MIKROCOMPUTER-LEHRGERAET

CL 8145

BESCHREIBUNG UND BEDIENUNGSANLEITUNG

CONATEX - DIDACTIC Lehrmittel GmbH
D - 66514 Neunkirchen - Postfach 1407
D - 66539 Neunkirchen - Rombachstraße 65

Telefon: 06821-4346
Telefax: 06821-4411
Telex: 444851 codid

Gliederung

0. Vorbemerkung

W I C H T I G

Vor Inbetriebnahme des Rechners folgendes beachten:

a) Für die Stromversorgung wird ein Netzteil mit einer stabilisierten Ausgangsspannung von 5.0 $\pm$ 0.25 Volt und einem Ausgangsstrom von ca. 1.6 A (nur für Rechner selbst) benötigt. Vorteilhaft ist ein Netzteil mit Überspannungsschutz, da eine Spannung von mehr als 5.5 Volt die Digitalbausteine des Rechners zerstören kann. Ist die Ausgangsspannung des Netzteils regelbar, so sorge man dafür, daß sie nicht versehentlich verändert werden kann. Eine einfache Schutzschaltung gegen Überspannung ist auf der Rechnerplatine vorhanden.

b) Man achte darauf, daß die Polarität der angelegten Versorgungsspannung stimmt. PLUS-Pol des Netzgerätes in die ROTE Buchse, MINUS-Pol in die BLAUE Buchse. Eine einfache Schutzschaltung gegen falsche Polarität der Versorgungsspannung ist auf der Rechnerplatine vorhanden. Bei falscher Polung sowie Überspannung wird die Feinsicherung 2.0 A zerstört und darf nur gegen eine vom gleichen Typ ersetzt werden.

c) Die mit Buchsen herausgeführten Ein- und Ausgänge entsprechen den Kenndaten von Standard-TTL-Schaltungen, d.h. Spannungen an Eingängen müssen zwischen -0.5 V und +5.5 V bezogen auf Rechner-Masse (Minuspol) liegen. Ausgänge sollten nicht länger als einige Sekunden kurzgeschlossen werden.

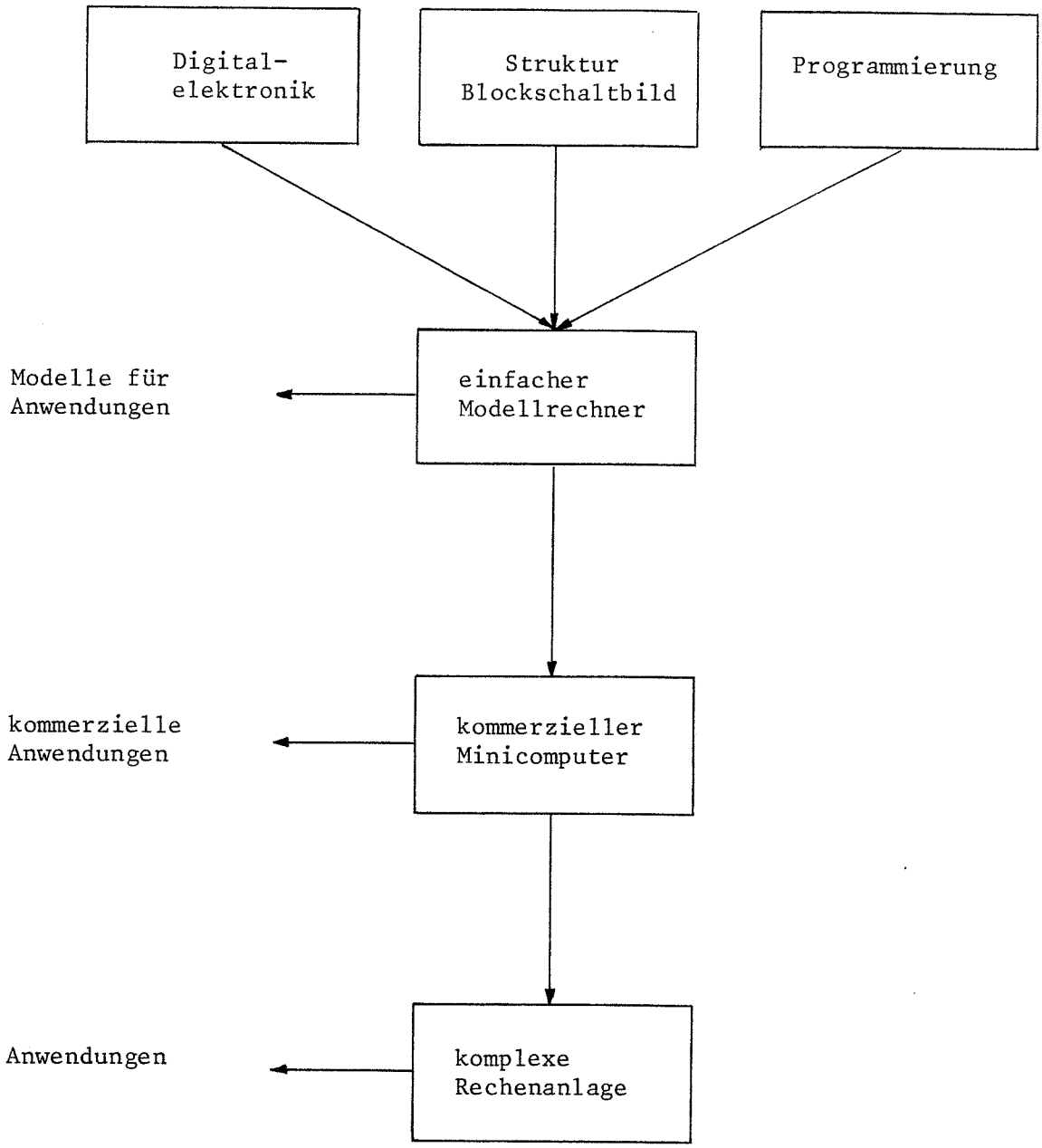
1. Einleitung

An diese Stelle eines kleinen Begleitbuches, das sich mit dem Aufbau, der Funktionsweise und der Anwendung eines Modellrechners beschäftigt, gehört üblicherweise der Hinweis darauf, wie wichtig es heutzutage ist, Vorstellungen von Begriffen wie Mikroprozessor, Mikrocomputer oder Digitalrechner zu haben. Der Mikroprozessor, als elektronisches Bauteil herausragender Vertreter des Fortschrittes in der Halbleitertechnologie hat heute weite Einsatzgebiete erobert und erschlossen. Sie reichen vom Einsatz in Prozeßrechnern zur Steuerung industrieller Herstellungsprozesse im weitesten Sinne bis hin zur Verwendung in unseren Waschmaschinen, Fernsehgeräten und manch anderen Dingen im täglichen Leben.

Will man also in der Diskussion um dieses vielseitige Bauelement und all die anderen Dinge, die damit zusammenhängen, in vernünftiger Art und Weise mitreden, muß man sich ein wenig mit dem Aufbau, der Funktionsweise von Digitalrechnern und ihren dadurch gegebenen (und dadurch begrenzten) Einsatzmöglichkeiten vertraut machen.

Dazu gehört nun nicht gerade ein Informatikstudium, es genügt aber auch ein gemütliches Stündchen nach einem guten Abendessen nicht, um sich aus einem Buch die notwendige Weisheit anzueignen. Der Weg zu einem gesunden Verständnis der Materie ist doch ein gutes Stück länger und gepflastert mit Bücherlesen, Programmeschreiben (sprich Fehler suchen) und nicht zuletzt mit Brandblasen an allen Fingern beim Zusammenlöten der integrierten Bausteine. Allerdings gibt es durchaus einen Mittelweg, den jeder beschreiten kann, der dafür etwas Neigung und Geduld mitbringt.

Im folgenden Diagramm ist ein Weg skizziert, den man einschlagen kann, um sich mit der Materie vertraut zu machen:


Digital-
elektronik
Struktur
Blockschaltbild
Programmierung
Modelle für
Anwendungen
einfacher
Modellrechner
kommerzielle
Anwendungen
kommerzieller
Minicomputer
Anwendungen
komplexe
Rechenanlage

Voraussetzungen, um die Funktionsweise eines Rechners verstehen zu können, sind:

- Grundkenntnisse in Digitalelektronik

  Hier reicht es schon, wenn der Leser weiß, was Binärzahlen sind, wie man mit ihnen einfache Grundoperationen ausführt, was ein logisches Gatter und ein Flipflop ist und wie man aus ihnen einen Zähler oder ein Register aufbaut. Diese Grundkenntnisse werden hier vorausgesetzt. Wenn sie an einigen Stellen nicht ausreichen sollten, kann er auf ein umfangreiches Angebot an Fachliteratur zurückgreifen.

- Grundkenntnisse im strukturellen Aufbau eines Rechners

  Sie sind nicht notwendigerweise Voraussetzung, da dem Leser Gelegenheit gegeben wird, diese Kenntnisse allerdings in sehr gestraffter Form zu erwerben. Es schadet nicht, wenn er sich auch hier zusätzlich mit der entsprechenden Literatur beschäftigt.

- Grundkenntnisse in maschinennaher Programmierung

  Sie sollten am wenigsten vorausgesetzt werden und müssen an dieser Stelle erworben werden.

Mit diesem Wissen sollte sich der vorliegende Modellrechner verstehen lassen. Einen Versuch in diese Richtung zu unternehmen, ist Aufgabe dieser Anleitung. Es empfiehlt sich, die einzelnen Abschnitte zunächst in der Reihenfolge durchzulesen, in der sie aufgeführt sind. Dabei sollte man den Modellrechner selbst immer neben sich liegen haben, weil er aufgrund seines optischen Designs sofort die Beziehung zum Blockschaltbild erkennen läßt.

Der Gliederungspunkt 4 gibt eine kompakte Einführung in die Baugruppen eines Mikrocomputers und deren Zusammenwirken. Im Punkt 5 werden die einzelnen Maschinenbefehle beschrieben, sodaß im Punkt 6 bereits der praktische Teil mit der Bedienung des Rechners beginnen kann. Unter Punkt 7 wird gezeigt, wie einfache Programmstrukturen mit Hilfe der Maschinenbefehle realisiert werden können. Im Punkt 8 wird in detaillierter Weise auf die Realisierung des Rechners mit Digitalbausteinen eingegangen, gefolgt von Hinweisen zum Anschluß externer Geräte, Programmbeispielen, einem Anhang und einem Literaturverzeichnis.

Um zu verdeutlichen, welche Gedanken hinter der Entwicklung dieses speziellen Mikrocomputer Lehrgerätes standen und warum es so und nicht anders realisiert wurde, möge der Leser das Gerät vor sich hinlegen und dabei folgende Kurzbeschreibung durchlesen:

Es handelt sich bei diesem Lehrgerät um einen vollständigen Mikrocomputer, der auf einer großflächigen Platine aufgebaut ist. Im Gegensatz zu anderen Mikrocomputer Lehrgeräten wurde kein Mikroprozessor als integrierter Baustein verwendet, sondern sämtliche Funktionen sind mit Hilfe einzelner digitaler TTL-Schaltkreise realisiert. Dies hat für den Lernenden mehrere Vorteile:

- Der Inhalt von Befehlszähler, Registern und Flipflops sowie die Information auf dem Datenbus werden jederzeit über Leuchtdioden angezeigt.

- Die räumliche Anordnung der Schaltkreise auf der Platine ist so gewählt, daß sie optisch unmittelbar den Zusammenhang zum Blockschaltbild erkennen läßt. Dieser wird dadurch noch vertieft, daß die zum Verständnis wichtigen Datenleitungen auf der Platinenoberfläche angeordnet sind. So wird in direkter Weise die vom Verständnis her schwierige Lücke zwischen dem Blockschaltbild und dessen technischer Realisierung mittels einfacher Grundbausteine überbrückt.

- Durch Zerlegung eines Maschinenbefehls in bis zu acht Teilschritte kann der zeitliche Ablauf eines einzigen Befehls erfaßt werden.

Diese Vorteile kann ein auf einem Mikroprozessor basierendes Lehrgerät prinzipiell nicht bieten, da hier alle komplexen Funktionen auf einem einzigen Halbleiterchip realisiert sind, in den man nicht "hineinschauen" kann.

Weitere Besonderheiten des vorgestellten Mikrocomputer Lehrgerätes sind:

- Die Taktfrequenz ist bis zu einer Höchstgrenze kontinuierlich einstellbar, d.h. der Rechner kann vor allem auch so langsam arbeiten, daß man die Abarbeitung der einzelnen Befehle in einem Programm und deren Auswirkungen auf Registerinhalte usw. anhand der zahlreichen Leuchtdioden Schritt für Schritt verfolgen kann.

- Die Bedeutung der Maschinenbefehle wird durch eine Matrix festgelegt, die mit diskreten Dioden bestückt ist und an einer Stelle verändert werden kann. Mit ihrer Hilfe läßt sich der Begriff der Mikroprogrammierbarkeit an einem Lehrgerät verstehen.

- Eine übersichtliche Ein- und Ausgabeorganisation gestattet es, die ebenfalls angebotenen Zusatzgeräte (Digital-Analogwandler, 7-Segmentanzeige, Eingabetastatur u.a.m.) in einfacher Weise vom Programm her anzusprechen und die Einsatzmöglichkeiten eines Rechners zur Prozeßsteuerung anhand praktischer Beispiele zu erfassen.

Dieses neue Mikrocomputer Lehrgerät erleichtert wesentlich das Verständnis des langen Weges zwischen Gattern als logischen Grundbausteinen und dem Mikrocomputer, seiner Programmierung und seinem praktischen Einsatz.

Ein weiterer Gedanke, der diesem Modellrechner Pate stand, war der Wunsch, eine möglichst einfache Schaltung zu entwickeln, die dennoch die wesentlichsten Eigenschaften eines Prozeßrechners besitzt. Derjenige, der sich in der Materie schon ein wenig auskennt, wird bemerken, wie erstaunlich leistungsfähig dieser Modellrechner ist und daß er den Vergleich mit Mikrocomputern auf der Basis kommerzieller Mikroprozessoren nur in wenigen Punkten scheuen muß. Sie sind dem Ziele, diesen Rechner Schülern, Studenten und anderen Interessierten zur Ausbildung zur Verfügung zu stellen, zum Opfer gefallen.

Zwei Dinge mußten vor allem vom notwendigen Aufwand her fallen gelassen werden. Einmal ist dies die Möglichkeit der indizierten Adressierung, zu der wenigstens ein Indexregister und damit ein umfangreicher Adressenverarbeitungsteil notwendig geworden wäre. Zweitens wurde auf eine Realisierung der Verarbeitung von Interupts auf Hardwareebene verzichtet. Dies ist besonders schwer gefallen, da solches für einen Prozeßrechner lebensnotwendig ist. Allerdings wurde hierfür durch eine vom Programm her abfragbare Leitung Ersatz geschaffen, sodaß das Prinzip des Interupts zumindest softwareseitig simuliert und begriffen werden kann.

2. Liste der Abkürzungen der Komponenten auf der Rechnerplatine

SEL1	Adressenselektor 1
RAR	Rücksprungadressenregister
PC	Befehlszähler, program counter
SEL2	Adressenselektor 2
S1	Datenschalter 1
RAM	Arbeitsspeicher, random access memory
S2	Datenschalter 2
ALU	arithmetisch-logische Einheit, arithmetic logic unit
ACC	Akkumulator, accumulator
TR	Hilfsregister, temporary register
CC	Condition code
OV	Overflow-Flipflop
Z	Zero-Flipflop
OUTO	OUTO-Register, Ausgaberegister 0, output 0
OUT1	OUT1-Register, Ausgaberegister 1, output 1
INO	INO-Eingabekanal, input 0
IN1	IN1-Eingabekanal, Schalterregister, input 1
ADR1	Adreßregister 1
ADRO	Adreßregister 0
IR	Befehlsregister, instruction register
CLOCK	Taktgenerator, clock
RUN	RUN-Flipflop
CL	Steuerlogik, control logic
WAIT	WAIT-Monoflop
DEC	Befehlsdekoder, decoder
MIC	Mikroprogrammatrix, microprogramm

Bedienungsfeld:

START	START-Taste
ET	Einzeltakt-Schalter
EB	Einzelbefehl-Schalter
RESET	Reset-Taste
MB	Schalter für Manuelle Befehlsausführung
URL	Schalter für Urladen

3. Technische Daten des Mikrocomputer-Lehrgerätes CL 8145/1

Typ: digitaler, mikroprogrammierter 4-Bit Mikrocomputer, nur aus TTL-Schaltkreisen auf einer Platine (45 x 40 cm) aufgebaut

Arbeitsspeicher: 16 Befehle, ein, zwei oder drei Worte lang; Bedeutung durch Diodenmatrix definiert, ein Befehl durch Ein- und Auslöten von Dioden veränderbar

Register: 4-Bit Akkumulator mit 4-Bit Hilfsregister und 4-Bit arithmetisch-logischer Recheneinheit, sowie Zero- und Overflow-Flipflop; zwei 4-Bit Ausgaberegister, ein 4-Bit Eingabekanal, ein 4-Bit Schalterregister zur manuellen Befehls- und Dateneingabe, 4-Bit Befehlsregister mit 8-Bit Adreßteil, 8-Bit Befehlszähler, 8-Bit Rücksprungadressenregister, ein Monoflop zum kurzzeitigen Stop des Rechners

Betriebsarten: Taktgenerator ermöglicht Betriebsarten Dauerlauf, Einzeltakt und Einzelbefehl; Taktfrequenz zwischen ca. 0.1 Hz bis ca. 1 MHz kontinuierlich einstellbar, bei 1 MHz beträgt die Ausführungszeit eines Maschinenbefehls 8 µs; Rechner kann in den Betriebsarten Urladen, manuelle Befehlsausführung und Normalbetrieb arbeiten

Aufbau und Maße: durchkontaktierte Epoxyd-Platine, Maße 400 x 450 mm, Leiterbahnen verzinnt, Platine auf beiden Seiten durch Plexiglasplatten geschützt; 42 integrierte TTL-Bausteine, 1 RAM, 84 verschiedenfarbige 5 mm-Leuchtdioden zur Anzeige von Registerinhalten usw., 9 Schalter, 2 Taster, zwei 4 mm Buchsen zur Stromversorgung, 21 2-mm Buchsen für Daten und Steuersignale

Stromversorgung: 5 V $\pm$ 0.25 V, ca. 1.6 A, einfacher Überspannungs- und Verpolungsschutz auf Platine vorhanden, Standard-TTL-Pegel für alle Ein- und Ausgänge

Literatur: Anleitung mit ca. 100 Seiten mit vollständiger Beschreibung und Programmbeispielen

Zusatzgeräte: 2 x 4-Bit Anzeigeeinheit (7-Segmentziffern), 2 x 4-Bit Digital-Analogwandler (0.. 3.5 V), 4-Bit Eingabetastatur, 4-Bit Analog-Digitalwandler

4. Baugruppen des Mikrocomputers

Im Grunde besteht jedes Rechnersystem aus drei Teilen:

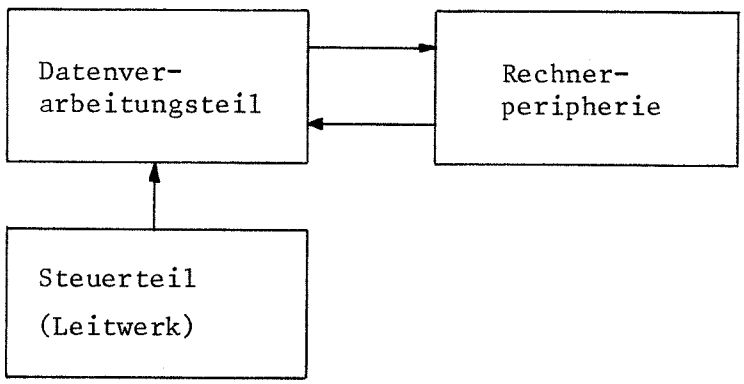
- dem Datenverarbeitungsteil)
- dem Steuerteil (Leitwerk)) Rechner
- der Rechnerperipherie

- Der Datenverarbeitungsteil umfaßt die Komponenten eines Rechnersystems, die zur Verarbeitung der eigentlichen Informationen dienen.

