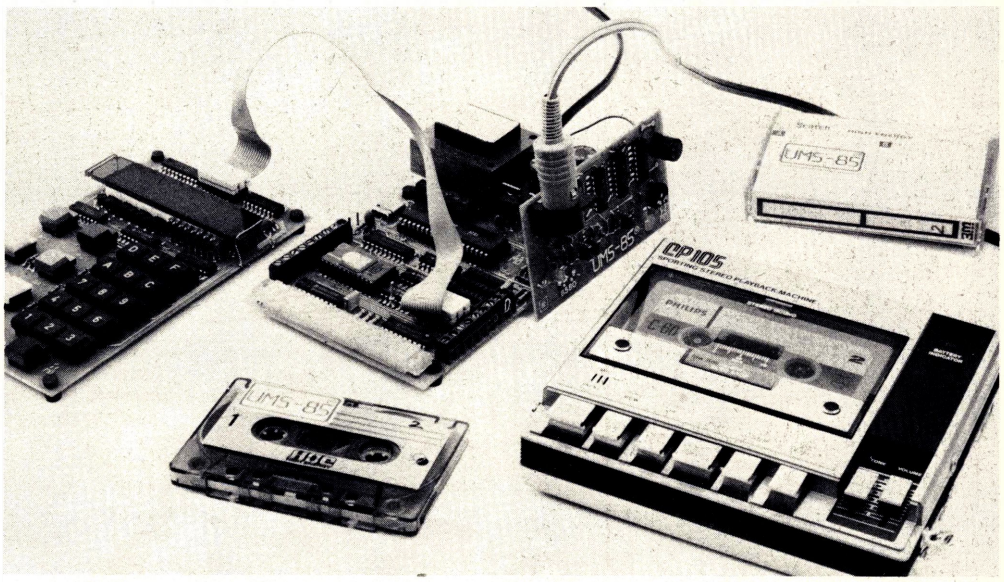


Bild 1: Mit dem Cassetten-Interface können handelsübliche Recorder als externer Massenspeicher benutzt werden.

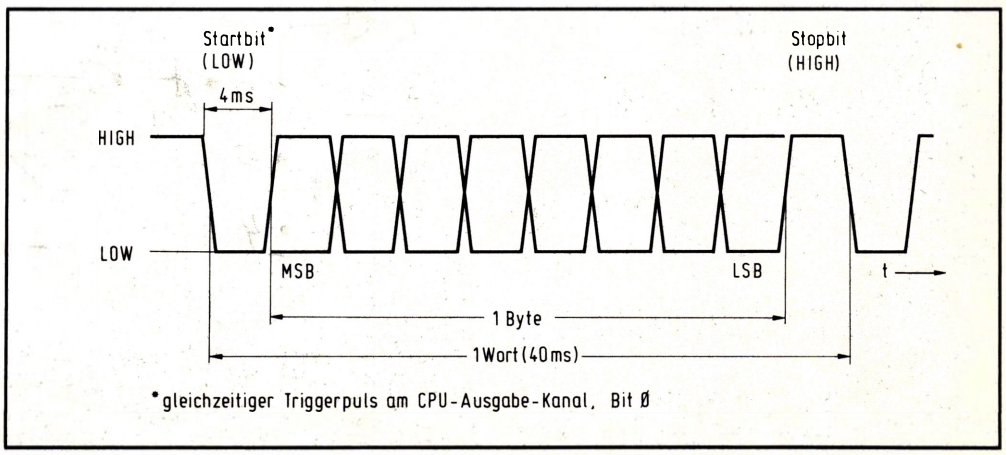


Bild 2. Bei der seriellen Datenübertragung werden pro Byte ein Start- und Stopbit ergänzt, was beim Wiedereinlesen die Synchronisation ermöglicht.

Wiedergabe definierte Anfangsbedingungen herzustellen, wird jedem Byte bei der Übertragung ein Startbit (LOW) voran- und ein Stopbit (HIGH) nachgestellt (**Bild 2**). Um ein störsicheres Verfahren zu erreichen, wollen wir für jedes Bit eine Zeitdauer von 4 ms spendieren, pro Byte also 40 ms einschließlich Start- und Stopbit.

Bild 3 zeigt die Schaltung des Cassetten-Interfaces, das aus dem Aufnahme- (obere Hälfte) und Wiedergabeteil (untere Hälfte) besteht. Der umschaltbare Oszillator für die beiden Tonfrequenzen wird vom seriellen CPU-Ausgang SEROT angesteuert. Ist SOD=SEROT auf HIGH, liefert die Schaltung ca. 3 kHz an die Tonbandbuchse, und bei LOW an SOD gelangt die halbe Frequenz dorthin. Das RC-Glied 2,7 kΩ/100 nF verschleift die steilen Rechteckflanken dieser Aufnahmefrequenzen, um wegen des hohen Eingangswiderstandes bei der Wiedergabe ein störendes Übersprechen zu verhindern.

Der Wiedergabezweig besteht aus einem Eingangsverstärker (rückgekoppeltes CMOS-Gatter) mit nachfolgender Impulsformung, die aus einem 3-kHz-Signal ausgangsseitig HIGH-Pegel und aus einem 1,5-kHz-Eingangssignal LOW-Pegel macht; das auf diese Weise entstehende TTL-Signal gelangt in den seriellen CPU-Eingang SERIN. In seinem Zeitverlauf entspricht es der bei der Ausgabe an SEROT erscheinenden Impulsfolge.

Mit dem Bestückungsplan (**Bild 4**) ist der Aufbau ein Kinderspiel. Die Verbindung zur Dreierklemme auf der CPU-Karte erfolgt über drei Lötstifte (SERIN – Masse – SEROT), die *von hinten* in die Interface-Karte eingelötet werden (**Bild 5**). Die +5-V-Versorgung holen Sie über eine Steckhülse vom entsprechenden Lötstift „+5 V" am „Nordende" der Dreierklemme.

Wenn Sie so weit sind, können Sie Ihr Interface abgleichen. Dazu überbrücken Sie an der Tonbandbuchse den Ein- und Ausgang (äußere Anschlüsse), drehen das 1-MΩ-Poti vom Rechtsanschlag langsam nach links und markieren die Stellung, bei der die Leuchtdiode *auf der Interface-Karte* gerade zu leuchten beginnt. Danach sorgen Sie für HIGH-Pegel an SEROT, indem Sie die Sequenz nach **Bild 7** eingeben und nach RESET und RUN starten (vgl. auch Teil 3 dieser Reihe). Nun drehen Sie das Poti wieder vom Rechtsanschlag nach links und markieren diejenige Stellung, bei der die Interface-LED anfängt zu leuchten (die SEROT-LED leuchtet wegen des dort hergestellten HIGH-Pegels jetzt dauernd). Bringen Sie nun das Poti in die Mitte zwischen beiden Markierungen, und das Interface ist betriebsbereit abgeglichen.

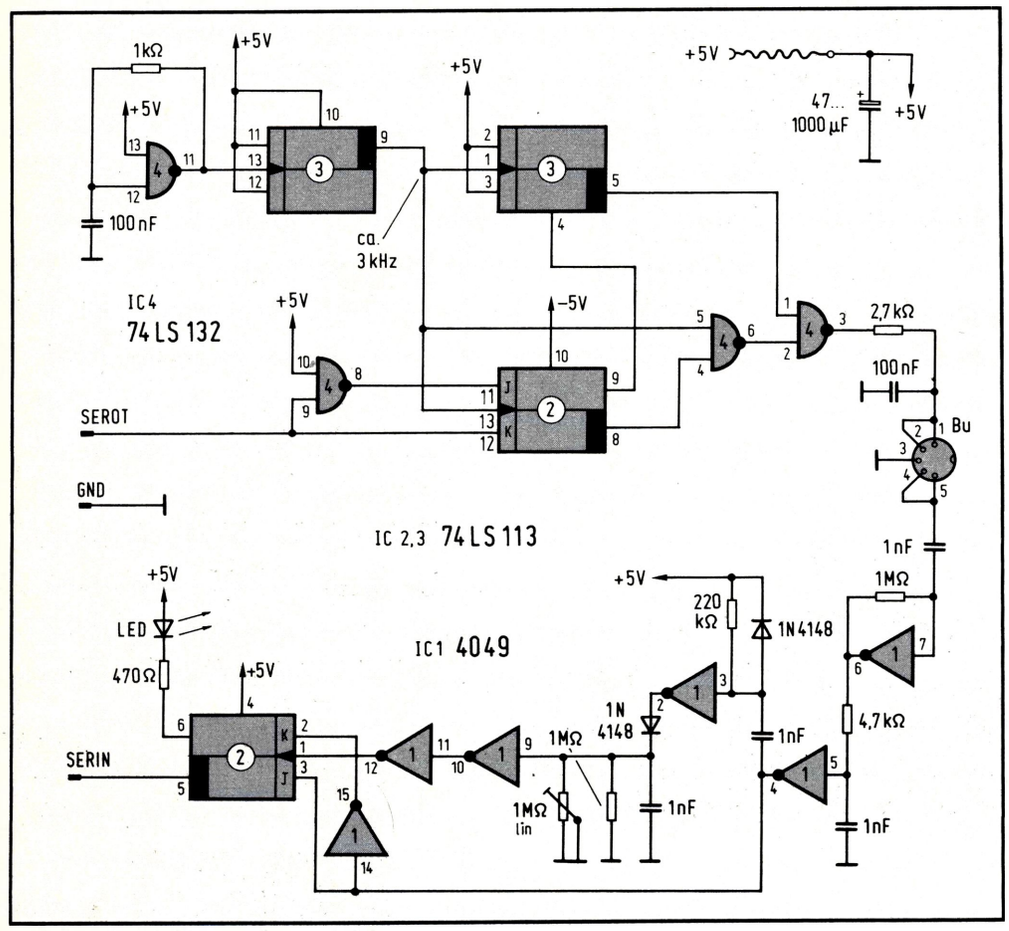


Bild 3: Schaltung des Cassetten-Interfaces.

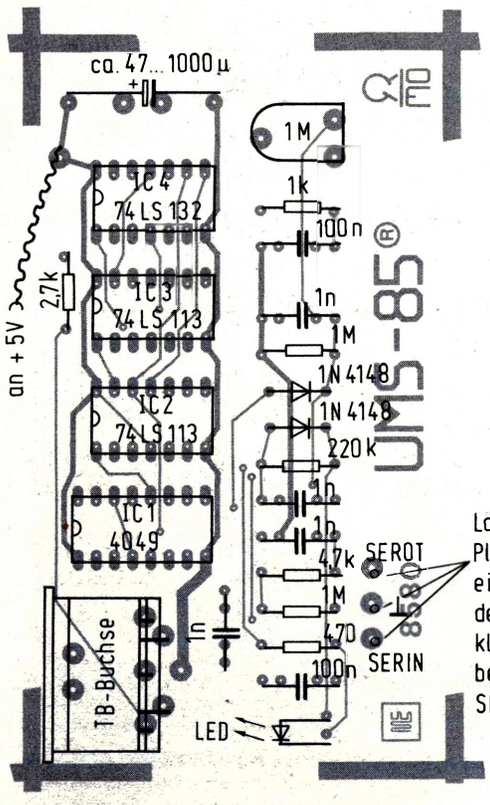
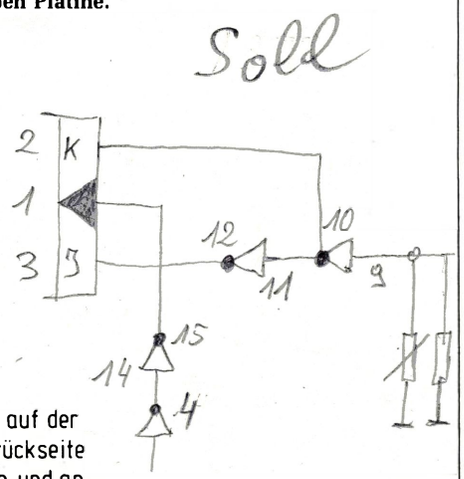


Bild 4: Bestückungsplan der nur 100 × 55 mm² großen Platine.



Lötstifte auf der Platinenrückseite einsetzen und an den drei Schraubklemmen der CPU befestigen (SERIN/SEROT)

Der große Bruder des Monitors

Zwei Unterprogramme CASOT und CASIN (sogenannte Software-Päckchen) übernehmen die Datenaufbereitung und Ablaufsteuerung beim Auslesen auf bzw. Einlesen vom Band. Beide Programme sind Bestandteil des erweiterten 2-K-Monitors, mit dem Sie Ihr UMS-85 ausbauen können (als Zube-

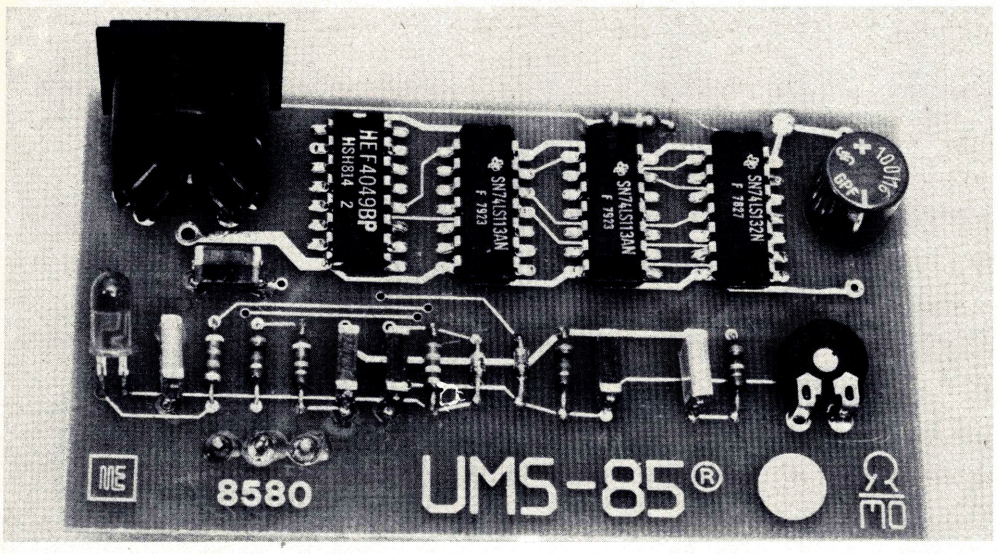


Bild 5: Ansicht des fertig bestückten Cassetten-Interfaces.

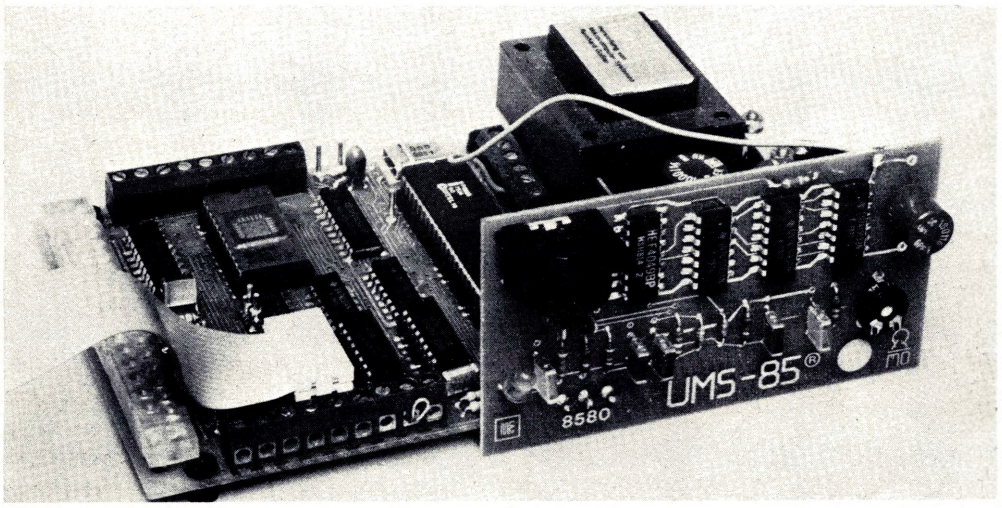


Bild 6: Die Interface-Platine wird mit drei Lötstiften in der Anschlußklemme der CPU-Karte festgeschraubt; die +5-V-Versorgung kommt über ein separates Kabel.

hör erhältlich); er wird in einem 2716-EPROM geliefert (2 K x 8 Bit Umfang) und anstelle des 1-K-Monitors der Grundausbaustufe in die 24polige Fassung eingesteckt.

Achtung! Beim Betrieb des 2716-EPROMS müssen die beiden Brücken oberhalb des Quarzes waagerecht eingelötet werden (Stellung A). Die andere Konfiguration (Stellung B für das 2708-EPROM) bringt einem 2716 den sicheren Tod!

Außer der Cassetten Verwaltung enthält das 2716-EPROM noch eine Reihe anderer Funktionen und Programmbeispiele, die bereits im Rahmen dieser Beitragsreihe vorgestellt wurden (Schaltwerk, Digitalvoltmeter) bzw. noch vorgestellt werden (z.B. Ansteuerung der sprechenden Uhr); wir kommen darauf noch im einzelnen zurück.

Ausgabe auf Magnetband

Die erste (Start-) und letzte (Stop-) Adresse des zu übertragenden Speicherbereichs laden Sie in die RAM-Paare OBFB/C und OBFD/E. Wenn 0800 beispielsweise die Start- und 08A7 die Stopadresse sind, wählen Sie OBFB an (Tasten „ADR – O – B – F – B – DAT" drücken) und geben dann nacheinander ein „O – 8 – NXT – O – O – NXT" (Startadresse), gefolgt von „O – 8 – NXT – A – 7 – NXT" (Stopadresse, das letzte NXT nicht vergessen!).
Bei angeschlossenem Tonbandgerät (in Ruhestellung) starten Sie nun CASOT einfach, indem Sie eingeben „FCT – O – NXT" (Funktionstaste O = Cassetten-Ausgabe). Unmittelbar danach müssen Sie Ihr Bandgerät in Stellung „Aufnahme" starten, denn 10 s nach dem Druck auf NXT beginnt die eigentliche Datenübertragung. Während dieser Zeit bleibt die Anzeige dunkel, und

```
0800   3E   MVI A, CO
0801   CO
0802   30   SIM
0803   76   HLT
0804   XX
```

Bild 7: Sequenz zum Umschalten des SEROT-Ausgangs auf HIGH.

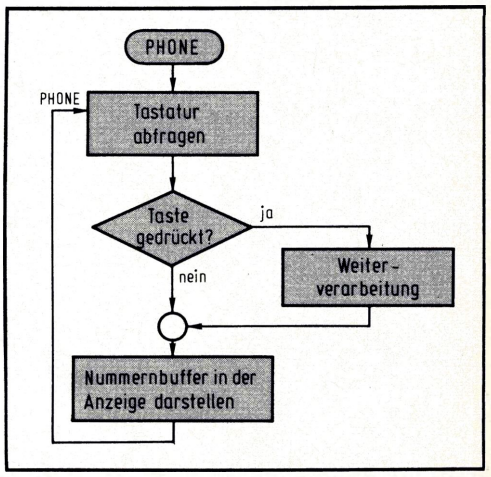


Bild 8: Das Hauptprogramm PHONE besteht aus einer Endlosschleife; der Sprung ins Unterprogramm erfolgt nur, wenn zuvor eine Taste gedrückt wird.

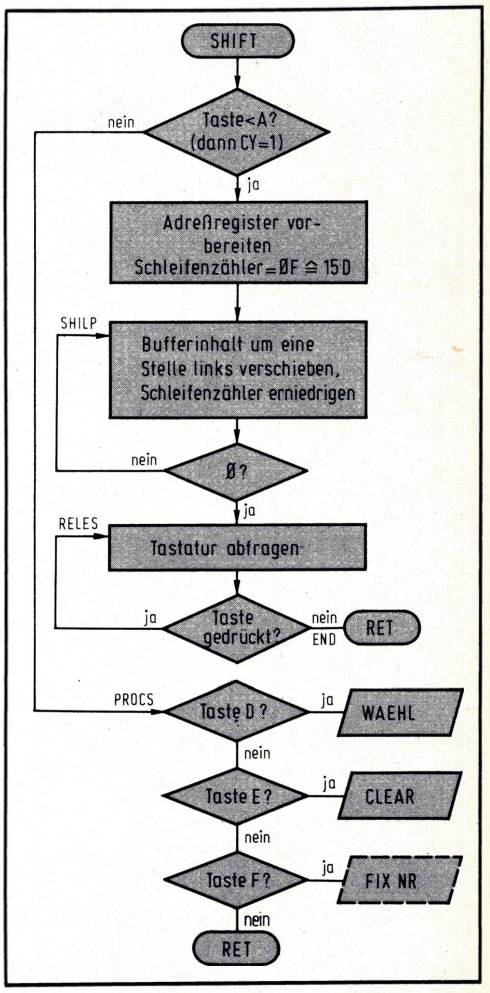


Bild 9: Bei Zifferntasten 0...9 verschiebt das Unterprogramm SHIFT den Nummernbuffer zyklisch nach links, und bei Befehlstasten verzweigt es zu den Programmteilen WAEHL bzw. CLEAR.

nur die SEROT-LED flackert im Rhythmus der ausgehenden Bits. Nach erfolgter Übertragung meldet sich das Monitor-Programm wieder mit der Anzeige „O800 XX".
Ohne daß Sie etwas davon bemerkt haben, hat CASOT eine Prüfsumme der gesamten übertragenen Daten erzeugt und mit aufgezeichnet. Nach dem Einlesen überprüft CASIN anhand dieser Information, ob die Übertragung fehlerfrei geklappt hat.

Einlesen vom Magnetband

Wie bei der Ausgabe beschrieben, geben Sie wiederum Start- und Stopadresse ein. Diese müssen nicht unbedingt dieselben sein wie bei der Ausgabe, wenn Sie beispielsweise Datenblöcke verschieden wollen. Die Fehlererkennung arbeitet allerdings nur dann, wenn die Blocklänge unabhängig von den Absolutadressen beim Ausgeben und Einlesen gleich ist.
Starten Sie Ihr Tonbandgerät in Stellung „Wiedergabe" innerhalb des 10-s-Vorspannbereichs (hoher Dauerton), der dem gewünschten Programm vorangeht (die Interface-LED leuchtet dabei kontinuierlich). Unmittelbar danach müssen Sie CASIN aufrufen (innerhalb des 10-s-Dauertons), wozu Sie nur folgende Sequenz ausführen: „FCT – 1 – NXT" (Funktionstaste 1 = Cassetten-Eingabe).
Bei Erreichen der Daten auf dem Band beginnt die Leuchtdiode im Rhythmus der ankommenden Bits zu flackern, und die daraus zusammengesetzten Bytes erscheinen zur Kontrolle in der LED-Zeile der Anzeige-Platine.
Wenn nach beendetem Einlesen (Erreichen der Stopadresse) die ermittelte Prüfsumme mit der beim Ausgeben aufgezeichneten übereinstimmt, meldet sich der Monitor automatisch wieder (Anzeige „O800 XX"); dies ist gleichzeitig die Gewähr für fehlerfreie Eingabe.
Bei Abweichungen erscheint die Fehlmeldung „Err" (Error) in der Anzeige. In diesem Fall stimmen ausgegebene und eingelesene Daten nicht überein, und Sie sollten die Prozedur des Einlesens wiederholen. Bitte beachten Sie, daß die Fehlermeldung auch dann erfolgt, wenn nur die Blocklänge bei Aus- und Eingabe nicht übereinstimmt, obwohl die eingelesenen Daten an sich fehlerfrei sind.
Sie können sich nun eine ganze Programmbibliothek auf Band zusammenstellen, über die Sie sorgfältig Buch führen sollten (Zählwerkstand notieren und/oder Kommentar auf Band sprechen). Bei Bedarf können Sie dann blitzschnell ins Regal greifen, um Ihren Computer außer „Wilhelm Tell" auch noch „Ännchen von Tharau" oder die Arie über „Colette von der Märkischen Heide" spielen lassen.

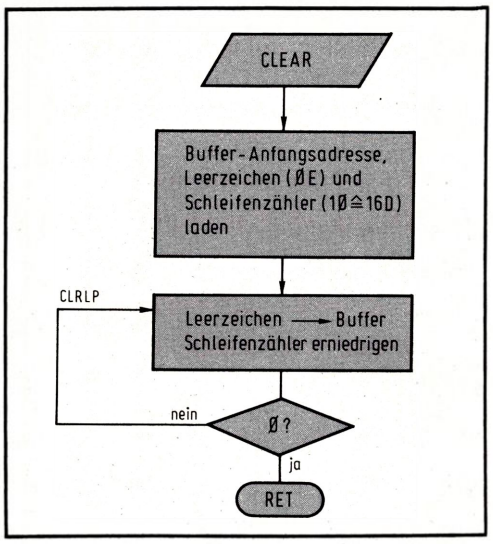


Bild 10: Programmteil CLEAR löscht den Nummernbuffer (Auffüllen mit 16 Leerzeichen 0E).

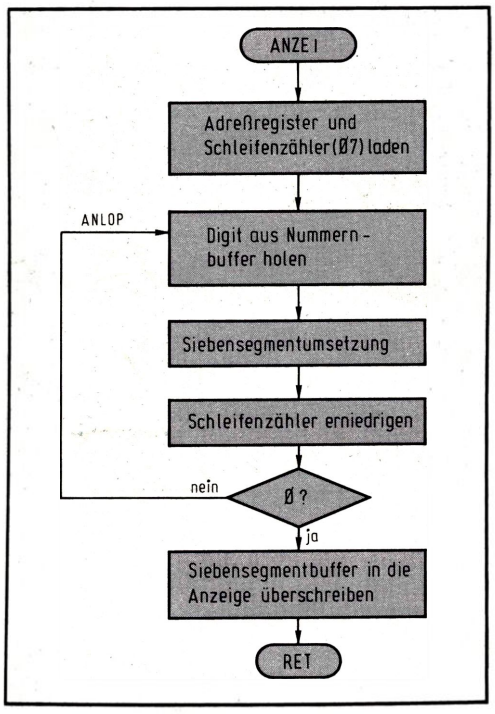


Bild 11: Vor dem Überschreiben des Siebensegmentbuffers in die Anzeige muß die Umsetzung in den Siebensegmentcode erfolgen.

Auf los geht's los

Spätestens jetzt erkennen Sie auch den Sinn der hinzugefügten Start- und Stopbits (vgl. Bild 2): Die auswertende Schaltung „weiß" beim Empfang (Einlesen vom Band), daß sie ihre Arbeit bei HIGH-Pegel beginnt (10-s-Vorspannbereich) und auf die erste negative Flanke (Startbit) warten muß. Genau 6 ms nach dieser Flanke (Mitte des ersten Datenbits) erfolgt die Abfrage des Eingangspegels, was danach alle 4 ms noch sieben Mal geschieht, um die Zustände aller Bits einzulesen und zu einem Byte zusammenzusetzen.
Nach Abfrage des LSB-Zustandes (34 ms nach der negativen Flanke) wartet die Auswerteschaltung weitere 4 ms (mindestens 2 ms), denn dann ist mit Sicherheit der HIGH-Pegel des Stopbits erreicht. Und danach geht das Spiel von vorn los, denn nun erfolgt wiederum das Warten auf die nächste negative Flanke (Startbit des folgenden Wortes). Durch diese ständig durchgeführte Synchronisation können pro Wort Zeitfehler (z.B. durch Gleichlaufschwankungen des Bandgeräts) von maximal ±½ Bitdauer verkraftet werden, was einer Abweichung von ±5% entspricht. Und weil so schlecht kein Tonbandgerät ist, arbeitet das Verfahren extrem störsicher.
Sie können den Bitstrom bei Ein- und Ausgabe übrigens sehr gut mit dem Oszillografen verfolgen, wenn Sie SEROT (bei der Ausgabe) bzw. SERIN (bei der Eingabe) darstellen und den Impuls an Bit 0 des CPU-Ausgabe-Kanals zum Triggern benutzen (Zeitablenkung: 4 ms pro Teilung). Den zehn Kästchen des Schirms können Sie gemäß Bild 2 die einzelnen Bits zuordnen, wobei Start- und Stopbit ihren Zustand definitionsgemäß nicht ändern.