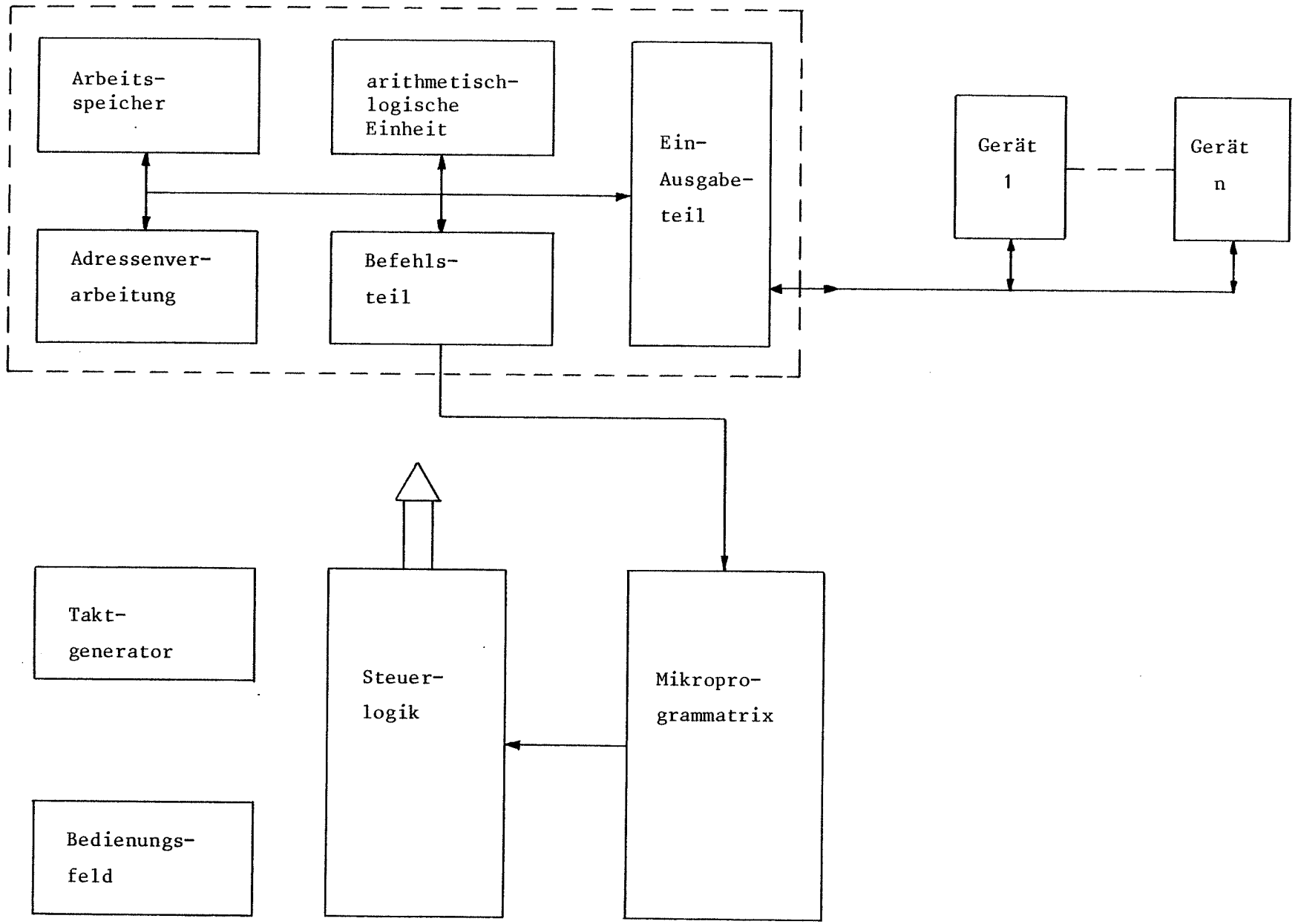
- Der Steuerteil (das Leitwerk) liefert dem Datenverarbeitungsteil all die Signale, die notwendig sind, um dort die Informationsverarbeitung organisiert ablaufen zu lassen.

- Die Rechnerperipherie erlaubt erst den praktischen Einsatz eines Rechners dadurch, daß sie die Verbindung des Rechners zur Außenwelt herstellt.

Schematisch ließe sich ein solches Rechnersystem folgendermaßen darstellen:



Nun sagt eine solch grobe Untergliederung noch nicht allzuviel aus und deswegen ist im nächsten Diagramm die Untergliederung etwas detaillierter ausgeführt:


Arbeits-
speicher
arithmetisch-
logische
Einheit
Ein-
Ausgabe-
teil
Gerät
1
Gerät
n
Adressenver-
arbeitung
Befehls-
teil
Takt-
generator
Steuer-
logik
Mikropro-
grammatrix
Bedienungs-
feld

DATENVERARBEITUNGSTEIL

ZENTRALER DATENBUS

S 1

S 2

ALU

CC

OV Z

RAM

SEL 1

ACC

TR

RAR

PC

SEL 2

ADR1

ADR0

IR

ZUM DECODER DEC

ZENTRALER DATENBUS

OUT 0

OUT 1

IN 0

IN 1

0 1 2 3 T(OUT 0)

0 1 2 3 T(OUT 1)

0 1 2 3

BY

S3

STEUERTEIL

CLOCK

GROB FEIN

FREQUENZ

RUN

$\overline{T7}$ $\overline{T6}$ $\overline{T5}$ $\overline{T4}$ $\overline{T3}$ $\overline{T2}$ $\overline{T1}$ $\overline{T0}$

CONTROL LOGIC

DEC

MIC MATRIX

T $\overline{OV}$ R $\overline{ST}$

START ET EB RESET MB URL

WAIT

INTEGRIERTER SCHALTKREIS

LEUCHTDIODEN

BUCHSEN

SCHALTER

TASTER

DATEN- ODER ADRESSBUS

(AUF PLATINENUNTERSEITE)

RICHTUNG DES DATENFLUSSES

BREITE DES BUS (4 BIT, 8 BIT)

BLOCKSCHALTBILD CL 8145 MIKROCOMPUTER-LEHRGERAET

CONATEX D6690 ST. WENDEL POSTF.1220

Der Datenverarbeitungsteil ist hier in mehrere Untereinheiten gegliedert:

1. Ein Arbeitsspeicher (RAM, random access memory) enthält die Befehle eines Programmes und Daten, die von diesem Programm verarbeitet werden. Der Datenschalter S1 dient dazu, die Daten aus dem RAM zur richtigen Zeit auf den gemeinsamen zentralen BUS zu schalten.

2. Der Arbeitsspeicher besteht aus vielen Speicherzellen, die durchnummeriert sind, d.h. eine Adresse haben. Sie müssen in einer bestimmten Reihenfolge angewählt werden und dazu ist ein Adressenverarbeitungsteil notwendig. Er besteht aus Befehlszähler (PC, program counter), Rücksprungadressenregister (RAR) und den beiden Adreßumschaltern SEL1 und SEL2.

3. Die eigentliche Verarbeitung der Daten geschieht in der arithmetisch-logischen Einheit (ALU, arithmetic logic unit) mit dem Akkumulator (ACC, accumulator), einem Hilfsregister (TR, temporary register) und zwei Flipflops, die gewisse Zustände der ALU festhalten (CC, condition code mit OVerflow- und Zero-Flipflop). Mit Hilfe des Datenschalters S2 werden die Daten aus der ALU bei Bedarf auf den BUS geschaltet.

4. Zur Ausführung eines Programmes, d.h. einer Folge von Maschinenbefehlen, muß ein Befehl nach dem anderen aus dem Arbeitsspeicher geholt und im Befehlsteil, der aus den Registern ADR1, ADRO und IR (instruction register) besteht, festgehalten werden.

5. Um Daten mit peripheren Geräten austauschen zu können, mußein Ein-Ausgabeteil (I/O, input/output) vorhanden sein, der hier aus zwei Ausgaberegistern OUTO und OUT1, sowie zwei Eingabekanälen INO und IN1 zusammengesetzt ist.

6. Ein zentraler Datenbus BUS stellt die Verbindung zwischen den verschiedenen Registern her.

Der Steuerteil (das Leitwerk) des Rechners ist ebenfalls in mehrere Untereinheiten gegliedert:

1. Ein Rechner arbeitet nicht beliebig schnell und zeitlich kontinuierlich, sondern er braucht einen Takt, d.h. ein periodisches Signal,

das den zeitlichen Ablauf eines Befehls und damit einer Befehlsfolge (Programm) bestimmt. Dieser Takt wird vom Taktgenerator (CLOCK) erzeugt.

2. Jeder Rechner benötigt ein Bedienungsfeld (CP, control panel), auf dem mit Hilfe von Schaltern und Tastern die Betriebsweise des Rechners festgelegt werden kann.

3. Die Steuersignale, die benötigt werden, um im Datenverarbeitungsteil alle Vorgänge organisiert ablaufen zu lassen, hängen von der Art des gerade abzuarbeitenden Befehls und von dem Zeitpunkt während der Ausführung dieses Befehls ab. Dazu ist eine aus Gattern bestehende Steuerlogik (CL, control logic) notwendig.

4. In der Mikroprogrammatrix wird zusammen mit dem Befehlsdekoder (DEC) die Bedeutung eines Maschinenbefehls festgelegt.

Zur Grundausstattung der Rechnerperipherie gehört:

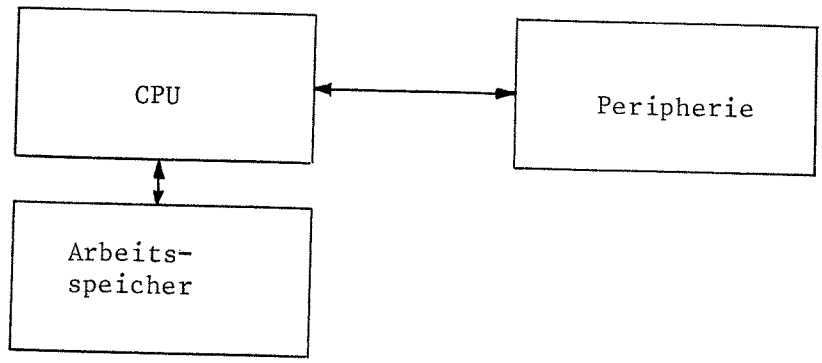
- ein Gerät, mit dem Daten (z.B. über Taster oder Schalter) dem Rechner eingegeben werden können,

- ein Gerät, mit dem der Rechner Daten in lesbarer Form (z.B. als Dezimalziffern) ausgeben kann und

- Geräte, die es gestatten, vom Rechner her irgendetwas zu steuern (z.B. Digital-Analogwandler usw.)

Nachdem wir die einzelnen Komponenten eines Rechners nun aufgezählt und kurz beschrieben haben, müssen wir uns damit beschäftigen wie sie miteinander zusammenarbeiten, wobei wir am besten die Rechnerplatine neben uns hinlegen und sie bei Bedarf mit dem Blockdiagramm vergleichen.

Schauen wir uns dabei noch einmal eine andere Unterteilung unseres Rechners an:



Unser Mikrocomputer besteht hier aus dem Arbeitsspeicher, dem Mikroprozessor, auch CPU (central processing unit) genannt und einer Ein- und Ausgabeperipherie. Auf der Platine des Modellrechners besteht der Arbeitsspeicher nur aus einem einzigen Baustein (RAM), der Rest gehört zur CPU. Um das Zusammenspiel dieser Teile zu verstehen, müssen wir nachsehen, wie die CPU mit dem Arbeitsspeicher zusammenarbeitet:

Damit eine CPU etwas sinnvolles tut, benötigt sie Befehle, die ihr Verhalten zu jeder Zeit bestimmen. Mit ihnen werden Daten, die sie aufnimmt, verarbeitet und danach in irgendeiner Form wieder ausgegeben. Die Befehle legen fest, wie diese Verarbeitung vor sich geht. Eine Folge von solchen Befehlen heißt Programm, die CPU (oder auch Zentraleinheit) holt diese Befehle nacheinander aus dem Speicher und führt sie aus.

Ein Arbeitsspeicher ist nun in viele Speicherzellen aufgeteilt. Solch eine Speicherzelle entspricht i.a. einem sog. Wort. Ein Wort ist im Arbeitsspeicher die hier kleinste adressierbare Einheit. Eine Speicherzelle des Modellrechners umfaßt 4 Bit, d.h. er hat eine Wortlänge von 4 Bit. Die Speicherzellen sind von Null beginnend fortlaufend durchnummeriert und werden mit Hilfe dieser Nummer oder Adresse angesprochen. Die Worte im Arbeitsspeicher unseres Mikrocomputers sind von 0 bis 255 nummeriert. Ein Programm wird nun in aufeinanderfolgenden Speicherzellen gespeichert, wobei jeder Befehl nun eine oder mehrere dieser Zellen beanspruchen kann. Ein kleines Programm sieht z. B. bei unserem Rechner folgendermaßen aus:

Adresse	Inhalt	mnemonische Abkürzung	Bedeutung
0	1101	INCA 1	Erhöhe Akkumulator-
1	0001		inhalt um 1
2	1001	OUT1	Gib Akkumulatorinhalt auf Kanal OUT1 aus
3	0000		
4	0000	JMP 0	Springe auf Adresse Null
5	0000		
6	...		
7	...		
.			
.			
.			

Das Programm erhöht den Inhalt des Akkumulators immer wieder um eins und gibt ihn auf dem Ausgabekanal OUT1 aus. Der erste Befehl besteht aus zwei Worten, der zweite nur aus einem und der dritte aus drei. In dieser Art und Weise ist ein Programm im Arbeitsspeicher eines Rechners in binär verschlüsselter Form abgelegt und Aufgabe der CPU ist es nun, in dieser Form vorliegende Befehle auszuführen. Damit die CPU weiß, welchen Befehl sie als nächsten ausführen muß, benötigt sie einen Befehlszähler PC (Programmzähler, program counter). Er ist wesentlicher Teil des Adressenverarbeitungsteils des Rechners. Der Befehlszähler muß eine Länge von 8 Bit haben, damit alle 256 Speicherzellen angesprochen werden können. Er zeigt immer auf die Adresse, auf der der nächste Befehl bzw. der nächste Teil des Befehls im Speicher steht. Ein Programm muß nun nicht nur so aufgebaut sein, daß seine Befehle alle hintereinander im Speicher stehen, sondern es besteht mit den sog. Sprungbefehlen die Möglichkeit, eine beliebige Adresse in den Befehlszähler zu befördern, sodaß ein Programm an anderer Stelle fortgesetzt werden kann.

Wird nun ein Befehl aus dem Speicher geholt, muß er im Befehlsteil des Rechners während seiner Ausführung festgehalten werden. Ein Befehl kann aus mehreren Teilen bestehen. In jedem Fall enthält ein Teil den Befehlscode, der in binärer Form festlegt, welcher Befehl vorliegt. In unserem Falle ist er ein Wort lang; mit diesen 4 Bit können 16 Befehle verschlüsselt werden. Dieser Befehlscode kommt nun aus dem Speicher in das Befehlsregister (Instruction register IR). Falls der Befehl einen sog. Adreßteil hat, der entweder einen Operanden für eine arithmetische oder logische Operation oder eine Speicheradresse enthält, kommt dieser Teil des Befehls in das Registerpaar ADR1/ADR0.

Wenn der Befehl in den Befehlsteil geladen ist, muß der Rechner erkennen, um welchen es sich handelt, d.h. er muß den Befehlscode analysieren. Dies geschieht mit Hilfe der Mikroprogrammatrix. Sie erzeugt eine Reihe von Signalen, den Mikrobefehlen, die die einzelnen Teilschritte innerhalb eines Befehles festlegen. Diese Mikrobefehle werden in der Steuerlogik (control logic CL) weiterverarbeitet und erzeugen dort alle Steuersignale, die einen Befehl ordnungsgemäß ablaufen lassen. Diese Steuersignale hängen ebenfalls von gewissen Systemtakten ab, die verschiedene Zeitpunkte während der Ausführung des Befehls definieren. Diese Systemtakte werden im Taktgenerator (CLOCK) erzeugt.

Was kann nun solch ein Befehl in der CPU bewirken?

Mit einigen Befehlen können arithmetische oder logische Operationen ausgeführt werden. Dies geschieht in der arithmetisch-logischen Einheit (arithmetic logic unit ALU) mit dem Akkumulator ACC und einem Hilfsregister TR. Der Akkumulator ist das zentrale Rechenregister und stellt den einen Operanden, das Hilfsregister den zweiten. In der ALU wird der Akkumulatorinhalt mit dem des Hilfsregisters verknüpft und auf den zentralen Datenbus gegeben. Der Datenbus, eine Gruppe von vier Datenleitungen, bildet das "Nervensystem" eines Rechners. Alle Daten, die nicht in den Adressenverarbeitungsteil geraten, werden hier zwischen Arbeitsspeicher, Registern und Ein/Ausgabekanälen hin- und hertransportiert. Dieser Datenbus wird nun nicht nur für diese arithmetischen oder logischen Befehle genutzt, sondern auch für eine Gruppe von anderen Befehlen, den Transportbefehlen. Sie gestatten es, Daten z.B. aus dem Arbeitsspeicher in eines der Register zu transportieren oder umgekehrt. Eine weitere in ganz besonderer Weise wichtige Befehlsgruppe sind die Sprungbefehle. Mit ihnen kann ein Programmablauf als Funktion einer Bedingung verändert werden. Ist z.B. das Ergebnis einer arithmetischen Operation gleich Null oder ist ein arithmetischer Überlauf passiert, kann mit ihnen das Programm an anderer Stelle fortgesetzt werden. Sprungbefehle ermöglichen es einem Rechner, Entscheidungen zu treffen und bilden zusammen mit der Schnelligkeit, mit der er die (im Grunde primitiven) Maschinenbefehle ausführt, seine eigentliche Existenzberechtigung.

Alle diese Strukturen tauchen bei Digitalrechnern in irgendeiner Form auf. Als Folge dier tiefgreifenden technologischen Entwicklung der elektronischen Bauteile und ihrer damit verbunden Kosten hat sich im wesentlichen nur das Gewicht der einzelnen Strukturen innerhalb eines Rechners ver-

ändert. Wo früher nur wenige Register in der CPU vorhanden waren, da sie teuere Elemente waren, gibt es nun bis zu einige hundert Register. Wo früher der Arbeitsspeicher ein besonders teueres Teil war, stellt er heute als Halbleiterspeicher einen weitaus kleineren Kostenfaktor dar. Wo früher die "Hardware", d.h. die Elektronik selbst, teuer war, wird die "Software", d.h. die Gesamtheit der Programme, insbesondere des Betriebssystems der bestimmende Kostenfaktor und Leistungsmaßstab.

Unser Modellrechner hat sowohl von den älteren klassischen Strukturen, als auch von recht neuen etwas. Im Grunde eine klassische Struktur, die jedoch in fast allen kommerziellen Mikroprozessoren vorkommt, ist der Aufbau der arithmetisch-logischen Einheit mit Akkumulator und Hilfsregister. Die Abwesenheit von Indexregistern unterscheidet ihn von den meisten anderen Rechnern und wirft ihn in seiner Entwicklungsstufe zurück. Dagegen hat er ein modernes Bussystem und wie nur sehr wenige Rechner heutzutage bietet er die Möglichkeit der Mikroprogrammierung. Die wesentlichsten Rechnerstrukturen können an ihm aufgezeigt werden, sodaß es sich lohnt, in genau kennenzulernen. Weniger Mühe und wenig prinzipiell Neues wird man erfahren, wenn man sich die Struktur anderer Rechner ansieht.

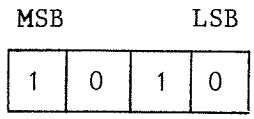
5. Maschinenbefehle des Mikrocomputers

Aufgrund einer Wortlänge von 4 Bit bietet sich ein Adreßraum von 8 Bit und ein Befehlscode von 4 Bit an, mit dem 16 Maschinenbefehle festgelegt werden können. Insgesamt belegt somit ein Befehl ein, zwei oder drei Worte im Arbeitsspeicher. Dies liegt in einem der Leistungsfähigkeit der CPU angepassten vernünftigen Verhältnis zur Größe des Arbeitsspeichers von 256 Worten.

Aus der Fülle der bei Digitalrechnern möglichen Befehle läßt sich damit natürlich nur eine sehr begrenzte Auswahl treffen. Sie muß besonders gut überlegt sein, da der Befehlssatz eines Rechners entscheidend seine Leistungsfähigkeit und Einsatzmöglichkeiten bestimmt. Es wird sich aber zeigen, daß mit der getroffenen Auswahl der 16 Befehle prinzipiell alle Programmierstrukturen dargestellt bzw. simuliert werden können. Dabei wird die Bedeutung dieser Befehle sehr viel schneller erfaßt als die der umfangreicheren Befehlssätze kommerzieller Mikroprozessoren. Die mnemotechnischen Abkürzungen sind an diejenigen üblicher Mikroprozessoren angelehnt.

5.1 Befehlsformat

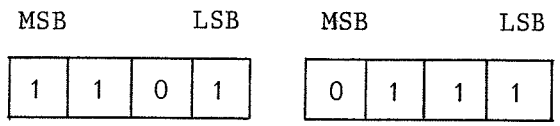
Ein-Wort-Befehle:



Beispiel: INA

Befehlscode
Adresse n

Zwei-Wort-Befehle:

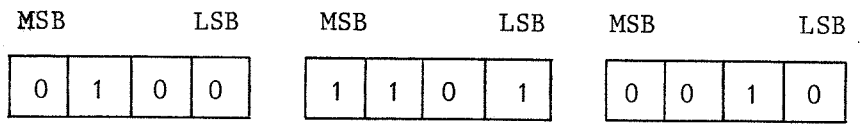


Beispiel: INCA 7

Befehlscode
Adresse n

Operand
Adresse n+1

Drei-Wort-Befehle:



Beispiel:
LDA 45

Befehlscode
Adresse n

unterer Adreßteil
Adresse n+1

oberer Adreßteil
Adresse n+2

Die 16 Maschinenbefehle können in 5 Befehlsgruppen aufgeteilt werden:

- Sprungbefehle
- Akkumulator-RAM-Befehle
- Ein- und Ausgabebefehle
- direkte Akkumulatorbefehle
- WAIT-Befehl

Die Befehle können ein, zwei oder drei Worte á 4 Bit lang sein. Die Ausführungszeit für die Befehle wird durch die Dauer von acht Takten des Taktge-

nerators festgelegt.

Zeichenerklärung:

⟨acc⟩	:	Inhalt des Akkumulators
⟨adr⟩	:	Inhalt der Speicherzelle mit der Adresse adr
n	:	4-Bit Muster mit dem dezimalen Wert n
:=	:	"wird ersetzt durch" oder "ergibt sich aus"

Bemerkung: Bei den Befehlen ADDA, INCA kann die Addition auch als Subtraktion aufgefaßt werden, wenn der zweite Operand im 2-er Komplement dargestellt wird.

5.2 Befehlstabelle

Befehlscode binär	mnemotechnische Abkürzung		Bedeutung	Anz. Worte
0000	JMP	adr	Jump: undbedingter Sprung auf Adresse adr	3
0001	JNZ	adr	Jump if not zero: Sprung auf Adresse adr, falls Zero-Flip-flop ungleich Null	3
0010	CALL	adr	Call: Unterprogrammsprung auf Adresse adr	3
0011	RET	-	Return: Unterprogrammrück-sprung	1
0100	LDA	adr	Load accumulator: ⟨acc⟩ := ⟨adr⟩	3
0101	STA	adr	Store accumulator: ⟨adr⟩ := ⟨acc⟩	3
0110	ADDA	adr	Add accumulator: ⟨acc⟩ := ⟨acc⟩ + ⟨adr⟩	3
0111	DECM	adr	Decrement memory: ⟨adr⟩ := ⟨adr⟩ - 1	3

Befehlscode binär	mnemotechnische Abkürzung		Bedeutung	Anz. Worte
1000	OUTO	n	Output 0: ⟨OUTO⟩ := n	2
1001	OUT 1	-	Output 1: ⟨OUT1⟩ := ⟨acc⟩	1
1010	INA	-	Input accumulator: ⟨acc⟩ := ⟨IN0/IN1⟩	1
1011	JBY	adr	Jump if busy: Sprung auf Adresse adr, falls Signal auf BY-Buchse low	3
1100	ENTA	n	Enter accumulator: ⟨acc⟩ := n	2
1101	INCA	n	Increment accumulator: ⟨acc⟩ := ⟨acc⟩ + n	2
1110	NANDA	n	Nand accumulator: ⟨acc⟩ := ⟨acc⟩ ∧ n bitweise NAND-Verknüpfung mit n	2
1111	WAIT	-	Wait: Stop des Rechners für eine einstellbare Zeitdauer	1

Dieser Befehl kann in der Mikroprogrammatrix verändert werden und mit einer anderen Bedeutung belegt werden.

Wie in der Tabelle schon angedeutet, läßt sich der Befehlssatz des Rechners in mehrere Untergruppen aufteilen.

5.3 Sprungbefehle

Mit diesen Befehlen kann in einem Programm auf jede beliebige Adresse gesprungen werden, d.h. das Programm wird von dieser Adresse ab weiter ausgeführt.

0000 JMP adr

Dieser Befehl bewirkt einen unbedingten Sprung auf die Speicheradresse adr ($0 \leq adr \leq 255$), d.h. nach Ausführung dieses Befehls läuft das Programm von Adresse adr an weiter, da der Befehlszähler auf diesen Wert gesetzt wurde. Dabei bleiben die Inhalte aller anderen Register und Flpflops unverändert.

0001 JNZ adr

Ein Sprung wird hier nur dann ausgeführt, wenn das Zero-Flip-flop ungleich Null ist. Ansonst wie JMP-Befehl.

0010 CALL adr

Hier wird ein unbedingter Unterprogrammsprung ausgeführt, wobei die Rücksprungadresse im Gegensatz zum JMP-Befehl im Rücksprungadressenregister RAR gespeichert wird. Die Rücksprungadresse ist die Adresse, die der Befehl hat, der dem CALL-Befehl folgt.

0011 RET

Dieser Befehl stellt einen unbedingten Sprung auf die Adresse dar, die im RAR gespeichert ist. Dies bleibt unverändert.

Mit den beiden Befehlen CALL adr und RET lassen sich sehr einfache und effizient Sprünge in ein Unterprogramm und wieder aus ihm heraus erzeugen.

5.4 Akkumulator-RAM-Befehle

Mit diesen Befehlen lassen sich Daten zwischen Akkumulator und Arbeitsspeicher austauschen und verknüpfen. Der Akkumulator spielt dabei die Rolle des zentralen Rechenregisters, in dem alle arithmetischen und logischen Operationen erfolgen.

0100 LDA adr

Mit diesem Befehl kann der Inhalt einer beliebigen Speicherzelle mit der Adresse adr in den Akkumulator gebracht werden (geladen werden). Das Zero-Flipflop wird dabei gesetzt, wenn der Inhalt des Akkumulators Null ist.

0101 STA adr

Dies ist der komplementäre Befehl zu LDA, hier wird der Inhalt des Akkumulators in eine beliebige Speicherzelle geschrieben. Das Zero-Flipflop wird dabei gesetzt, wenn der Inhalt der Speicherzelle danach Null ist.

0110 ADDA adr

Dies ist ein "klassischer" Akkumulatorbefehl, der dem Akkumulator den Namen gegeben hat. Der Inhalt der Speicherzelle mit der Adresse adr wird zum Inhalt des Akkumulators hinzuaddiert und das Ergebnis im Akkumulator "Akkumuliert". Ist das Ergebnis Null, wird das Zero-Flipflop gesetzt, außerdem das Overflow-Flipflop, wenn ein Überlauf erfolgt ist.

0111 DECM adr

Mit diesem Befehl kann der Inhalt der Speicherzelle mit der Adresse adr um 1 erniedrigt werden. Es wird sich zeigen, daß hiermit sehr einfach Schleifen programmiert werden können. Das Zero-Flipflop wird gesetzt, wenn das Ergebnis der Operation Null ist. Der Inhalt des Akkumulators wird nicht verändert.

Bemerkung: Bei den folgenden Maschinenbefehlen ENTA, OUTO und NANDA, bei denen kein arithmetischer Überlauf möglich ist, wird das Overflow-Flipflop dann gesetzt, wenn der Operand in diesen Befehlen ungleich Null ist. Dies liegt an der Konstruktion der ALU.

5.5 Ein- und Ausgabebefehle

Für einen Prozeßrechner spielen Ein- und Ausgabebefehle eine besonders wichtige Rolle, da er intensiven Kontakt mit seiner Umgebung, d.h. dem zu steuernden Prozeß haben muß.

1000 OUTO n

Der Inhalt des Ausgaberegisters OUTO wird gleich dem Wert n gesetzt, der als Operand ein Teil des Befehls selbst ist. Danach kann unmittelbar am Ausgang des Registers über Buchsen das Bitmuster mit dem Wert n abgegriffen werden. Dieses Register stellt einen der beiden Ausgabekanäle ("Ports") des Rechners dar.

1001 OUT1

Der Inhalt des Ausgaberegisters OUT1 wird gleich dem Inhalt des Akkumulators gesetzt. Die Information kann ebenfalls über Buchsen abgegriffen werden. OUT1 stellt den zweiten Ausgabekanal (Port) des Rechners dar.

1010 INA

Mit diesem Befehl kann die Information, die an den Eingabekanälen IN0 oder IN1 liegt, in den Akkumulator gebracht werden. Ob die Information von IN0 oder IN1 genommen wird, hängt von der Stellung des Schalters S3 auf dem Datenverarbeitungsteil der Platine ab.

1011 JBY adr

Durch diesen Befehl wird das logische Signal, das z.B. von einem externen Gerät an die Buchse BY gelegt ist, als Bedingung dafür genommen, ob ein Sprung auf die Adresse adr erfolgt oder nicht. Bei Low-Pegel an BY (< 0.8 V) erfolgt ein Sprung. Dieser Befehl erlaubt es z.B. die Bereitschaft eines Gerätes abzufragen, Daten an den Rechner zu liefern bzw. vom Rechner aufzunehmen ("Busy waiting"). Dies ist besonders wichtig, wenn das Gerät langsamer ist als der Rechner.

Durch eine Kabelverbindung zwischen der Buchse BY und der Buchse $\overline{OV}$ im Steuerteil des Rechners erhält der Befehl die Bedeutung: "Springe, wenn Überlauf erfolgt ist". Die $\overline{OV}$-Buchse ist mit dem Ausgang des Overflow-Flipflops verbunden. Mit dieser Modifikation lassen sich Programme für doppelt-lange Arithmetik schreiben.

5.6 Direkte Akkumulatorbefehle

Diese Befehle gestatten eine Veränderung der Daten im Akkumulator. Dabei ist der erste Operand der Akkumulator, der zweite ist Bestandteil des Befehls selbst.

1100 ENTA n

Der Inhalt des Akkumulators wird gleich n gesetzt. Ist er Null, wird das Zero-Flipflop gesetzt.

1101 INCA n

Der Inhalt des Akkumulators wird um n erhöht. Ist er danach Null, wird das Zero-Flipflop gesetzt, sowie das Overflow-Flipflop, wenn ein Überlauf eingetreten ist.

1110 NANDA n

Der Inhalt des Akkumulators ergibt sich nach der Ausführung des Befehls aus der bitweisen NAND-Verknüpfung des Akkumulatorinhaltes mit dem Bitmuster n. Mit diesem Befehl kann der Akkumulatorinhalt auf den Zustand jedes seiner 4 Bit abgefragt und auch invertiert werden.

5.7 WAIT-Befehl

1111 WAIT

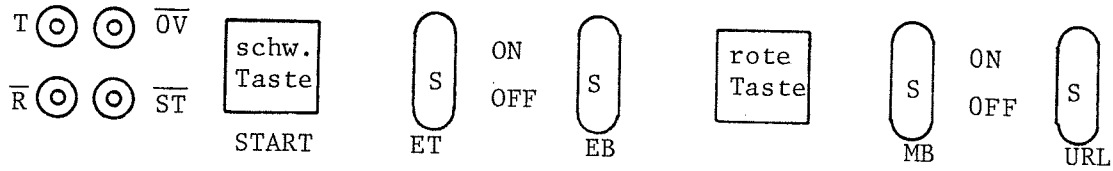
Dieser Befehl ist für kommerzielle Rechner unüblich. Er bewirkt den Ausführungsstop eines Programmes für eine mit einem Trimmpotentiometer einstelllbare Zeitdauer von ca. 0.1 bis ca. 1 sec. Da dabei eine Leuchtdiode aufleuchtet, kann damit optisch ein Programmende signalisiert werden, oder es kann in einfacher Art und Weise auf ein langsames E/A-Gerät gewartet werden.

Diese 16 Befehle gestatten es, alle Programmierstrukturen darzustellen bzw. zu simulieren, die es auf kommerziellen Rechnern geben kann. Gewisse Befehle, wie z.B. Verschiebe-(Shift)Befehle, Interruptbefehle, Befehle mit indizierter Adressierung, Befehle mit anderen logischen Verknüpfungen, die hardwaremäßig nicht implementiert sind, können mit Klimmzügen als Makrobefehle dargestellt werden. Eine Reihe solcher Befehle könnte sogar hardwaremäßig nur durch einfaches Verändern der Mikroprogrammatrix implementiert werden. (Siehe Abschnitt "Realisierung des R. mit Digitalb.")

6. Bedienung des Mikrocomputers

Zur Bedienung des Rechners steht links unten auf der Platine ein Feld mit Tastern, Schaltern und Buchsen zur Verfügung, sowie auf der rechten Seite im Datenverarbeitungsteil Schalter zur Daten- und Befehlseingabe sowie ebenfalls eine Anzahl von Buchsen. Wenn wir mit dem Rechner arbeiten wollen, müssen wir uns im einzelnen mit der Bedeutung dieser Elemente vertraut machen.

Schauen wir uns zunächst einmal das Bedienungsfeld links unten an:



Die Start-Taste, der Einzeltakt-Schalter sowie der Einzelbefehl-Schalter beeinflussen nur die Arbeitsweise des Taktgenerators:

Start-Taste: Sie startet den Taktgenerator. Die erzeugte Taktfolge hängt von der Stellung der beiden Schalter ab. Sie hat keine Wirkung, wenn der Taktgenerator schon läuft.

Einzeltakt-Schalter: Ist dieser Schalter auf Stellung "ON", wird beim Drücken der Start-Taste nur ein einziger Takt erzeugt. Damit kann der Ablauf eines einzigen Maschinenbefehls, der aus acht Takten besteht, genau verfolgt werden.

Einzelbefehls-Schalter: Ist er auf Stellung "ON", wird beim Drücken der Start-Taste genau eine Folge von acht Takten erzeugt, d.h. es wird ein einziger Maschinenbefehl ausgeführt. War der Taktgenerator nicht in Grundstellung, werden soviele Takte erzeugt, bis er in Grundstellung, d.h. am Anfang des nächsten Befehls steht.

Ist weder der Einzeltakt- noch der Einzelbefehl-Schalter eingeschaltet, so erzeugt der Taktgenerator eine kontinuierliche Anzahl von Pulsen, was der normalen Arbeitsweise des Taktgenerators während der Ausführung eines Programmes entspricht.

Ist sowohl der Einzeltakt- als auch der Einzelbefehl-Schalter eingeschaltet, so verhält sich der Taktgenerator so, als wenn nur der Einzeltakt-Schalter eingeschaltet wäre.

Bemerkung: Gestoppt wird ein Programm zweckmäßigerweise durch Einschalten des Einzelbefehl-Schalters. Es hält dann vor dem nächsten auszuführenden Befehl. Es kann danach jederzeit durch Betätigen der Start-Taste in jeder Betriebsart wieder gestartet werden.

Die Taktfrequenz selbst kann in groben Stufen mit dem Drehschalter, fein und kontinuierlich mit dem Potentiometer eingestellt werden. Dies kann auch während der Ausführung eines Programmes in jeder Betriebsweise des Taktgenerators geschehen, ohne daß Störungen auftreten, auch wenn die Taktfrequenz beim Umschalten kurzzeitig hochschnellt. Die Taktfrequenz läßt sich im Bereich von ca. 0.1 Hz bis ca. 1 MHz variieren. Da ein Befehl acht Takte benötigt, entspricht dies einer Ausführungszeit pro Befehl zwischen 80 sec (!) und 8 µsec, d.h. es werden bei 1 MHz 125000 Befehle pro Sekunde ausgeführt. Damit ist der Rechner nicht langsamer im Vergleich zu Rechnern auf der Basis kommerzieller Ein-Chip-Mikroprozessoren. Dies trotz der aus didaktischen Gründen nicht besonders optimierten Auslegung von Steuerwerk und Arbeitsspeicher.

Erzeugt der Taktgenerator Pulse, leuchtet die Leuchtdiode (LED) des RUN-Flipflops auf. Acht Systemtakte $\overline{T0}$ bis $\overline{T7}$, die vom Taktsignal abgezweigt werden, sind durch verschiedenfarbige LED's angezeigt. Sie entsprechen den Teilschritten in der Ausführung eines Maschinenbefehls und werden nacheinander durchlaufen.

Die vier Buchsen ganz am linken unteren Rand haben folgende Bedeutung:

Buchse T: Ausgang Taktsignal T des Taktgenerators. Das Signal kann Test- und Synchronisierungszwecken dienen. Beim Herausnehmen der Lötbrücke B kann der Rechner über diese Buchse mit einem externen Takt betrieben werden. (Siehe Taktgenerator mit Bedienungsfeld)

Buchse $\overline{ST}$: Eingang für ein Start-Signal des Taktgenerators. Wird hier ein Low-Signal angelegt, bewirkt dies den Start des Taktgenerators genau wie beim Betätigen der Start-Taste.

Buchse $\overline{R}$: Ausgang Reset-Signal. Wird die Reset-Taste gedrückt, liegt an dieser Buchse Low-Signal, ansonst High-Signal. Sie liefert ein (nicht entprelltes) Reset-Signal z.B. für externe Geräte.

Buchse $\overline{OV}$: Ausgang des Overflow-Flipflops. Durch Verbinden dieser Buchse mit der BY-Buchse kann der Maschinenbefehl JBY als Sprungbefehl bei eingetretenem arithmetischem Überlauf betrachtet werden (siehe Beschreibung von JBY).

Reset-Taste: Bei Betätigen der Reset-Taste werden der Befehlszähler und alle Register auf Null gesetzt. Das Overflow-Flipflop wird ebenfalls auf Null gesetzt, während das Zero-Flipflop auf High gesetzt wird, da alle Register ja Null sind. Speicherinhalte werden dadurch nicht beeinflußt. Die Reset-Taste zeigt nur eine Wirkung, wenn der Taktgenerator ruht. Damit ist ausgeschlossen, daß die Reset-Taste versehentlich während der Ausführung eines Programmes bei Betätigung eine Wirkung zeigt und dadurch Befehlszähler und Registerinhalte gelöscht werden. Der Taktgenerator wird nicht in Grundstellung gebracht.

Um ein Programm ausführen zu können, muß es erst einmal in den Arbeitsspeicher gebracht werden. Dies muß insbesondere deswegen möglichst bequem sein, da als Arbeitsspeicher ein Halbleiterspeicher benutzt wird, der beim Ausfall bzw. Abschalten der Stromversorgung seine Information verliert. Wird in einen noch leeren Arbeitsspeicher ein erstes Programm gebracht, nennt man dies "Urladen". Weiterhin muß man bei einem Rechner die Möglichkeit haben, ohne softwareseitige Hilfsmittel (Monitorprogramme, Betriebssysteme) Inhalte von Registern und Speicherzellen auch mitten in der Ausführung eines Programmes zu verändern oder anzusehen. Diese beiden Betriebsweisen werden durch die Schalter Urladen (URL) und manuelle Befehlsausführung (MB) eingestellt.

Der Rechner hat somit insgesamt drei verschiedene Betriebsarten, die man genau verstehen muß:

Normalbetrieb: Schalter Urladen und manuelle Befehlsausführung auf "OFF"

Diese Betriebsart ist der Normalfall. Hier liegt ein Programm im Arbeitsspeicher vor und wird ausgeführt, indem Befehl für Befehl aus ihm herausgeholt und abgearbeitet wird.

Manuelle Befehlsausführung: Schalter Urladen auf "OFF", Schalter manuelle Befehlsausführung auf "ON"

Diese Betriebsart unterscheidet sich vom Normalbetrieb dadurch, daß der abzuarbeitende Befehl von einem der Eingabekanäle geholt wird und nicht aus dem Arbeitsspeicher. Da mit dem Schalterregister z.B. nur ein 4-Bit Wort zur Verfügung steht, ein Befehl aber bis zu drei 4-Bit Worte lang sein kann,

muß mit dem Taktgenerator in Einzeltakt-Stellung ein Wort nach dem anderen in das Befehlsregister und seinen Adreßteil gebracht werden. Dabei muß man die durch acht Leuchtdioden angezeigten Systemtakte $\overline{T0}$ bis $\overline{T7}$ beobachten, um zu wissen, wann und wielange man welchen Teil des auszuführenden Maschinenbefehls am Eingabekanal anlegen muß. Bei den Systemtakten $\overline{T0}$, $\overline{T2}$, $\overline{T4}$ muß jeweils das Wort, das ins IR, ADR0, ADR1 soll, an den Bus gelegt werden. Der Befehlszähler wird bei Eingabe der bis zu drei Befehlsteile nicht jedesmal um 1 erhöht, wie dies in der Betriebsweise Normalbetrieb der Fall sein muß. Er kann nur durch Sprungbefehle verändert werden. Jeder Befehl kann in dieser Betriebsart ausgeführt werden.

Urladen: Schalter Urladen auf "ON",
Schalter Manuelle Befehlsausführung auf "OFF"

In dieser Betriebsart werden Daten und Programme in den Arbeitsspeicher geladen, die an einem der Eingabekanäle (z.B. dem Schalterregister) anstehen. Nach Abspeichern eines Wortes im Einzelbefehl-Modus wird der Befehlszähler jeweils um 1 erhöht und zeigt somit auf die Adresse der nächsten zu ladenden Speicherzelle.

Sind beide Schalter Urladen und manuelle Befehlsausführung auf "ON", so gilt die Betriebsart Urladen.