Telefon total

In dem Bewußtsein, alle folgenden (und früheren) Programme jeweils nur noch ein einziges Mal eintippen zu müssen, weil Sie sie nun per Cassetten-Interface ablegen und wieder einlesen können, gehen Sie mit frischem Mut an Ihr halbfertiges Tastentelefon. Der im vorigen Abschnitt dieser Reihe erreichte Stand war der, daß das Unterprogramm WAEHL eine Ziffernfolge aus dem Nummernbuffer ausliest und daraus die entsprechenden Impulse formt, um diese Ziffern als Telefonnummer zu verarbeiten, die dann automatisch gewählt wird.
Was hierbei stört, ist das plumpe und umständliche Eingeben der einzelnen Digits mit Hilfe des Monitors; außerdem fehlt die Darstellung der eingetasteten Ziffern in der Anzeige, womit wir unser Anwendungsbeispiel nun komplettieren wollen. Das Hauptprogramm PHONE **(Bild 8)** durchläuft daher in einer Endlosschleife die beiden Aktivitäten „Tastatur abfragen und Eingaben verarbeiten" (Unterprogramme HHKEY aus dem Monitor sowie PROCS, s.u.) und „Inhalt des Nummernbuffers anzeigen" (ANZEI, s.u.).

HHKEY fragt die HEX-Tastatur des UMS-85 ab; bei gedrückter Taste ist beim Rücksprung das CY-Bit auf HIGH, und die zur jeweiligen Taste gehörende Information 0...F steht in den Registern A und B; das Unterprogramm SHIFT wird also nur aufgerufen, wenn CY = 1 ist (bedingter CALL-Befehl). SHIFT übernimmt zwei Aufgaben (**Bild 9**): Ist eine Taste 0...9 gedrückt (Abfrage: kleiner A?), wird die Ziffer in den Nummernbuffer nachgeschoben. Den Tasten D...F sind Sonderfunktionen zugeordnet (Verzweigung imTeil PROCS), während A...C hier ohne Bedeutung bleiben.

Bei **D** (wie **D**urchwahl) springt das Programm zu WAEHL; dies ist also die Starttaste, um das Wählen der zuvor eingetasteten Rufnummer zu veranlassen. Dieses Unterprogramm haben wir im letzten Beitrag vorgestellt.

Beim Betätigen von *E* (wie *Erase* = löschen) wird der Nummernbuffer mit dem Leerzeichen „E" (wie *Empty* = leer) vollgeschrieben, was vereinbarungsgemäß dem Löschen gleichkommt. Das Programm verzweigt in diesem Fall zu CLEAR (**Bild 10**; kein neuer Unterprogramm-Aufruf, sondern Programmsprung!).

Und der Taste F können Sie beispielsweise die Funktion „Festnummer" zuordnen, bei der Sie eine Zahl von 00...99 eintippen, unter der Sie in einem separaten Speicherbereich hundert feste Telefonnummern von guten Freundinnen, Nachbarn oder Kunden abgelegt haben, die Sie dann nicht mehr umständlich suchen und wählen müssen, sondern sie per Kennziffer abrufen können. Diese Zusatzfunktion würde allerdings den hier gesteckten Rahmen übersteigen, so daß wir hier auf eine detaillierte Vorstellung verzichten wollen und Sie es als Ansporn werten sollten, diesen Teil einmal selbst zu realisieren.

Das Unterprogramm ANZEI setzt die HEX-Digits des Nummernbuffers unter Zuhilfenahme des Monitor-Programms SSGCV in den Siebensegmentcode um und bringt diesen mittels DISPL zur Anzeige (**Bild 11**). Die vollständigen Listings aller Programmteile finden Sie in **Bild 12**, und nach deren Eingabe ins RAM können Sie Ihre Mikrocomputer-Hardware bereits sehr vielseitig einsetzen: Über die Tastatur geben Sie Daten (0...9) oder Befehle ein (Wählen, Löschen, Festnummer abrufen), und in der Anzeige lassen Sie sich die eingetasteten Informationen darstellen. Auf Knopfdruck schließlich veranlassen Sie die Weiterverarbeitung der eingegebenen Daten, was hier am Beispiel des Telefons (Aktivierung zweier Relais) veranschaulicht wurde.

Bild 13 zeigt ein solchermaßen „aufgebohrtes" UMS-85, das in der Lage ist, bis zu 150 feste Telefonnummern mit maximal 16 Stellen zu speichern und abzurufen. Es handelt sich hierbei um eine Studie, die die Vielseitigkeit dieses kleinen Systems zeigen soll, und für die daher auch keine postalische Zulassung besteht.

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
8000	CD	61	00	PHONE	CALL HHKEY	1
03	DC	10	08		CC SHIFT	2
06	CD	40	08		CALL ANZEI	3
09	C3	00	00		JMP PHONE	4
						5
810	FE	0A		SHIFT	CPI 0A	6
12	D2	2D	08		JNC PROCS	7
15	21	BF	0B		LXI H,0BBF	8
18	11	BE	0B		LXI D,0BBE	9
1B	0E	0F			MVI C,0F	10
81D	1A			SHILP	LDAX D	11
1E	77				MOV M,A	12
1F	2B				DCX H	13
20	1B				DCX D	14
21	0D				DCR C	15
22	C2	1D	08		JNZ SHILP	16
25	70				MOV M,B	17
8226	CD	61	00	RELES	CALL HHKEY	18
29	D0			END	RNC	19
2A	C3	26	08		JMP RELES	20
82D	FE	0D		PROCS	CPI 0D	21
2F	CA	65	08		JZ WAEHL	22
32	FE	0E			CPI 0E	23
34	CA	58	08		JZ CLEAR	24
37	FE	0F			CPI 0F	25
39	00	00	00		NOPs	1
3C	C9				RET	2
						3
8400	21	B0	0B	ANZEI	LXI H,0BB0	4
43	11	F1	0B		LXI D,0BF1	5
46	0E	07			MVI C,07	6
8486	7E			ANLOP	MOV A,M	7
49	E5				PUSH H	8
4A	CD	E2	00		CALL SSGCV	9
4D	E1				POP H	10
4E	23				INX H	11
4F	0D				DCR C	12
50	C2	48	08		JNZ ANLOP	13
53	CD	1F	01		CALL DISPL	14
56	C9				RET	15
						16
8558	21	B0	0B	CLEAR	LXI H,0BB0	17
5B	01	10	0E		LXI B,0E10	18
85E	70			CLRLP	MOV M,B	19
5F	23				INX H	20
60	0D				DCR C	21
61	C2	5E	08		JNZ CLRLP	22
64	C9				RET	23
						24
						25

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
2000	55	07		PHONE	READE R1,07	1
02	3E	02	0C		BSTA SHIFT	2
05	3F	02	35		BSTA ANZEI	3
08	1F	02	00		BCTA PHONE	4
						5
200C	45	0F		SHIFT	ANDI R1,0F	6
0E	E5	0A			COMI R1,0A	7
109	A1	2			BCFR PROCS	8
120	7	0F			LODI R3,0F	9
2140	F6	2	AF	SHILP	LODA R3*	10
17	CF	62	B0		STRA R3*	11
1A	FB	78			BDRR SHILP	12
1C	CD	02	B0		STRA R1	13
21F	54	07		RELES	READE R0,07	14
211	A	7C			BCTR RELES	15
213	17			END	RETC	16
224	E5	0D		PROCS	COMI R1,0D	17
26	1C	02	65		BCTA WAEHL	18
29	E5	0E			COMI R1,0E	19
2B	1C	02	58		BCTA CLEAR	20
2E	E5	0F			COMI R1,0F	21
30	C0	C0	C0		NOPs	22
33	17				RETC	23
						24
						25
						23
						24
						25
235	07	07		ANZEI	LODI R3,07	1
237	0F	62	AF	ANLOP	LODA R3*	2
3A	3F	00	DE		BSTA CONVT	3
3D	CF	62	BF		STRA R3*	4
40	FB	75			BDRR ANLOP	5
42	0C	02	C3		LODA R0	6
45	D	40	00		WRTE R0,00	7
47	0C	02	C2		LODA R0	8
4A	D4	01			WRTE R0,01	9
4C	0C	02	C1		LODA R0	10
4F	D4	02			WRTE R0,02	11
51	0C	02	C0		LODA R0	12
54	D4	03			WRTE R0,03	13
56	17				RETC	14
						15
258	07	10		CLEAR	LODI R3,10	16
5A	0	40	0E		LODI R0,0E	17
5C	CF	62	AF	CLRLP	STRA R3*	18
5F	FB	7B			BDRR CLRLP	19
61	17				RETC	20
						21
						22

Bild 12: Die Auflistung der vorgestellten Programmteile; das Unterprogramm WAEHL muß ebenfalls eingegeben werden (vgl. vorigen Beitrag).

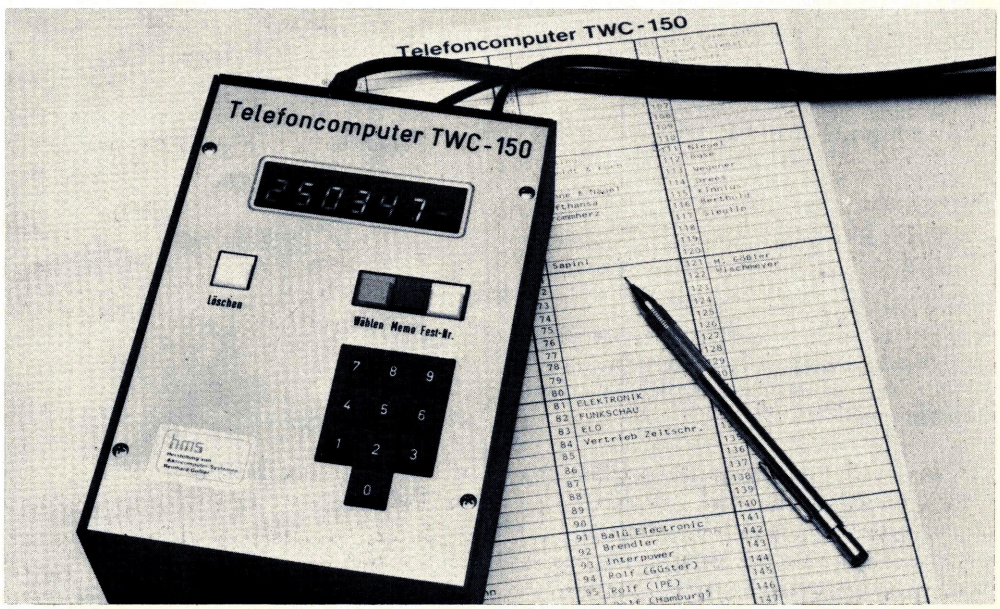


Bild 13: Entwicklungsstudie für einen automatischen Rufnummernspeicher und -geber auf der Basis des UMS-85, der bis zu 150 Festnummern auf Abruf bereithalten kann.

Daten zu Papier gebracht

Wenn man – oft nach langen Mühen – ein Mikrocomputer-Programm endlich zum Laufen gebracht hat, gehört es zu den wichtigsten Aufgaben, den erreichten Stand zu fixieren, d.h. die Befehlsfolge zu dokumentieren. Hierzu leistet ein Drucker wertvolle Hilfe, und nach dem Magnetband-Interface als externer Massenspeicher (letzte Folge) wollen wir hier einen Normalpapier-Drukker vorstellen, der als leistungsfähiges Peripheriegerät unsere Mikrocomputer UMS-85 bzw. ELDO ergänzt (**Bild 1**). Das Interface eigent sich aber auch zum Anschluß an alle anderen Mikrocomputer mit Parallelschnittstelle; es ist als Bausatz oder fertig montiert lieferbar, während die Druckwerk-Mechanik in jedem Fall zusammengebaut und getestet geliefert wird.

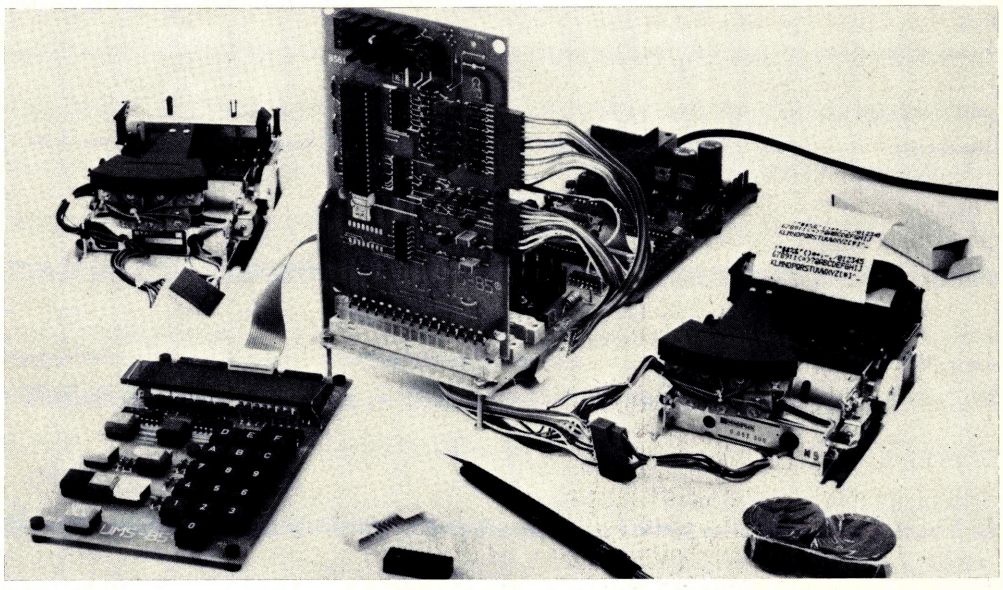


Bild 1: Mit dem Druckwerk und zugehörigem Interface wird der Mikrocomputer zum richtigen Entwicklungsplatz.

Ein eigener Computer für das Druckwerk

Aus dem Blockschaltbild (**Bild 2**) erkennen Sie die funktionelle Trennung zwischen Mechanik (Druckwerk) und Ansteuer-Elektronik (Interface). Im Druckwerk sind sieben vertikal untereinander angeordnete Nadeln dafür verantwortlich, daß durch deren geeignete Aktivierung die Zeichen und Buchstaben aufs Papier kommen. Da jedes Zeichen aus einer 5 x 7-Punktmatrix besteht, müssen die Nadeln pro Zeichen fünf Mal – jeweils um eine Punktbreite versetzt – aktiviert werden (**Bild 3**). Weil dies zu-

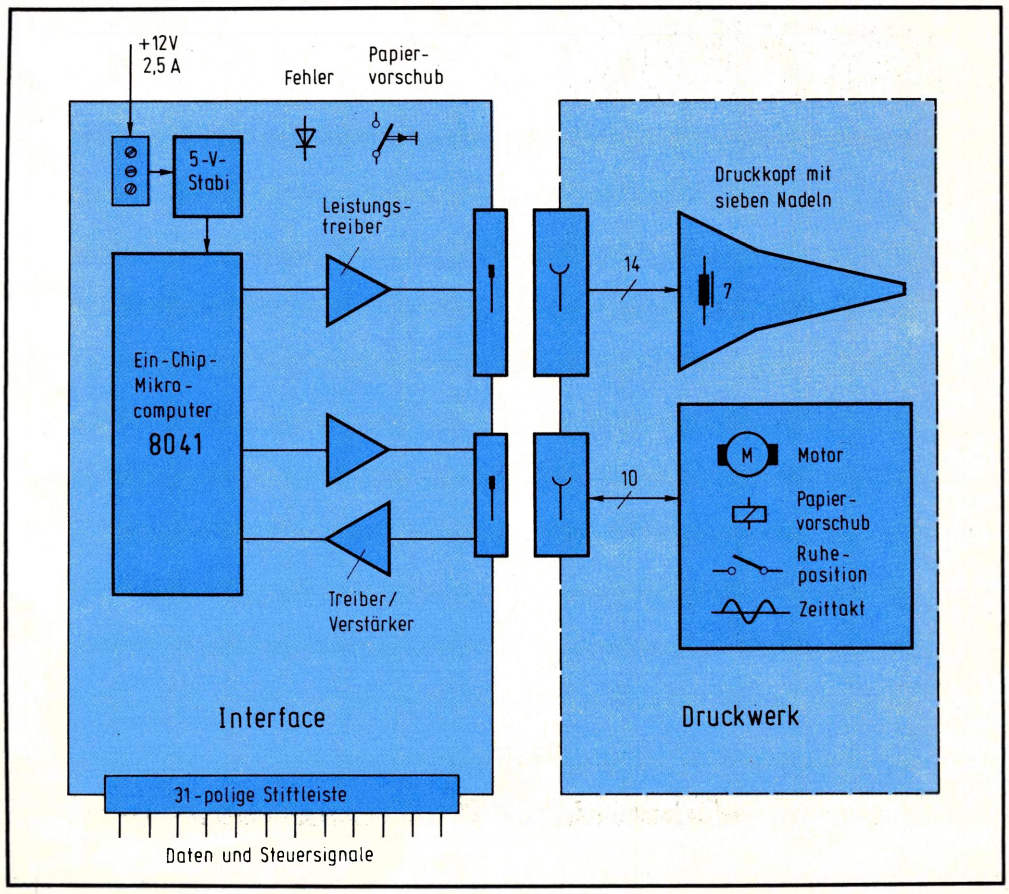


Bild 2: Die Blockschaltung zeigt, daß zur Ansteuerung des Druckwerks eine eigene Interface-Karte erforderlich ist.

```
DAS DRUCKBILD IST TOP
DANK NORMALPAPIER UND
5*7-PUNKTMATRIX
```

Bild 3: Durch die Verwendung von Farbband und Normalpapier entsteht ein Druckbild ausgezeichneter Lesbarkeit.

sammen mit der Motor- und Zeittakt-Steuerung ein aufwendiges Unternehmen ist, übernimmt diese Fleißaufgabe ein eigener Ein-Chip-Mikrocomputer (8041 mit internem RAM und ROM). Der bekommt vom UMS-85 bzw. ELDO eingangsseitig ASCII-

Informationen angeboten (Details entnehmen Sie bitte dem Sonderheft „Dem Mikrocomputer auf's Bit geschaut") und erzeugt daraus die benötigten Steuerimpulse für das Druckwerk. Ehe wir darauf näher eingehen, wollen wir die Schaltung (**Bild 4**) und den Bestückungsplan (**Bild 5**) der Interface-Karte vorstellen.

Die Karte findet entweder in der CPU-Buchsenleiste Platz, oder Sie stecken sie beim Betrieb mit der Bus- und Speichererweiterung dort in die hinterste (der CPU am nächsten liegende) Buchsenleiste (Interface-Adresse: 0C03 im Memory-Mapped-Betrieb; vgl. Seite 20). Je nach Mikrocomputer-Typ sind die Brücken A + F (UMS-85) bzw. B + E (ELDO) zu verdrahten. Beim Einsetzen des MOS-Bausteins 8041 achten Sie bitte wieder darauf, daß Sie ihn nicht mit 10 000 V Ihrer statischen (Körper-)Aufladung zerschießen; alle übrigen Teile brauchen nur am richtigen Ort plaziert und eingelötet zu werden.

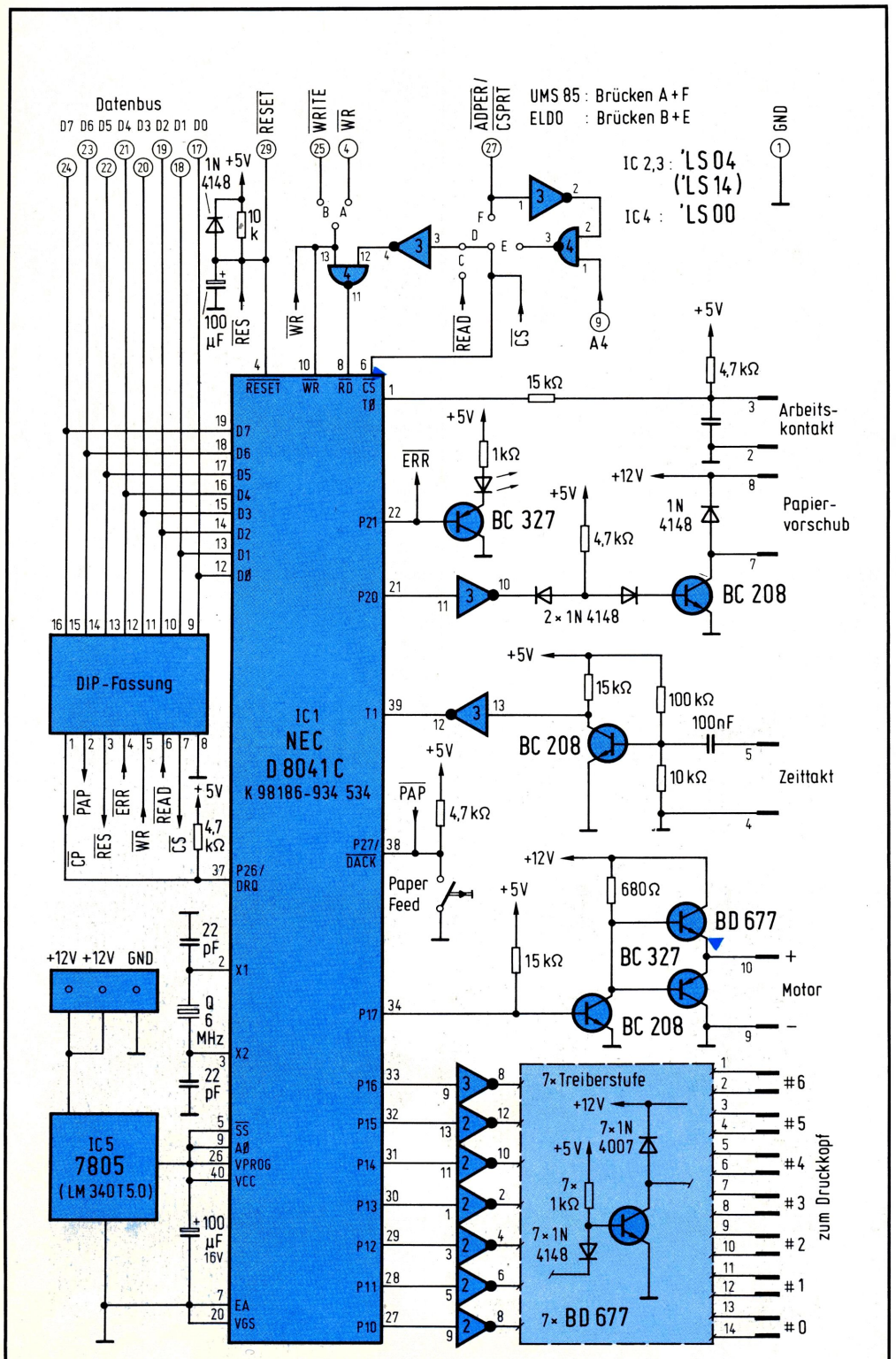


Bild 4: Die Detailschaltung der Interface-Karte.

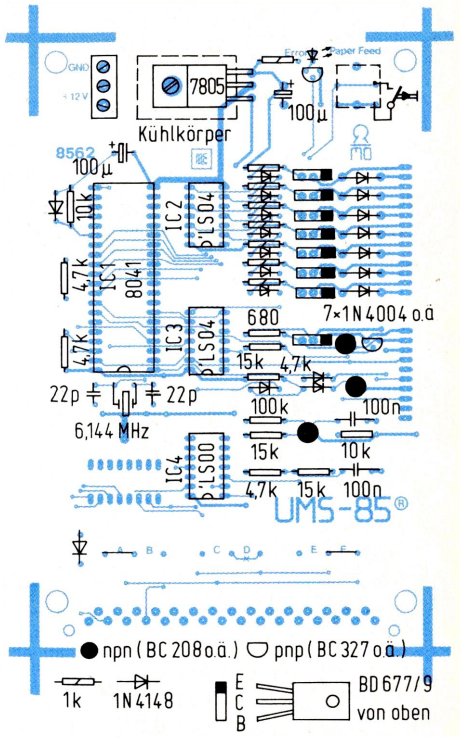


Bild 5: Auf dem Bestückungsplan ist die übersichtliche Anordnung der Bauteile zu erkennen.

Der Anschluß des Druckwerks erfolgt über die Buchsenleisten, deren Nase zur Bauteilseite der Platine zeigen muß. Da die Anschlußkabel etwas kurz geraten sind (Herstellergeiz), liegt der Lieferung ein Flachbandkabel bei, das Sie dreiteilen und zur Verlängerung benutzen können. In diesem Fall löten Sie platinenseitig das Kabel ein und löten am anderen Ende die 14- bzw. 10polige Steckerhülse an, die dem Druckwerk beiliegen.

Das Farbband legen Sie bitte so ein, daß es um die beiden Umlenkrollen geführt wird; die drei Transportdorne der Farbbandrollen sind mit der Nase nach unten einzusetzen (**Bild 6**). Außerdem müssen Sie dafür sorgen, daß das Farbband unter den beiden Andruckfedern links und rechts vom Druckkopf hindurchgeführt wird.

Betreiben Sie den Drucker niemals ohne eingelegtes Papier!

Das Papier (58 mm Normalpapier aus dem Bürohandel) schneiden Sie am besten spitz zu, ehe Sie es in den Zuführungsschlitz (von hinten) einführen; wenn Sie das Ny-

lon-Hemmrad auf der linken Seite des Druckwerks (bei Draufsicht) dann manuell drehen, wird das Papier erfaßt und tritt oben hinter dem Farbband wieder aus. Später dient zum Papiertransport die entsprechende Taste auf der Interface-Karte.
Das Einsetzen der Interface-Karte erfolgt selbstverständlich im stromlosen Zustand (sowohl CPU- als auch Interface-Versorgung abgeschaltet). Erst danach können Sie dem Mikrocomputer „Saft" zuführen und dann ein geeignetes 12-V-Netzteil (2,5 A Dauerstrom) mit der Interface-Karte verbinden. Testen Sie Ihren Aufbau nun wie folgt: Computer-RESET drücken (und festhalten), dann Taste „Papiervorschub" drücken und dabei RESET wieder loslassen; nach ca. 3 s muß der Drucker mit seinem Eigentest-Programm losrattern und seinen Zeichenvorrat zum Ausdruck bringen (**Bild 7**).

```
!"#$%&'()*+,-./012345
6789:;<=>?@ABCDEFGHIJ
KLMNOPQRSTUVWXYZ[¥]^_
```

Bild 7: Beim Eigentest druckt der Controller seinen ASCII-Zeichenvorrat aus.