Durch die prinzipielle Bedeutung dieser Betriebsarten, darf die Umschaltung zwischen diesen Betriebsarten Normalbetrieb, Urladen und manuelle Befehlsausführung nur dann erfolgen, wenn der Taktgenerator zweckmäßigerweise durch Einschalten des Einzelbefehls-Schalters gestoppt wurde. Wird z.B. während der Ausführung eines Programmes im Normalbetrieb auf Urladen umgeschaltet, werden Speicherzellen durch die an einem der Eingabekanäle anstehende Information überschrieben.

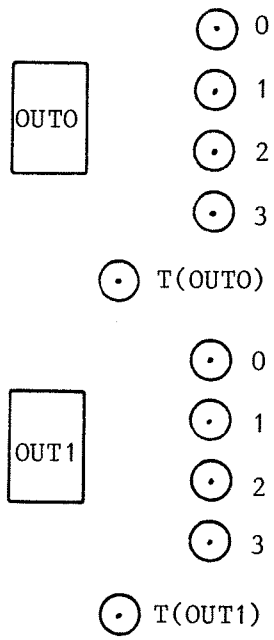
Diese Betriebsarten werden durch einfache Beispiele am Ende des Abschnittes "Programmierung des Mikrocomputers" verdeutlicht.

Schauen wir uns zunächst noch die anderen Bedienungselemente des Rechners an. Sie sind auf dem Ein/Ausgabeteil im Datenverarbeitungsteil rechts auf der Platine angeordnet.

Wichtig: Was die beiden Anschlüsse zur Stromversorgung des Rechners betrifft, so beachte man unbedingt die Hinweise am Anfang der Anleitung.

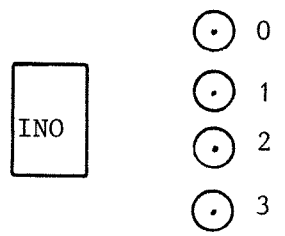
Alle Buchsen stellen Ein-, Ausgänge oder Steuersignale dar, an die die Rechnerperipherie, d.h. irgendwelche externen Geräte angeschlossen werden können. Damit der Datenaustausch überhaupt möglich ist, muß diese Schnittstelle zur Außenwelt sowohl von der funktionellen als auch technischen Seite genau definiert sein. Betrachten wir uns diese Schnittstelle nun etwas genauer:

6.1 OUT0 und OUT1 Register (Ports)

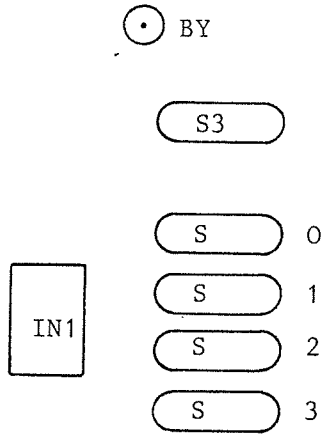


Beide Register gestatten es, vom Programm her über die Befehle OUT0 n und OUT1 Daten (jeweils 4 Bit) auszugeben, die dann an den Buchsen anliegen. Wurde der Befehl OUT0 '1o11' gegeben, so liegt danach am Ausgang des OUT0 Registers die binäre Information '1011' in positiver Logik an. Eine log. 1 entspricht einem TTL-Spannungspegel von > 2.0 V, eine log. 0 einem Spannungspegel von < 0.8 V. Die Ausführung des OUT0 bzw. OUT 1 Befehls bewirkt außerdem, daß an den Steuerausgängen T (OUT0) bzw. T (OUT1) ein Signal erscheint, das gestattet, die anstehende Information in den geräteeigenen Speicher zu übertragen (siehe "Anschluß ext. Geräte").

6.2 IN0 und IN1 Eingabekanäle (Ports)



Beide Kanäle ermöglichen es, vom Programm her 4-Bit Daten mittels des INA Befehls in den Akkumulator zu laden. Über vier Buchsen kann Information von externen Geräten an den IN0 Kanal angelegt werden.



Die Information am IN1 Kanal ist durch die Stellung der vier Schalter festgelegt. Da Schalter speichernde Elemente sind, kann dieser Eingabekanal als Schalterregister betrachtet werden.
Die BY-Buchse ist ein Eingang, an den ein Signal angelegt werden kann, das sich im Programm über den JBY-Befehl abfragen läßt.
Der Schalter S 3 legt fest, welcher der Eingabekanäle INO oder IN1 durch den INA Befehl angesprochen wird.

7. Programmierung des Mikrocomputers

Nachdem wir in den vorangegangenen Abschnitten gesehen haben, welcher Befehlssatz uns zur Verfügung steht und wie der Rechner bedient wird, können wir damit anfangen, Programme zu schreiben und ablaufen zu lassen. Zuvor sollten wir uns jedoch den Ablauf eines einzigen Maschinenbefehls vor Augen führen.

Die Ausführung eines Maschinenbefehls wird durch die Systemtakte $\overline{T0}$ bis $\overline{T7}$ in acht Teilschritte unterteilt. Sie werden durch die acht verschiedenfarbigen Leuchtdioden (LED's) im Taktgenerator angezeigt, wobei jeder Teilschritt eine genau definierte Bedeutung hat. Ein Teilschritt wird dann ausgeführt, wenn die dazugehörende Leuchtdiode erlischt und die des nächsten Teilschrittes aufleuchtet.

Grob kann man bei der Ausführung eines Maschinenbefehls zwei Phasen unterscheiden:

- die Fetch-Phase, das Laden des auszuführenden Befehls aus dem Arbeitsspeicher (rote und grüne Leuchtdioden) und

- die Execute-Phase, das eigentliche Ausführen des Befehls (gelbe Leuchtdioden).

Fetch-Phase:

Die ersten 6 Teilschritte werden dazu benutzt, den Befehl selbst aus dem Arbeitsspeicher zu holen:

rote LED: $\overline{T0}$

Der Befehlszählerinhalt wird als Adresse über SEL1 an den Speicher gelegt und zeigt auf das erste Wort des Befehls, das den Befehlscode enthält. Dadurch wird der Inhalt dieser Speicherzelle auf den Bus gelegt und dort angezeigt. Beim Wechsel des Systemtaktes $\overline{T0}$ auf $\overline{T1}$ wird er in das Befehlsregister IR übernommen und sofort dekodiert. Die diesem Befehl entsprechende Leuchtdiode in der Mikroprogrammatrix leuchtet auf, und der Rechner kann sofort erkennen, um welchen Befehl es sich handelt.

grüne LED: $\overline{T1}$

Der Inhalt des Befehlszählers wird beim Wechsel von $\overline{T1}$ auf $\overline{T2}$ um eins erhöht, damit er auf die nächste Speicheradresse zeigt. Da der Rechner bei $\overline{T0}$ schon erkannt hat, ob der Befehl ein, zwei oder drei Worte lang ist, muß er nun entsprechend reagieren.

rote LED: $\overline{T2}$

Ein-Wort-Befehl: keine Wirkung, da der Befehl ja schon geladen ist.

Zwei-Wort-Befehl: Enthält der Befehl im Adreßteil den Operanden für eine arithmetische oder logische Operation, wird das zweite Wort beim Übergang von $\overline{T2}$ auf $\overline{T3}$ in das Hilfsregister TR übernommen.

Zwei-Wort- und Drei-Wort-Befehl: Da die Adresse des zweiten Wortes des Befehls über SEL1 schon am Speicher anliegt, ist der Inhalt dieser Zelle auf dem Bus und wird beim Übergang von $\overline{T2}$ auf $\overline{T3}$ in den Adreßteil ADR0 (niederwertige 4 Bit) übernommen.

grüne LED: $\overline{T3}$

Ein-Wort-Befehl: keine Wirkung, da Befehl schon geladen ist.

Zwei-Wort- und Drei-Wort-Befehl: Der Inhalt des Befehlszählers wird beim Wechsel von $\overline{T3}$ auf $\overline{T4}$ um eins erhöht, damit er auf die nächste Speicheradresse zeigt.

rote LED: $\overline{T4}$

Ein-Wort- und Zwei-Wort-Befehl: keine Wirkung, da Befehl schon geladen ist.

Drei-Wort-Befehl: Da die Adresse des dritten Wortes des Befehls schon am Speicher anliegt, ist der Inhalt dieser Speicherzelle schon auf dem Bus und wird beim Übergang von $\overline{T4}$ auf $\overline{T5}$ in den Adreßteil ADR1 (höherwertige 4 Bit) übernommen.

grüne LED: Ein-Wort- und
$\overline{T5}$ Zwei-Wort-Befehl: keine Wirkung, da Befehl schon geladen ist.

Drei-Wort-Befehl: Alle Drei-Wort-Befehle enthalten in ihrem Adreßteil eine Speicheradresse. Sie wird nun als Adresse an den Speicher gelegt (vorher war es die Befehlszähleradresse), der Inhalt der adressierten Speicherzelle erscheint auf dem Bus. Muß der Inhalt dieser Speicherzelle noch in der ALU verarbeitet werden, wird er beim Wechsel von $\overline{T5}$ auf $\overline{T6}$ in das Hilfsregister übernommen. Gleichzeitig muß noch der Inhalt des Befehlszählers um eins erhöht werden.

Nach diesen Teilschritten ist der Befehl vollständig aus dem Arbeitsspeicher geholt und der Befehlszähler zeigt auf die Adresse des nächsten Befehls (falls es sich nicht um einen Sprungbefehl handelt, der ausgeführt wird). Einige Vorbereitungen, z.B. das Laden des Hilfsregisters wurden ebenfalls getroffen. Bei den mit roten Led's gekennzeichneten Takten wird jeweils die Businformation in das Register IR, ADR0 oder ADR1 übernommen, bei den mit grünen Led's gekennzeichneten Teilschritten wird der Befehlszähler erhöht.

Nun kann die eigentliche Ausführungsphase beginnen:

Execute-Phase:

gelbe LED: Sprungbefehle: Ist die Sprungbedingung nicht erfüllt, so bleibt
$\overline{T6}$ dieser Schritt ohne Wirkung. Ist die Sprungbedingung erfüllt, so wird beim Übergang von $\overline{T6}$ auf $\overline{T7}$ der Befehlszählerinhalt durch den Inhalt des Adreßteils ersetzt. Handelt es sich um den CALL Befehl, wird zuvor der Befehlszählerstand (er entspricht der Rücksprungadresse) ins Rücksprungadressenregister RAR gerettet.

übrige Befehle
außer WAIT: Die ALU verknüpft den Inhalt des Akkumulators mit dem des Hilfsregisters. Es kann sich auch nur um bloßes Durchschalten des Akkumulator-Hilfsregister- oder Eingabekanalinhaltes auf den Bus handeln. Das Ergebnis wird beim Übergang von $\overline{T6}$ auf $\overline{T7}$ in den Speicher oder das betreffende Register übernommen. Evtl. wird das Overflow- und Zero-Flipflop gesetzt.

gelbe LED: $\overline{T7}$ — Beim Übergang von $\overline{T7}$ auf $\overline{T0}$ wird der Inhalt des Befehlsregisters IR, seines Adreßteils ADR1/ADR0 und der Inhalt des Hilfsregisters gelöscht. Dieser Schritt wird nur aus didaktischen Gründen durchgeführt, da die Inhalte dieser Register ja beim nächsten Befehl sowieso überschrieben werden würden.

Die Leuchtdiode des Systemtaktes $\overline{T0}$ leuchtet nun wieder auf und der nächste Befehl kann ausgeführt werden.

Die oben gegebene Beschreibung eines Befehlsablaufes trifft nur auf die Betriebsart Normalbetrieb zu. Die anderen Betriebsarten manuelle Befehlsausführung und Urladen unterscheidet sich davon folgendermaßen:

Manuelle Befehlsausführung

Der Ablauf eines Maschinenbefehls gleicht dem in der Betriebsart Normalbetrieb mit folgenden Ausnahmen:

- Während der Teilschritte $\overline{T0}$, $\overline{T2}$ und $\overline{T4}$ (rote Led's) liegt nicht der Inhalt einer Speicherzelle auf dem Bus, sondern einer der beiden Eingabekanäle. Zu diesen Zeitpunkten muß der richtige Teil eines Befehls auf den Bus gelegt werden.

- Beim Wechsel $\overline{T1}/\overline{T2}$, $\overline{T3}/\overline{T4}$ und $\overline{T5}/\overline{T6}$ wird der Befehlszählerinhalt nicht um eins erhöht, da der Befehl ja nicht aus dem Speicher geholt wurde.

Urladen

Hier wird während der Teilschritte $\overline{T0}$ bis $\overline{T7}$ der Inhalt einer der beiden Eingabekanäle auf den Bus gelegt und beim Übergang von $\overline{T7}$ auf $\overline{T0}$ in den Speicher übernommen. Gleichzeitig wird der Befehlszähler um eins erhöht und zeigt nun auf die Adresse der nächsten Speicherzelle, die geladen werden soll. Alle Register sind gelöscht, das Zero-Flipflop gesetzt.

Nun wird es höchste Zeit, sich all diese Schritte anhand eines praktischen Beispiels klar zu machen. Schauen wir uns zunächst einmal das folgende aus nur zwei Befehlen bestehende Programm an:

Adresse	mnemotechnische Abkürzung	Code	Bedeutung
0	INCA 1	1101	Erhöhe Akkumulator um 1
1		0001	
2	JMP 0	0000	Springe nach Adresse 0
3		0000	
4		0000	

Ganz links ist die Speicheradresse angegeben, danach folgt die symbolische Bezeichnung für den Maschinenbefehl (mnemonischer Code), danach der Inhalt der entsprechenden Speicherzelle als Bitmuster und ganz zum Schluß ein eventueller Kommentar. An diese bei Rechnern übliche Notation bei der Programmierung in einer Maschinensprache sollte man sich halten.

Zunächst muß dieses Programm erst einmal in den Arbeitsspeicher geladen werden. Dies geschieht, wie im Abschnitt "Bedienung" beschrieben, folgendermaßen:

- Drehschalter Taktfrequenz Stufe 2, Poti ganz rechts
- Einzeltakt-Schalter aus
- Einzelbefehl-Schalter ein
- falls Taktgenerator noch nicht in Grundstellung, Drücken der Start-Taste
- Schalter für manuelle Befehlsausführung aus
- Schalter Urladen ein
- Schalter S3 für IN0/IN1 Auswahl auf IN1 (Schalterregister)
- Reset-Taste drücken

Danach muß der Taktgenerator in Grundstellung sein, alle Register sind auf Null gesetzt und die Information, die am Schalterregister ansteht, muß auf den Datenbus geschaltet sein.

Stellen wir nun am Schalterregister das erste 4-Bit Wort '1101' ein (Reihenfolge der Bit's beachten!), so erscheint es sofort auf dem Bus und wird nach Drücken der Start-Taste in den Speicher geladen. Danach muß der Befehlszähler den Inhalt '00000001' anzeigen, d.h. die nächste zu ladende Adresse ist eins. Danach wird das nächste Wort '0001' am Schalterregister eingestellt, wieder die Start-Taste gedrückt u.s.f. Wurde das letzte Wort auf diese Weise in Adresse vier geladen, ist das Programm im Arbeitsspeicher.

Nun kann es ausgeführt werden und dies geschieht folgendermaßen:

- Einzeltakt-Schalter aus
- Einzelbefehl-Schalter aus

Der Taktgenerator muß in Grundstellung sein

- Schalter manuelle Befehlsausführung aus
- Schalter Urladen aus
- Reset-Taste drücken
- Start-Taste drücken

Nach dem Drücken der Start-Taste wird das Programm gestartet und läuft solange, bis es durch Einschalten des Einzelbefehls-Schalters gestoppt wird. Das Programm erhöht laufend den Inhalt des Akkumulators.

Um ein Gefühl für die Arbeitsgeschwindigkeit zu bekommen, variiere man die Taktfrequenz mit dem Drehschalter und dem Potentiometer.

- Einzelbefehl-Schalter ein

Das Programm hält nun entweder auf Adresse Null (INCA 1 Befehl) oder auf Adresse 2 (JMP 0 Befehl). Will man die Betriebsart Einzelbefehl ausprobieren, so muß folgendes geschehen:

- Einzelbefehl-Schalter ein (ist ja schon getan)
- Einzeltakt-Schalter aus
- Reset-Taste drücken (um sicher auf Adresse Null zu starten)

Nach dem ersten Drücken der Start-Taste wird der Akkumulatorinhalt um eins erhöht und der Befehlszähler zeigt nun auf den nächsten Befehl, der auf Adresse zwei liegt. Drückt man wiederum die Start-Taste, wird dieser Befehl, der Sprungbefehl auf Adresse Null, ausgeführt, d.h. der Befehlszähler wird auf Null gesetzt. Beim nächsten Drücken der Start-Taste wird der Akkumulatorinhalt wiederum um eins erhöht usw..

Schauen wir uns nun einmal die Ausführung des Programmes in der Betriebsart Einzeltakt an:

- Reset-Taste drücken
- Schalter Einzelbefehl aus
- Schalter Einzeltakt ein

Ist der Taktgenerator in Grundstellung ($\overline{T0}$), liegt die Adresse Null am Speicher und der Inhalt dieses Wortes '1101' liegt auf dem Bus (Anzeige!). Nach Drücken der Start-Taste wird diese Information in das IR übernommen. Die Spalte 13 der Mikroprogrammatrix wird aktiviert (Led!), der Rechner hat somit erkannt, daß es sich um den INCA-Befehl handelt. Beim nächsten Drücken der Start-Taste wird der Befehlszähler um eins erhöht und zeigt auf das zweite Wort des INCA-Befehls mit dem Inhalt '0001', das nun auf dem Bus liegt. Nach erneutem Drücken der Start-Taste wird die Businformation in das ADRO und das Hilfsregister TR übernommen und wieder nach neuer Betätigung der Befehlszähler um eins erhöht. Der INCA-Befehl besteht nun nur aus zwei Worten, sodaß sich beim nächsten und übernächsten Drücken der Start-Taste in den Registern nichts tut. Auf dem Bus kann die Information wechseln, was aber in diesem Falle bedeutungslos ist. Während des Systemtaktes $\overline{T6}$ liegt die Summe aus Akkumulatorinhalt (noch Null) und Hilfsregisterinhalt (eins) auf dem Bus und wird bei Drücken der Start-Taste in den Akkumulator übernommen. Nachdem die Taste nochmals betätigt wurde, wird IR, ADRO, ADR1 und TR gelöscht, der Taktgenerator befindet sich wiederum in Grundstellung und der nächste Befehl auf Adresse zwei (JMP 0) kann ausgeführt werden.

Es würde nun zu weit führen, die Ausführung eines jeden der 16 Maschinenbefehle in Einzeltakt-Betriebsweise zu beschreiben. Der Leser möge die anschließenden Beispiele in dieser Betriebsweise durchführen. Dabei kann er sich die am Anfang des Abschnittes detailliert beschriebene Ausführung eines Befehls anhand der 8 Systemtakte klarmachen. Ist ihm dies gelungen, wird er keine Schwierigkeiten mehr haben, z.B. den zeitlichen Ablauf von Maschinenbefehlen auch in anderen Rechnern ("Timing-Diagramme") zu verstehen.

7.1 Einfache Programme zur Verdeutlichung der Maschinenbefehle

1. Schauen wir uns zunächst noch einmal das folgende aus nur zwei Befehlen bestehende Programm an:

```
0    INCA  1      1101      Erhöhe Akkumulator um 1
1                 0001

2    JMP   0      0000      Springe nach Adresse 0
3                 0000
4                 0000
```

Das Programm bewirkt eine laufende Erhöhung des Akkumulatorinhaltes um eins. Man ändere auch die Arbeitsfrequenz und lasse das Programm in der Betriebsart Einzeltakt laufen, um den Befehlsablauf anhand des zeitlichen Ablaufes eines Befehls zu verstehen.

2. Um den WAIT-Befehl kennenzulernen:

0	WAIT		1111	kurze Zeit warten
1	INCA	15	1101	Subtrahiere vom Akkumulator eine 1
2			1111	(1 entspricht im 2er-Kompl. 15)
3	JMP	0	0000	Springe nach Adresse 0
4			0000	
5			0000	

Hier drehe man am Potentiometer des WAIT-Flipflops.

3. Die Befehle LDA adr, STA adr und ENTA n:

0	ENTA	12	1100	Setze Akkumulatorinhalt auf 12
1			1100	
2	STA	255	0101	Speichere Akkumulatorinhalt nach
3			1111	Adresse 255
4			1111	
5	ENTA	0	1100	Setze Akkumulatorinhalt auf 0
6			0000	
7	LDA	255	0100	Lade den Akkumulator mit Inhalt von
8			1111	Adresse 255. Ergebnis: die 12 ist
9			1111	wieder im Akkumulator.
10	JMP	10	0000	Springe nach Adresse 10 !! Trick !!
11			1010	Unendliche Schleife, "auf der Stelle
12			0000	treten", mögliche Version eines Stop-Befehles.

4. Der NANDA-Befehl bietet die Möglichkeit, den Akkumulator bit-weise abzufragen:

0	ENTA	'1101'	1100	Setze den Akkumulator auf '1101'
1			1101	
2	NANDA	'0110'	1110	Bitweise NAND-Verkn. mit '0110':
3			0110	danach '1011' im Akkumulator.
4	NANDA	'1111'	1110	Bitweise NAND-Verkn. mit '1111':
5			1111	entspricht Invertierung des ACC.

```
6       JMP     6       0000    Springe nach 6    , unendliche
7                       0110    Schleife
8                       0000
```

5. Was macht der JBY-Befehl?

```
0       INCA    1       1101    Erhöhe Akkumulator um 1
1                       0001

2       JBY     2       1011    Springe nach 2, wenn BY Low ist
3                       0010    (dann unendliche Schleife!)
4                       0000

5       JMP     0       0000    Springe nach Null
6                       0000
7                       0000
```

Der Akkumulatorinhalt wird nur dann, wenn die Buchse BY auf High liegt, erhöht. Liegt sie auf Low, ist die Sprungbedingung des JBY-Befehls erfüllt und der Rechner läuft auf Adresse 2 in einer Schleife solange, bis BY-Signal wieder auf High ist.

6. Ein- und Ausgabe von Daten, INA, OUTO, OUT1:

```
0       OUTO    '1101'  1000    Setze OUTO auf '1101'
1                       1101

2       INA             1010    Hole Daten von INO oder IN1 (S3)

3       OUT1            1001    Gib ACC auf OUT1 aus

4       JMP     0       0000    Springe nach Adresse 0
5                       0000
6                       0000
```

Mit diesem Programm wird die Information, die, je nach Stellung von Schalter S3, an INO oder IN1 ansteht, über den Akkumulator am OUT1-Register wieder ausgegeben.

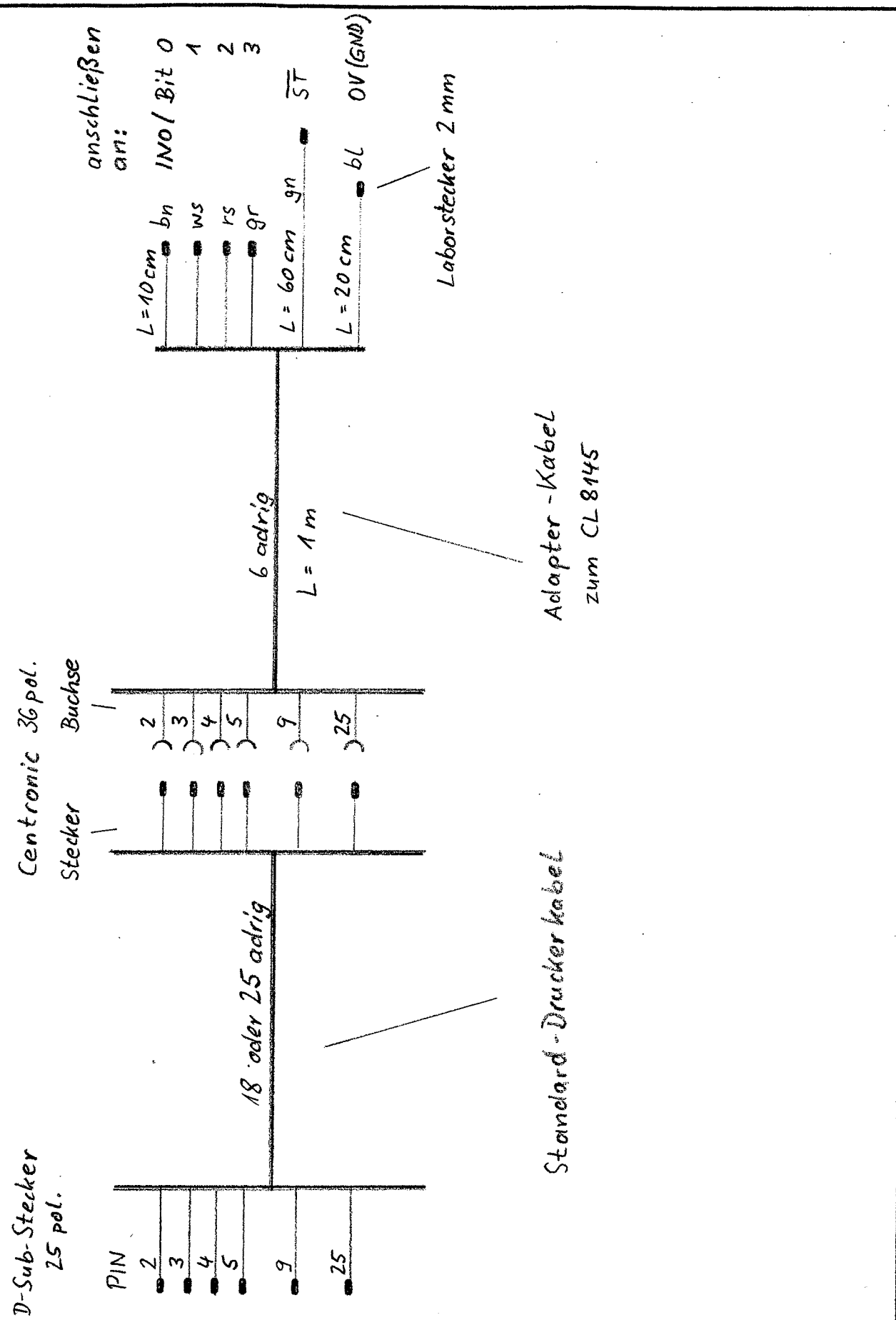
7. Bedingter Sprung mit JNZ adr:

```
0       OUT1            1001

1       INCA    1       1101
2                       0001

3       JNZ     0       0001
4                       0000
5                       0000

6       WAIT            1111
```



Maßstab	Gezeichnet	26.7.96	Wi	Änd. Zust.	Datum	Name
	Geprüft			a)	28.7.96	Wi
	Normgepr.					

CONATEX

Maße ohne Toleranz-angabe nach: DIN 7168 mittel	Adapter-Kabel PC – CL 8145	Zeichnungs-Nr.	Insges.	Blatt	Blatt-Nr.

7	JMP	0	0000
8			0000
9			0000

Das Programm zählt im Register OUT1 von 0 bis 15 und stoppt dann kurz.

8. Unterprogramm mit Rücksprungadresse im RAR:

0	OUT1		1001	
1	INCA	1	1101	
2			0001	
3	JNZ	0	0001	
4			0000	
5			0000	
6	CALL	128	0010	Sprung auf Adresse 128, auf der
7			0000	sich das Unterprogramm befindet.
8			1000	Rücksprungadr. 9 ist im RAR.
9	JMP	0	0000	
10			0000	
11			0000	
.			.	
.			.	
.			.	
128	WAIT		1111	
129	RET		0011	Rücksprung ins Hauptprogramm Adresse 9

Die Befehle WAIT und RET müssen von Adresse 128 an in den Arbeitsspeicher geladen werden:
In der Betriebsart manuelle Befehlsausführung wird der Befehl JMP 128 ausgeführt (siehe Anfang des Abschnittes und Kapitel "Programmbeispiele"). Danach wird in der Urlade-Betriebsart der WAIT und RET Befehl in den Arbeitsspeicher gebracht.

8. Realisierung des Mikrocomputers mit Digitalbausteinen

Der Rechner ist vorwiegend mit integrierten Digitalbausteinen und nur mit wenigen diskreten Bauteilen wie Widerständen und Kondensatoren aufgebaut. Zur Anzeige der logischen Zustände von Befehlszähler, Register, Flipflops und Datenleitungen wurden ausschließlich Leuchtdioden benutzt. Die Mikroprogrammatrix, die mit Hilfe von integrierten Digitalbausteinen hätte realisiert werden können, wurde aus didaktischen Gründen als Matrix mit diskreten

Siliziumdioden bestückt.

Aus dem Angebot der Logikfamilien wurde die Standard-TTL-Serie ausgewählt, weil sie mehrere Eigenschaften besitzt, die sie zur Realisierung des Modellrechners auszeichnet:

- im Vergleich zu anderen Logikfamilien große Auswahl an Digitalbausteinen von vielen Herstellern
- geeignete Bausteine für die 4 Bit Struktur des Rechners
- genügend große Strombelastbarkeit der Ausgänge insbesondere zur direkten und vollen Aussteuerung der Leuchtdioden
- einfach im Handel zu beschaffen und relativ preiswert
- relativ robust im praktischen Umgang (unempfindlich statischen Aufladungen gegenüber usw.)
- eine einzige Versorgungsspannung von + 5 V
- relativ schnell

Die Nachteile der Standard-TTL-Serie anderen Logikfamilien gegenüber, wie größerer Stromverbrauch, kleinerem Störspannungsabstand sporadisch auftretenden Störungen bei einfachen Netzteilen, ungeeigneter Leiterbahnführung und mangelnder Abschirmung, können den gewichtigen Vorteilen gegenüber vernachlässigt werden.

An dieser Stelle sei auf die umfangreiche Literatur, insbesondere auf die Datenblätter der Hersteller hingewiesen, die sich mit dieser und anderen Logikfamilien beschäftigen. Kurz zusammengefaßt sind im folgenden die wichtigsten elektrischen Eigenschaften:

Standard-TTL-Serie (TTL: Transistor-Transistor-Logik)

- Versorgungsspannung von + 5 V $\pm$ 0.25 V
- logische Pegel: 0 V ..... 0.8 V = log. "0" oder "low"
  2.0 V ..... 5.0 V = log. "1" oder "high"
- Strombelastbarkeit der Ausgänge
  log. "0" : max. 16 mA
  log. "1" : max. 400 µA

Die meisten Ausgänge sind von ihrer Strombelastbarkeit so ausgelegt, daß sie bis zu 10 Eingänge ansteuern können. Hat ein Ausgang dieses sog. FAN-OUT von 10, kann er 10 Eingänge mit einem FAN-IN von 1 ansteuern oder 5 Eingänge mit einem FAN-IN von 2. (siehe Datenblätter)

Zur Kennzeichnung der digitalen Bausteine wenden die meisten Hersteller die von Texas Instruments eingeführte Bezeichnung an:

SN 74 xx N,	Arbeitstemperaturbereich	0 ..... + 70°C
SN 84 xx N,	Arbeitstemperaturbereich	-25 C ..... + 85°C
SN 54 xx N,	Arbeitstemperaturbereich	-55 C ..... + 125°C

wobei xx für eine zwei- oder dreistellige Zahl steht, die den Typ kennzeichnet.

Der einzige integrierte Digitalbaustein, der nicht der TTL-Serie angehört, ist der Arbeitsspeicher, dessen Spannungsversorgung, Ein- und Ausgänge jedoch alle TTL-kompatibel sind.

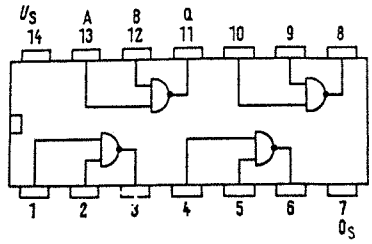
Im Folgenden sind die digitalen Bausteine, die zur Realisierung des Rechners herangezogen wurden, aufgelistet, anschließend einzeln aufgeführt (Anschlußbelegung) und, soweit notwendig, kurz beschrieben:

74 00 74 02 74 08 74 10 7425 7432 74125 74154 74155 74157 74181 74 93 74161	einfache Gatter komplexere Gatterschaltungen Zähler	74 74 74175 74123	Flipflops Monoflops Arbeitsspeicher: statisches RAM, 2101, 1024 Bit 256 x 4 Bit

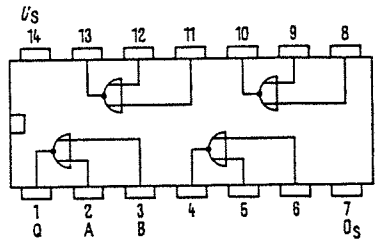
Einfache logische Gatter

Alle Gattereingänge haben ein FANIN von 1, alle Ausgänge ein FANOUT von 10.

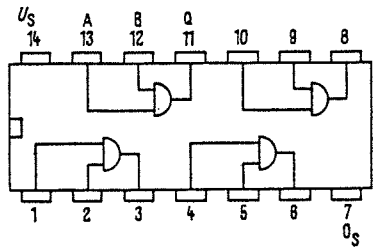
7400: Vier NAND-Glieder mit
je zwei Eingängen



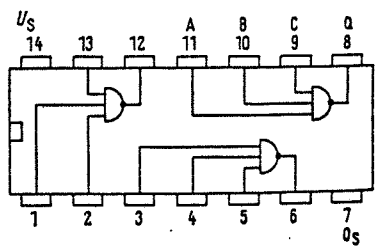
7402: Vier NOR-Glieder mit
je zwei Eingängen



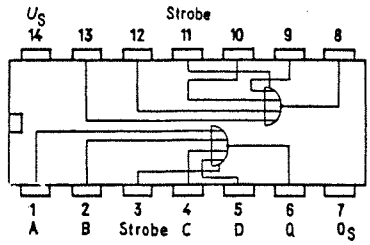
7408: Vier UND-Glieder mit
je zwei Eingängen



7410: Drei NAND-Glieder mit
je drei Eingängen



7425: Zwei NOR-Glieder mit je vier Eingängen und Strobe



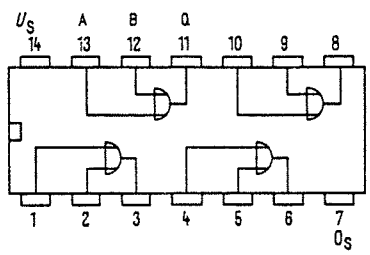
Das Strobe-Signal wird bei der Anwendung im Modellrechner immer auf High gehalten, sodaß sich die beiden Gatter wie NOR-Glieder mit je vier Eingängen verhalten.

Logisches Verhalten

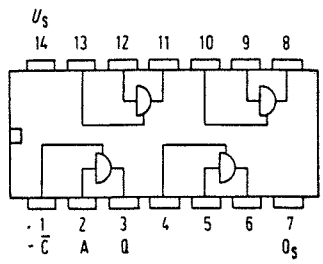
Eingänge					Ausgang
A	B	C	D	Strobe	Q
H	X	X	X	H	L
X	H	X	X	H	L
X	X	H	X	H	L
X	X	X	H	H	L
L	L	L	L	X	H
X	X	X	X	L	H

X≙L- oder H-Signal

7432: Vier ODER-Glieder mit je zwei Eingängen



74125: Vier UND-Stufen mit je einem Eingang, Kontrolleingang und Tri-State-Ausgängen

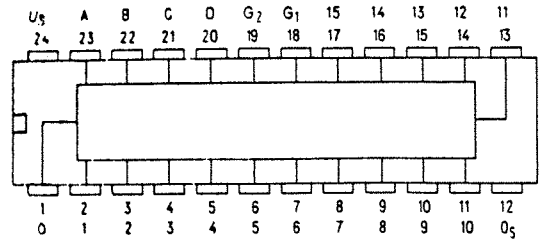


Diese Schaltelemente werden dazu benutzt, 4 Bit Daten auf den Datenbus durchzuschalten. Bei High-Signal an den Kontrolleingängen werden die Ausgänge hochohmig gesperrt, d.h. die Daten werden nicht auf den Bus gegeben.

Logisches Verhalten

Eingang A	Kontroll-Eingang C	Ausgang Q
H	L	H
L	L	L
H	H	hochohmig
L	H	hochohmig

74154: 4 Bit Binär-Dekoder / Demultiplexer



Anschlußanordnung
Ansicht von oben

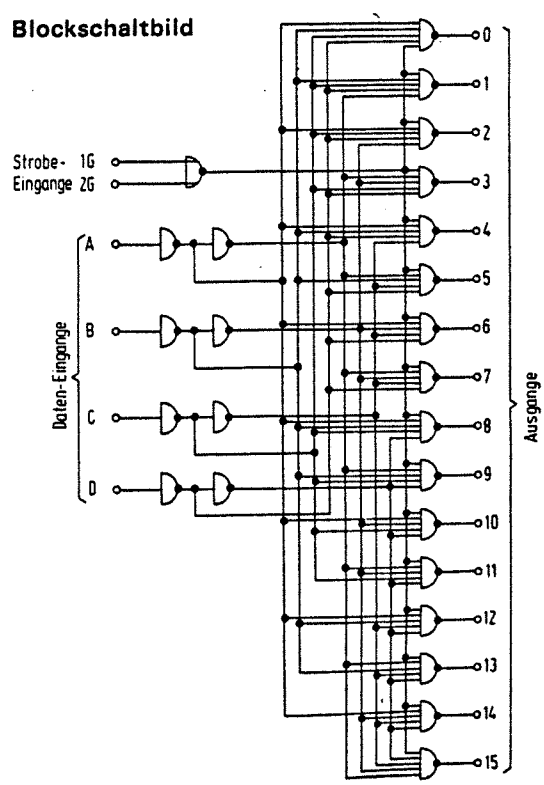
Logisches Verhalten

Eingänge						Ausgänge														
G_1	G_2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

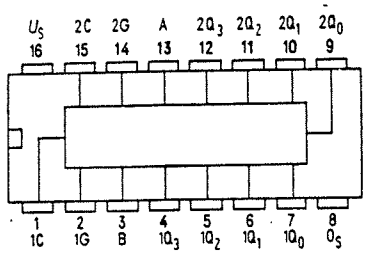
X = H- oder L-Signal

Bei G_1 und G_2 auf Low-Signal wird genau einer der durch die Adresse DCBA angewählten Ausgänge auf Low-Signal geschaltet. Dieser Baustein wird als Spaltendekoder des Mikrobefehls -ROM's verwendet.

Blockschaltbild

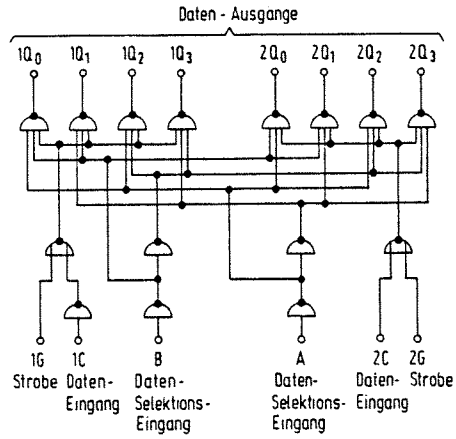


74155: 3 Bit Binärdekoder/Demultiplexer



Anschlußanordnung
Ansicht von oben

Blockschaltbild



Logisches Verhalten als 3-Bit-Binärdekoder und Einfacher Demultiplexer

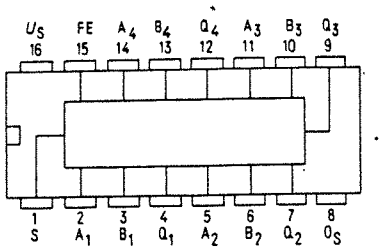
Daten-Selektions-Eingang			Strobe oder Daten-Eing.	Ausgänge							
C[1])	B	A	G[2])	$2Q_0$	$2Q_1$	$2Q_2$	$2Q_3$	$1Q_0$	$1Q_1$	$1Q_2$	$1Q_3$
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	L

X = H- oder L-Signal

C = 1 C und 2 C extern verknüpft
G = 1 G und 2 G extern verknüpft

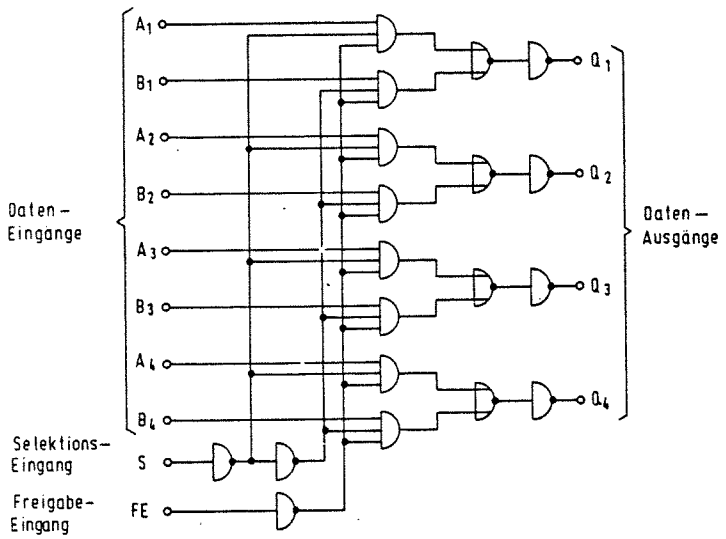
Die Funktionsweise ist analog der des 74154, hier können mit einer drei Bit Adresse jedoch nur acht Ausgänge angesteuert werden. Dieser Schaltkreis wird zur Darstellung der acht Systemtakte im Steuerwerk benutzt.