Lassen Sie sich Visitenkarten drucken

Um den Drucker dazu zu bringen, einen Text Ihrer Wahl auszudrucken, gehen Sie folgendermaßen vor: Sie laden das interne RAM des 8041 mit den gewünschten Zeichen (ASCII-Code) einer ganzen Zeile (21 Zeichen) und geben dann den Druckbefehl („0A") an den Controller aus. Die auszudruckenden Zeichen plaziert man vorher vorteilhafterweise nacheinander in einen separaten RAM-Bereich des Mikrocomputer-Arbeitsspeichers (Zeichenbuffer). Das Unterprogramm PRINT (**Bild 8**) übernimmt dann das sequentielle Auslesen aus dem Zeichenbuffer und zeitrichtige Überschreiben in den Controller; vor dem Einsprung nach PRINT muß das Registerpaar D&E mit der Anfangsadresse des Zeichenbuffers geladen werden.
Baut man sich dazu ein winziges Hauptprogramm DRUCK (**Bild 9**), kann man damit bereits ganze Texte ausdrucken lassen. Zu Beginn lädt man die gewünschte Zeilenzahl nach Register C (Schleifenzähler), definiert den Beginn des Zeichenbuffers (D&E = 0800) und ruft das Unterprogramm PRINT in einer Schleife so oft auf, wie es der Zähler in C vorgibt.

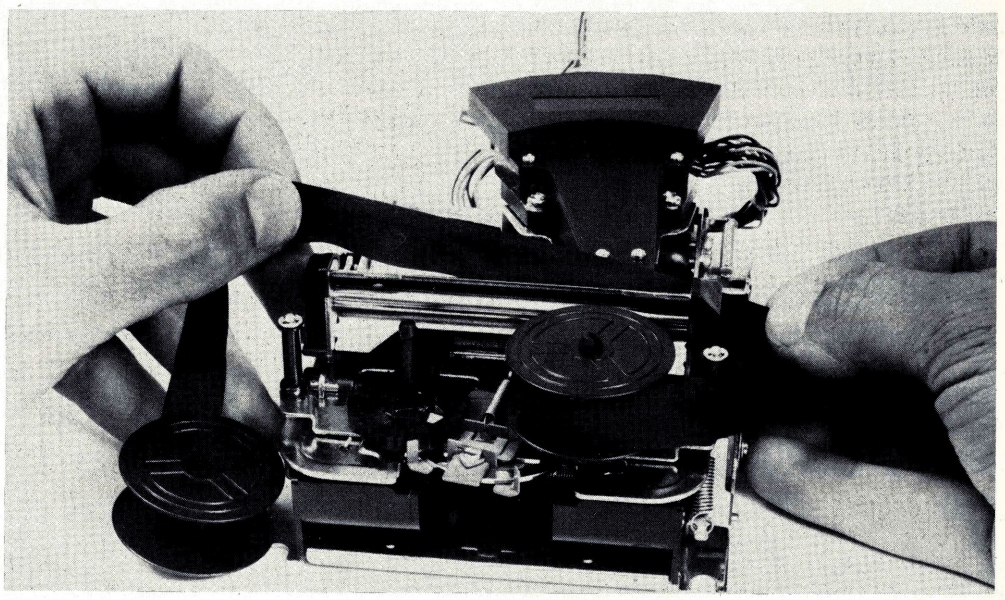


Bild 6: Beim Einsetzen des Farbbandes ist auf richtige Führung zu achten.

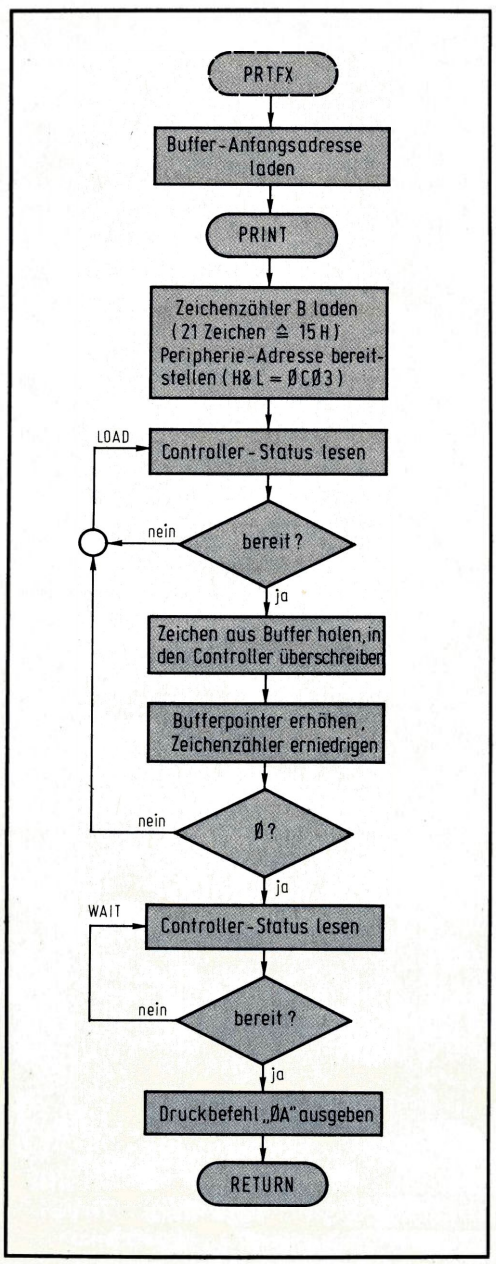
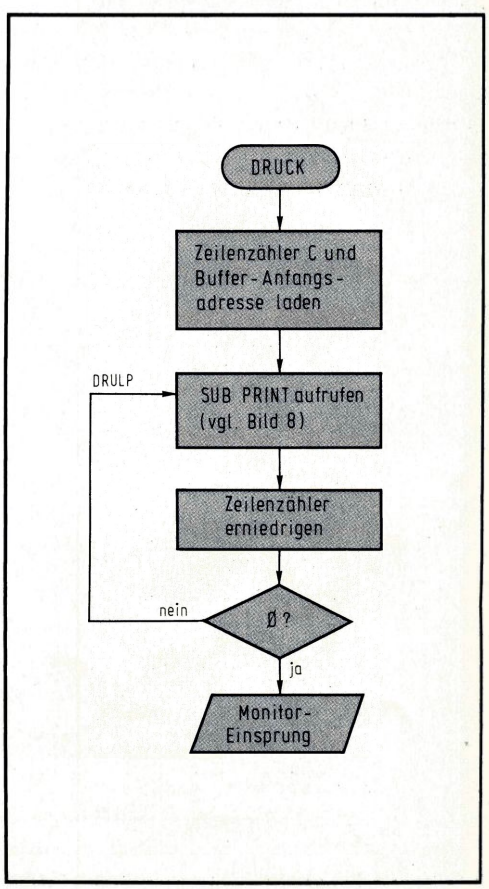


Bild 8: Unterprogramm PRINT, das 21 Zeichen aus dem RAM ausliest, in den Controller überschreibt und deren Ausdruck veranlaßt.

Bild 9: Mit dem übergeordneten Programm DRUCK kann man mehrere Zeilen nacheinander ausdrucken.



Im Prinzip wissen Sie nun ja schon, wie Sie Ihrem Drucker das Drucken beibringen können. Jetzt muß nur noch die Software geladen werden und Sie können schwarz auf weiß nachlesen, was Ihr Computer in seinen kleinen (Speicher-)Zellen gerade so „denkt".
Wie gewohnt – und wie sich's gehört –, sind die entsprechenden Programme für den UMS 85 und den ELDO getrennt aufgeführt.

Nun müssen Sie nur noch wissen, mit welchem Zeichen Sie Ihren Buffer füllen müssen, um zum gewünschten Text zu kommen. Schauen Sie sich dazu die **Tabelle 1** an, die diesen Zusammenhang darstellt.
Um beispielsweise ein „A" auszudrucken, muß man die „41" in den Buffer laden; bei hexadezimal „2A" druckt der Drucker ein Sternchen, und um eine „9" zu erhalten, geben Sie „39" in den Buffer ein. Bei „20" passiert gar nichts, denn da macht der Drucker ein Leerzeichen (Blank); dem Code „5C" schließlich ist nicht, wie üblich, das Dollarzeichen „$" zugeordnet, sondern jenes für den Yen „¥" (weil der Controller japanische Eltern hat).
Das Maschinenprogramm von DRUCK und PRINT zeigt **Bild 10**. Auch wenn Sie die Bus- und Speichererweiterung nicht besitzen, legen Sie die hier vorgestellten Programme unter den angegeben Adressen ab; in diesem Fall landen Adreßeingaben „OBXX" automatisch im Bereich „08XX" (vgl. „Der Monitor denkt mit" auf Seite 22).
Zum Starten müssen Sie die Anfangsadresse eingeben (ADR-B-9-0), gefolgt von RUN, und schon knattern die Nadeln über das Papier und drucken aus, was immer Sie zuvor nach 0800 ff. geladen haben.

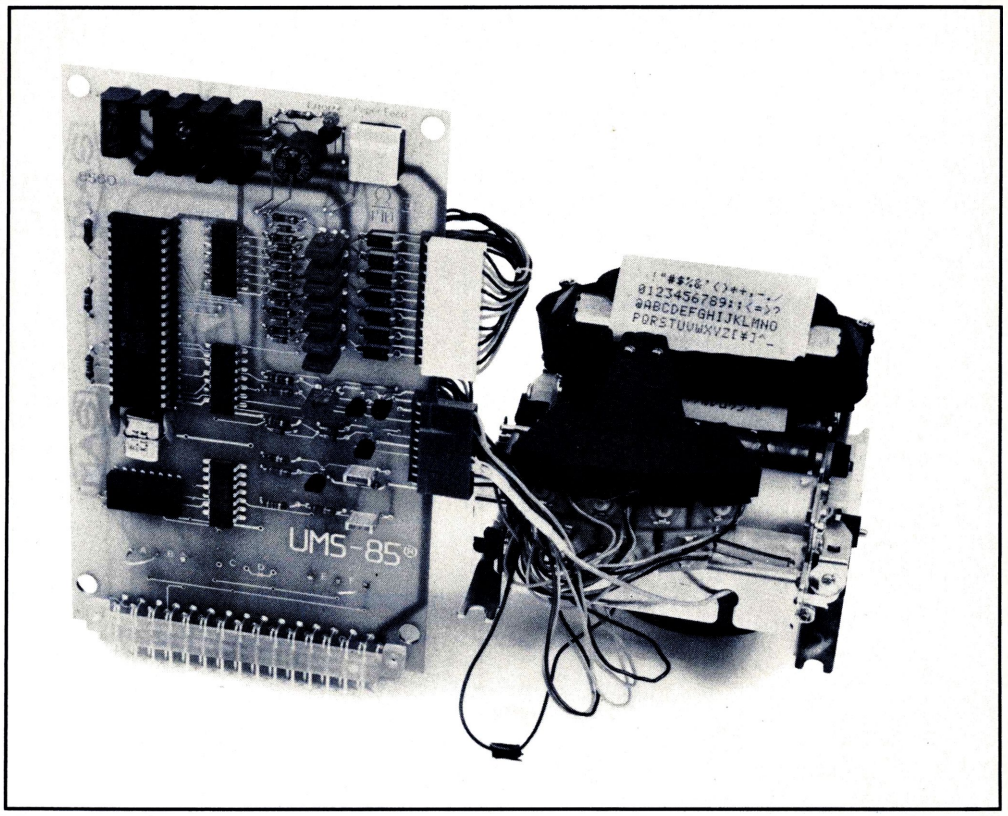
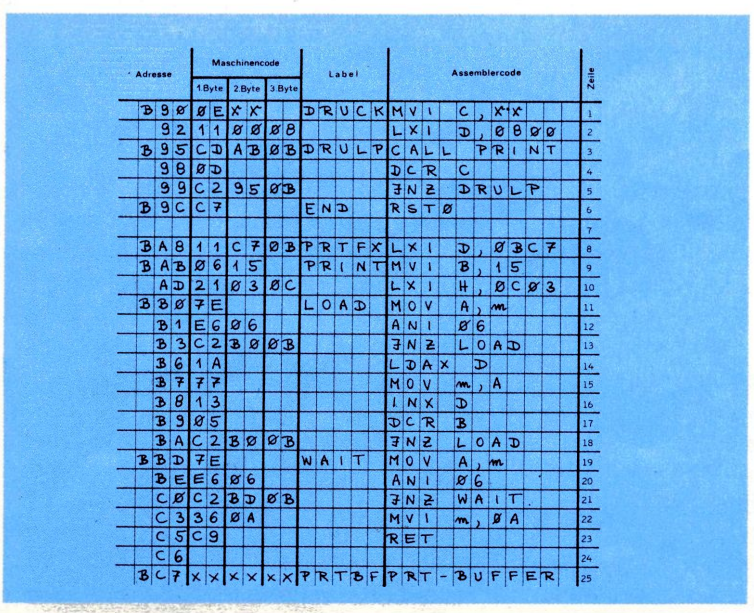
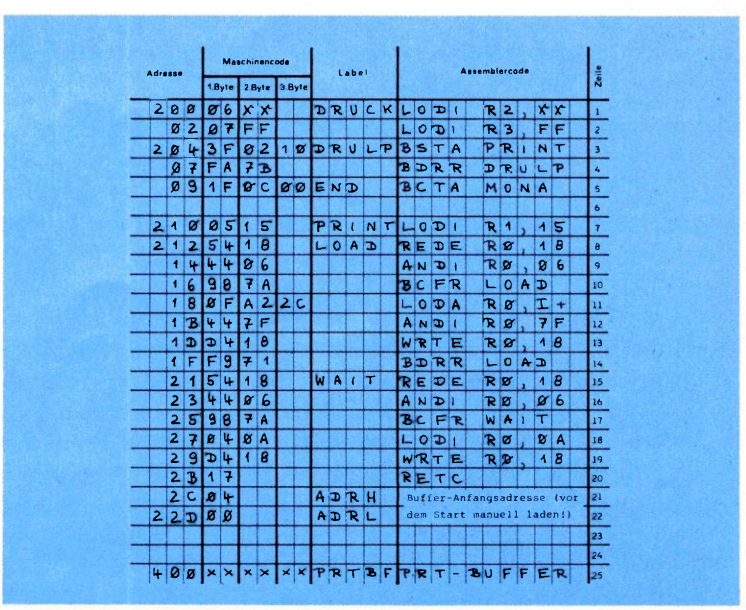


Tabelle 1: ASCII-Codes (untere Zeile) für die Standard-Druckzeichen.

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
0	1	2	3	4	5	6	7	8	9	:	;	[	=	]	?
30	31	32	33	34	35	36	37	38	39	3A	3B	3Ç	3D	3E	3F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
P	Q	R	S	T	U	V	W	X	Y	Z	I	¥	I	^	_
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F



Adresse	Maschinencode 1 Byte	2 Byte	3 Byte	Label	Assemblercode	Zeile
B98Ø	ØE	XX		DRUCK	MVI C, XX	1
92	11	ØØ	ØB		LXI D, ØBØØ	2
B95	CD	AB	ØB	DRULP	CALL PRINT	3
98	ØD				DCR C	4
99	C2	95	ØB		JNZ DRULP	5
B9C	C7			END	RST Ø	6
						7
BA8	11	C7	ØB	PRTFX	LXI D, ØBC7	8
BAB	Ø6	15		PRINT	MVI B, 15	9
AD	21	Ø3	ØC		LXI H, ØC Ø3	10
BBØ	7E			LOAD	MOV A, m	11
B1	E6	Ø6			ANI Ø6	12
B3	C2	BØ	ØB		JNZ LOAD	13
B6	1A				LDAX D	14
B7	77				MOV m, A	15
B8	13				INX D	16
B9	Ø5				DCR B	17
BAC	2	BØ	ØB		JNZ LOAD	18
BBD	7E			WAIT	MOV A, m	19
BE	E6	Ø6			ANI Ø6	20
CØC	2	BD	ØB		JNZ WAIT	21
C3	36	ØA			MVI m, ØA	22
C5	C9				RET	23
C6						24
BC7	XX	XX	XX	PRTBF	PRT-BUFFER	25



Adresse	Maschinencode 1 Byte	2 Byte	3 Byte	Label	Assemblercode	Zeile
2ØØ	Ø6	XX		DRUCK	LODI R2, XX	1
Ø2	Ø7	FF			LODI R3, FF	2
2Ø4	3F	Ø2	1Ø	DRULP	BSTA PRINT	3
Ø7	FA	7B			BDRR DRULP	4
Ø9	1F	ØC	ØØ	END	BCTA MONA	5
						6
21Ø	Ø5	15		PRINT	LODI R1, 15	7
212	54	18		LOAD	REDE RØ, 18	8
14	44	Ø6			ANDI RØ, Ø6	9
16	98	7A			BCFR LOAD	10
18	ØF	A2	2C		LODA RØ, I+	11
1B	44	7F			ANDI RØ, 7F	12
1D	D4	18			WRTE RØ, 18	13
1F	F9	71			BDRR LOAD	14
215	54	18		WAIT	REDE RØ, 18	15
23	44	Ø6			ANDI RØ, Ø6	16
25	98	7A			BCFR WAIT	17
27	Ø4	ØA			LODI RØ, ØA	18
29	D4	18			WRTE RØ, 18	19
2B	17				RETC	20
2C	Ø4			ADRH	Buffer-Anfangsadresse (vor	21
2D	ØØ			ADRL	dem Start manuell laden!)	22
						23
						24
4ØØ	XX	XX	XX	PRTBF	PRT-BUFFER	25

Bild 10: Das Maschinenprogramm zu den Bildern 8 und 9, für UMS und ELDO getrennt aufgeführt.

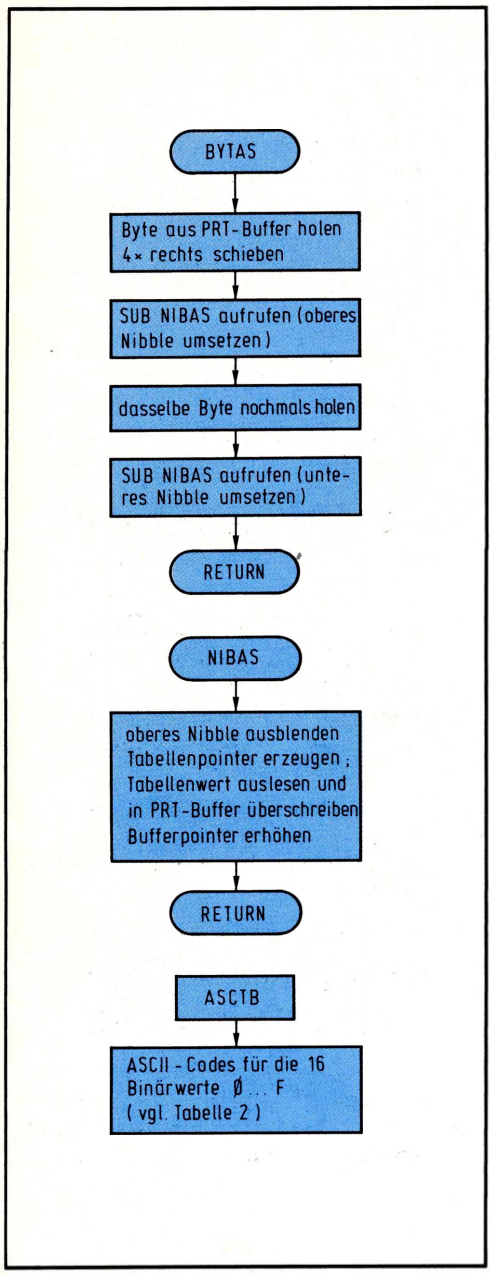


Bild 11: Zur Umsetzung eines Datenbytes in zwei ASCII-Zeichen dienen die beiden Unterprogramme BYTAS und NIBAS, die die Umsetztabelle ASCTB benutzen.

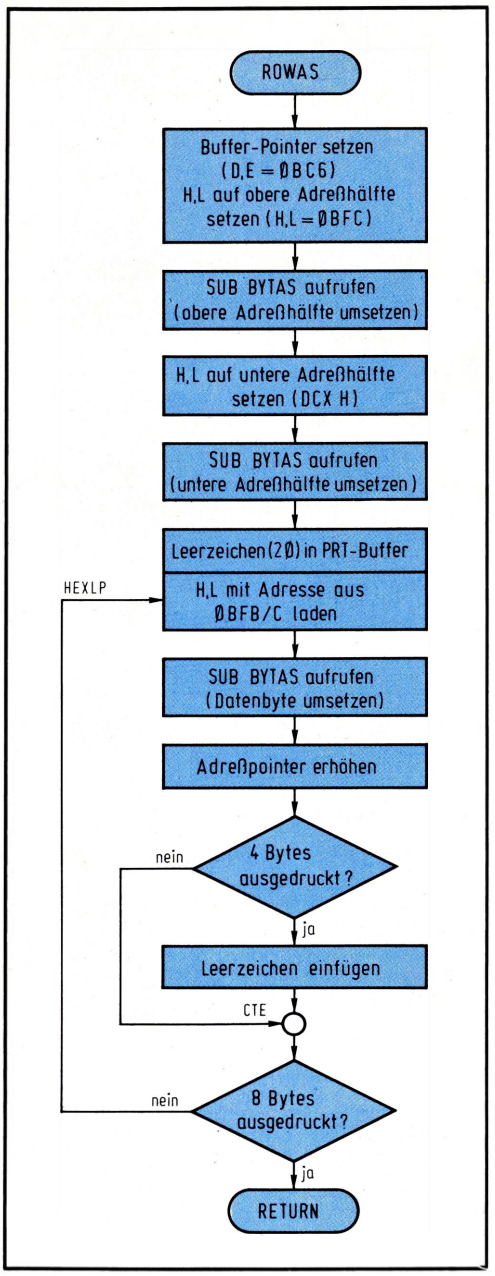


Bild 12: Unter Zuhilfenahme der Unterprogramme aus Bild 8 und 9 führt ROWAS die ASCII-Umsetzung von 8 aufeinanderfolgenden Bytes durch, denen die Anfangsadresse vorangestellt wird.

Quasi am Rande registrieren Sie den Programmschluß von DRUCK: Dort steht kein herkömmlicher JUMP-Befehl an irgendeiner Stelle, sondern einer der acht RESTART-Befehle (RST0...7; vgl. Befehlsliste im 2. Teil, S. 12). Dies sind Ein-Wort-Sprungbefehle mit fester Zieladresse, und bei der Ausführung von RST0 erfolgt ein unbedingter Sprung zur Adresse 0000 (entspricht der Software-RESET-Funktion).

Der Speicherinhalt schwarz auf weiß

Ein Mikrocomputer-Profi (und ein jeder, der auch nur halbwegs dorthin strebt) möchte sich natürlich nicht nur Texte, sondern vor allem seine Programme ausdrucken lassen (Listings anfertigen). Da das Drucker-Interface aber die (weltweit standardisierte) ASCII-Schnittstelle besitzt, müssen die binären Daten zuvor in den ASCII-Code umgesetzt werden. Für den ELDO ist dies bereits in dem oben zitierten Sonderheft beschrieben worden, so daß wir uns hier dem entsprechenden 8085-Programm zuwenden wollen.
Das Unterprogramm BYTAS (**Bild 11** oben) holt sich dazu ein Byte aus dem Speicher (Byte-Adresse steht in H&L), setzt dieses in zwei HEX-Digits (ASCII-Code) um und überschreibt diese in den PRINT-Buffer, von wo aus sie später in den Controller geladen werden. BYTAS ruft das Unterprogramm NIBAS auf (**Bild 11** unten), welches das untere Halbbyte (Nibble) des im Akkumulator stehenden Wortes in den ASCII-Code umsetzt; NIBAS entnimmt die Zuordnung zwischen binärer und ASCII-Information der Tabelle ASCTB (**Tabelle 2**; die müssen Sie ebenfalls ins RAM laden).
Um eine ganze Druckzeile ASCII-gerecht zu erzeugen, baut man das übergeordnete Programm ROWAS auf (**Bild 12**). Es stellt die Anfangsadresse eines 8-Byte-Blocks dar, gefolgt von den acht Bytes selbst; Beim Einsprung nach ROWAS muß die Anfangsadresse des umzusetzenden und auszudruckenden Speicherbereichs in H&L stehen.

Tabelle 2: ASCII-Codes für die Binärwerte 0...F (ab Adresse 0B98 ins RAM laden).

0B98	30	Code für 0
0B99	31	1
0B9A	32	2
0B9B	33	3
0B9C	34	4
0B9D	35	5
0B9E	36	6
0B9F	37	7
0BA0	38	8
0BA1	39	9
0BA2	41	A
0BA3	42	B
0BA4	43	C
0BA5	44	D
0BA6	45	E
0BA7	46	F

Speicherabzug zeilenweise

Wenn wir diese Programmfolge, die ja nur bescheidene acht Bytes druckgerecht aufbereitet hat, um geringe Zusätze ergänzen, entsteht das Programm LIST (**Bild 13**), das Ihnen – wenn Sie wollen – den Speicherausdruck meterweise liefert. Es ist so organisiert, daß vor dem Einsprung die Anfangsadresse des gewünschten Speicherblocks in den RAM-Zellen 0BFB/C und in 0BFD die Anzahl der auszudruckenden Zeilen stehen muß (ähnliche Organisationen wie bei der Cassetten-Ein- und -Ausgabe). Da die Adreßeingabe sinnfälligerweise mit der oberen Hälfte beginnt, der 8085 es aber

andersherum besser findet, vertauscht LIST zunächst oberes und unteres Byte der in 0BFB/C stehenden Adresse. Die dazu mit eingebauten Befehle LHLD YY XX (Load H&L Direct) bzw. SHLD (Store H&L Direct) laden das Registerpaar H&L mit derjenigen Information, die in der im Befehl angegebnen Adresse (XX YY) und (XX YY + 1) steht (bzw. umgekehrt bei SHLD). Danach folgen der Aufruf von ROWAS (Umsetzung der ersten 8 Bytes) und PRINT (Überschreiben der ASCII-Daten in den Controller und Ausdruck), was sich so oft wiederholt, wie es der Schleifenzähler in 0BFD angibt.
Laden Sie nun diese gesamte Sequenz in den Speicher (**Bild 14**) und lassen Sie sich einen Speicherausdruck anfertigen, beispielsweise von dem eben geladenen Programm (das ist genauso, wie wenn ein Arzt seine eigene Krankheit behandelt). Dazu geben Sie die Speicher-Anfangsadresse 0B30 nach 0BFB/C ein (ADR-B-F-B-DAT-0-B-NXT-3-0-NXT), gefolgt von der gewünschten Zeilenzahl, z.B. 22D = 16H nach 0BFD (einfach weitermachen mit 1-6-NXT). Zum Starten des LIST-Programms gehen Sie zu dessen Anfangsadresse 0B30 (ADR-B-3-0), gefolgt von RUN, und Sie erhalten einen Ausdruck gemäß **Bild 15**. Darauf erkennen Sie, wie sich artig die Bytes 2A-FB-0B-7D usf. aneinanderreihen, wie Sie sie ja eben auch dorthin geladen haben.

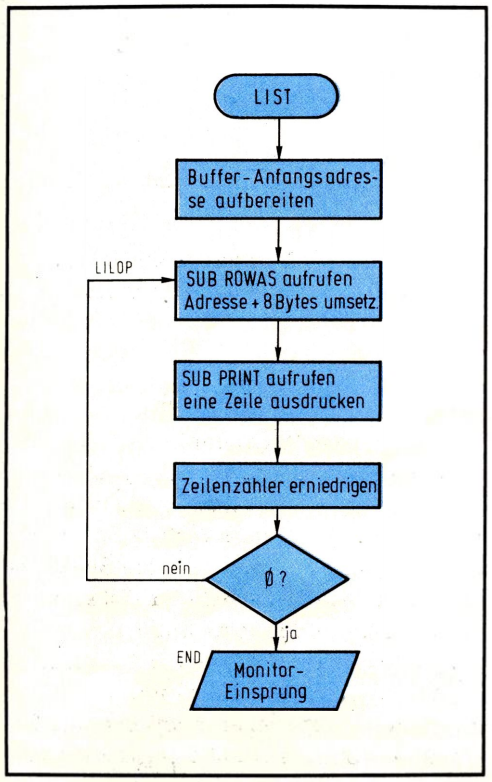


Bild 13: Das Hauptprogramm LIST kann den Speicherinhalt in hexadezimaler Form ausdrucken (vgl. Bild 15).