74157: Vierfach 1 Bit Datenselektor/Multiplexer



Anschlußanordnungen
Ansicht von oben

Blockschaltbild



Logisches Verhalten

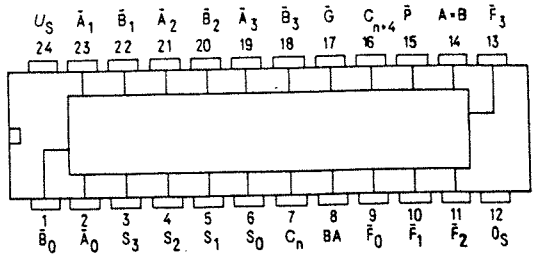
Freigabe-Eingang FE	Selektions-Eingang S	Daten-Eingänge A	Daten-Eingänge B	Daten-Ausgänge Q
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

X=H- oder L-Signal

Mit Hilfe des Selektionseinganges S gestattet es dieser Baustein, eine der beiden 4 Bit Gruppen A1-A4 oder B1-B4 auszuwählen und auf den Ausgang Q1-Q4 durchzuschalten. Im Rechner dienen diese Bausteine als Adreßschalter im Adressenverarbeitungsteil (SEL1 und SEL2).

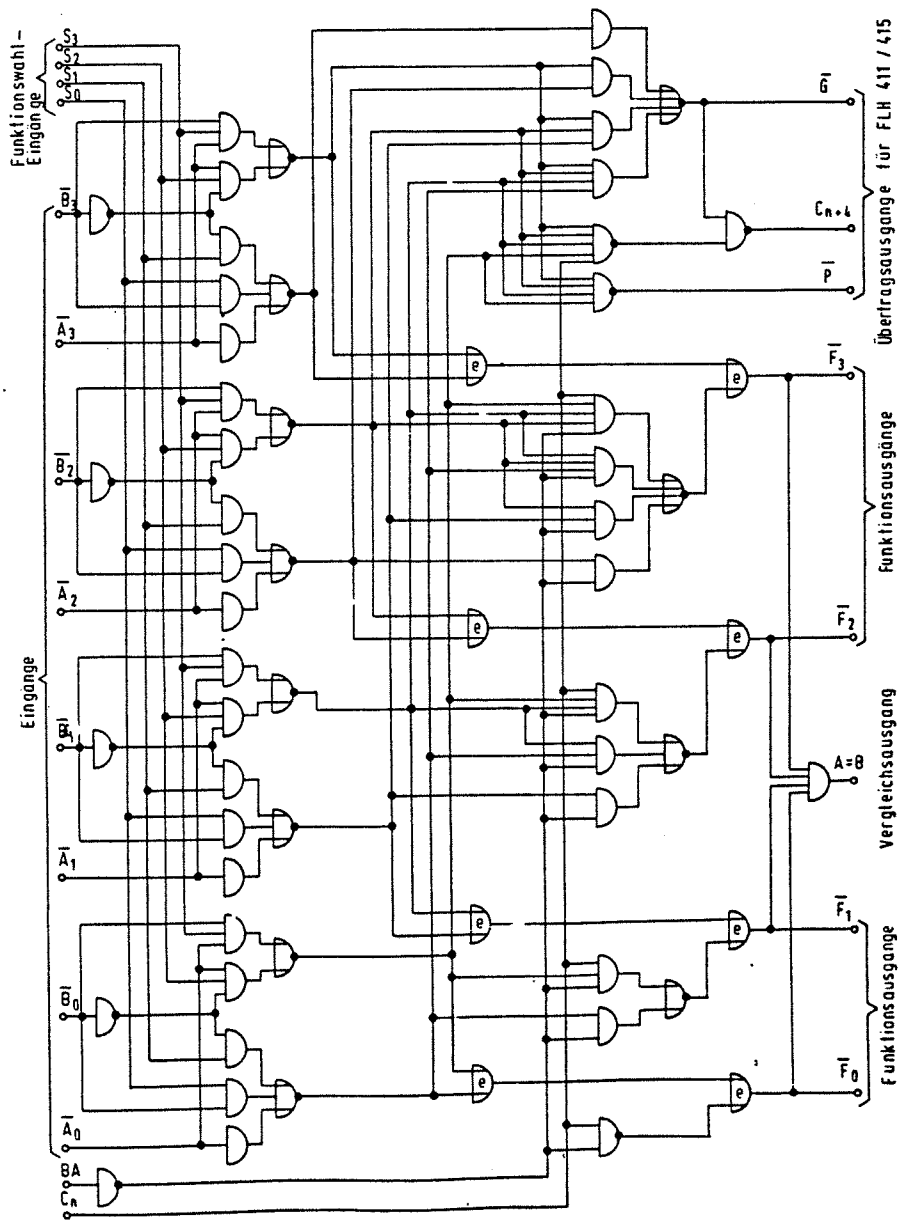
74181:

4 Bit arithmetische Logikeinheit



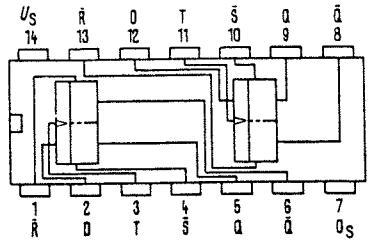
Funktionswahl S_3 S_2 S_1 S_0	Logische Betriebsart BA=H, C_n=X	Arithmetische Betriebsart; BA=L	
		C_n=H=kein Übertrag	C_n=L=Übertrag
L L L L	$F=\bar{A}$	F=A	F=A plus 1
L L L H	$F=\overline{\bar{A} \vee \bar{B}}$	$F=A \vee B$	$F=(A \vee \bar{B})$ plus 1
L L H L	$F=\bar{A} \wedge B$	$F=A \vee \bar{B}$	$F=(A \vee \bar{B})$ plus 1
L L H H	F=L	F=H	F=L
L H L L	$F=\overline{\bar{A} \wedge \bar{B}}$	$F=A$ plus $(A \wedge \bar{B})$	F=A plus $(A \wedge \bar{B})$ plus 1
L H L H	$F=\bar{B}$	$F=(A \vee B)$ plus $(A \wedge \bar{B})$	$F=(A \vee B)$ plus $(A \wedge \bar{B})$ plus 1
L H H L	$F=(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$	F=A minus B minus 1	F=A minus B
L H H H	$F=A \wedge \bar{B}$	$F=(A \wedge \bar{B})$ minus 1	$F=A \wedge \bar{B}$
H L L L	$F=\bar{A} \vee V$	F=A plus $(A \wedge B)$	F=A plus $(A \wedge B)$ plus 1e
H L L H	$F=\overline{(A \wedge \bar{B}) \vee (\bar{A} \wedge B)}$	F=A plus B	F=A plus B plus 1
H L H L	F=B	$F=(A \vee \bar{B})$ plus $(A \wedge B)$	$F=(A \vee \bar{B})$ plus $(A \wedge B)$ plus 1
H L H H	$F=A \wedge B$	$F=(A \wedge B)$ minus 1	$F=A \wedge B$
H H L L	F=H	F=A plus A*	F=A plus A plus 1
H H L H	$F=A \vee \bar{B}$	$F=(A \vee B)$ plus A	$F=(A \vee B)$ plus A plus 1
H H H L	$F=A \vee B$	$F=(A \vee \bar{B})$ plus A	$F=(A \vee \bar{B})$ plus A plus 1
H H H H	F=A	F=A minus 1	F=A

Schaltschema



Diese Schaltung stellt die arithmetisch-logische Einheit (ALU) unseres Rechners dar. Sie verknüpft die Eingänge, die mit A0 bis A3 bezeichnet sind mit den Eingängen Bo bis B3 und gibt das Ergebnis an den Ausgängen F0 bis F3 aus. Die Verknüpfung wird durch die Funktionswahleingänge S0 bis S3, BA und C_n bestimmt.

Flipflops

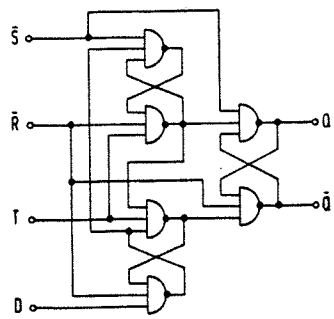


Anschlußanordnung
Ansicht von oben

D = Informationseingang
Q, $\bar{Q}$ = Ausgänge
$\bar{R}$ = Rückstelleingang
$\bar{S}$ = Stelleingang
T = Takteingang

7474:

2 D-Flipflops



Blockschaltbild (ein Flipflop)

Das Zero-Flipflop und Overflow-Flipflop wird durch solch einen Baustein dargestellt. Die anstehende Information wird bei der positiven Taktflanke übernommen.

Logisches Verhalten
(jedes Flipflop)

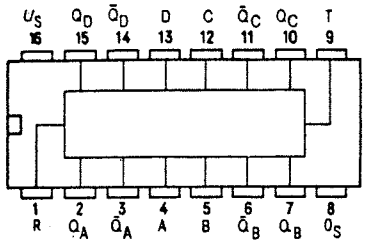
t_n	t_{n+1}	
D	Q	$\bar{Q}$
L	L	H
H	H	L

t_n = Zeitpunkt vor dem Taktimpuls
t_{n+1} = Zeitpunkt nach dem Taktimpuls

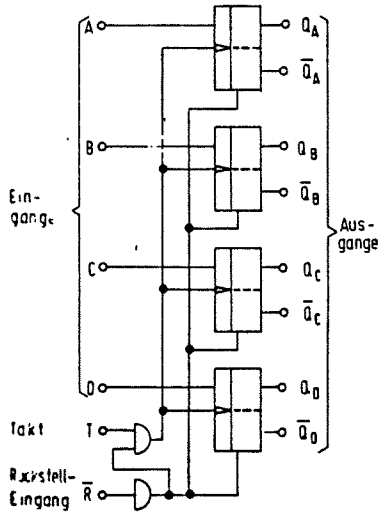
L-Potential an $\bar{R}$ bringt Q auf L-Signal
L-Potential an $\bar{S}$ bringt Q auf H-Signal
$\bar{R}$ und $\bar{S}$ arbeiten unabhängig von T

74175:

Dieser Baustein wird für alle Register des Modellrechners benutzt. An der positiven Taktflanke des gemeinsamen Takteinganges wird die Information, die an den Eingängen ansteht, übernommen und bis zum nächsten Taktsignal gespeichert.



Blockschaltbild



Logisches Verhalten

Eingang bei t_n	Ausgänge bei t_{n+1}	
D	Q	$\bar{Q}$
H	H	L
L	L	H

t_n = Zeitpunkt vor dem Taktimpuls
t_{n+1} = Zeitpunkt nach dem Taktimpuls

L-Potential an $\bar{R}$ legt Ausgänge Q auf L-Signal.
$\bar{R}$ arbeitet unabhängig von T.

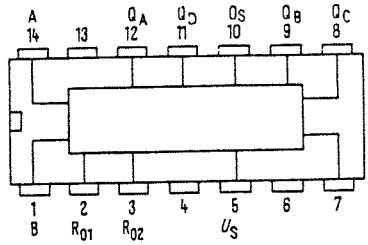
Zähler:

7493:

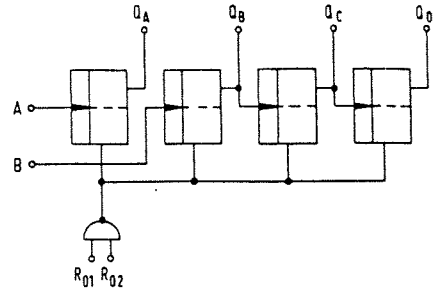
4 Bit Binärzähler

Der Zähler wird im Taktgenerator im Zusammenspiel mit dem Baustein 74155 zur Erzeugung der acht Systemtakte $\overline{T0}$ bis $\overline{T7}$ eingesetzt. Der Zähler zählt an der negativen Taktflanke.

A, B = Zähleingänge
R_{01}, R_{02} = Rückstelleingänge
Q = Ausgänge



Blockschaltbild



Logisches Verhalten (Q_A mit B verbunden)

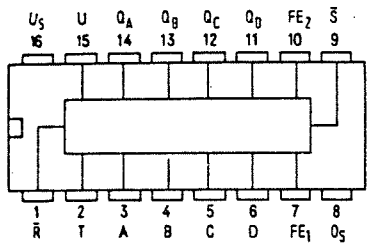
Zählfolge	Ausgänge Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

Anmerkungen:
Um alle Ausgänge auf L-Signal zu setzen, müssen R_{01} und R_{02} auf H-Signal sein.

74161:

Synchroner Binärzähler mit Stelleingängen und Rücksetzeingang.

Mit zwei dieser Bausteine ist der Befehlszähler realisiert.

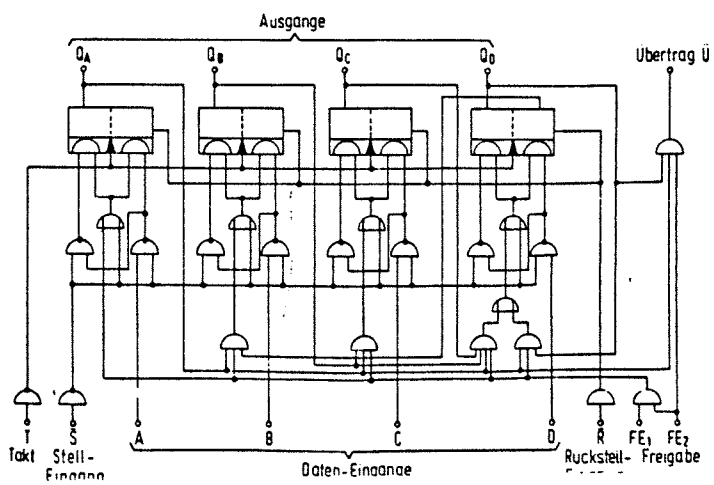


Anschlußanordnung
Ansicht von oben

Logisches Verhalten

		obere Grenze A
H-Ausgangslastfaktor pro Ausgang	F_{QH}	20
L-Ausgangslastfaktor pro Ausgang	F_{QL}	10
Eingangslastfaktor T oder FE_2	F_I	2
übrige Eingänge	F_I	1

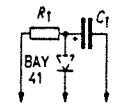
Blockschaltbild

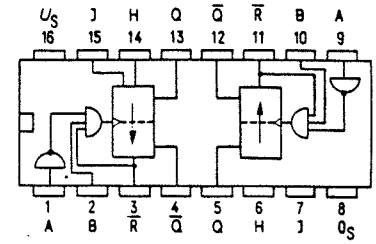


Monoflops

74123: zwei nachtriggerbare monostabile Kippstufen

Sie werden sowohl für den Taktgenerator als auch für das WAIT-Monoflop benutzt.
Die Zeitdauer, die sich der Ausgang Q auf High befindet, wird durch das R_tC_t-Glied bestimmt. Die Diode schützt die Eingänge bei höheren Kapazitäten.

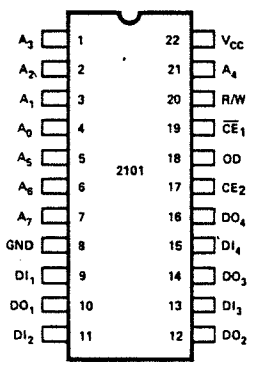




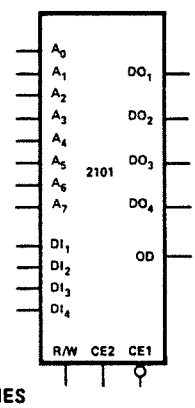
Arbeitsspeicher

Er ist der einzige Baustein, der nicht zur TTL-Serie gehört. Es handelt sich um einen N-MOS, statischen 1024 Bit Speicher, der in 256 Worten zu je 4 Bit organisiert ist. Außerdem besitzt er getrennte Ein- und Ausgänge. Alle Ein- und Ausgänge sind TTL-kompatibel, ebenfalls die Versorgungsspannung von 5 Volt.

PIN CONFIGURATION



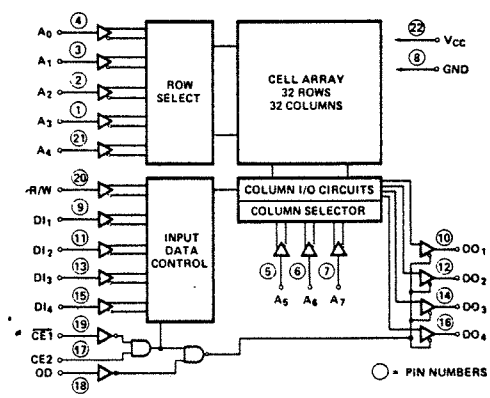
LOGIC SYMBOL



PIN NAMES

DI_1-DI_4	DATA INPUT	$\overline{CE}_2$	CHIP ENABLE 2
A_0-A_7	ADDRESS INPUTS	OD	OUTPUT DISABLE
R/W	READ/WRITE INPUT	DO_1-DO_4	DATA OUTPUT
$\overline{CE}_1$	CHIP ENABLE 1	V_{CC}	POWER (+5V)

BLOCK DIAGRAM



Ein Wort noch zur Nomenklatur der Signalleitungen, die einer Systematik bedarf, da zwischen den integrierten Digitalbausteinen eines solch komplexen Systems wie das eines Rechners sehr viele Querverbindungen bestehen. Allgemein sieht die Bezeichnung einer Signalleitung folgendermaßen aus:

X (Y) oder $\overline{X}$ (Y) oder einfach X oder $\overline{X}$

wobei X stellvertretend für eine Reihe von ein bis vier Buchstaben oder Ziffern steht, die die Bedeutung einer Signalleitung kennzeichnen. Y ist eine Kurzbezeichnung der logischen Einheit, an der das Signal X anliegt.

Beispiel: $\overline{R}$ (ACC) : Rückstell(Reset)eingang des Akkumulators
T (IR) : Takteingang des Instruction Registers

Eine Übersicht aller Steuersignale befindet sich am Ende des Abschnittes.

Wenden wir uns nun der detaillierten Beschreibung des Rechners zu. Auf den Seiten 54 + 55 ist die gesamte Schaltung aufgezeichnet und in Funktionsblöcke aufgeteilt. Diese Funktionsblöcke schauen wir uns nun im einzelnen an.

8.1 Taktgenerator mit Bedienungsfeld

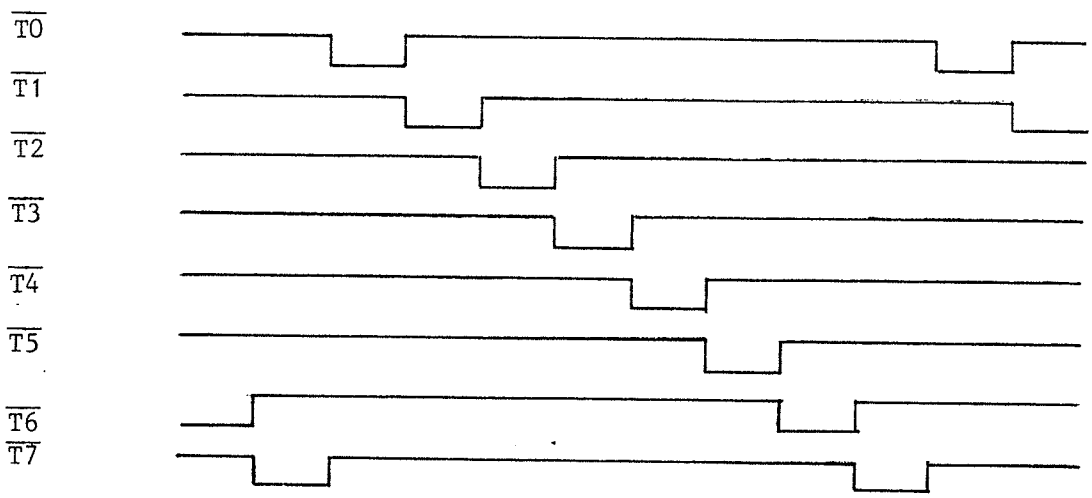
Die Start-Taste wird zunächst durch zwei als RS-Flipflop geschaltet NAND-Gatter zuverlässig entprellt. Um einen kurzen Setzimpuls für das RUN-Flipflop zu bekommen, der einmal unabhängig ist von der Zeitdauer, die die Start-Taste gedrückt ist und der außerdem kürzer sein muß als die Ausführungszeit eines Maschinenbefehls, wird das entprellte Signal über zwei NAND-Gatter zum Setzeingang des RUN-Flipflops geleitet. Das erste dieser beiden NAND's erzeugt zusammen mit dem 500 pF Kondensator eine Laufzeitverzögerung des ankommenden Signals derart, daß für eine Zeit von einigen zehn Nanosekunden beide Eingänge des zweiten NAND's auf High sind, und damit sein Ausgang kurzzeitig auf Low. Dadurch erhält beim Drücken der Start-Taste das RUN-Flipflop einen kurzen Setzimpuls, die RUN-Leuchtdiode leuchtet auf und der nachfolgende Oszillator wird gestartet.