Adresse	Maschinencode 1 Byte	2 Byte	3 Byte	Label	Assemblercode	Zeile
B30	2A	FB	0B	LIST	LHLD 0BFB	1
33	7D				MOV A,L	2
34	6C				MOV L,H	3
35	67				MOV H,A	4
36	22	FB	0B		SHLD 0BFB	5
B39	CD	54	0B	LILOP	CALL ROWAS	6
3C	CD	A8	0B		CALL PRINT	7
3F	21	FD	0B		LXI H,0BFD	8
42	35				DCR M	9
43	C2	39	0B		JNZ LILOP	10
B46	C7			END	RST 0	11
						12
B54	11	C6	0B	ROWAS	LXI D,0BC6	13
57	21	FC	0B		LXI H,0BFC	14
5A	CD	80	0B		CALL BYTAS	15
5D	2B				DCX H	16
5E	CD	80	0B		CALL BYTAS	17
61	3E	20			MVI A,20	18
63	12				STAX D	19
64	13				INX D	20
B65	2A	FB	0B	HEXLP	LHLD 0BFB	21
68	CD	80	0B		CALL BYTAS	22
6B	23				INX H	23
6C	22	FB	0B		SHLD	24
6F	3E	D3			MVI A,D3	25
71	BB				CMP E	1
72	C2	79	0B		JNZ CTE	2
75	3E	20			MVI A,20	3
77	12				STAX D	4
78	13				INX D	5
B79	3E	DC		CTE	MVI A,DC	6
7B	BB				CMP E	7
7C	C2	65	0B		JNZ HEXLP	8
7F	C9				RET	9
B80	7E			BYTAS	MOV A,M	10
81	0F				RRC	11
82	0F				RRC	12
83	0F				RRC	13
84	0F				RRC	14
85	CD	8D	0B		CALL NIBAS	15
88	7E				MOV A,M	16
89	CD	8D	0B		CALL NIBAS	17
8C	C9				RET	18
B8D	E6	0F		NIBAS	ANI 0F	19
8F	C6	90			ADI 90	20
91	4F				MOV C,A	21
92	06	0B			MVI B,0B	22
94	0A				LDAX B	23
95	12				STAX D	24
96	13				INX D	25
B97	C9				RET	

Bild 14: Das vollständige Maschinenprogramm der vorgestellten Haupt- und Unterprogramme wird noch durch die Umsetztabelle (vgl. Tabelle 2) und PRINT (vgl. Bild 10) ergänzt.

```
B30 2AFB0B7D 6C6722FB
B38 0BCD540B CDA80B21
B40 FD0B35C2 390BC700
B48 00000000 00000000
B50 00000000 11C60B21
B58 FC0BCD80 0B2BCD80
B60 0B3E2012 132AFB0B
B68 CD800B23 22FB0B3E
B70 D3BBC279 0B3E2012
B78 133EDCBB C2650BC9
B80 7E0F0F0F 0FCD8D0B
B88 7ECD8D0B C9E60FC6
B90 984F060B 0A1213C9
B98 30313233 34353637
BA0 38394142 43444546
BA8 11C70B06 1521030C
BB0 7EE606C2 B00B1A77
BB8 1305C2B0 0B7EE606
BC0 C2BD0B36 0AC93042
BC8 43382034 33333832
BD0 30333420 33333333
BD8 33333333 805B880B
```

Bild 15: Ein mit Hilfe von LIST angefertigter Speicherabzug der in Bild 14 vorgestellten Programmteile.

```
SCHRIFTBREITE 18 Z
SCHRIFTBREITE 21 Z/Z.
```

Bild 16: An einem Anschluß des Controllers kann man die Schreibbreite von 21 Zeichen pro Zeile auf 18 Z/Z umschalten.

Zwei Besonderheiten der Ansteuer-Elektronik sollen Sie noch erfahren, um bei Bedarf darauf zurückgreifen zu können. Wenn Sie Stift 37 des Controllers erden, druckt Ihr Druckwerk statt 21 Zeichen pro Zeile nur noch 18 Z/Z (**Bild 16**). Wenn Sie wollen, können Sie diese Schriftverbreiterung sogar per Programm umschalten, indem Sie ein Bit des CPU-Ausgabe-Kanals dafür verwenden.
Bei Störungen im Druckablauf kommt der Zeittakt vom Druckwerk nicht mehr in der richtigen Sequenz zum Controller, was diesen sofort veranlaßt, alle Beine von sich zu strecken: Er schaltet seine Ausgänge in den hochohmigen Zustand, aktiviert die Fehler-Leuchtdiode („Error") und verharrt in dieser Lage, bis ihn ein (manueller) RESET-Impuls wieder erlöst.

Achtung:

Nach dem Druck auf RESET bleibt das Drucker-Interface noch ca. 3 s lang inaktiv, damit sich die übrige Schaltung mit Sicherheit richtig eingestellt hat; während dieser Zeit können Sie kein Programm starten, das den Drucker anspricht, weil der dann noch taub ist.
Nun sind Sie in der Lage, vom Klartext bis zum Speicherinhalt jede gewünschte Information auszudrucken, also beispielsweise auch den Wert einer Spannung, den Sie zuvor mit dem Analog-Interface digitalisiert haben. Sie sehen, daß Sie Ihr Mikrocomputer-System mit dem Drucker zu einem kompletten kleinen Entwicklungsplatz ausbauen, der sogar für Aufgaben der Meßdatenerfassung geeignet ist.

Digitalwecker per Computer

Eine der schönsten Aufgaben für einen Computer ist es, als Uhr eingesetzt zu werden. Die hierfür erforderlichen Tätigkeiten des Zählens, Formatierens, Verzweigens usw. sind ihm so recht auf den Leib geschnitten, und Sie werden schnell sehen, wie diese simpel anmutende Aufgabe bereits recht komplexe Programmtechniken erfordert. Diese stehen bei dem folgenden Beispiel auch im Vordergrund und bilden lediglich die Grundlage für das später folgende Programmbeispiel der sprechenden Computer-Uhr. Machen Sie sich darum hier zunächst mit der Programmorganisation vertraut, ehe wir gemeinsam darangehen, den Computer (im wahrsten Sinne des Wortes) zum Sprechen zu bringen.

Unorthodoxes Zählen

Würde man heute die Uhr erfinden, hätten die Minuten 100 Sekunden, die Stunde 100 Minuten und der Tag 10 Stunden; das Programm zum Weiterzählen einer solchen (Computer-)Uhr wäre wahnsinnig einfach, und wir würden kaum wagen, Ihnen so etwas Leichtes vorzusetzen. Darum ist es ein Glück, daß es bei der richtigen Uhr kunterbunt durcheinandergeht: Übertrag der Sekunden- und Minuten-Einer bei 10, der Zehner bei 60; Übertrag des Stunden-Einers meistens bei 10, manchmal aber auch bei 4 (um 24 Uhr). Da lohnt es sich schon eher, einen Mops heranzunehmen und dem so ein unorthodoxes Verhalten zu übertragen (**Bild 1**). Dementsprechend wollen wir unseren Mikrocomputer programmieren und die Anzeige dazu benutzen, die Zeit (Stunden, Minuten und Sekunden) darzustellen.
Es ist hierzu sinnvoll, einen separaten RAM-Bereich (Uhrzeit-Buffer UHRBF) zu reservieren, in dem die sechs Zähler für Sekunden-Einer... Stunden-Zehner angeordnet sind (**Bild 2**). Obwohl jeder Zähler höchstens eine Neun haben kann (der Stunden-Zehner sogar nur eine Zwei), spendieren wir jeweils ein ganzes 8-Bit-Wort, dessen obere Hälfte immer Null ist.
Sie wissen noch, daß Sie auch ohne Speichererweiterung getrost Adressen „OBXX" laden können und diese im Bereich „O8XX" landen? (Vgl. „Der Monitor denkt mit" im 5. Teil, Seite 25.)

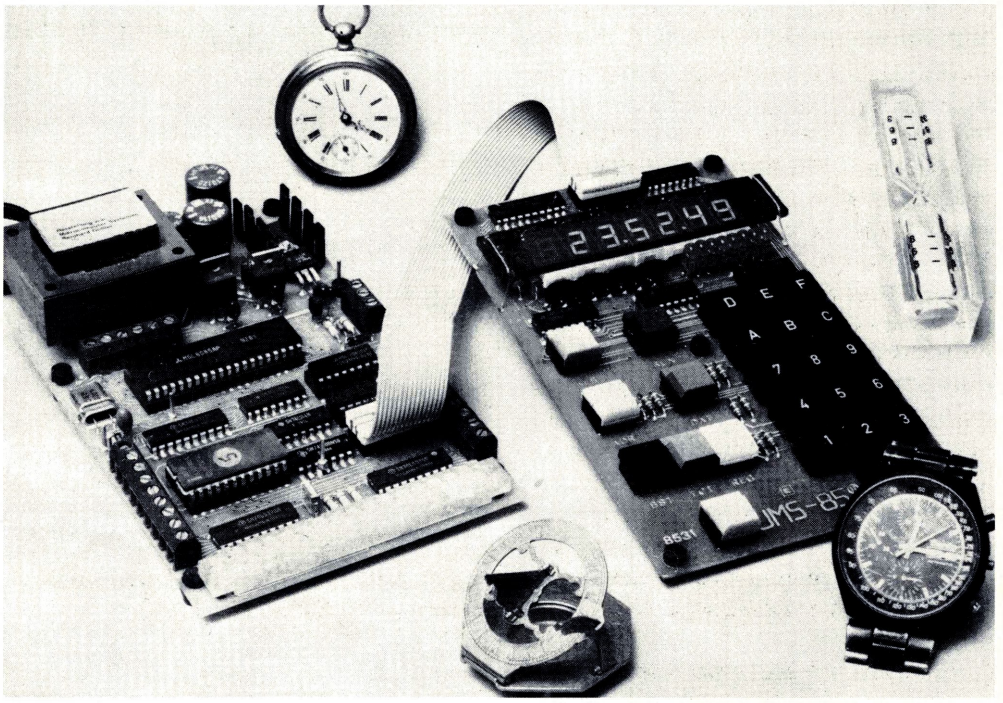


Bild 1: Bei geeigneter Programmierung wird der Mikrocomputer zur Digitaluhr – bei Bedarf sogar mit komfortabler Weckeinrichtung.

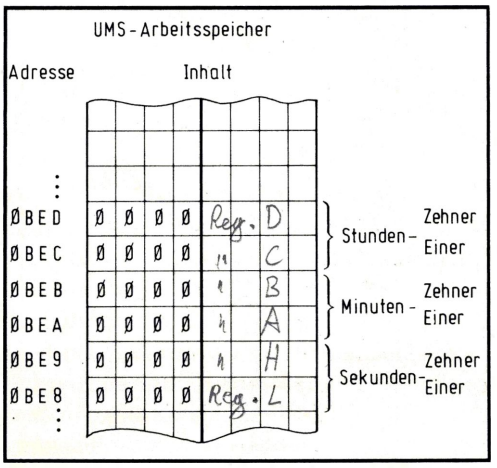


Bild 2: Sechs RAM-Zellen aus dem Arbeitsspeicher sind für die Sekunden-, Minuten- und Stunden-Zähler reserviert.

Drei Taten für das Programm

Das Digitaluhr-Programm DIGUR muß prinzipiell drei Aufgaben übernehmen (**Bild 3**):
1. Hochzählen des Uhrzeit-Buffers unter Berücksichtigung des jeweiligen Übertragungswertes, d. h. die nächsthöhere Stelle

wird nur dann weitergezählt, wenn bei der vorhergehenden ein Überlauf aufgetreten ist; 2. Umsetzen der BCD-Daten des Uhrzeit-Buffers in den Siebensegmentcode und 3. Überschreiben dieses Codes in die Anzeige sowie Erzeugen eines definierten Zeittaktes, um das Hochzählen im Sekundenrhythmus sicherzustellen.

Der Mops macht Mathematik

Das Unterprogramm UHR ist ein Musterbeispiel für eine universelle Struktur (**Bild 4**). Es ruft mehrfach nacheinander ein weiteres Zählprogramm CNT59 bzw. COUNT auf, das jeweils ein Paar BCD-Digits (also Sekunden oder Minuten oder Stunden) als Einheit behandelt. Dazu werden die beiden Digits in eine Binärzahl umgesetzt (weiteres Unterprogramm BCBIN), die vom nächsten Unterprogramm INCRM ganz einfach erhöht wird. Jetzt vergleicht INCRM die so entstandene Binärzahl mit dem Übertragswert, der vor dem Einsprung nach COUNT gesetzt worden ist; sind hochgezählter und Übertragswert gleich (z. B. Sekunden =3C≙60D), wird die Binärzahl auf Null gesetzt, andernfalls bleibt sie erhalten. Nach dieser Prozedur (die sich im Mikrosekundenbereich abspielt) erzeugt INCRM aus der Binärzahl wieder zwei BCD-Digits, die zurück an ihren Platz im Uhrzeit-Buffer transportiert werden. Schließlich signalisiert INCRM dem übergeordneten Programm noch, ob ein Übertrag vorgelegen hat; denn dann (und nur dann) muß auch das nächsthöhere Digit-Paar noch hochgezählt werden.
Das alles hört sich sehr dramatisch an, läßt sich aber auf folgende einfache Formel bringen: Um die verschiedenen Übertragungswerte, die bei den Stunden sogar noch wechseln (13 auf 14 Uhr, aber 23 auf 00 Uhr!), erfassen zu können, ist es programmtechnisch besonders einfach und elegant, Sekunden, Minuten und Stunden jeweils als Einheit (= *eine* Binärzahl) zu betrachten und diese zu modifizieren. Dazu sind zwei zusätzliche Programmteile zur BCD/Binär- bzw. Binär/BCD-Umsetzung erforderlich, die aber gegenüber dem kleckerweisen Hochzählen einzelner Digits mit umständlicher Abfragerei geradezu lächerlich wirken. Der Vorteil dieser Struktur wird noch deutlicher, wenn nach den Stunden auch noch Wochen- und Kalendertag, Monat und Jahr hochgezählt werden sollen. Im computergesteuerten Uhrenanlagen (aus denen diese Programmteile stammen) müssen derartige Randbedingungen sehr sorgfältig berücksichtigt werden, denn Sie können sich die

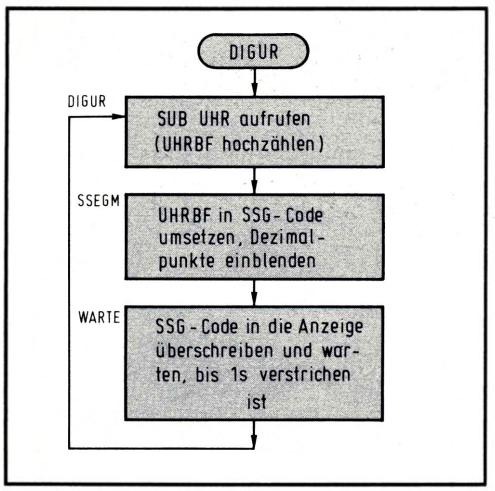


Bild 3: Das Digitaluhr-Programm besteht aus drei Aktivitäten (vgl. Bilder 4, 8 und 9).

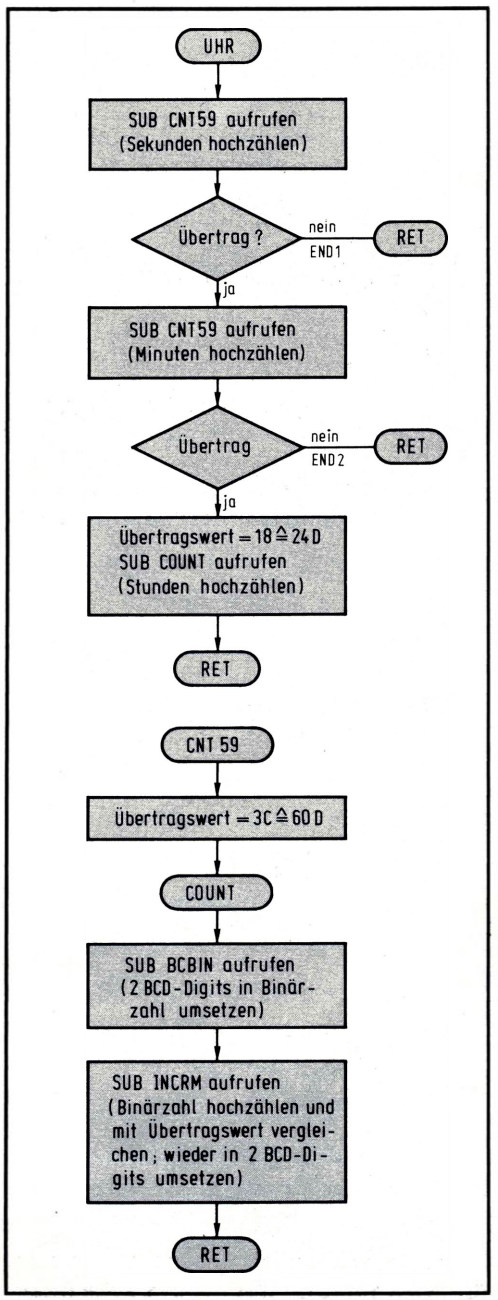


Bild 4: Unterprogramm UHR aktiviert ein weiteres Zähl-Unterprogramm CNT59 bzw. COUNT.

doofen Gesichter vorstellen, wenn eine – noch dazu computerisierte! – Uhr den 30. Februar anzeigt...

Binär geht's leichter

Das Problem der Datenumsetzung von einem Format ins andere tritt beim Programmieren häufig auf; Sie sollen daher die prinzipielle Vorgehensweise kennenlernen, die Sie gern überspringen können, wenn Sie das im Augenblick nicht interessiert. Sobald Sie aber selbst einmal mit dem Problem konfrontiert werden, erinnern Sie sich an *diesen* Abschnitt, in dem steht, wie's gemacht wird.
Das Unterprogramm BCBIN wandelt ein BCD-Paar (Zehner plus Einer) in die entsprechende Binärzahl um, d. h. aus 01/05 (=15 D) wird demnach 0F (**Bild 5**). Beim Einsprung steht in rp H&L die Adresse des BCD-Digits mit der *Zehner*-Wertigkeit. Das Löschen des Übertragsflags ist eine Vorbereitung für die spätere Rückwandlung (vgl. Bild 6). Die eigentliche Binärumsetzung geschieht nach folgendem Schema: Zehner-Digit holen und so lange (dezimal) Zehn abziehen, wie der Rest dabei nicht negativ wird; für jeden Subtraktionsvorgang summiert REG A (binär) Zehn (=0A) auf, und nach diesem Vorgang steht in A der Binärwert des BCD-Zehner-Digits. Dazu wurde einfach so oft die (binäre) Zehn (=0A) addiert, wie es das Zehner-Digit vorgab. Zählt man dazu nun noch das Einer-Digit hinzu, ist die Binärumsetzung abgeschlossen. Hätten Sie gedacht, daß eine geschwollene BCD/Binärumsetzung so einfach ist?

Zählen bis zum Überlauf

Das Unterprogramm INCRM zählt zunächst den in REG A stehenden Wert hoch (**Bild 6**); beim Einsprung stehen in REG D der Übertragungswert (also z. B. 3C≙60D für Sekunden und Minuten) und in REG E der Anfangswert, auf den beim Überlauf zurückgesetzt werden soll (00 bei der Uhrzeit, 01 z. B. beim Kalendertag oder Monat).
Nach dem Hochzählen erfolgt der Vergleich mit REG D (Übertragswert), der ohne Auswirkungen bleibt, solange (A) < (D) ist. Bei Erreichen des Übertragungswertes passieren zwei Dinge: Erstens wird A mit dem Anfangswert aus E geladen (= zurückgesetzt) und zweitens wird das Übertragsflag modifiziert; es zeigt dem übergeordneten Programm durch einen Wert ≠ 0 an, daß während des aktuellen Zählvorgangs ein Überlauf aufgetreten ist (vgl. Bild 4). In A steht demnach in jedem Fall die aufdatierte

Binärzahl, und das Übertragungsflag signalisiert zusätzlich, ob die übergeordneten Digits noch weitergezählt werden sollen oder nicht.

Zurück ins BCD-Format

Nahtlos geht es zum Programmteil BINBC über, der die Rückwandlung der in A stehenden Binärzahl in zwei BCD-Digits übernimmt. Das vollzieht sich nach folgendem Schema: A zunächst nach C retten, dann von A (binär) Zehn (=0A) abziehen; das wiederholt sich so oft, wie A dadurch nicht negativ wird, und jede Subtraktion wird in B mitgezählt. War ein Subtraktionsvorgang zu viel, d. h. wurde A dadurch negativ, stehen in B und C die BCD-Digits für Zehner und Einer der ursprünglichen Binärzahl. Artig wandern sie in den UHRBF zurück, womit der Zählvorgang für ein BCD-Digit abgeschlossen ist. Hat ein Überlauf vorgelegen, sorgt das Programm UHR dafür, daß auch das nächste Digit-Paar noch hochgezählt wird. Das alles erscheint fürchterlich kompliziert, ist es aber in Wirklichkeit gar nicht, was Sie spätestens nach dem dritten Durchlesen bestätigen werden.
Das Assembler-Listing dieses Programmteils finden Sie in **Bild 7**. UHR ist Bestandteil des 2-K-Monitors für das UMS-85 (im 2716-EPROM erhältlich) und dementsprechend im Adreßbereich 0653 ff. abgelegt. Wer das Programm manuell (ins RAM) laden will, kann dies natürlich ohne weiteres z. B. ab 0853 tun und muß dazu lediglich das obere Byte der Sprungadressen entsprechend ändern.
UHR allein ist nur ein Unterprogramm, das zwar RAM-Zellen hochzählen, deren Inhalt aber nicht anzeigen kann; ein Versuch, dieses Programm per RUN zu starten, sollte schon allein deshalb unterbleiben, weil es (wenn es im RAM steht) zerstört werden kann (beim ersten RET-Befehl fehlt die richtige Rücksprungadresse).

Der Schritt zur Digitaluhr

Ein Blick zurück auf Bild 3 zeigt, daß der erste (und dickste) Brocken des Digitaluhr-Programms erledigt ist; denn die restlichen beiden Aufgaben der Siebensegment-Umsetzung (SSEGM) und des Wartens (WARTE) können wir recht einfach unter Zuhilfenahme von Monitor-Unterprogrammen lösen.
SSEGM benutzt das Monitor-Unterprogramm SSGCV (#00E2) zur Umsetzung eines BCD-Digits in den äquivalenten Siebenseg-

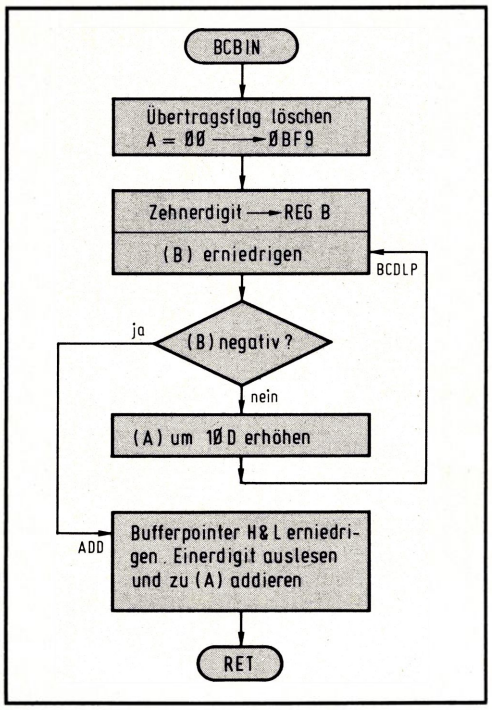


Bild 5: Die Binärumsetzung zweier BCD-Digits erfordert nur diese kurze Programmsequenz.

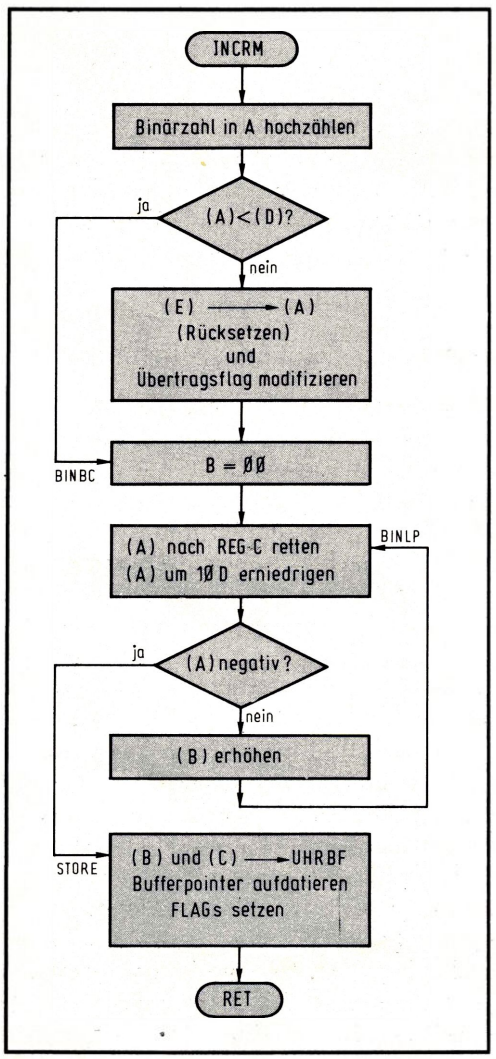


Bild 6: INCRM übernimmt das Hochzählen mit Vergleichen und BINBC die Rückwandlung in zwei BCD-Digits.

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode
653	21	E9	ØB	UHR	LXI H, ØBE9
6	CD	65	Ø6		CALL CNT59
9	C8			END1	RZ
A	CD	65	Ø6		CALL CNT59
D	C8			END2	RZ
E	11	ØØ	18		LXI D, 18ØØ
61	CD	68	Ø6		CALL COUNT
4	C9			END3	RET
665	11	ØØ	3C	CNT59	LXI D, 3CØØ
8	CD	6F	Ø6		CALL BCBIN
B	CD	8Ø	Ø6		CALL INCRM
E	C9				RET
66F	AF			BCBIN	XRA A (CLR A)
7Ø	32	F9	ØB		STA ØBF9
3	46				MOV B, m
74	Ø5			BCDLP	DCR B
5	FA	7D	Ø6		JM ADD
8	C6	ØA			ADI ØA
A	C3	74	Ø6		JMP BCDLP
7D	2B			ADD	DCX H
E	86				ADD m
F	C9				RET
68Ø	3C			INCRM	INR A
1	BA				CMP D
2	DA	89	Ø6		JC BINBC
5	32	F9	ØB		STA ØBF9
8	7B				MOV A, E
9	Ø4			BINBC	INR B
A	4F			BINLP	MOV C, A
B	D6	ØA			SUI ØA
D	FA	94	Ø6		JM STORE
9Ø	Ø4				INR B
1	C3	8A	Ø6		JMP BINLP
94	71			STORE	MOV m, C
5	23				INX H
6	7Ø				MOV m, B
7	23				INX H
8	23				INX H
9	3A	F9	ØB		LDA ØBF9
C	B7				ORA A
69D	C9				RET

Bild 7: Trotz der Arithmetik-Funktionen ist das komplette Maschinenprogramm für UHR nicht sehr lang geworden.

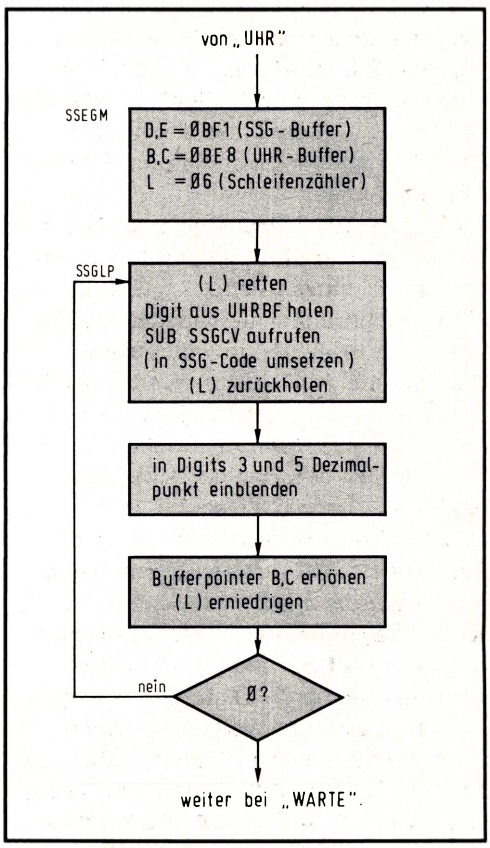


Bild 8: Die Siebensegmentumsetzung der sechs Zähler vollzieht sich über den Aufruf von SSGCV.

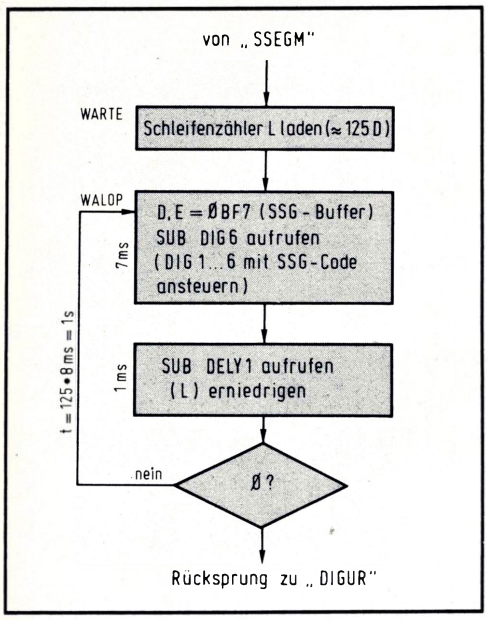
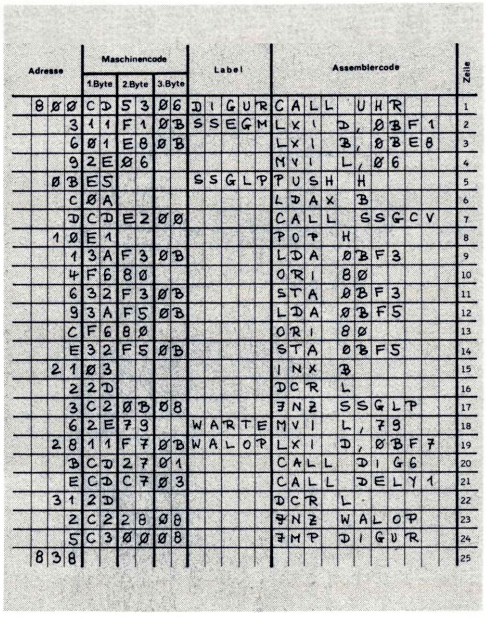


Bild 9: Während der Wartezeit wird das Unterprogramm DIG6 ständig aufgerufen, um die Anzeige dynamisch anzusteuern (softwaremäßige Erzeugung der Multiplex-Frequenz).



Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
ØØØ	CD	53	Ø6	DIGUR	CALL UHR	1
	31	F1	ØB	SSEGM	LXI D,ØBF1	2
	Ø1	E8	ØB		LXI B,ØBE8	3
	2E	Ø6			MVI L,Ø6	4
ØB	E5			SSGLP	PUSH H	5
	ØA				LDAX B	6
	CD	E2	ØØ		CALL SSGCV	7
1Ø	E1				POP H	8
	3A	F3	ØB		LDA ØBF3	9
	F6	8Ø			ORI 8Ø	10
	32	F3	ØB		STA ØBF3	11
	3A	F5	ØB		LDA ØBF5	12
	F6	8Ø			ORI 8Ø	13
	32	F5	ØB		STA ØBF5	14
21	Ø3				INX B	15
	2D				DCR L	16
	C2	ØB	Ø8		JNZ SSGLP	17
	2E	79		WARTE	MVI L,79	18
28	11	F7	ØB	WALOP	LXI D,ØBF7	19
	CD	27	Ø1		CALL DIG6	20
	CD	C7	Ø3		CALL DELY1	21
31	2D				DCR L	22
	C2	28	Ø8		JNZ WALOP	23
	C3	ØØ	Ø8		JMP DIGUR	24
838						25

Bild 10: Das Maschinenprogramm für DIGUR (vgl. Bild 3).

mentcode (**Bild 8**). Dies vollzieht sich über eine Tabelle, ähnlich dem Musikprogramm im 2. Teil bzw. dem Schaltautomaten im 3. Teil dieser Reihe, wobei der Schleifenzähler REG L dafür sorgt, daß SSGCV auch sechsmal durchlaufen wird (pro Digit einmal). Das Einblenden der Dezimalpunkte zwischen Stunden/Minuten und Minuten/Sekunden (im 5. bzw. 3. Digit) erfolgt nach der Generierung des Siebensegmentcodes durch Setzen des MSB (OR-Immediate-Befehl mit „80").
Bleibt übrig, den Siebensegmentcode in die Anzeige zu überschreiben und 1 s zu warten, ehe der Uhrzeit-Buffer erneut hochgezählt wird (**Bild 9**). Unterprogramm DIG6 (#0127 im 2-K-Monitor) transportiert den Siebensegmentcode der sechs Uhrzeit-Digits aus dem Buffer 0BE1...0BE6 in die Anzeige; es hat eine Laufzeit von 7 ms, die wir durch den anschließenden Aufruf von DELY1 (= 1 ms) auf 8 ms erhöhen. Kleiden wir diese Sequenz nun in eine Schleife ein, die 125mal durchlaufen wird, erhalten wir gleichzeitig die dynamische Ansteuerung der Anzeige und die gewünschte Taktzeit von 1 s. In **Bild 10** sehen Sie das Maschinenprogramm für die Digitaluhr, die vor dem Start „gestellt" werden muß, indem Sie die gewünschte Zeit nach 0BE8...0BED eingeben (z. B. ADR – 0BE8 – DAT – 09 – NXT – 04 – NXT – 02 – NXT – 05 – NXT – 03 – NXT – 02 – NXT für 23.52.49 Uhr; Sekunden zuerst eingeben!). Mit **ADR (!)** und RUN starten Sie Ihre Uhr und können beglückt feststellen, daß sich hinter dem simplen „Ticken" ein erklecklicher Programmaufwand verbirgt, den Sie selbst realisiert haben (**Bild 11**). **Achtung!** Wenn Sie UHR ins RAM geladen haben, muß der Unterprogramm-Aufruf CALL UHR in 0800 ff. auch dorthin springen!

Wollen Sie einen Computer-Wecker?

Wenn Sie nach Durchlaufen von WARTE nicht sofort zum Programmanfang zurückspringen, sondern ein zusätzliches Vergleichsprogramm einfügen, können Sie die aktuelle Zeit im UHRBF mit einer voreingestellten Weckzeit (in einem weiteren Buffer, den Sie definieren) vergleichen, um bei Übereinstimmung beispielsweise „Wilhelm Tell" aufzurufen (vgl. 2. Teil) oder über ein Ausgabe-Bit ein Relais mit angeschlossenem Radio zu aktivieren. Auf diese Weise entsteht ein Computer-Wecker, der bei Bedarf z. B. auch acht verschiedene Weck-Buffer überwacht, von denen jeder einem separaten Ausgangs-Bit zugeordnet ist. So könnte man bei Erreichen einer gestellten Zeit Musik ertönen lassen, bei einer anderen Licht einschalten und bei einer dritten schließlich die Kaffeemaschine aktivieren; ganz raffiniert wird es, wenn Sie vor einem ungeliebten Besuch eine Telefonklingel programmieren, um zur vorbestimmten Zeit urplötzlich abgerufen zu werden. Genausogut können Sie eine Stoppuhr aufbauen (SUB UHR nur bedingt aufrufen) oder die Auflösung auf ⅒ s bzw. 1/100 s erhöhen – wir sind sicher, daß Ihnen dazu noch eine Menge anderer Beispiele einfallen wird!
Achtung, Genauigkeitsfanatiker! Der 1-Hz-Takt für diese Uhr wird rein softwaremäßig durch Programmlaufzeiten erzeugt, was naturgemäß nur annäherungsweise gelingt. Dementsprechend groß sind die Gangabweichungen dieser „Uhr", was wir aber gelassen hinnehmen; denn vereinbarungsgemäß ist dieses Beispiel nur die Grundlage weiterführender Programmtechniken, die zielstrebig zur sprechenden (und genauer gehenden) Uhr führen sollen.
Zum Abschluß noch ein Hinweis: Das komplette Digitaluhr-Programm ist im großen 2-K-Monitor enthalten und kann dort wie folgt aktiviert werden: Laden der Zeit nach 0BFB...0BFE (beginnend mit Stunden-Zehner; Minuten-Einer nach 0BFE, Sekunden werden automatisch auf 00 gesetzt), Funktionstaste FCT drücken, gefolgt von 6 – NXT, und schon läuft die Uhr los. Das Programm im EPROM sortiert vor dem Erreichen von DIGUHR die eingegebene Zeit (aus 0BFB ff. nach 0BE8 ff), ist also etwas länger als das hier abgedruckte.

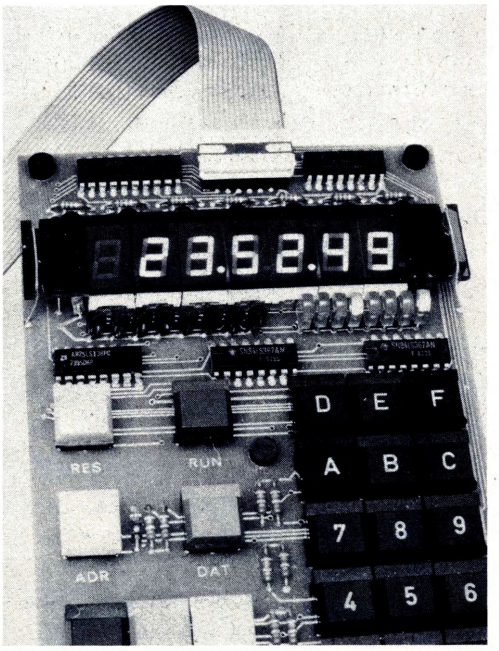


Bild 11: Die Zeit erscheint in der Anzeige und kann bei Bedarf auch Stoppuhr-Funktionen oder höhere Auflösungen liefern.

Die sprechende Computeruhr

Daß Ihr Mikrocomputer als automatisches Schaltwerk, Digitalvoltmeter oder gar Musikbox arbeiten kann, wußten Sie längst und haben dies sicher anhand unserer Einführungsreihen auch schon praktisch ausprobiert. Aber können Sie sich vorstellen, daß Sie Ihren Mops auch dazu bringen, richtig zu sprechen? Wir meinen damit nicht etwa das langweilige Vorspielen einer Tonbandpassage, sondern die wohlartikulierte und dem Menschen verständliche Erzeugung von Wörtern und ganzen Sätzen. Wenn Sie gleich erfahren, wie das im einzelnen vor sich geht, dann sollten Sie sich diesen Entwicklungsstand einmal deutlich vor Augen führen: Es ist tatsächlich soweit, daß die Maschine *sprechen*, sich uns klar und deutlich verständlich machen kann; der erste Schritt zur regelrechten Unterhaltung mit dem Computer ist damit getan, und die Frage nach dem Sinn erhält eine Fülle von Antworten; denken Sie z. B. nur daran, daß diese Form der Kommunikation blinden Menschen in kaum vorstellbarer Weise hilft, und für einen ohnehin überlasteten Autofahrer würde die Sprach-Ein- und -Ausgabe eine wesentliche Entlastung bei der Bedienung des Autos bedeuten. Im Augenblick befinden wir uns auf diesem Sektor noch im Anfangsstadium, das es uns ermöglicht, den Computer erst einmal die Uhrzeit ansagen zu lassen (**Bild 1**).

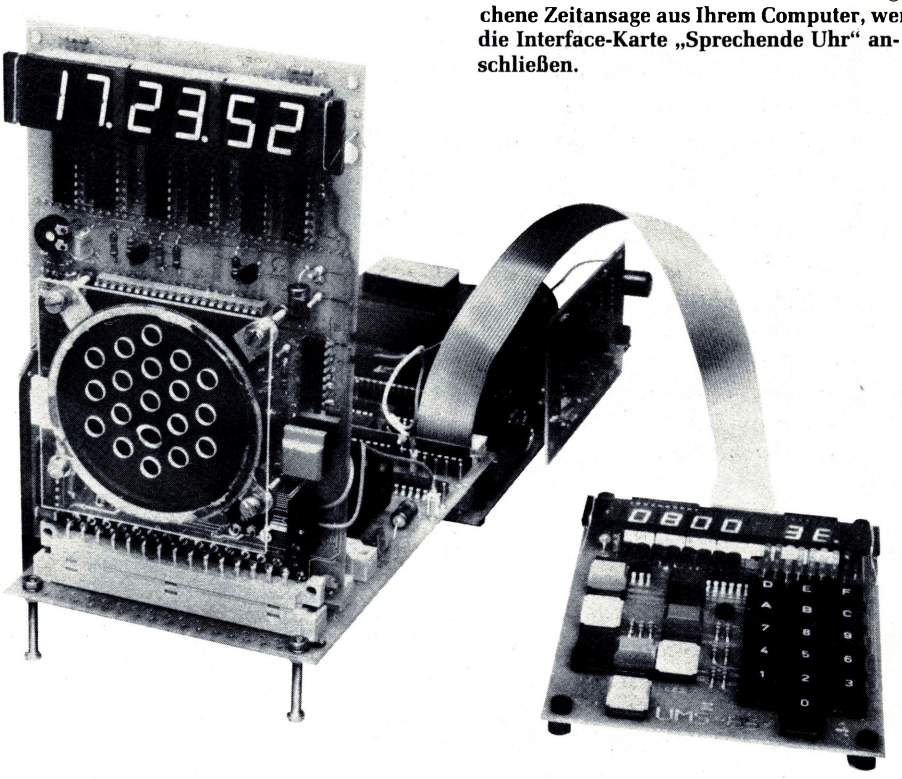


Bild 1: Laut und vernehmlich ertönt die gesprochene Zeitansage aus Ihrem Computer, wenn Sie die Interface-Karte „Sprechende Uhr" anschließen.

Von außen ins Programm gelangt

Das Problem umfaßt zwei Aufgabenbereiche: Im ersten gilt es, die Uhrfunktion bereitzustellen, d. h. für ein zeit- und übertragsrichtiges Weiterzählen zu sorgen; dies soll Software-orientiert erfolgen, ähnlich dem Beispiel mit der Digitaluhr im vorigen Beitrag dieser Reihe. Die zweite Aufgabe besteht darin, die in Form von losen Bits herumliegende Uhrzeit in Sprache umzusetzen und zu Gehör zu bringen; dies erfolgt Hardware-orientiert auf einer eigenen Interface-Baugruppe, die einen hochintegrierten Sprachgenerator beinhaltet.

Beginnen wir mit der Uhrfunktion, die wir prinzipiell bereits gelöst haben. Dieses Beispiel hatte allerdings einen kleinen Schönheitsfehler, weil die dort vorgestellte Lösung zu recht großen Gangabweichungen der Uhr führt. Das resultiert ganz einfach daraus, daß dort der Zeittakt aus Software-Zeitverzögerungen abgeleitet wird, was naturgemäß nur näherungsweise gelingt. Und da eine Uhr – anders als sonstige Meßgeräte wie z. B. ein Voltmeter – ihren Gangfehler summiert, entsteht eine Abweichung, die mit der Zeit immer größer wird. Sie sollen darum einen (noch dazu hocheleganten!) Weg kennenlernen, dieses Manko auszubügeln. Man schafft dies über externe Programmunterbrechungen, sogenannte **Interrupts** („Interrappts"), bei denen sich folgendes abspielt: Der Prozessor arbeitet in aller Seelenruhe irgendein Programm ab, das überhaupt nichts mit der Uhr zu tun haben muß. In festen Zeitabständen, die von einem Quarztakt abgeleitet sind, wird die CPU von außen „angestoßen", d. h. es wird per TTL-Impuls der Interrupt-Eingang

aktiviert. Resultierend daraus unterbricht die CPU das laufende Programm, springt zu einem dafür vorgesehenen Programmteil (der Interrupt-Service-Routine, ISR) und führt diejenige Aktivität aus, die wir für den Fall eines Interrupts programmiert haben. Im Beispiel der Uhr wäre das das zeitrichtige Weiterzählen eines Uhrzeit-Buffers. Unmittelbar danach geht der Prozessor artig zurück an diejenige Programmstelle, die er bei Eintreffen des Interrupts verlassen hatte,

Ablenkung für den Computer

Sie müssen sich den Interrupt für den Computer genauso vorstellen wie einen Telefonanruf, der einen bei der laufenden Büroarbeit (kurzzeitig) unterbricht. Natürlich müssen dabei gewisse Grenzen gewahrt bleiben, damit das System nicht zusammenbricht: Zwischen den einzelnen Anrufen muß genügend Zeit für die normale Arbeit bleiben, und die Zeit, die einem der Anrufer stiehlt, sollte in einem vernünftigen Verhältnis zur eigentlichen Arbeitszeit stehen.
Der Vorteil dieser Organisation liegt auf der Hand: Während der Prozessor bei der Software-Zeitverzögerung zu 100% mit dieser (noch dazu unfruchtbaren) Tätigkeit beschäftigt ist, geht diese Belastung bis auf wenige Prozent zurück, wenn die kurze Service-Routine nur hin und wieder, z. B. nur jede Sekunde, durchlaufen wird.

Das unbekannte Phänomen

Bei der Flut der angebotenen Mikrocomputer-Literatur finden Sie nur relativ selten etwas über die Interrupt-Technik. Das liegt daran, daß die meisten Leute diese Zusammenhänge nicht verstehen und froh sind, wenn ein Programm endlich läuft; es dann mutwillig wieder unterbrechen hieße, das getane Werk wieder zu vernichten. Es soll hier nicht verschwiegen werden, daß die Interrupt-Technik tatsächlich recht komplex (insbesondere beim 8085) und nicht ohne Probleme handhabbar ist; um uns hier auf das Wesentliche zu konzentrieren, beschränken wir uns auf einige Auszüge, weil sonst mehr als eine ganze ELO voll von Interrupt-Gerede wäre.
Von den zahlreichen Interrupt-Möglichkeiten des 8085 benutzt das Mikrocomputer-System UMS-85 nur eine, nämlich den Interrupt-Eingang RST7.5 (Restart 7,5; Pin 7 der CPU). Liegt an diesem Anschlußstift HIGH-Potential an (ein kurzer Puls genügt), springt der Prozessor zur Adresse 003C; diesen festen Zusammenhang gibt der Halbleiterhersteller vor. In diesem Adreßbereich

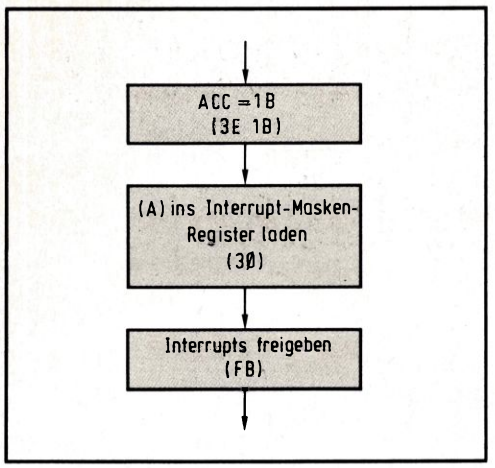


Bild 2: Bevor ein Interrupt akzeptiert werden kann, muß diese Sequenz mindestens einmal durchlaufen werden.

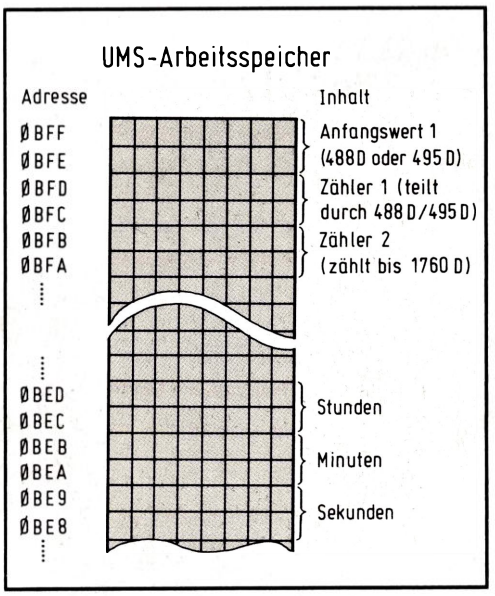


Bild 3: Die softwaremäßige Teilung der eingespeisten Interrupt-Frequenz erfordert noch einmal sechs RAM-Zellen.

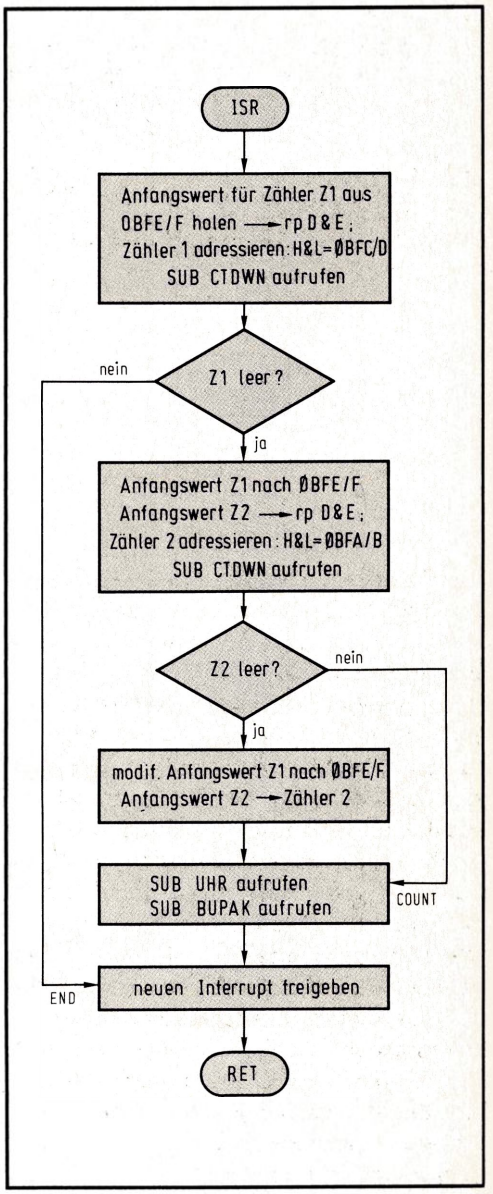


Bild 4: Die Interrupt-Service-Routine sorgt dafür, daß das Hochzählen des Uhrzeit-Buffers nur einmal pro Sekunde erfolgt.

ist, wie Sie wissen, der Monitor angesiedelt, der in 003C ff. einen Sprungbefehl nach 083C programmiert hat; d. h., daß beim UMS-85 ein Interrupt dazu führt, auf dem Umweg über 003C immer nach 083C zu springen (wo die ISR stehen muß), den dort abgelegten Programmteil als Unterprogramm abzuarbeiten und bei Erreichen von RETURN an die zuvor verlassene Stelle zurückzuspringen. Ist die ISR anderswo abgelegt, muß in 083C ein weiterer Sprungbefehl eben dorthin stehen.
Allerdings akzeptiert die CPU Interrupts nur dann, wenn diese nach RESET mindestens einmal initialisiert worden sind, andernfalls bleiben sie ohne jede Auswirkung (**Bild 2**). Dazu wird ein bestimmtes Bitmuster (in unserem Fall „1B") ins Interrupt-Maskenregister geladen (per SIM=30), gefolgt von der Interrupt-Freigabe (*Enable Interrupt*; EI=FB). Der Ladebefehl SIM ist übrigens derselbe, der auch den seriellen CPU-Ausgang SOD modifiziert (vgl. 2. und 6. Teil).

Ein Hardware-Teiler hilft dem Programm

Als eifriger Programmierer haben Sie jetzt bereits weitergedacht und schlagen einen Interrupt-Takt von 1 Hz vor, bei dessen Eintreffen unser Digitaluhr-Programm aktiviert wird. Ist es abgearbeitet (was einige 10 µs dauert), kann sich die CPU mit etwas anderem beschäftigen, bis dann eine Ewigkeit später (knapp 1 s) der nächste Inter-

rupt-Puls eintrifft, und der Uhrzeit-Buffer erneut hochgezählt wird. Sie hören bereits das „Aber" gegen dieses Konzept: Woher wollen Sie den 1-Hz-Takt bekommen? Mit einem separaten Oszillator und Teiler, oder mit endlos vielen Teilerstufen aus dem CPU-Takt abgeleitet? Alles zu aufwendig, es geht auch einfacher. Wenn wir ein einziges Teiler-IC spendieren (beispielsweise den 12stufigen Teiler 4040) und in dieses den 2-MHz-Systemtakt einspeisen, kommt „hinten" (an Q12) schon eine ganz schön niedrige Frequenz heraus, die 488,281 Hz beträgt. Daraus per Teiler, Flipflop und Gatter 1 Hz zu generieren, ist für einen Mikrocomputer-Spezi unter jeglicher Würde. Nehmen wir doch einfach diesen Takt, legen ihn an den CPU-Interrupt-Eingang und überlassen die restliche komplizierte Teilerei dem Programm, genauer gesagt, der Interrupt-Service-Routine.

Diese muß den Uhrzeit-Buffer nicht bei *jedem* Interrupt hochzählen, sondern nur bei jedem 488. (genauer: alle 488,281... Interrupts). Da der Prozessor die Nachkomma-Bruchteile nicht verarbeiten kann, lassen wir ihn getrost durch 488 teilen und fügen gelegentlich (alle 1760 Durchläufe) einen Zyklus ein, bei dem er statt durch 488 durch 983 teilt. Über alles gesehen teilt er damit im Mittel genau durch 488,281... und zählt dadurch den Uhrzeit-Buffer im gewünschten Sekundentakt hoch.

Für diese komplizierte Teilerei reserviert sich die CPU sechs weitere RAM-Zellen für die Teiler durch 488 (bzw. 983) und 1760 sowie für den wechselnden Anfangswert für Teiler 1 (**Bild 3**). Die eigentliche ISR (**Bild 4**) führt in Verbindung mit dem Unterprogramm CTDWN (**Bild 5**) das Teilen durch, indem die Zähler leergezählt und entsprechend neu gesetzt werden. Alle 488 (bzw. 983) Durchläufe erfolgt das Aktivieren der Unterprogramme UHR (vgl. vorigen Beitrag dieser Reihe) und BUPAK, das die Aufgabe übernimmt, den Inhalt des Uhrzeit-Buffers zu packen und in die sprechende Uhr zu überschreiben (**Bild 6**).