Dieser besteht aus zwei Monoflops (74123), die sich gegenseitig triggern, solange das RUN-Flipflop gesetzt ist. Die Zeitkonstante des ersten Mono-

flops ist über ein Potentiometer kontinuierlich und über den Drehschalter mit mehreren Kondensatoren in groben Stufen einstellbar. Die Zeitkonstante des zweiten Monoflops ist mit einem Trimmpotentiometer fest eingestellt. Die Arbeitsfrequenz des Rechners kann somit mit dem Potentiometer und dem Drehschalter variiert werden, wobei die maximale Arbeitsfrequenz durch die feste Zeitkonstante des zweiten Monoflops begrenzt wird. Sie ergibt sich im wesentlichen aus der Zugriffszeit des Arbeitsspeichers.

An der Buchse $\overline{T}$ kann der Takt abgegriffen werden. Wird die Brücke B herausgenommen, ist der Taktgenerator des Rechners abgetrennt und er kann über diese Buchse mit einem externen Takt betrieben werden. Die Funktion der Start-Taste, des Einzeltakt- und Einzelbefehls-Schalters ist abeschaltet.

Die mit dem Oszillator erzeugte Taktfrequenz wird einem 4 Bit Binärzähler zugeführt. Da dieser aber wegen der acht benötigten Zeitabschnitte für einen Befehl nur von Null bis Sieben zählen darf, wird die höchstwertige Stelle direkt auf den Rücksetzeingang gelegt. Dies hat zur Folge, daß dieses Bit nur sehr kurze Zeit (Zeit für das Rücksetzen des Zählers) auf High ist. Die drei niederwertigen Bits des Zählers werden auf einem Demultiplexer geleitet, an dessen acht Ausgängen die im Folgenden aufgeführten Taktsignale durch Leuchtdioden angezeigt werden und als Systemtakte $\overline{T0}$ bis $\overline{T7}$ für die Erzeugung der weiteren Steuersignale zur Verfügung stehen.



Um den Ablauf eines Programmes zu verstehen, möchte man nun auch die Möglichkeit haben, nicht nur eine kontinuierliche Taktfolge zu erzeugen, sondern auch nur einen Befehl ausführen zu lassen. Dafür darf der Taktgenerator nur eine Folge von acht Takten erzeugen und muß danach stehen bleiben. Dies wird dadurch bewirkt, daß das nach acht Takten auftretende Rücksetzsignal des

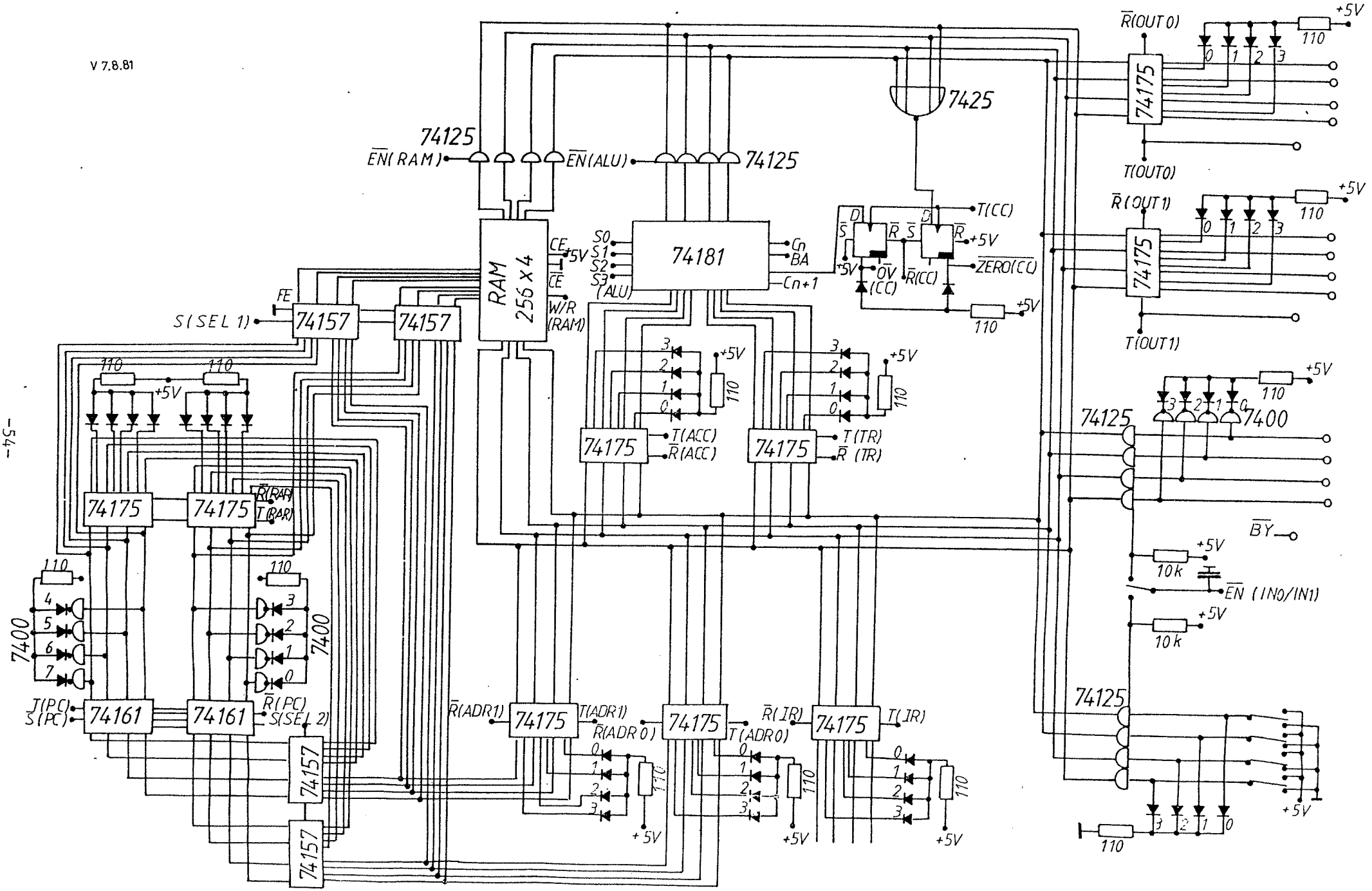
Binärzählers über ein NAND-Gatter den Rücksetzeingang des RUN-Flipflops erreicht, wenn der Schalter Einzelbefehl entsprechend gesetzt ist.

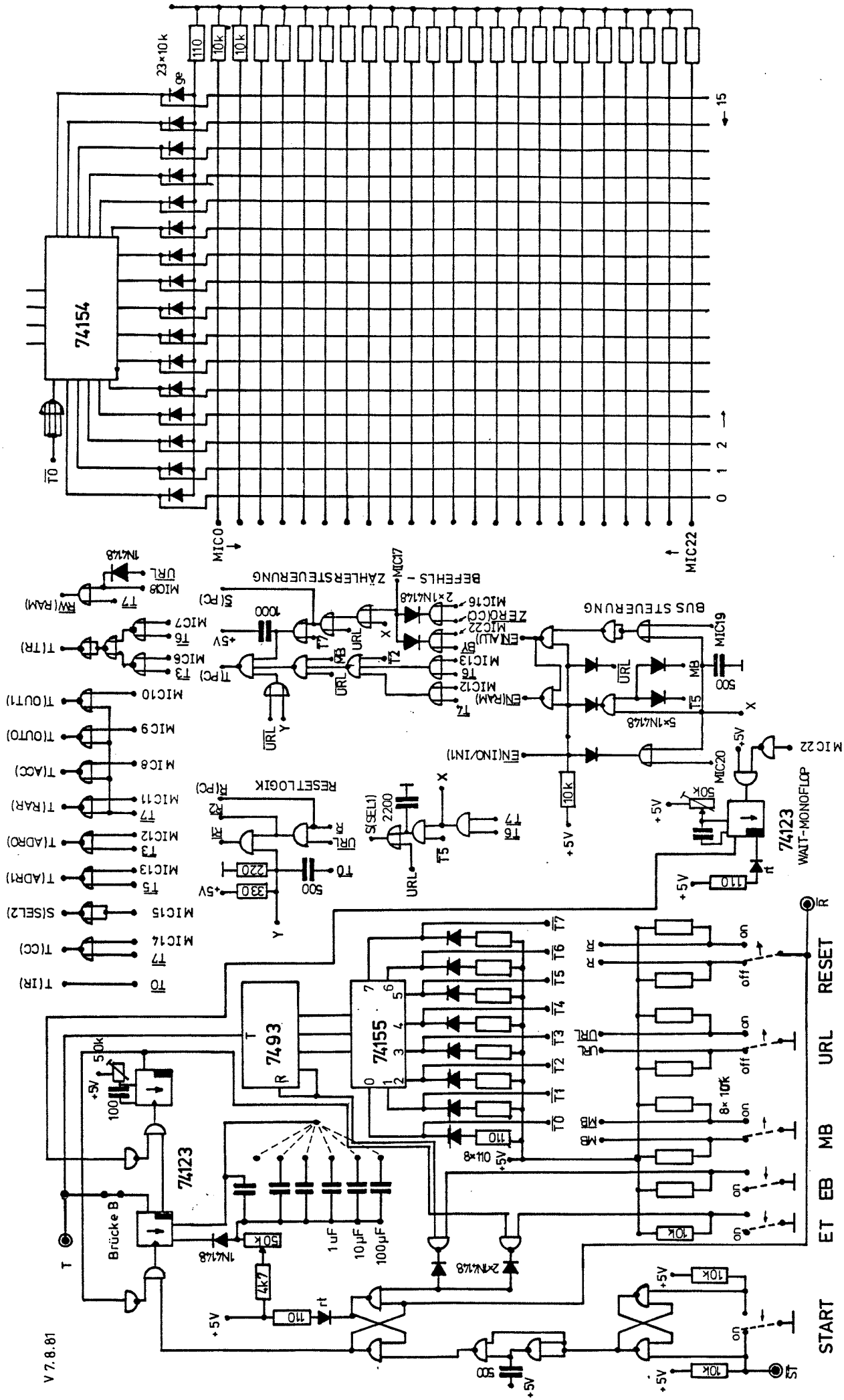
Die Ausgabe eines einzelnen Taktes des Taktgenerators wird analog dadruch bewirkt, daß gleich nach Erzeugung des ersten Taktes durch die Oszillatorstufe über ein anderes NAND-Gatter der Rücksetzeingang des RUN-Flipflops Low-Signal erhält, wenn der Schalter Einzeltakt entsprechend eingestellt ist.

Die Oszillatorstufe kann durch das WAIT-Monoflop, das beim WAIT-Befehl getriggert wird, für eine durch ein Trimmpotentiometer einstellbare Zeit gestoppt werden.

Die Schalter manuelle Befehlsausführung, Urladen und der Taster Reset brauchen nicht entprellt werden, sie erhalten 10 kOhm Pull-up-widerstände, die die jeweils offenen Kontakte mit Sicherheit auf High halten. Durch die Verbindung der Reset-Taste mit dem RUN-Flipflop wird ausgeschlossen, daß die RESET-Taste eine Wirkung zeigt, wenn der Taktgenerator läuft.

V 7.8.81


74125
EN(RAM)
EN(ALU)
74125
7425
RAM 256 x 4
CE
T
CE
W/R (RAM)
74181
S0
S1
S2
S3 (ALU)
Cn
BA
Cn+1
T(CC)
R(CC)
OV (CC)
ZERO(CC)
+5V
110
FE
S(SEL 1)
74157
74157
74157
74157
74175
74175
74175
74175
74175
74175
T(ACC)
R(ACC)
T(TR)
R(TR)
R(BAR)
T(BAR)
74161
74161
T(PC)
S(PC)
R(PC)
S(SEL 2)
7400
7400
R(ADR1)
T(ADR1)
R(ADR0)
T(ADR0)
R(IR)
T(IR)
R(OUT0)
T(OUT0)
R(OUT1)
T(OUT1)
74175
74175
74125
7400
BY
10k
EN (IN0/IN1)
74125


74154
74155
74123
74123
7493
WAIT-MONOFLOP
BUS STEUERUNG
ZÄHLERSTEUERUNG
BEFEHLS -
RESETLOGIK
Brücke B
RESET
URL
MB
EB
ET
START
V 7.8.81

8.2 Mikroprogrammatrix mit Befehlsdekoder

Alle Signale, die die Informationsverarbeitung im Datenverarbeitungsteil des Rechners organisiert ablaufen lassen, wie Takt-, Reset-, Selektions- und Enable-Signale, hängen prinzipiell von zwei verschiedenartigen Größen ab:

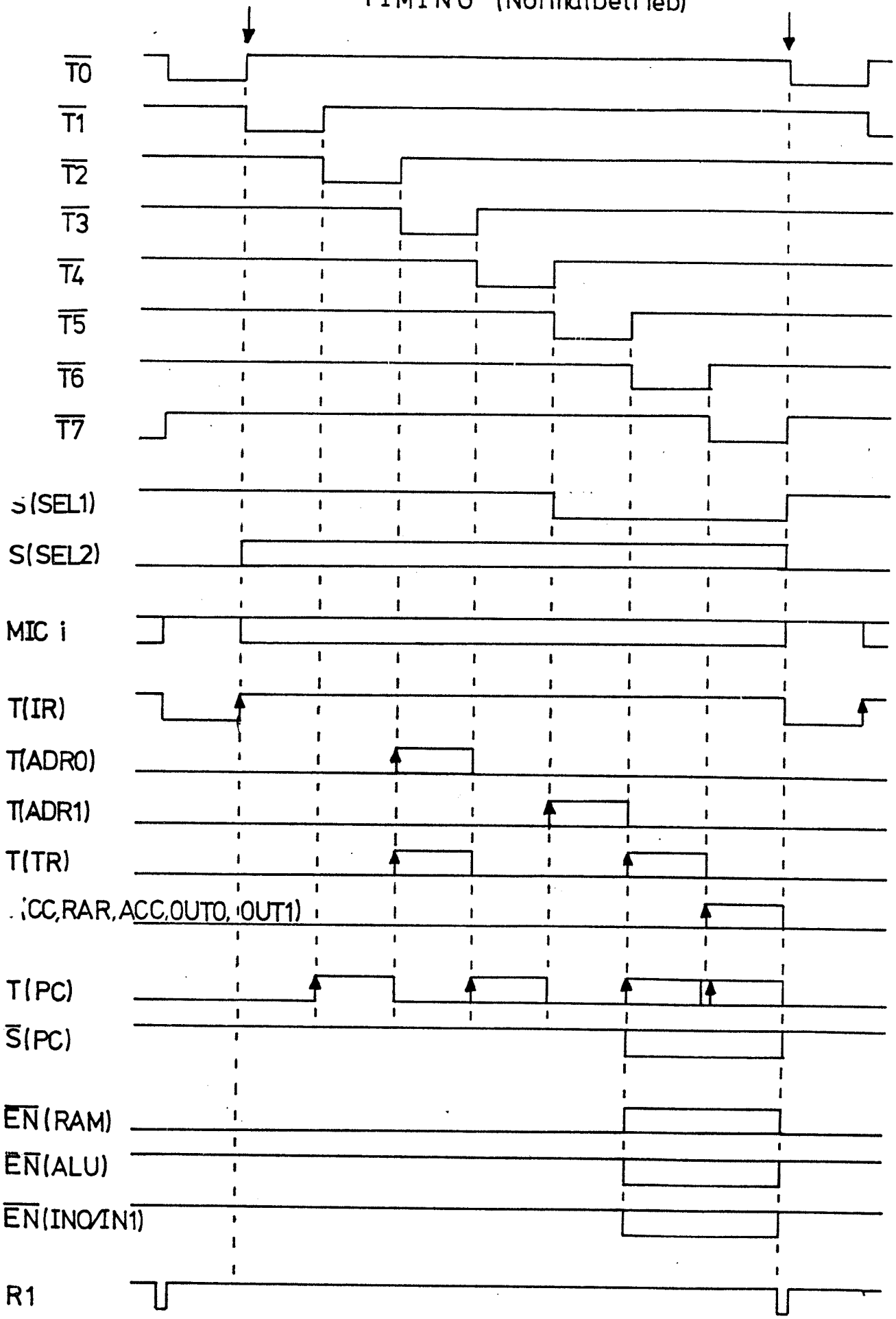
- von der Art des Befehls, der ausgeführt werden soll
- vom Zeitpunkt innerhalb der Ausführungszeit des Befehls

Die erste Größe hängt nur von der Art des Befehls ab und ist innerhalb der Ausführungszeit des Befehls eine statische Größe, d.h. sie verändert sich nicht. Sie hängt deswegen nur vom Befehlscode ab, der im Befehlsregister IR während der Befehlsausführung gespeichert ist.

Die dynamischen Größen, die Zeitpunkte innerhalb der Ausführungszeit definieren, sind die Systemtakte $\overline{T0}$ bis $\overline{T7}$, die vom Taktgenerator erzeugt werden.

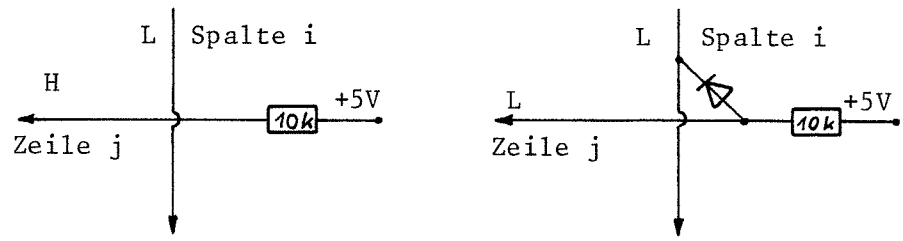
Folgendes Schema soll noch einmal das Zusammenspiel der statischen und dynamischen Größen klar machen, die in der Steuerlogik verknüpft werden und somit befehls- und zeitabhängige Signale liefern:

TIMING (Normalbetrieb)



Im Folgenden schauen wir uns die Erzeugung der statischen Größen, die hier Mikrobefehle genannt werden sollen, näher an.

Die 4 Bit dies Befehlsregisters IR, die den Befehlscode darstellen, werden auf die Eingänge eines Binärdekoder-Demultiplexers gegeben. Je nachdem welche 4 Bit Kombination angelegt ist, wird der dazugehörende Ausgang, genau einer von insgesamt 16 Ausgängen, auf Low geschaltet, während alle anderen 15 Ausgänge High-Pegel besitzen. Diese 16 Ausgänge bilden die Spalten der Mikroprogrammatrix. Genau eine dieser Spalten wird bei der Ausführung des dazugehörenden Befehles also angesteuert (aktiviert), was durch eine Leuchtdiode angezeigt wird. Die Zeilen stellen die Ausgangsgrößen dar und werden Mikrobefehle genannt. An den Kreuzungspunkten von Spalten und Zeilen können Dioden sitzen, je nachdem welchen Pegel die Zeile bei Aktivierung der Spalte haben soll. Schauen wir uns nun einen Kreuzungspunkt näher an.



Ist an dem Kreuzungspunkt keine Diode, so liegt Zeile j immer über den Widerstand auf High-Pegel, wenn die Spalte i aktiviert, d.h. Low ist.

Mit einer in Durchlaßrichtung geschalteten Diode jedoch, liegt Zeile j ebenfalls auf Low, wenn Spalte i Low ist.

Bei Aktivierung einer Spalte der Mikroprogrammatrix durch den entsprechenden Befehl werden nur die Dioden signifikant, die in dieser Spalte sitzen. Alle anderen kann man vergessen, da die Spannungspegel an ihnen so anliegen, daß sie gesperrt sind. Damit erhalten wir für jeden Befehl (Spalte) einen Satz von Mikrobefehlen, der davon abhängt, an welchen Kreuzungspunkten Dioden sitzen oder nicht. Haben wir dort eine Diode, erhält der entsprechende Mikrobefehl (Zeile) einen Low-Pegel, und er wird ausgeführt. Um eine Vorstellung zu bekommen, welche Bedeutung diese Mikrobefehle haben, schauen wir uns einmal folgende Liste an:

8.3 Bedeutung der Mikrobefehle

Mikrobefehle (Zeilennummer)	Bedeutung des Mikrobefehls
MIC 0	BA (ALU)
MIC 1	C_n (ALU)
MIC 2	S_o (ALU)
MIC 3	S_1 (ALU)
MIC 4	S_2 (ALU)
MIC 5	S_3 (ALU)
MIC 6	TR bei $\overline{T2}/\overline{T3}$ takten
MIC 7	TR bei $\overline{T5}/\overline{T6}$ takten
MIC 8	ACC bei $\overline{T6}/\overline{T7}$ takten
MIC 9	OUT0 bei $\overline{T6}/\overline{T7}$ takten
MIC 10	OUT1 bei $\overline{T6}/\overline{T7}$ takten
MIC 11	RAR bei $\overline{T6}/\overline{T7}$ takten
MIC 12	ADR0 bei $\overline{T2}/\overline{T3}$ und PC bei $\overline{T3}/\overline{T4}$ takten
MIC 13	ADR1 bei $\overline{T4}/\overline{T5}$ und PC bei $\overline{T5}/\overline{T6}$ takten
MIC 14	CC bei $\overline{T6}/\overline{T7}$ takten
MIC 15	Adresse vom RAR zum PC durchschalten (SEL2)
MIC 16	PC bei 'Zero-Flipflop gleich Null' takten $\overline{T6}/\overline{T7}$
MIC 17	PC bei $\overline{T6}/\overline{T7}$ takten
MIC 18	RAM bei $\overline{T6}/\overline{T7}$ takten (Schreibvorgang)
MIC 19	ALU bei $\overline{T6}$ und $\overline{T7}$ auf Bus geben
MIC 20	IN0/IN1 bei T6 und T7 auf Bus geben
MIC 21	WAIT-Monoflop bei $\overline{T2}/\overline{T3}$ takten
MIC 22	PC bei 'BY gleich Low' takten $\overline{T6}/\overline{T7}$

8.4 Belegung der Mikroprogrammatrix

Befehl:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Mikro-befehl:																
MIC 0							o	o						o		
MIC 1																
MIC 2						o				o					o	
MIC 3							o							o	o	
MIC 4						o	o			o				o		
MIC 5															o	
MIC 6									o				o	o	o	
MIC 7							o	o								
MIC 8					o		o				o		o	o	o	
MIC 9									o							
MIC 10										o						
MIC 11			o													
MIC 12	o	o	o		o	o	o	o	o			o	o	o	o	
MIC 13	o	o	o		o	o	o	o				o				
MIC 14					o	o	o	o			o		o	o	o	
MIC 15				o												
MIC 16		o														
MIC 17	o		o	o												
MIC 18						o		o								
MIC 19						o	o	o	o	o			o	o	o	
MIC 20											o					
MIC 21																o
MIC 22												o				

o: Mikrobefehl wird ausgeführt. An dieser Stelle sitzt in der Matrix eine Diode.