Auch diese Programmteile sind Bestandteil des erweiterten 2-K-Monitors für das UMS-85; sie sind dort unter den Adressen abgelegt, die auch in den Assembler-Listings eingetragen sind (**Bild 7...9**). Wenn Sie diese Software in RAM laden wollen, müssen Sie die Sprungadressen entsprechend anpassen.

Auf jeden Fall muß in 083C ff. der Sprungbefehl zur ISR (hier in 0610 ff.) stehen (C3 10 06).

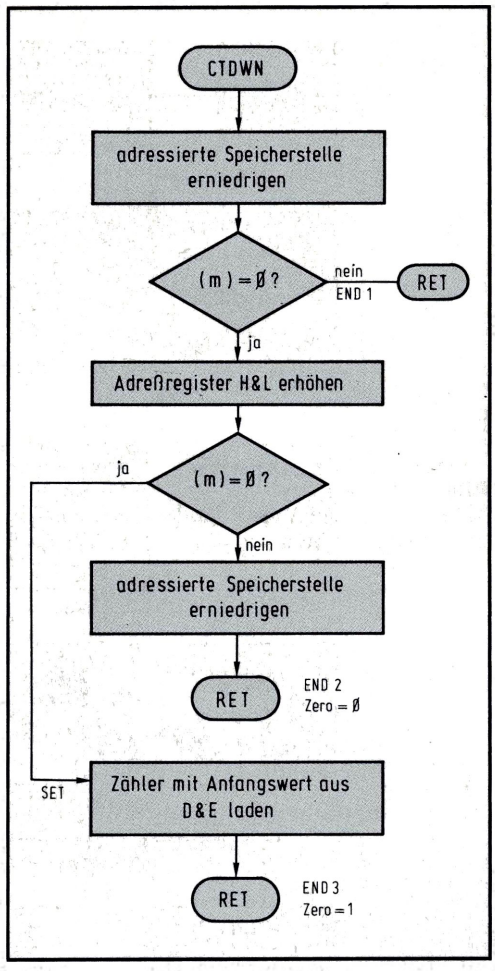


Bild 5: SUB Countdown erniedrigt einen 16-Bit-Zähler (gebildet aus zwei Bytes im RAM) und setzt beim Nullzustand das Zero-FLAG.

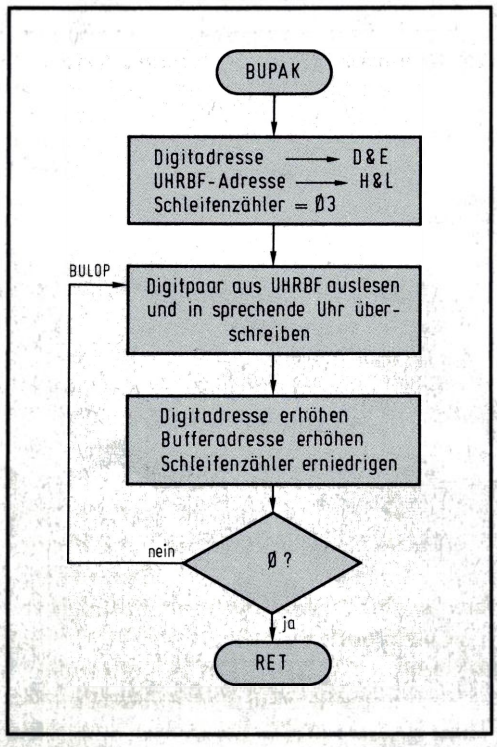


Bild 6: Mit Hilfe dieses Unterprogramms wird die Uhrzeit in die Eingangsspeicher der sprechenden Uhr überschrieben.

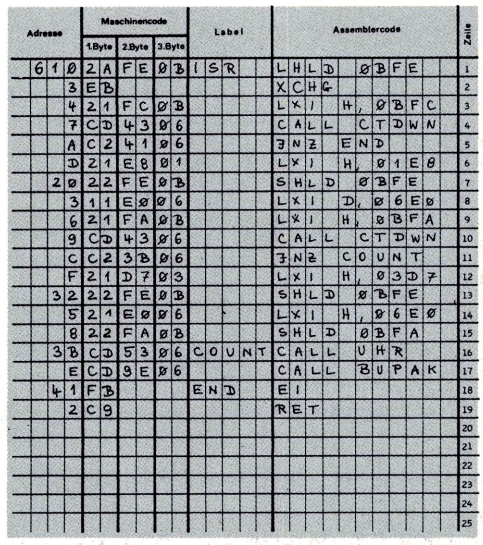


Bild 7: Maschinenprogramm der Interrupt-Service-Routine.

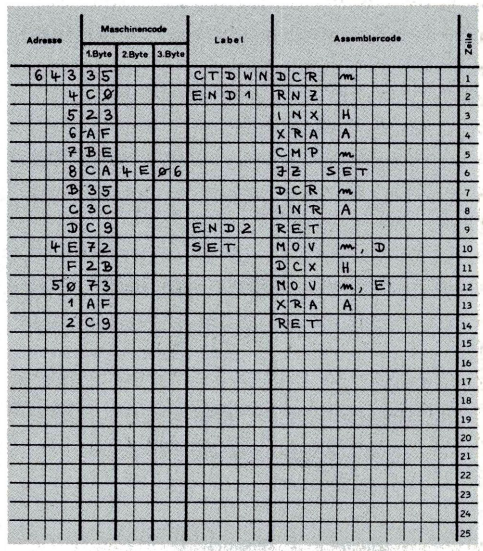


Bild 8: Maschinenprogramm zu Bild 5.

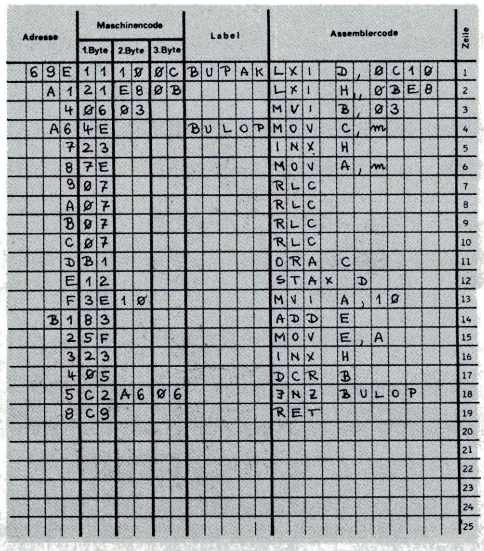


Bild 9: Maschinenprogramm zu Bild 6.

Das Futter für die sprechende Uhr

Die vorgestellte Software ist (in Verbindung mit dem winzigen Hardware-Zusatz-Teiler) in der Lage, den Uhrzeit-Buffer quarzgenau und zeitrichtig hochzuzählen; das Unterprogramm BUPAK packt jeweils Einer- und Zehner-Digit von Sekunden, Minuten und Stunden zusammen in ein Byte und überschreibt dies in die sprechende Uhr, deren Blockschaltung Sie in **Bild 10** finden: Große und helle Siebensegmentanzeigen übernehmen die Darstellung der Zeit; sie werden von ICs vom Typ 9368 angesteuert, von denen jedes einen 4-Bit-Speicher, Decodierer und sieben Stromtreiber besitzt (**Bild 11**). Die Eingangsspeicher nehmen die von BUPAK gelieferten Daten auf, und nach der Decodierung im 9368 werden direkt (und ohne Vorwiderstände!) die einzelnen LED-Segmente angesteuert. Dies erfolgt statisch, also nicht im Multiplex-Betrieb, weil der nachfolgende Sprachgenerator die Eingangsinformationen so braucht: deshalb können wir hierfür auch nicht die UMS-Anzeige benutzen.

Dieser ITT-Sprachgenerator UAA1003 gehört zu den komplexesten ICs, die heute überhaupt hergestellt werden (**Bild 12**). Vereinfacht dargestellt besitzt er einen internen Festwertspeicher (ROM), der eine Fülle winziger, digitalisierter Amplitudenwerte der menschlichen Sprache enthält. Geeignet zusammengesetzt und einem Digital/Analog-Umsetzer zugeführt entsteht ein Signalverlauf, der der menschlichen Sprache sehr ähnlich ist. Die Grundlagen der

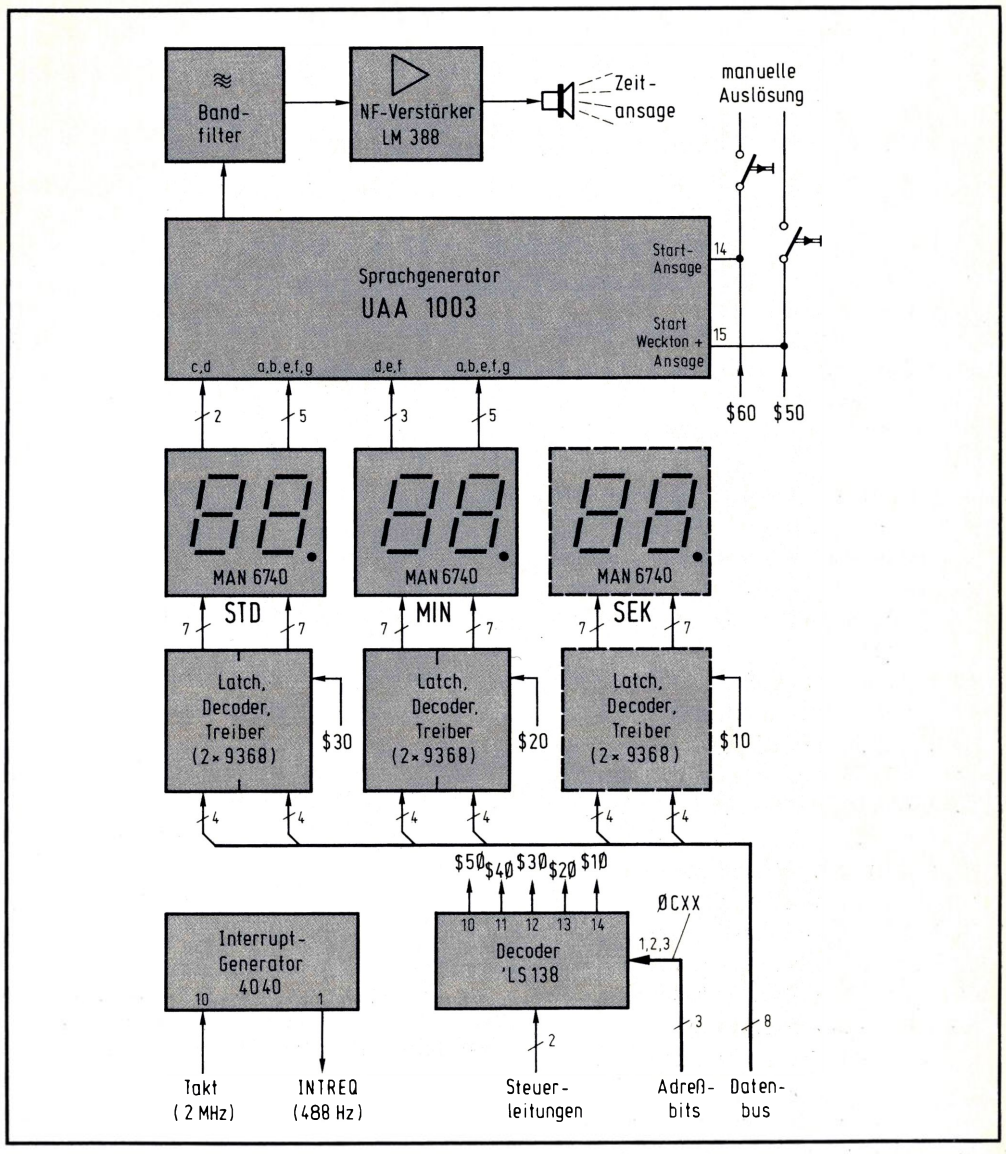
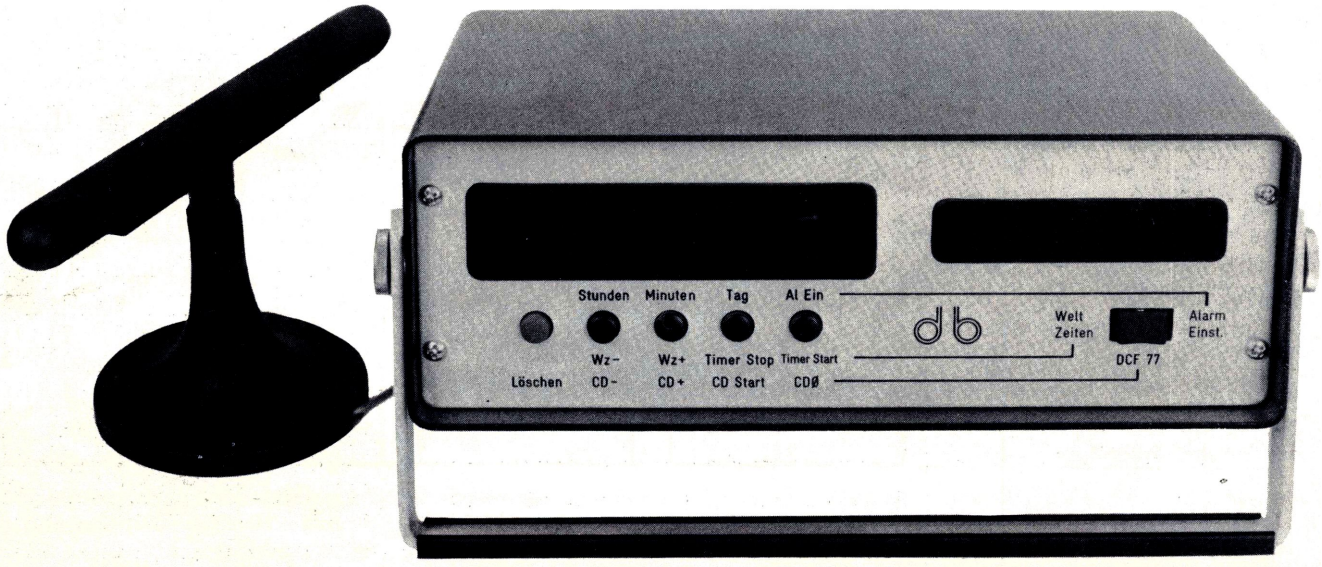


Bild 10: Blockschaltung der Interface-Karte „Sprechende Uhr".



A/D- und D/A -Umsetzung haben wir ausführlich im 4. Teil dieser Reihe vorgestellt (vgl. Seite 19), während es den Rahmen bei weitem sprengen würde, auf die Sprach-Analyse und -Synthese detailliert einzugehen.

Sprache aus der Retorte

Wir wollen uns mit der Beschaltung des ICs begnügen, die folgendermaßen aussieht (**Bild 13**): An den 15 Signaleingängen liegen die Zustände der Anzeigen-Elemente an (nur Stunden und Minuten, da Sekunden nicht mit angesagt werden), aus denen die Uhrzeit eindeutig erkennbar ist. P1 dient zur Einstellung der internen Taktfrequenz, die unmittelbar die Tonhöhe der erzeugten Sprache bestimmt; sie reicht von Micky-Maus-Sprache über den offiziellen Ton ei-

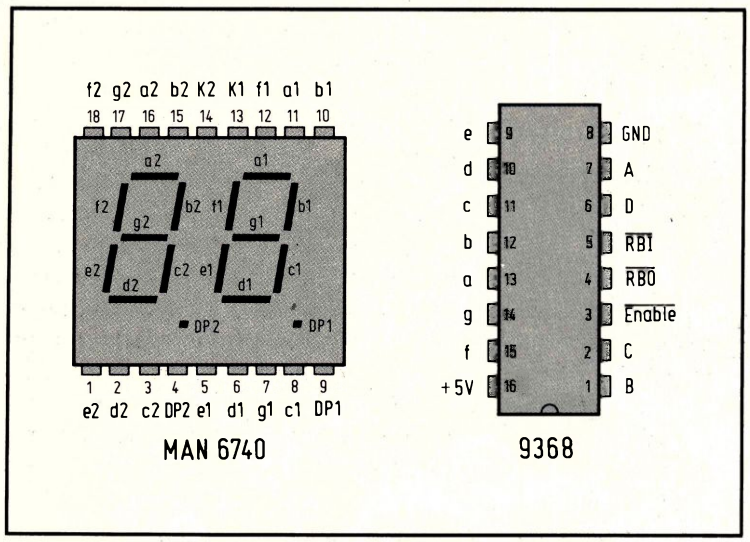


Bild 11: Der Decoder 9368 besitzt einen 4-Bit-Eingangsspeicher; seine Ausgänge treiben die Segmente direkt ohne Vorwiderstände.

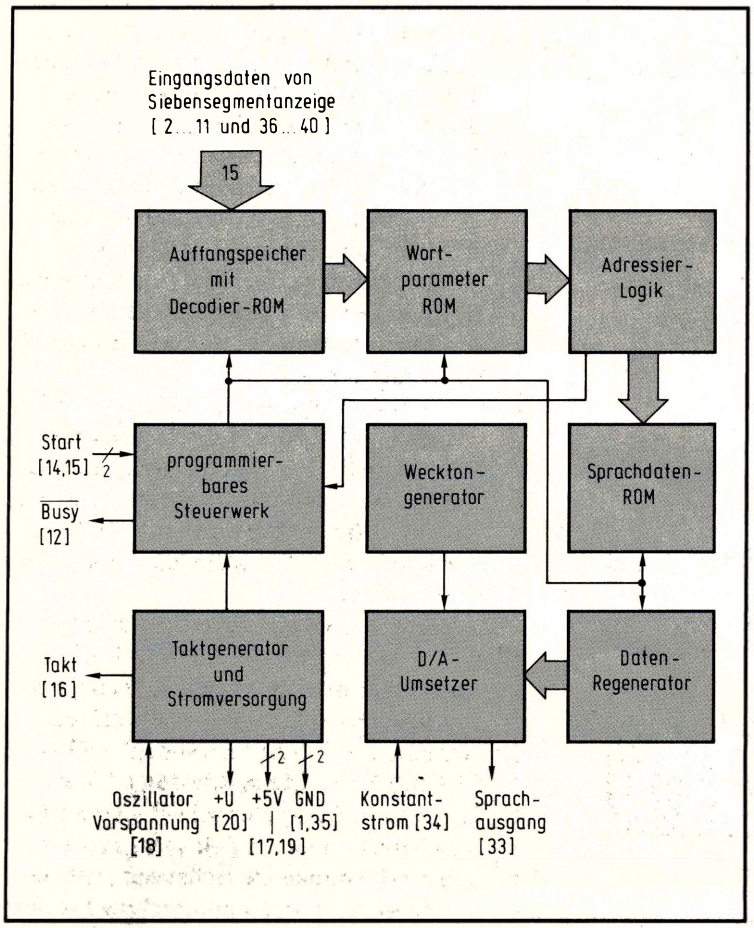


Bild 12: Interner Aufbau des Sprachgenerators UAA 1003.

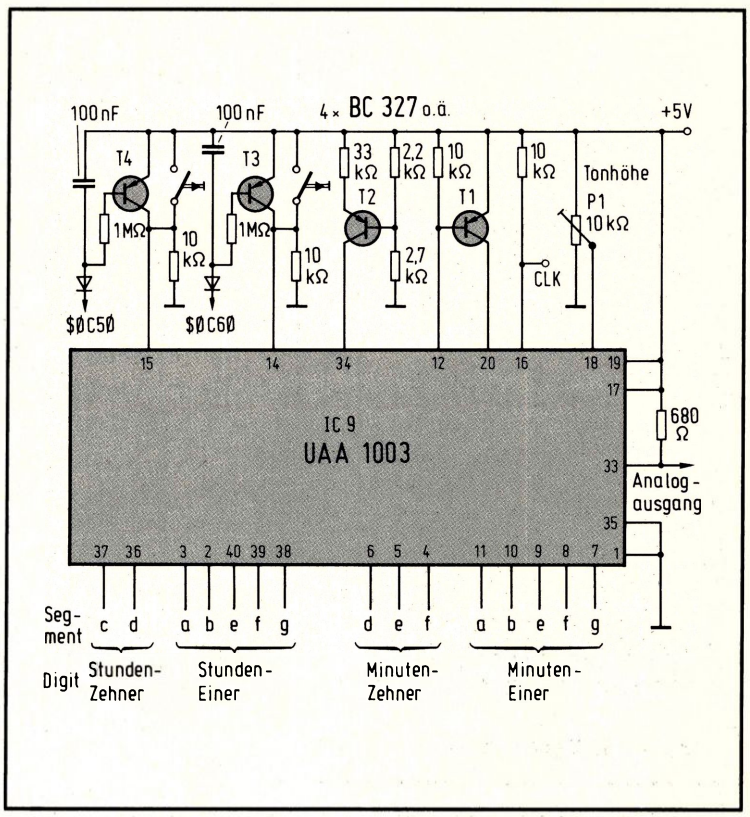


Bild 13: Beschaltung des Sprachgenerators, dessen analoges Ausgangssignal noch gefiltert und verstärkt werden muß.

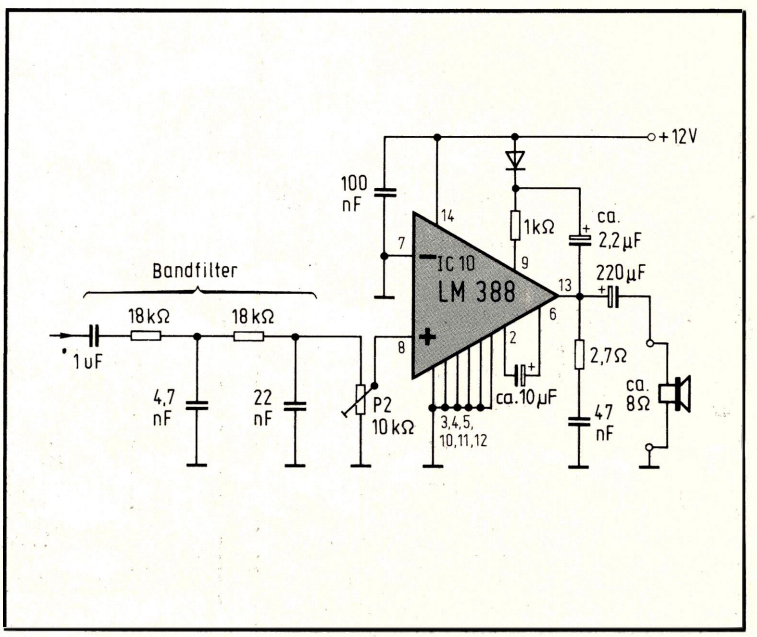


Bild 14: Ein integrierter NF-Verstärker sorgt dafür, daß die Zeitansage laut genug ertönt.

nes Tagesschau-Sprechers bis zu abgrundtiefem Gequake. T3 und T4 können vom Programm aktiviert werden (durch Ausgabe der entsprechenden Adresse 0C50 bzw. 0C60, vgl. Bild 18), um die Ansage auszulösen; das kann manuell auch über die parallelliegenden Taster erfolgen. Der T4-Eingang startet nur die Zeitansage, während beim Auslösen des anderen Eingangs ein markerschütternder Weckton vorausgeht, um beispielsweise auch die letzte Schlafmütze beim Feierabend zu wecken.

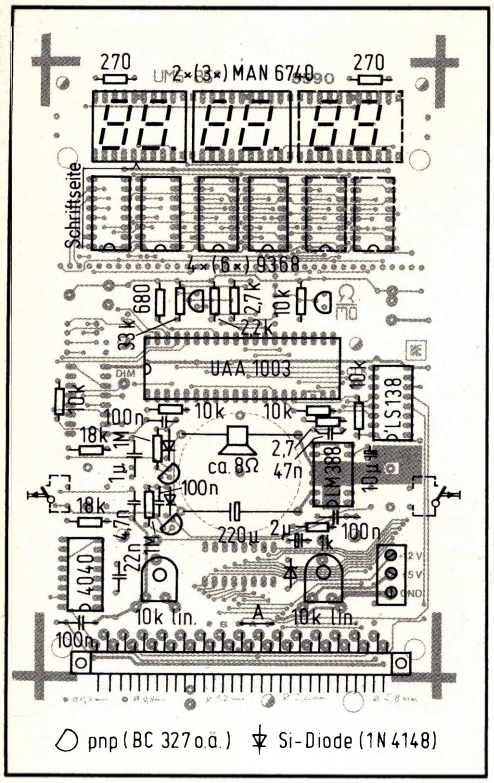


Bild 15: Bestückungsplan der sprechenden Uhr.

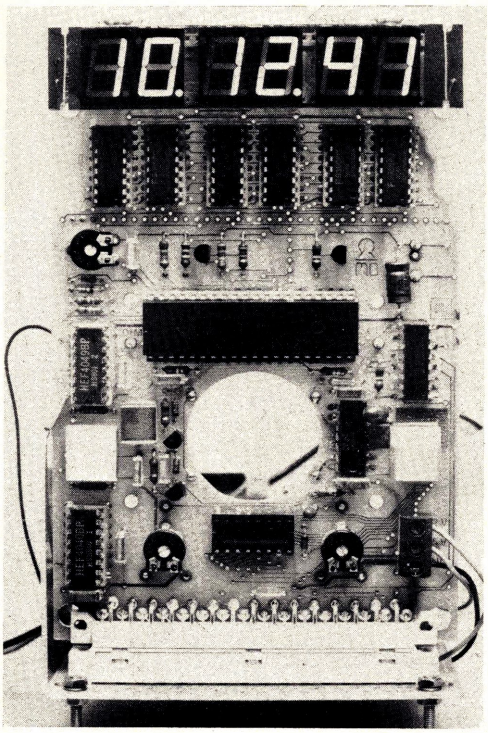


Bild 16: Die fertig bestückte Baugruppe; in der Mitte kann ein Loch herausgesägt werden, um einen Lautsprecher zentral zu befestigen.

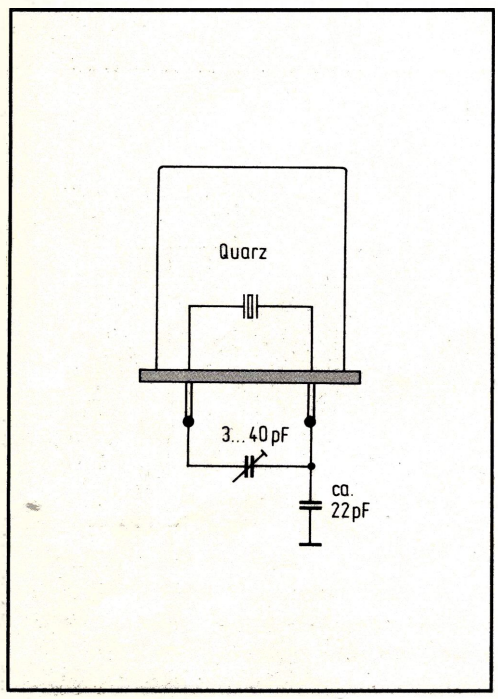


Bild 17: Mit dieser Kondensator-Beschaltung läßt sich der Quarzoszillator des 8085 exakt abgleichen.

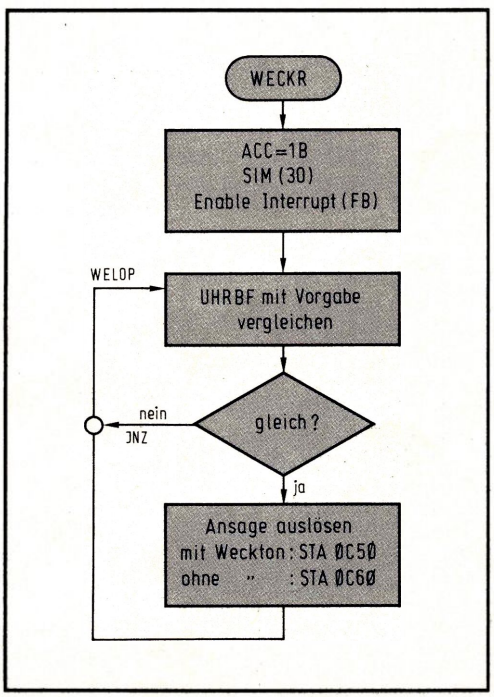


Bild 18: Programmvorschlag für einen Wecker, der die Weckzeit ansagen kann.

Den letzten Schliff erhält die Sprache in einem Bandfilter, das die Sprachcharakteristik bestimmt (**Bild 14**); der nachfolgende NF-Verstärker sorgt für eine Lautstärke, die mit Sicherheit wohnungsdurchdringend ist.

Und nun werden Sie fragen, wie sich das eigentlich anhört, wenn der Computer „spricht". Es klingt absolut natürlich, wenn er beispielweise sagt: „Es ist siebzehn Uhr dreiundzwanzig" Die Ansage erfolgt im wahrsten Sinne monoton, d. h. ohne Sprachmelodie und Betonung, die der natürlichen Sprache Klang und Ausdruckskraft verleihen. Aber die wohlgesittete Artikulation durch den Computer hat nichts Unsympathisches an sich, im Gegenteil: Gernot, der Fotograf, stellte bei der ersten Konfrontation verblüfft fest: „Der spricht ja genauso wie ich"! – Der ganz leichte schwäbische Dialekt, den die Freiburger Entwickler des ICs mit hineingebracht haben sollen, ist allerdings vom ungeübten Ohr nicht herauszuhören...

Kein stummer Diener mehr

Wenn Sie sich an den Nachbau machen wollen, dürfte dies mit dem Bestückungsplan in **Bild 15** keine Schwierigkeiten bereiten. In der Grundausbaustufe enthält der angebotene Bausatz nur die Minuten- und Stundenanzeigen; die fertige Baugruppe (**Bild 16**) kann in eine Buchsenleiste des UMS-85 (auf der CPU oder der Bus- und Speichererweiterung) eingesteckt werden, aber natürlich läßt sie sich auch an jeden anderen Computer mit 8-Bit-Parallel-Schnittstelle anschließen. Zur Stromversorgung sind +5 V/650 mA (bei sechs Anzeigen) und +12 V/ca. 50 mA (je nach Lautstärke) anzuschließen.

Die Ganggenauigkeit hängt nun unmittelbar vom Quarz ab. Genauigkeitsfanatiker können die Oszillatorfrequenz direkt am 8085 abgleichen, indem sie die Beschaltung nach **Bild 17** ergänzen. Am Testpunkt CLK müssen dann exakt 2,000 MHz (bzw. 500 ns) zu messen sein.

Sie haben nun die Möglichkeit, in einem Hauptprogramm den Uhrzeit-Buffer beispielsweise mit einer eingegebenen Weckzeit zu vergleichen, um beim Erreichen die Ansage auszulösen (dazu genügt der Befehl STA 0C50 bzw. 0C60; **Bild 18**); oder Sie lassen Ihren Computer bei jeder vollen Minute bzw. Stunde die Zeit ansagen – auf jeden Fall können Sie sicher sein, daß Sie mit dieser tönenden Zeitmaschine Freunde und Bekannte frappieren, und das Ganze ist mit Sicherheit **der** Knüller für Ihre nächste Party!

Trickreiches Thermometer

Praktische Einsatz- und Anwendungsbeispiele für Ihren Mikrocomputer vermittelt Ihnen diese Reihe, und Sie lernen heute eine Anwendung kennen, der gerade im Rahmen der Energie-Einsparung eine große Bedeutung zukommt: Die Temperaturmessung per Computer (**Bild 1**). Dies ist ein klassisches Beispiel für die Meßwertaufbereitung eines analogen Signals mit anschließender Analog/Digital-Umsetzung und Weiterverarbeitung durch den Computer. Sie benötigen hierzu außer dem Computer das Analog-Interface, das wir ausführlich im 4. Teil vorgestellt haben. Außerdem sind für die Temperaturmessung ein paar zusätzliche Standard-Bauteile erforderlich, die Sie aber sicherlich in Ihrem Bestand vorfinden.

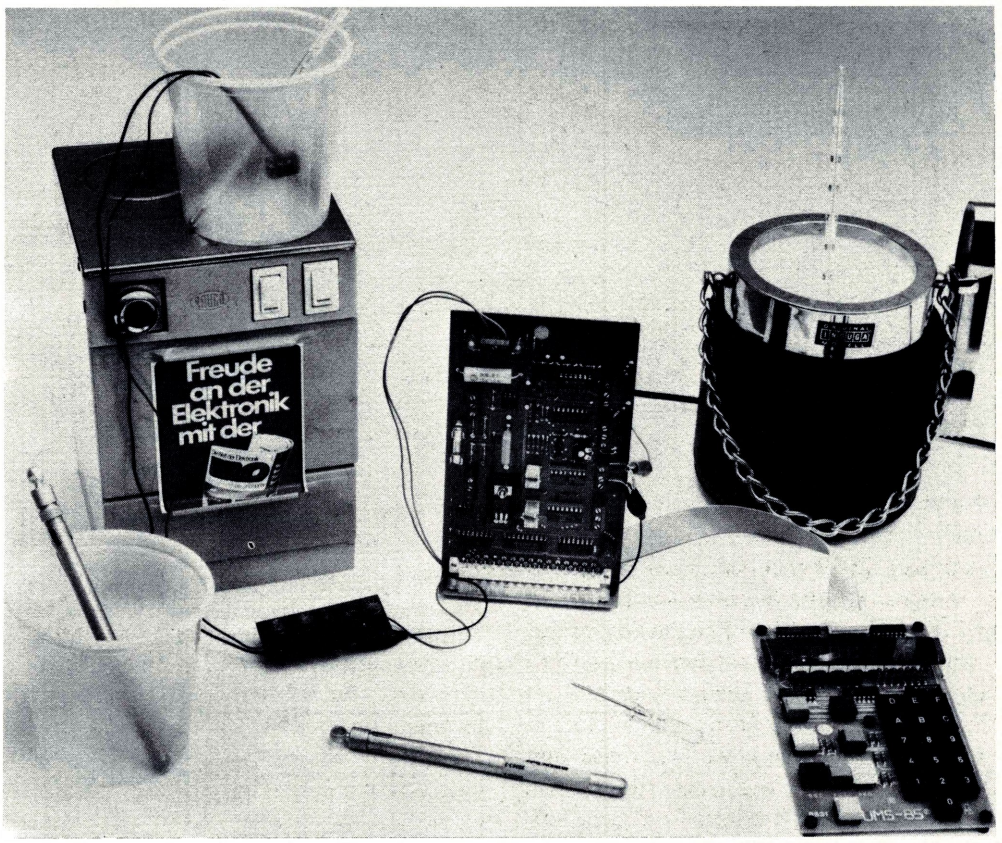


Bild 1: Mit einer simplen Siliziumdiode als Sensor arbeitet das Digitalthermometer, dessen Realisierung dieser Beitrag beschreibt.

Ein Winzling wird zum Sensor

Es bedarf für Sie keiner Erwähnung mehr, daß der Computer ausschließlich digitale Daten annehmen und verarbeiten kann. Bei der Temperaturmessung treten daher zwei grundsätzliche Probleme auf: Erstens wird ein Meßfühler (Sensor) benötigt, der ein von der Temperatur abhängiges Ausgangssignal (möglichst eine Spannung) liefert, und zweitens brauchen wir eine Einheit, die diese analoge Meßgröße digitalisiert (**Bild 2**).

Die zweite Aufgabe übernimmt das Analog-Interface, dessen Funktion und Arbeitsweise wir ausführlich im vierten Teil dieser Reihe beschrieben haben. Und für den Meßfühler selbst gibt es verschiedene Möglichkeiten, die vom Thermo-Element bis hin zum Platin-Widerstand reichen. Die billigste und nicht einmal schlechteste Lösung stellt aber eine simple Siliziumdiode dar, die geradezu wie geschaffen scheint, um als Temperatursensor zu arbeiten. Man macht sich hierbei die (ansonsten nicht gerade wünschenswerte) Eigenschaft der Siliziumdiode zunutze, daß sie ihre Durchlaßspannung mit jedem Grad Temperaturänderung um (ziemlich genau) 2 mV verändert. Verstärkt man diese Gleichspannungsänderungen geeignet, dann kann man sie unmittelbar in das Analog-Interface einspeisen, wo sie weiterverarbeitet und mit Hilfe des Mikrocomputers auch digitalisiert wird.

Sie sehen an diesem Beispiel, welche erstaunlichen Eigenschaften in den sattsam bekannten Standard-Bauteilen stecken können. Sie werden verblüfft sein, wie funktionstüchtig und genau unser Thermometer arbeitet, das eine simple Diode 1N4148 als Temperaturfühler verwendet!

Von eiskalt bis siedendheiß

Vor der eigentlichen Meßwertverarbeitung müssen wir noch einige Randbedingungen klären, zu denen insbesondere die Festlegung des Meßbereichs gehört; wie Sie mittlerweile längst wissen, ist diese genaue Spezifikation der gestellten Aufgabe unerläßlich für eine effektive Vorgehensweise bei der Programmierung.

Unsere Überlegungen beginnen damit, daß das verwendete IC zur Analog/Digital-Umsetzung eine Auflösung von 8 Bit besitzt, womit 256 binäre Werte darstellbar sind.

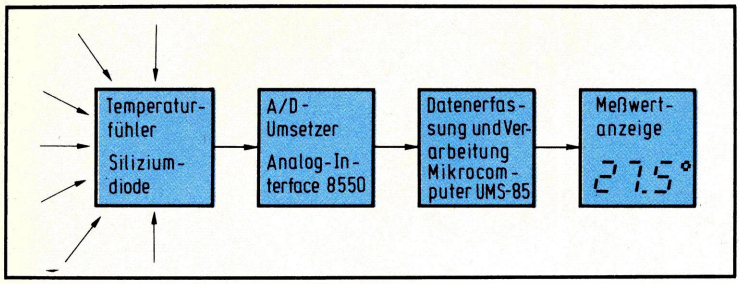


Bild 2: Blockschaltung des Thermometers auf Mikrocomputer-Basis.

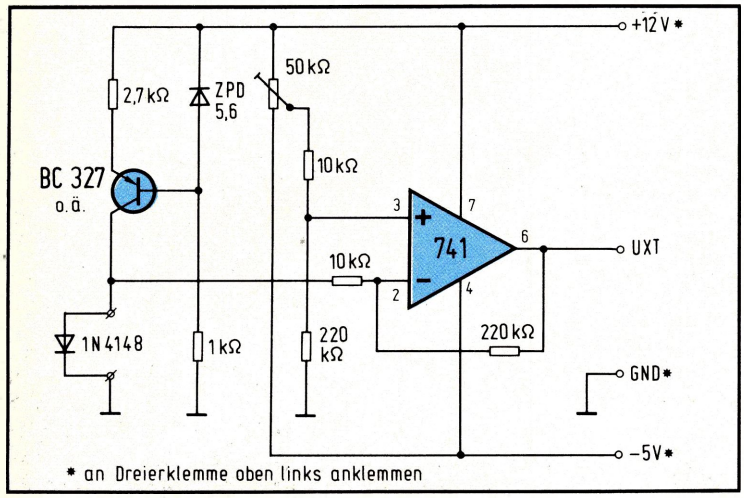


Bild 3: Konstantstromquelle und Operationsverstärker bereiten das von der Diode gelieferte Signal auf.

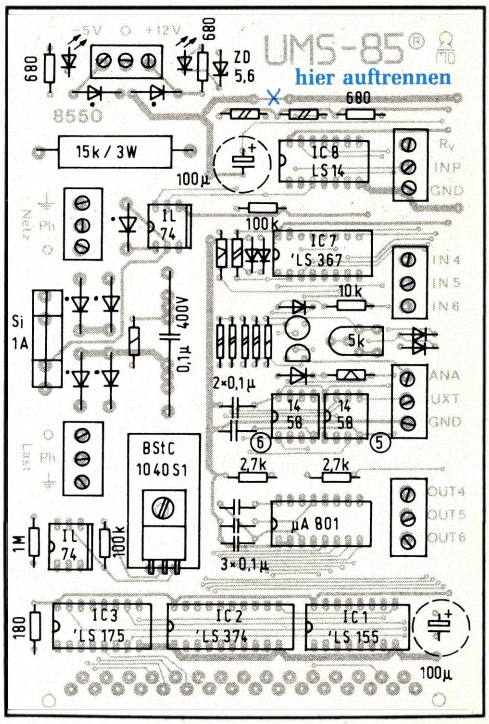


Bild 4: Mit der Trennung von digitaler und analoger Masse erreicht man eine verbesserte Störspannungsunterdrückung.

Man könnte also beispielsweise hergehen und damit einen Meßbereich von 0...255 °C realisieren. Da es aber bei diesen Temperaturen schon reichlich schwül wird im Hobby-Labor, und die Lötstellen am Sensor dabei bereits schmelzen würden, wollen wir uns mit dem halben Bereich von 0...127,5 °C begnügen. Aus dieser Festlegung folgt unmittelbar die Auflösung von 0,5 °C pro digitalem Schritt, oder anders ausgedrückt ändert sich der binäre Meßwert pro Grad um zwei Schritte. Das wiederum hat unmittelbare Auswirkungen auf das spätere Programm; denn wenn der in binärer Form vorliegende Meßwert gerade ist (niedrigstwertiges Bit LSB = 0), kommt in die Nachkommastelle bei der Meßwertdarstellung eine Null. Bei ungeraden Binärwerten (LSB = 1) muß nach dem Komma eine Fünf erscheinen.

Den Ausschlag vergrößern

Im gewählten Meßbereich ändert sich die Dioden-Durchlaßspannung um 127 · 2 mV = 0,25 V. Diese Spannungs*änderung* wollen wir verstärken, um sie optimal an den Eingangsbereich der A/D-Umsetzung anzupassen. Die auf dem Analog-Interface angeordneten Operationsverstärker können maximal 8...10 V Eingangsspannung verkraften, ohne diese zu verzerren. Verstärken wir nun die gelieferte Spannungsdifferenz von 0,25 V mit einem Faktor von etwa 25, kommen wir maximal auf ca. 6,25 V am Analog-Eingang UXT, was dieser klaglos verarbeitet. **Bild 3** zeigt einen einfachen Meßverstärker, der unmittelbar aus dem Mikrocomputer-Netzteil gespeist werden kann. Transistor und Z-Diode bilden eine Konstantstromquelle, die die Diode fortwährend mit reichlich 1 mA speist. Die mit steigender Temperatur sinkende Dioden-Durchlaßspannung (negativer Temperaturkoeffizient) wird vom Operationsverstärker invertierend verstärkt (bei der gewählten Beschaltung 22fach). Das Poti dient zum Abgleich bei 0 °C, wobei die Dioden-Durchlaßspannung natürlich nicht 0 V beträgt. Hier sollte man – falls die finanziellen Mittel reichen – ein Zehn-Gang-Poti einsetzen, um feinfühlig abgleichen zu können.
Abweichend von früheren Beschreibungen soll das Analog-Interface in diesem Anwendungsfall außer mit +12 V und −5 V auch noch mit Masse (GND) vom Mikrocomputer-Netzteil verbunden werden. Die Masseleitung auf der Interface-Karte ist dann an der in **Bild 4** gezeigten Stelle aufzutrennen. Durch die getrennte Führung von analoger und digitaler Masse vermeidet man bei empfindlichen Meßaufbauten die Einstreuung von Störspitzen aus dem Digitalteil (die digitale Masse gelangt über Stift 1 des Steckers an die Platine).

Nun ist der Mops an der Reihe

Das Temperaturprogramm TEMPR besteht im wesentlichen aus drei Blöcken (**Bild 5**): Zuerst erfolgt die Digitalisierung des analogen Meßwertes, was detailliert beim Digitalvoltmeter-Programm in Teil 4 beschrieben worden ist (einschließlich Unterprogramm OUTDA, **Bild 6**). Danach werden die einzelnen Digits hergenommen, nach ihrer BCD-Wandlung in den Siebensegmentcode umgesetzt und in den Anzeige-Buffer 0BF1...5 überschrieben. Bei Digit 2 kann, wie eingangs erläutert, nur eine „5" oder „0" auftreten (SSG-Code „6D" bzw. „3F"), und das Bitmuster „63" im Digit 1 (ganz rechts) symbolisiert das Grad-Zeichen (hochgestellte kleine Null).
Da die Aktivierung der Anzeige (SUB DISPL) ca. 8 ms dauert, würde beim kontinuierlichen Durchlauf von A/D-Wandlung und Meßwertdarstellung eine flackernde Anzeige entstehen. Wir fügen daher eine Schleife von ca. 125 DISPL-Durchläufen ein (Schleifenzähler auf 80 HEX setzen), mit der sich der Umsetz- und Anzeige-Zyklus nur etwa einmal pro Sekunde abspielt. Die zugehörigen Maschinenprogramme finden Sie in **Bild 7** und **8**, und Sie sollten vor der nachfolgenden Eichung die paar Bytes in den angegebenen Adreßbereich Ihres Computers laden.

Wasser hilft uns eichen

Mit dem eingesetzten und angeschlossenen Analog-Interface erfolgt nun die Eichung. Dazu geben Sie „FF" (= Vollausschlag) an den D/A-Umsetzer aus (3E – FF – 32 – 00 – 0C – 76; vgl. Teil 4) und verdrehen das Poti auf dem Analog-Interface derart, daß an den Klemmen „ANA" und „GND" eine Spannung von 5,61 V entsteht. Dieser Wert für den Vollausschlag ergibt sich aus der Multiplikation des Meßbereich-Endwertes von 127,5 °C mit dem Temperaturkoeffizienten 2 mV/°C und dem Verstärkungsfaktor 22 des Meßverstärkers (s. o.).

Nun eichen Sie den Nullpunkt Ihres Thermometers bei exakt 0°C, einer Temperatur, die Sie sich wie folgt problemlos selbst „machen" können: Geben Sie in ein Glas reichlich Eiswürfel ein (so viele, wie man für einen ganz schlechten Whisky braucht) und füllen Sie diese mit kaltem Wasser auf. Gut umgerührt stellen sich in diesem Napf genau 0°C ein, weil sich die Temperatur bei diesem Fixpunkt des Wassers so lange nicht erhöht, wie noch Eis vorhanden ist, das jedes Quäntchen Wärme erst zum Schmelzen „verbrät", ehe eine Erwärmung stattfindet. Das Poti *am Meßverstärker* verdrehen Sie nun so, daß in der Anzeige gerade „000.0" erscheint (mit einem leichten Trend hin zu „000.5"). Diese Einstellung verändern sie anschließend bitte nicht mehr.

Den Abgleich für Vollausschlag haben Sie grob bereits vorgenommen (5,61-V-Einstellung, s. o.). Um dies exakt nachzujustieren, sollten Sie wieder Ihren Wassernapf hernehmen und den Inhalt diesmal zum Kochen bringen. Solange noch nicht alles Wasser verdampft ist, haben Sie ein ausgezeichnetes Eichnormal für 100 °C, und mit dem Poti *auf dem Analog-Interface* gleichen Sie diesen Wert ab (bei eingetauchtem Meßfühler, versteht sich).

Sie verfügen nun über ein recht genaues und komfortables Digitalthermometer auf Mikrocomputer-Basis, das Grundlage Ihrer privaten Wetterstation oder Raumüberwachung werden kann. Natürlich haben Sie Verständnis dafür, daß weder die Redaktion noch die Autoren aus zeitlichen Gründen in der Lage sind, Programmerweiterungen oder gar Ihre Heizungsregelung zu realisieren; aber vielleicht versuchen Sie sich einmal daran und lassen uns Ihre Ergebnisse wissen. Auf jeden Fall wünschen wir Ihnen viel Spaß und guten Erfolg!

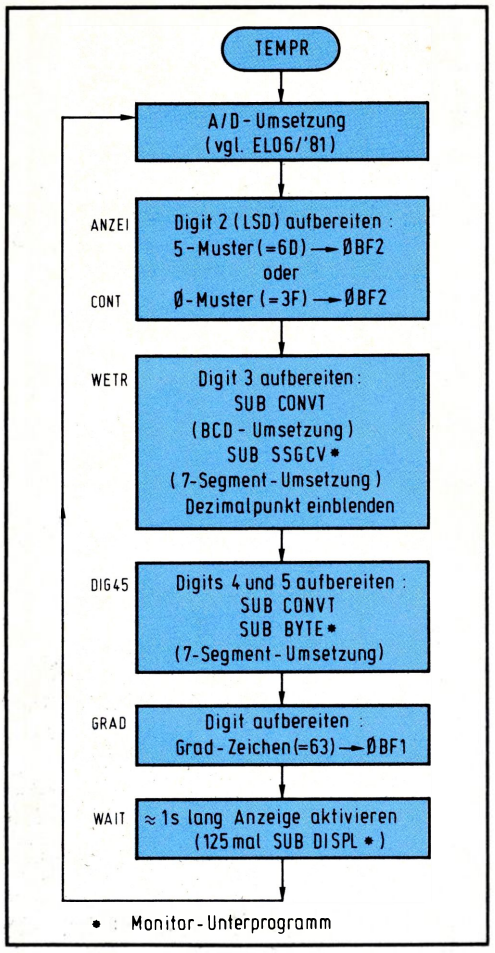


Bild 5: Im wesentlichen weist das Hauptprogramm TEMPR eine lineare Struktur auf.

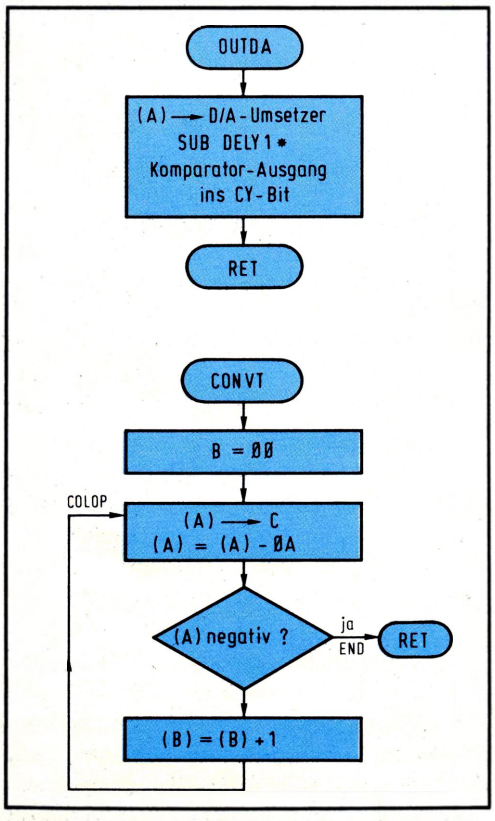


Bild 6: Zwei kleine Unterprogramme werden vom Hauptprogramm aufgerufen; OUTDA spricht den D/A-Umsetzer an, während CONVT zur BCD-Umsetzung dient.

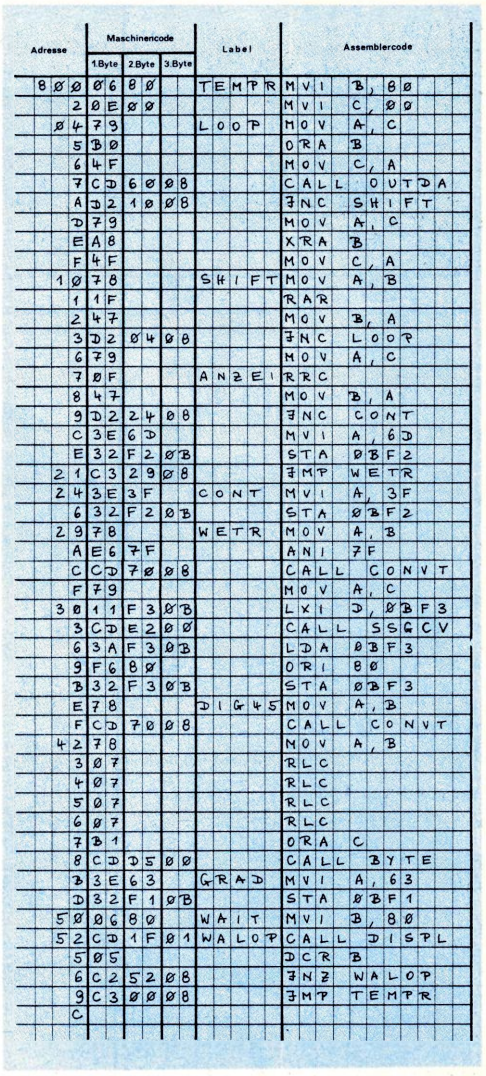


Bild 7: Maschinenprogramm zu Bild 5.

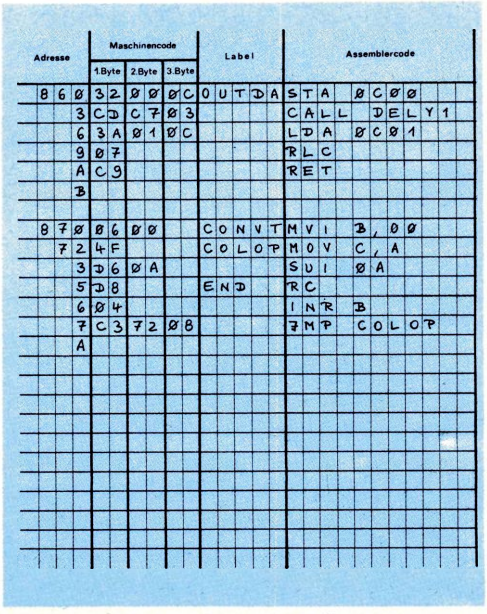


Bild 8: Vergessen Sie nicht, auch diese beiden Programmteile noch einzugeben.

Dem Mops geht ein Licht auf

Wenn Sie sich im Rahmen unserer Mikrocomputer-Einführung bis hierhin durchgearbeitet haben, kennen Sie von den Grundlagen der Programmierung so ziemlich alles, was Rang und Namen hat. Als Vervollkommnung erfahren Sie in diesem Teil, wie Sie Ihren Mikrocomputer dazu bringen, Lasten im 220-V-Netz zu schalten, was einem so zartbesaiteten MOS-IC sicherlich einiges abverlangt. Natürlich nehmen wir hierfür nicht etwa ein Relais, das von einem Bit des Ausgabe-Kanals angesteuert wird; diese Lösungsmöglichkeit ist Ihnen nicht neu, denn wir haben verschiedentlich bereits darauf verwiesen (z. B. beim Schaltautomaten im 3. Teil oder beim Telefon im 6. Teil). Wir wollen Ihnen hier eine Möglichkeit vorstellen, bei der der Mops über einen kontaktlosen Halbleiterschalter auf Lasten im Netz einwirkt. Das wird sich nicht nur auf das bloße Ein- und Ausschalten beschränken, sondern so weit gehen, daß Sie einen richtigen Mikrocomputer-Dimmer aufbauen können, der Ihnen auf Tastendruck eine 100-W-Glühbirne auf jede gewünschte Helligkeit bringt.

Vorsicht beim Umgang mit der Netzspannung! Niemals netzspannungsführende Teile unbeaufsichtigt lassen und wenn irgend möglich, einen Trenntrafo verwenden!

Zurück zu einem alten Bekannten

Für das hier beschriebene Anwendungsbeispiel verwenden wir diejenigen Schaltungsteile, die bei der Vorstellung des Analog-Interfaces im 4. Teil noch unberücksichtigt geblieben sind (**Bild 1**). Schlagen Sie dazu bitte noch einmal das Bild 2 aus Teil 4 auf

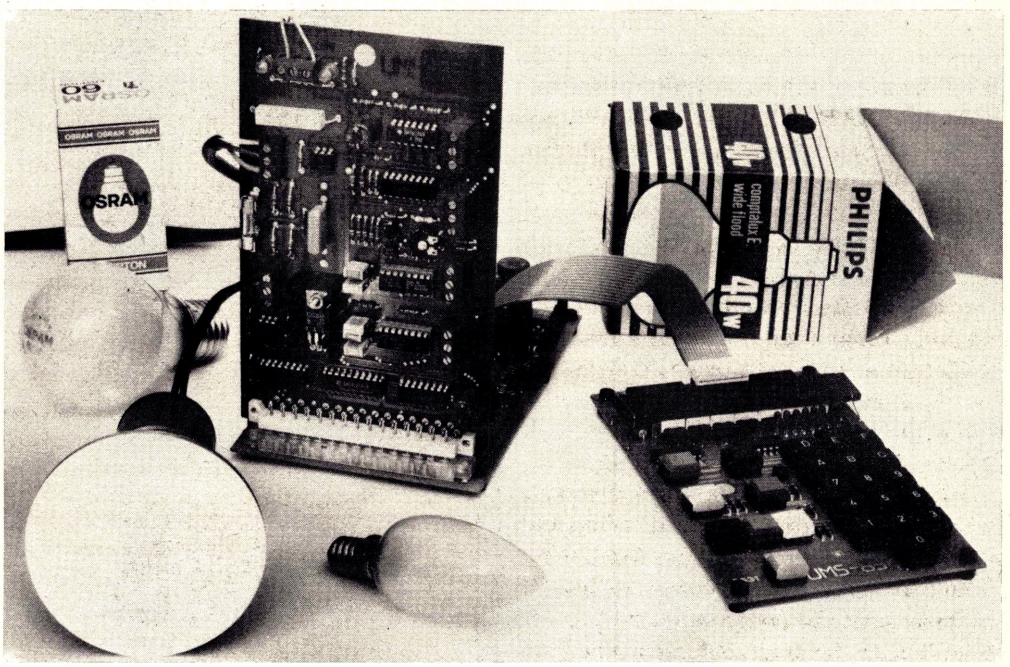


Bild 1: Über einen kontaktlosen Halbleiterschalter besteht sogar die Möglichkeit, Lasten im 220-V-Netz zu schalten.

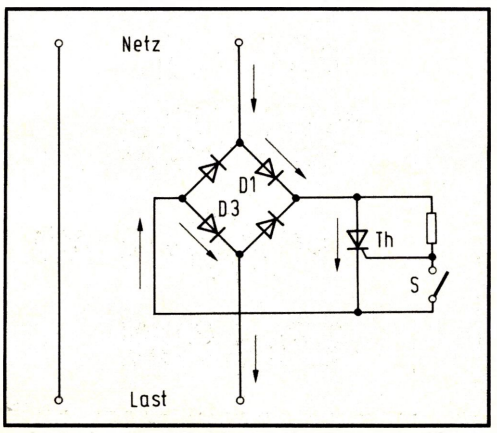


Bild 2: Als Halbleiterrelais fungiert eine Diodenbrücke mit nachgeschaltetem Thyristor, der wegen der galvanischen Trennung vom Netz über einen Optokoppler angesteuert wird.

und betrachten den unteren Schaltungsteil. Der bildet im Prinzip einen Halbleiterschalter, bei dem im Weg vom Netz zum Verbraucher eine Diodenbrücke liegt. Über diese Brücke kann nun aber nur dann Strom fließen, wenn der Thyristor leitend ist. Für die positive Halbwelle der Wechselspannung zeigen die Pfeile in **Bild 2** den Stromlauf über D1 und D3 an, bei der negativen Halbwelle ist entsprechend das andere Diodenpaar leitend. Voraussetzung für einen leitenden Thyristor ist ein offener Schalter S, weil in diesem Zustand Steuerstrom ins Gate fließen kann.

Auf der Platine wird der Schalter S durch einen Optokoppler ersetzt. Dadurch entfallen nicht nur sämtliche mechanisch beweg-

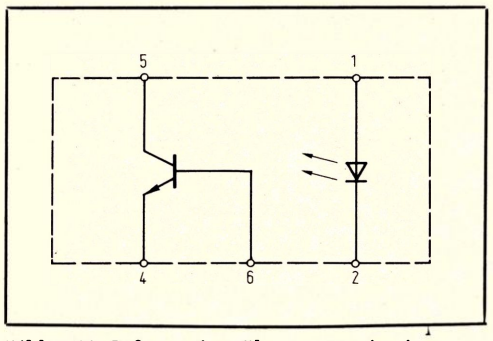


Bild 3: Die Informationsübertragung in einem Optokoppler erfolgt kontakt- und berührungslos mittels Licht.

ten Teile, sondern es besteht auch keine leitende Verbindung zwischen dem Netz und dem Computer (galvanische Trennung). Dies ist die Grundvoraussetzung dafür, daß man derartige Anwendungen überhaupt realisieren kann, weil im anderen Fall bei der Bedienung der Elektronik die Gefahr besteht, direkt mit dem Netz in Berührung zu kommen.

Wirkung per Lichteinfall

Der ansteuernde Optokoppler beinhaltet eine Leuchtdiode und einen Fototransistor, der immer dann leitet, wenn die Diode stromdurchflossen ist (**Bild 3**). Legt man, wie es in der Schaltung getan wurde, die Anode über einen Vorwiderstand an +5 V, so ist der Transistor immer dann leitend, wenn die Katode (Stift 2) auf LOW liegt; dies entspricht einem geschlossenen Schalter S in Bild 2. Um diesen Zustand herzustellen, braucht man nur an die Adresse C01 ein Datenwort auszugeben, bei dem das höchstwertige Bit (MSB) Null ist. Im 4-Bit-Speicher IC3 werden die oberen vier Bits dieses Datenwortes zwischengespeichert, und das MSB geht als Signal THYR an die Katode im Optokoppler. Im umgekehrten Fall (MSB auf HIGH) sperrt der Fototransistor (offener Schalter S), der Thyristor leitet, und der Verbraucher bekommt Strom zugeführt.

Vorsicht ist geboten

Das wollen wir uns einmal in der Praxis ansehen. Stecken Sie dazu das Analog-Interface in die CPU-Buchsenleiste (oder in die vorderste der Bus- und Speichererweiterung), schließen die Stromversorgung an und holen sich eine Glühbirne mit Fassung und Anschlußdrähten. Diese Last kommt an die untere Dreierklemme am linken Platinenrand, wo Null, Phase und Schutzleiter-

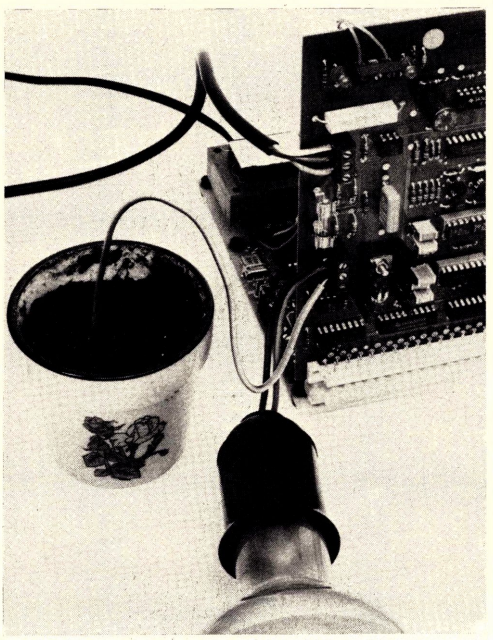


Bild 4: Last und Netz werden an die entsprechend gekennzeichneten Schraubklemmen angeschlossen.

anschluß deutlich gekennzeichnet sind (**Bild 4**). Entsprechend klemmen Sie ein Schukokabel an die obere Dreierklemme an und achten bitte sorgfältig darauf, daß der Schutzleiter richtig angeschlossen wird; die Polung von Phase und Null ist selbstverständlich unkritisch. Wenn Sie jetzt das Schukokabel mit dem Netz verbinden, machen Sie mit Ihren Händen einen weiten Bogen um diesen linken, Netzspannung führenden Teil der Platine. Den restlichen Computer können Sie gefahrlos bedienen. Nach RESET geht das Signal THYR automatisch auf LOW, was die Lampe zum Verlöschen bringt (geschlossener Schalter S). Wenn Sie nun das Datenwort 80 (MSB auf HIGH) nach C01 laden (Zieladresse des Zwischenspeichers IC3 auf dem Analog-Interface), geht THYR auf HIGH, der Fototransistor im Optokoppler sperrt, und resultierend daraus leuchtet die Glühbirne. Die Sequenz, mit der Sie das schaffen, umfaßt drei Befehle: Akku mit 80 laden (3E–80), Akku-Inhalt nach 0C01 transportieren (32–01–0C) und Mops anhalten (76). Laden Sie diese paar Bytes einmal in Ihren Arbeitsspeicher und starten das Prográmmchen mittels RUN, um die Wirkung am Erstrahlen der Glühbirne zu verfolgen. RESET schaltet die Lampe jedesmal aus, und RUN bringt sie zum Leuchten.

Nach dem Experimentieren ziehen Sie vorsichtshalber sofort den Netzstecker wieder heraus!

Der Effekt an sich ist sicherlich nicht berauschend, der Weg dazu aber schon eher; denn schließlich ist es ein kleines, unscheinbares Bit, das auf dem Weg über Zwi-

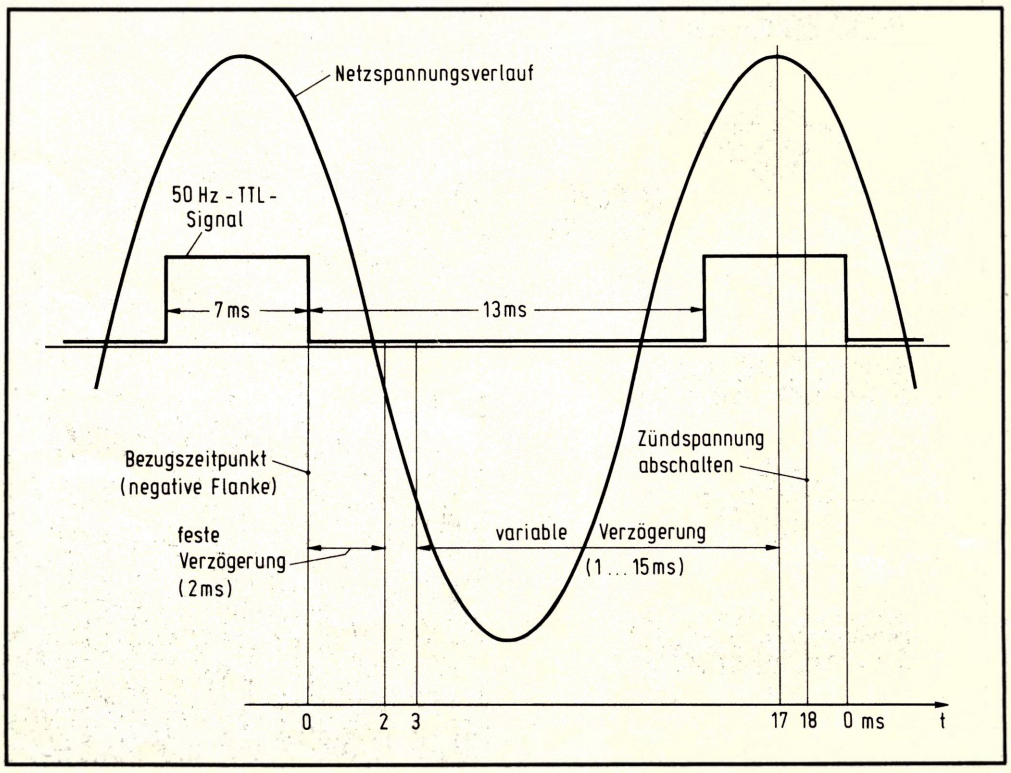


Bild 5: Über einen zweiten Optokoppler wird ein netzspannungssynchrones TTL-Signal bereitgestellt.

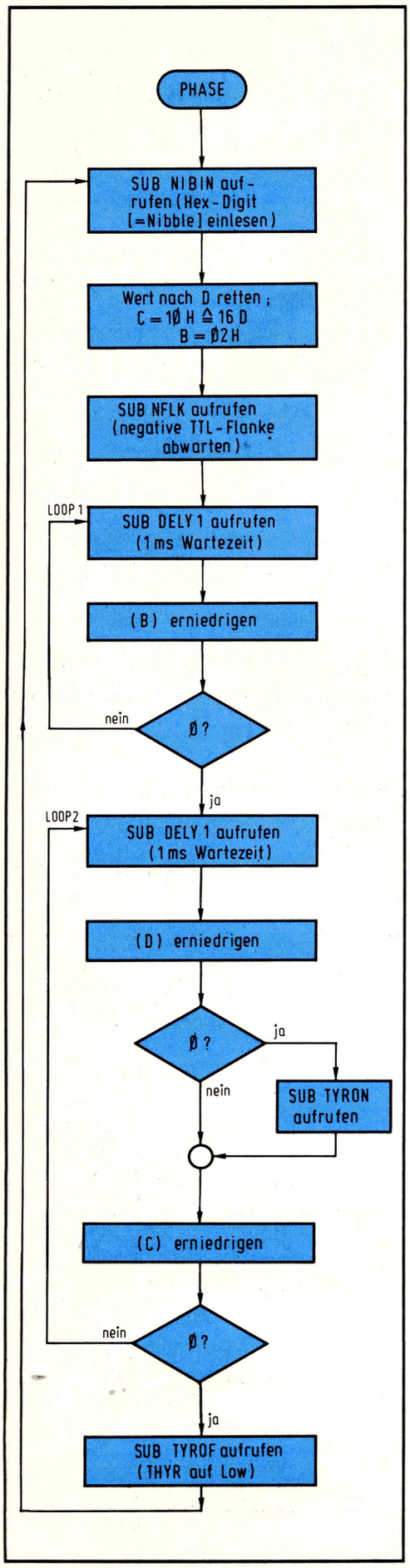


Bild 6: Flußdiagramm für das Programm der Phasenanschnittsteuerung.

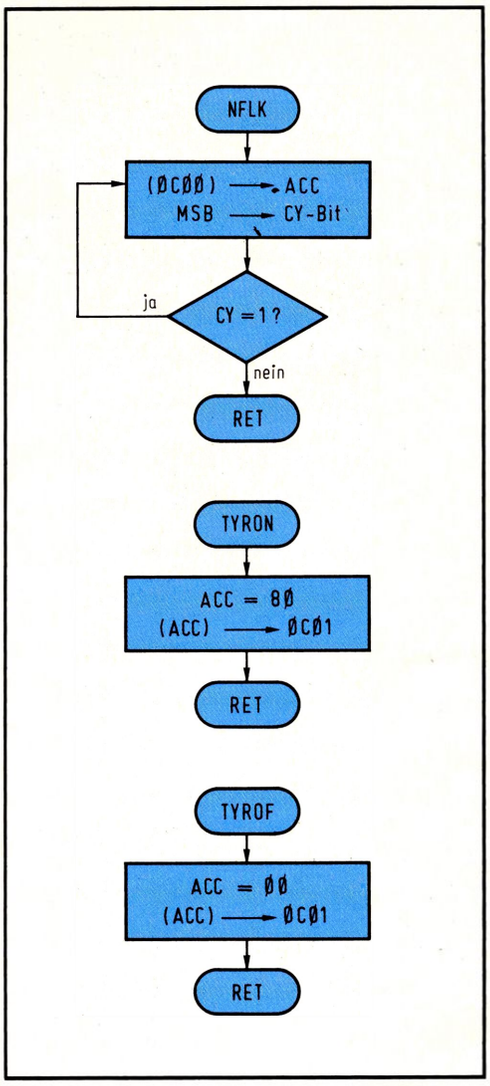


Bild 7: Das Hauptprogramm verwendet drei kleine Unterprogramme zur Flankenerkennung (NFLK) sowie zum Ein- und Ausschalten des Thyristors (TYRON bzw. TYROF).

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
ØØØ	CD	4Ø	Ø3	PHASE	CALL NIBIN	1
Ø35	7				MOV D,A	2
Ø41	1Ø	Ø2			LXI B,Ø21Ø	3
Ø7	CD	3Ø	Ø8		CALL NFLK	4
ØA	CD	C7	Ø3	LOOP1	CALL DELY1	5
ØD	Ø5				DCR B	6
ØE	C2	ØA	Ø8		JNZ LOOP1	7
11	CD	C7	Ø3	LOOP2	CALL DELY1	8
14	15				DCR D	9
15	CC	38	Ø8		CZ TYRON	10
18	ØD				DCR C	11
19	C2	11	Ø8		JNZ LOOP2	12
1C	CD	3E	Ø8		CALL TYROF	13
ØB1F	C3	ØØ	Ø8		JMP PHASE	14
						15
ØB3Ø	3A	ØØ	ØC	NFLK	LDA ØCØØ	16
33	17				RAL	17
34	DØ			END	RNC	18
35	C3	3Ø	Ø8		JMP NFLK	19
ØB38	3E	8Ø		TYRON	MVI A,8Ø	20
3A	32	Ø1	ØC		STA ØCØ1	21
3D	C9				RET	22
ØB3E	AF			TYROF	XRA A	23
3F	32	Ø1	ØC		STA ØCØ1	24
Ø842	C9				RET	25

Bild 8: Maschinenprogramm zu den Bildern 6 und 7.

schenspeicher, Optokoppler und Thyristor eine Glühbirne ein- und ausschaltet, mit der Sie unter Umständen auch Eier braten können! Als maximale Leistung sind 220 W schaltbar, bedingt durch den höchstzulässigen Strom für den Thyristor.

Dem Netz auf die Phasenlage geschaut

Es ist Ihrer Aufmerksamkeit mit Sicherheit nicht entgangen, daß auf der Platine ein zweiter Optokoppler sein Dasein fristet, für das Sie eine Erklärung erwarten; hier ist sie: über einen Vorwiderstand liegt die interne Leuchtdiode im Stromkreis der Netzspannung, und während der positiven Halbwellen leitet der sekundärseitige Fototransistor, der einen Inverter in IC8 ansteuert. An dessen Ausgang entsteht ein Rechtecksignal, das synchron zur Netzspannung verläuft (**Bild 5**). Während der negativen Halbwellen ist der Optokoppler inaktiv, weil dann die antiparallel geschaltete Schutzdiode leitet.
Mit der Kenntnis dieser Zusammenhänge können Sie jetzt ohne weiteres einen computergesteuerten Dimmer aufbauen, der im Echtzeitbetrieb die 50-Hz-Frequenz des Netzes verarbeitet. Dazu muß der Thyristor nur gezielt zu einem definierbaren Zeitpunkt innerhalb einer Halbwelle eingeschaltet werden; das Sperren erfolgt jeweils automatisch beim nächsten Nulldurchgang der Netzspannung. Wird der Thyristor danach wieder gezündet (durch HIGH-Potential an THYR), leuchtet die angeschlossene Glühbirne wieder, was wegen der schnellen Aufeinanderfolge (100mal pro Sekunde) vom Auge als kontinuierlicher Vorgang wahrgenommen wird.

Dosierter Verbraucherstrom

Der Spannungsnulldurchgang erfolgt rund 2 ms nach der negativen Flanke des TTL-Signals (vgl. Bild 5). Dies soll der frühestmögliche Zündzeitpunkt sein, bei dem der Verbraucher die meiste Leistung bekommt (rund 100 %). Verzögert man die Zündung um 1...15 ms (vereinfachende Annahme), geht die Stromflußdauer bis auf etwa 25 % zurück, was dem gewünschten Dimm-Effekt entspricht. Unser Programm liest dazu in einer Endlosschleife die gewünschte Verzögerungszeit von der Tastatur ein und zündet den Thyristor entsprechend, nachdem es die negative TTL-Flanke abgewartet hat. Spätestens 18 ms nach dieser Flanke (oder

16 ms nach dem Nulldurchgang) wird das THYR-Signal wieder auf LOW gebracht, damit der Thyristor beim folgenden Spannungsnulldurchgang automatisch sperrt. Danach vollzieht sich das Spielchen aufs neue, und wenn keine neue Eingabe über die Tastatur erfolgt, bleibt die eingestellte Helligkeit erhalten.
Im Flußdiagramm (**Bild 6**) erkennen Sie, daß zunächst das Monitor-Unterprogramm NIBIN (340) aufgerufen wird; es liest ein HEX-Digit (Nibble) von der Tastatur ein und überschreibt es in den Akku, nachdem die Taste NXT gedrückt und losgelassen worden ist. Register D speichert dieses HEX-Digit zwischen, damit der Akku die Hände frei behält für andere Aufgaben. Die Vorgabe von dezimal 16 in Reg C bestimmt den Ausschaltzeitpunkt, und Reg B hält den Wert für die 2-ms-Wartezeit nach dem Nulldurchgang. Das Unterprogramm NFLK (**Bild 7**) wartet auf die negative Flanke des TTL-Signals, das beim Einlesen der Adresse 0C00 im höchstwertigen Bit des Akkus abgefragt werden kann. Wie das vor sich geht, sehen Sie beim Blick auf das Flußdiagramm: Nach dem Einlesen per LDA-Befehl steht der Pegel des TTL-Signals im höchstwertigen Akku-Bit; durch den Links-Schiebebefehl RAL gelangt diese Information ins CY-Bit, wo es mit dem bedingten Rücksprungbefehl RETURN ON CARRY (Rücksprung bei C=1) abgefragt wird. Eine andere, Ihnen bekannte Möglichkeit der Einzelbit-Abfrage besteht im Maskieren mit anschließender Nullabfrage.

Nun dimmt mal schön

Das Monitor-Unterprogramm DELY1 (3C7) hat eine feste Laufzeit von 1 ms, die mit den einzelnen Zeitfaktoren jeweils in einer Schleife multipliziert wird. Das zugehörige Maschinenprogramm finden Sie in **Bild 8**. Aufgrund der gewählten Anordnung ergibt sich ein Abfrage- und Zündzyklus von 20 ms, also genau halb so oft wie bei einem „normalen" Hardware-Dimmer. Durch die Einkleidung in die Endlosschleife können Sie die Helligkeit Ihrer Lampe jederzeit modifizieren, indem Sie einfach einen neuen Wert für die Verzögerungszeit eingeben; mit dem Druck auf NXT wird er in den Akku übernommen und bestimmt fortan, wieviel Strom vom Thyristor durchgelassen wird. Ihr Mikrocomputer-Dimmer ist damit funktionstüchtig, und Sie haben ein Anwendungsbeispiel bekommen, bei dem Sie per Programm sogar auf Lasten im 220-V-Netz einwirken können.

Reinhard Gößler

Funktionen des 2-K-Monitors

Der 2-K-Monitor ist als Zubehör in einem 2716-EPROM erhältlich. Neben den Standard-Funktionen der Grundversion enthält die Monitor-Erweiterung die Software zur Verwaltung des Cassetten-Interfaces sowie eine Reihe der in der ELO beschriebenen Programm- und Anwendungsbeispiele.

Beim Einsatz des 2716-EPROMs müssen die beiden Drahtbrücken auf der CPU (rechts oberhalb des Quarzes) unbedingt in Stellung „A" eingelötet werden (waagerecht).

Als Sonderfunktion stellt der 2-K-Monitor den Inhalt des Datenfeldes auch noch als Bitmuster in der LED-Zeile dar; deshalb bestehen geringfügige Abweichungen zwischen Monitor-Dokumentation und dem Inhalt des 2716; an den beschriebenen Unterprogramm-Adressen ändert sich dadurch nichts.
Die Sonderfunktionen werden durch die Funktionstaste FCT aktiviert, gefolgt von der entsprechenden Ziffer 1...7 und den Abschluß durch NXT.
Die Eingabe von Adressen erfolgt immer in der natürlichen Reihenfolge, d. h. erst die obere, dann die untere Hälfte der Adresse (z. B. 08 – NXT , 40 NXT für Adresse 0840).

FCT 0: SEROT (Cassetten-Ausgabe)

Ansteuerung des Cassetten-Interfaces bei der Ausgabe von Programmen und Daten; nach OBFB/C die gewünschte Start- und nach OBFD/E die Stopadresse des zu übertragenden Datenblocks eingeben (vgl. „Vom Bit zum Beispiel", Teil 6).

FCT 1: SERIN (Cassetten-Eingabe)

Ansteuerung des Cassetten-Interfaces bei der Eingabe von Programmen und Daten; nach OBFB/C die gewünschte Start- und nach OBFD/E die Stopadresse des zu übertragenden Datenblocks eingeben, die nicht dieselben sein müssen wie bei der Ausgabe. SERIN im 10-s-Vorspannbereich (hoher Dauerton) starten. Fehlerfreies Einlesen: Anzeige 0800 XX; andernfalls Fehlermeldung „Err" (vgl. Teil 6).

FCT 2: LIST (Speicherabzug ausdrucken)

nach OBFB/C die Startadresse des Datenblocks und nach OBFD die Anzahl der gewünschten Druckzeilen eingeben; pro Zeile werden acht Bytes ausgedruckt (vgl. Teil 7).

FCT 3: SCHAM (Schaltautomat)

Nach OBFB/C die Startadresse des Datenbuffers und nach OBFD die hexadezimale Schrittzahl eingeben; automatisches Weiterschalten im Sekundentakt bei offenem SERIN. Liegt SERIN auf LOW, muß zum Weiterschalten ein beliebiges Bit des CPU-Eingabe-Kanals kurzzeitig geerdet werden (vgl. Teil 3).

FCT 4: DVM (Digitalvoltmeter)

Betrieb des Analog-Interfaces als Digitalvoltmeter mit einer Anzeige von 0,0...9,9 V. Eingangsspannung an UXT anlegen (maximal +10 V), Ausgang zuvor mit „64" auf 10 V eichen (vgl. Teil 4).

FCT 5: PHONE (Automatischer Rufnummerngeber)

Rufnummer eintasten und zum Starten „D" (= Durchwahl) drücken. Zum Löschen „E" (= Erase) drücken (vgl. Teil 5 und 6).

FCT 6: DIGUHR (Digitaluhr)

Nach OBFB/C die Stunden (z. B. 01 – 05 = 15 h) und nach OBFD/E die Minuten (z. B. 04 – 05 = 45 min) eingeben; Sekunden werden automatisch auf Null gesetzt (vgl. Teil 8).

FCT 7: SPUHR (Sprechende Uhr)

Nach 083Cff. eingeben C3 – 10 – 06 (Zieladresse für Interrupt-Service-Routine); Eingabe der Zeit wie bei FCT 6. Zum Betrieb ist die entsprechende Interface-Karte erforderlich (vgl. Teil 9).

Hinweis: Das Weiterzählen der Uhr erfolgt automatisch per Interrupt (CPU-Belastung unter 3 %), so daß nebenbei ein eigenes Hauptprogramm bearbeitet werden kann. Dies muß beginnen mit 3E – 1B – 30 – FB (Interrupts freigeben) und muß in O83C die Sequenz C3 – 10 – 06 vorfinden; in diesem Fall das Hauptprogramm über RUN starten und nicht über FCT 7. Die Uhrzeit kann bei Bedarf aus OBE8...OBEF (Stunden-Zehner bis Sekunden-Einer) ausgelesen werden.

MUSIK (Musikautomat)

Registerpaar H&L mit der Anfangsadresse der Notentabelle laden, dann Unterprogramm bei 2A4 aufrufen. Mit H&L=O2EO Abspielen der „Wilhelm-Tell-Ouvertüre".