Jeder dieser Mikrobefehle definiert eine Tätigkeit, die während der Abarbeitung eines Maschinenbefehls ausgeführt werden muß, wenn die entsprechende Stelle mit einer Diode besetzt ist. Jeder Maschinenbefehl setzt sich somit aus einem Satz solcher Mikrobefehle zusammen. Anhand der Belegung der Mikroprogrammatrix kann man sich nun klar machen, wie jeder Befehl zusammengesetzt ist.

Ist solch eine Matrix, die nichts anderes als ein ROM (read only memory) darstellt, in einem Rechner veränderbar, heißt der Rechner mikroprogrammierbar. Unser Modellrechner ist insofern mikroprogrammierbar, als die Matrix durch Herausnehmen bzw. Einsetzen von Dioden verändert werden kann. Der Befehlssatz unseres Rechners ist nun allerdings so ausgesucht, daß nur wenige der vorhandenen Befehle verändert werden können, ohne die Leistungsfähigkeit erheblich zu beeinträchtigen. Der WAIT-Befehl z.B. könnte jedoch ohne weiteres durch einen selbst zusammengestellten Maschinenbefehl ersetzt werden, da auf der Platine an allen Stellen Bohrungen für Dioden vorgesehen sind. Bevor man dies jedoch unternimmt, sollte man den Rechner und den Ablauf eines jeden Befehls genau verstanden haben.

8.5 Der Ein- und Ausgabeteil

Die Ausgabe von Daten geschieht mittels der beiden Register OUT0 und OUT1. Sie bestehen jeweils aus 4 D-Flipflops 74175 und werden über die Befehle OUT0 n und OUT1 angesprochen. Die auszugebende Information wird an der positiven Taktflanke (Übergang Low auf High) übernommen, gespeichert und kann an den Buchsen abgegriffen werden. Um den externen Geräten den Zeitpunkt mitzuteilen, zu dem gültige Information in diesen Registern vorhanden ist, können die beiden Registertakte ebenfalls an zwei Buchsen abgegriffen werden.

Zur Eingabe von Daten stehen zwei Kanäle zur Verfügung. Mit Hilfe des IN0-Kanals kann externe Information, die an den zugehörenden Buchsen anliegt, mittels des INA-Befehls in den Akkumulator gebracht werden. Der zweite Eingabekanal ist das 4 Bit Schalterregister. Der Umschalter S3 legt fest, welcher der beiden Eingabekanäle durch den INA-Befehl angesprochen wird.

Die Steuerlogik liefert folgende Signale:

$T(OUT0) = \overline{\overline{T7} \vee MIC9}$, $\overline{R}(OUT0) = \overline{URL} \wedge \overline{R}$

$T(OUT1) = \overline{\overline{T7} \vee MIC10}$, $\overline{R}(OUT1) = \overline{URL} \wedge \overline{R}$

$\overline{EN}(IN0/IN1)$ = siehe Steuerlogik

Das OUT0 bzw. OUT1 Register muß genau dann getaktet werden, wenn:
- der Befehl OUT0 n bzw. OUT1 vorliegt, d.h. wenn die Mikrobefehle MIC9 bzw. MIC10 ausgeführt werden sollen und
- wenn der Übergang $\overline{T6}/\overline{T7}$ erfolgt.

Zur Beschaltung der Reset-Eingänge schaue man im Teilabschnitt Steuerlogik nach.

Die Information am IN0 und IN1 Eingabekanal wird über die Gatter 74125 mit Tri-state-Ausgang auf den Bus gelegt. Das Enable-Signal, das über den Schalter S3 an IN0 oder IN1 gelegt werden kann und das die Durchschaltung ermöglicht, ist etwas komplizierter zusammengesetzt, da es mit den anderen beiden Enable-Signalen für ALU und RAM verknüpft ist und wird ebenfalls im Teilabschnitt Steuerlogik als Bussteuerung behandelt.

8.6 Der Befehlsteil

Der Befehlsteil des Rechners besteht aus drei 4 Bit Registern 74175 (je vier D-Flipflops), die den Befehlscode (IR) und den aus einem (ADR0) oder zwei 4 Bit Worten (ADR1/ADR0) zusammengesetzten Operanden oder Adreßteil aufnehmen und während der Ausführung des Befehls speichern. Diese Register sind folgendermaßen beschaltet:

$T(IR) = \overline{T0}$, $\overline{R}(IR) = \overline{URL} \wedge \overline{R} \wedge \overline{Y}$

$T(ADR0) = \overline{MIC12 \vee \overline{T3}}$, $\overline{R}(ADR0) = \overline{URL} \wedge \overline{R} \wedge \overline{Y}$

$T(ADR1) = \overline{MIC13 \vee \overline{T5}}$, $\overline{R}(ADR1) = \overline{URL} \wedge \overline{R} \wedge \overline{Y}$

Das erste 4 Bit Wort eines Befehls enthält den Befehlscode und wird beim Übergang $\overline{T0}/\overline{T1}$, d.h. an der positiven Flanke von $\overline{T0}$ in das IR übernommen. Nach Erhöhen des Befehlszählers um eins wird das nächste 4 Bit Wort des Befehls (falls es ein zwei oder drei-Wort-Befehl ist) bei $\overline{T2}/\overline{T3}$ in das ADR0

getaktet. Der Befehlszähler wird wiederum erhöht und das dritte Wort des Befehls (nur bei drei-Wort-Befehlen) bei $\overline{T4}/\overline{T5}$ im ADR1 gespeichert.

8.7 Die arithmetisch-logische Einheit

Die arithmetisch-logische Einheit (ALU) verknüpft die Daten, die im Akkumulator ACC und dem Hilfsregister TR liegen, miteinander und gibt das Ergebnis über die Schaltergatter des Datenschalters S2 bei Bedarf auf den Bus. Außderdem liefert sie die Information, ob das Ergebnis der arithmetischen oder logischen Operation Null ist oder ob ein Überlauf (Übertrag) zu vermerken ist. Die Steuereingänge sind folgendermaßen beschaltet:

T	(ACC)	=	$\overline{MIC8 \vee \overline{T7}}$	, $\overline{R}$ (ACC)	=	$\overline{URL} \wedge \overline{R}$
T	(TR)	=	$\overline{MIC6 \vee \overline{T3}} \vee \overline{MIC7 \vee \overline{T6}}$	, $\overline{R}$ (TR)	=	$\overline{URL} \wedge \overline{R} \wedge \overline{Y}$
BA	(ALU)	=	MIC0			
C_n	(ALU)	=	MIC1			
S_0	(ALU)	=	MIC2			
S_1	(ALU)	=	MIC3			
S_2	(ALU)	=	MIC4			
S_3	(ALU)	=	MIC5			

Sowohl Akkumulator als auch Hilfsregister bestehen aus vier D-Flipflops 74175, die Daten an der positiven Taktflanke übernehmen. Daten müssen dann in den Akkumulator getaktet werden, wenn:

- der Befehl es erfordert, d.h. wenn der Mikrobefehl MIC8 gegeben wird und
- wenn der Übergang $\overline{T6}/\overline{T7}$ erfolgt.

Der Zeitpunkt für die Übernahme von Daten in das Hilfsregister hängt von der Art des Befehls ab. Es müssen zwei Möglichkeiten unterschieden werden:

1. Es liegt ein Befehl vor, der aus zwei Worten besteht, d.h. dessen zweites Wort als Operand betrachtet wird (OUTO, ENTA, INCA, NANDA). Nachdem das erste Wort des Befehls mit dem Befehlscode in des Befehlsregister IR gebracht und der Befehlszähler PC um eins erhöht wurde, steht bei diesen Befehlen mit dem zweiten Befehlswort der Operand auf dem Bus und muß in das Hilfsregister übernommen werden (Übergang $\overline{T2}/\overline{T3}$).

Gleichzeitig wird dieselbe Information in das ADRO übernommen. Dies wäre im Prinzip unnötig, da sie an dieser Stelle nicht weiterverarbeitet werden kann, wurde aber aus didaktischen Gründen beibehalten.

Der Speicher besitzt sog. Tri-state-Ausgänge, d.h. Ausgänge, die bie Low-Signal am OD-Eingang hochohmig geschaltet werden können. Dies gestattet das Parallelschalten von mehreren dieser Bausteine. Diese Möglichkeit wird jedoch nicht ausgenutzt, sodaß der Eingang OD konstant mit High-Pegel, die Chip-Enable-Eingänge $\overline{CE1}$ und CE2 ebenfalls mit festem Low- bzw. High-Pegel beschaltet sind. Die Information, die im Wort mit der angelegten Adresse vorhanden ist, kann jederzeit an den Ausgängen abgegriffen werden. Verändert werden kann sie, wenn:

- der Mikrobefehl MIC18 gegeben wird und wenn
- der Übergang $\overline{T6}/\overline{T7}$ erfolgt.

So wird die Information bei $\overline{R}/\overline{W}$ gleich Low eingespeichert.

Die Daten aus dem RAM werden bei Bedarf über den Datenschalter S1 auf den Bus geschaltet.

8.8 Der Adressenverarbeitungsteil

Der Adressenverarbeitungsteil des Rechners besteht aus mehreren Komponenten:

- dem Adressenselektor SEL1,
- dem Rücksprungadressenregister RAR
- dem Befehlszähler PC und
- dem Adressenselektor SEL2

Diese Komponenten bestimmen nun, welche Adresse zu welchem Zeitpunkt am Arbeitsspeicher RAM anliegt. Prinzipiell können zwei verschiedene Adressen an den Speicher gelegt werden:

- die Befehlszähleradresse - Sie liegt dann am Speicher, wenn der nächste Befehl oder der nächste Teil eines Befehls geholt werden soll.
- die Adresse im Adreßteil eines Befehls (ADR1/ADRO), wenn ein Operand aus dem Speicher geholt bzw. in den Speicher geschrieben werden soll.

Demnach haben die vier Komponenten des Adressenverarbeitungsteils folgende Aufgaben:

- der Adressenselektor SEL1 dient als Schalter, der entweder die Befehlszähleradresse oder die Adresse im Befehlsadreßteil zum Speicher durchschaltet. Für seinen Selektionseingang gilt:

  $S\ (SEL) = URL \vee (\overline{T5} \wedge \overline{T6} \wedge \overline{T7})$

2. Bei drei-Wort-Befehlen, die das Hilfsregister ansprechen (ADDA, DECM), muß das Hilfsregister die Information bekommen, die in der Speicherzelle mit der im Befehl selbst angegebenen Adresse steht. Diese Information kann erst nach Laden des gesamten Befehls vorhanden sein, sodaß das Hilfsregister erst beim Übergang $\overline{T5}/\overline{T6}$ getaktet werden darf.

Auf Null gesetzt wird das Hilfsregister nach Ausführung eines jeden Befehls sowie bei Betätigung der Reset-Taste.

Die Steuersignale, die benötigt werden, um in der ALU 74181 die arithmetischen oder logischen Operationen festzulegen, müssen sich während der Ausführung des Befehls nicht ändern. Sie sind somit statische Größen und können direkt als Mikrobefehle aus der Mikroprogrammatrix betrachtet werden. Für die Belegung dieser Signale schaue man sich die Beschreibung des 74181 an.

Der Condition Code CC besteht aus dem Zero- und Overflow-Flipflop. Liegt auf dem Datenbus eine Null auf allen vier Leitungen, so hat der Ausgang des NOR-Gatters 7425 mit vier Eingängen High-Pegel. Beim Takten des Zero-Flipflops wird dieser Pegel in das Zero-Flipflop übernommen. Über den bedingten Sprungbefehl JNZ wird der Zustand dieses Flipflops als Sprungbedingung genommen.

Das Overflow-Flipflop erhält seinen Zustand direkt aus der ALU. Beide Flipflops werden getaktet, wenn:

- ein arithmetischer Befehl vorliegt, d.h. der Mikrobefehl MIC14 ausgeführt werden soll und
- wenn der Übergang $\overline{T6}/\overline{T7}$ vorliegt.

8.9 Der Arbeitsspeicher

Der Arbeitsspeicher besteht aus einem einzigen integrierten Baustein. Er enthält 1024 Bit, die in 256 Worten zu 4 Bit organisiert sind. Er ist ein statischer Speicher, d.h. seine einzelnen Speicherzellen sind Flipflops, die ihre Information solange behalten, wie die Spannungsversorgung gesichert ist. Um die 256 Worte adressieren zu können, besitzt er acht Adreßeingänge, weiterhin stehen zum Abfragen eines Wortes vier Ausgänge, und zum Verändern des Inhaltes eines Wortes vier Eingänge zur Verfügung. Beschaltet sind seine Steuereingänge folgendermaßen:

$OD\ (RAM) = High,\ \overline{CE1} = Low,\ CE2 = High,\ \overline{R}/\overline{W} = \overline{(MIC18 \wedge URL) \vee \overline{T7}}$

Sowohl in der Betriebsart Urladen als auch während der Systemtaktzeiten $\overline{T0}$ bis $\overline{T4}$ in den anderen Betriebsarten muß immer die Befehlszähleradresse am Speicher liegen, während in den Systemtaktzeiten $\overline{T5}$ bis $\overline{T7}$ bei den Betriebsarten manuelle Befehlsausführung und Normalbetrieb die Adresse im Befehlsadreßteil am Speicher liegen muß.

- Am Ausgang des Befehlszählers PC liegt das Rücksprungadressenregister RAR. Liegt ein CALL-Befehl vor, so wird vor Ausführung des Sprunges die Adresse des dem CALL-Befehl folgenden Befehles im RAR abgespeichert und bis zum nächsten CALL-Befehl aufbewahrt. Über den Befehl RET wird diese Adresse wieder in den Befehlszähler gebracht. Der Befehl steht am Ende eines Unterprogrammes und bewirkt den Rücksprung ins Hauptprogramm auf die im RAR abgespeicherte Adresse.

- Der Adressenselektor SEL2 legt je nach Signal am Selektionseingang entweder die RAR-Adresse (bei RET-Befehl) oder die Adresse im Befehlsadreßteil an den Befehlszähler.

- Der Befehlszähler selbst zeigt immer die Adresse des nächsten Befehls bzw. die des nächsten Teil des Befehls an, wenn der Befehl mehrere Worte lang ist.

Die Beschaltung der Steuereingänge des Befehlszählers ist im Abschnitt Steuerlogik unter 'Befehlszählersteuerung' aufgeführt. Für die anderen Steuersignale gilt:

$$S\ (SEL2) = \overline{MIC15}$$

$$T\ (RAR) = \overline{MIC11 \vee \overline{T7}}\ ,\quad \overline{R}\ (RAR) = \overline{\overline{URL} \wedge \overline{R}}$$

8.10 Die Steuerlogik

Die Steuerlogik verknüpft die statischen Größen eines Befehls, d.h. die Mikrobefehle MIC0 bis MIC22 mit den dynamischen, d.h. den Systemtakten $\overline{T0}$ bis $\overline{T7}$. Sie besteht nur aus Gattern und enthält keine Flipflops. Ihre Aufgabe ist es, alle Signale zu liefern, die notwendig sind, den Datenverarbeitungsteil zu steuern. Die einfacheren Bestandteile der Steuerlogik wurden schon bei der Beschreibung der einzelnen Funktionsgruppen behandelt. Etwas komplizierter aufgebaut sind:

- die Reset-Logik,
- die Bus-Steuerung,
- die Befehlszählersteuerung.

Diese Teile der Steuerlogik werden nun im folgenden behandelt:

8.11 Die Reset-Logik

Diese Logik hat die Aufgabe, folgende Reset-Signale zu liefern:
$\overline{R}$ (TR), $\overline{R}$ (ACC), $\overline{R}$(CC), $\overline{R}$(IR), $\overline{R}$(ADR0), $\overline{R}$(ADR1), $\overline{R}$(PC), $\overline{R}$(RAR), $\overline{R}$(OUT0), $\overline{R}$(OUT1).
Das Zurücksetzen von Befehlszähler, Registern und Flipflops hängt von drei Größen ab: dem Systemtakt $\overline{T0}$, der Stellung des Urlade-Schalters und dem Zustand der Reset-Taste.

Wird die Reset-Taste betätigt, liegt am Eingang des ersten UND-Gatters Low-Signal, das sofort bewirkt, daß der Befehlszähler auf Null gesetzt wird. Der Ausgang dieses Gatters ist dann ebenfalls Low, sodaß der Inhalt von Akkumulator ACC, von OUT0 und OUT1 Register, sowohl Overflow- als auch Zero-Flipflop initialisiert werden. Das Zero-Flipflop wird jedoch gesetzt, da ja alle Register Null sind. Der Ausgang dieses Gatters ist nun Eingang eines weiteren UND-Gatters, dessen Ausgang nun gleichfalls Low-Signal erhält und damit das Reset-Signal für die Register IR, ADR0, ADR1 und TR liefert. Bei Betätigen der Reset-Taste wird somit alles initialisiert.

In der Betriebsart Urladen müssen auch alle Register und Flipflops jedoch mit Ausnahme des Befehlszählers, der ja hochgezählt wird, in einem definierten Ausgangszustand gebracht werden. In diesem Falle ist $\overline{URL}$ Low, sowie die beiden Ausgänge der UND-Gatter.

Nach der Ausführung eines Maschinenbefehls ist es aus didaktischen Gründen wünschenswert, daß der Inhalt all der Register, die nur Daten enthalten, die während der Ausführung dieses Maschinenbefehls von Bedeutung sind, gelöscht wird. Dies gilt natürlich nicht z.B. für den Akkumulator, der seinen Inhalt über mehrere Befehle hinweg behalten muß. Beim Wechsel des Systemtaktes $\overline{T0}$ von High auf Low erhält der Eingang des zweiten UND-Gatters über einen 500 pF-Kondensator einen kurzen Low-Puls, sodaß sein Ausgang, an dem die Reset-Signale für IR, ADR0, ADR1 und TR hängen, kurzzeitig ebenfalls Low ist. Die beiden Widerstände dienen zur Festlegung der Pegel für Low- und High-Signal. Den Kondensator und diese beiden Widerstände könnte man weglassen, wenn man einen Schönheitsfehler bei der Bedienung des Urlade-Schalters in Kauf nähme (siehe Befehlszählersteuerung).

8.12 Die BUS-Steuerung

Die 4-Bit-Daten auf dem zentralen Datenbus des Rechners müssen ihrer Herkunft nach unterschieden werden. Es gibt drei Möglichkeiten:

- Sie werden aus dem Arbeitsspeicher RAM über den Datenschalter S1 auf den Bus geschaltet oder
- sie erreichen den Bus von der arithmetisch-logischen Einheit ALU über den Datenschalter S2 oder
- sie kommen aus einem durch den Schalter S3 ausgewählten Eingabekanal INO oder IN1.

Unmittelbar einzusehen ist, daß niemals Daten aus diesen drei Quellen gleichzeitig auf den Bus gelangen dürfen. Dies darf schon aus technischen Gründen nicht vorkommen. Einzusehen ist ebenfalls, daß die Herkunft der Daten auf dem Bus davon abhängt, in welcher Betriebsart der Rechner läuft, welcher Befehl gerade ausgeführt wird und welcher Systemtakt gerade erreicht ist. Folgende Tabelle gibt an, welche Information auf dem Bus vorhanden sein muß:

$\overline{T0}$	$\overline{T1}$	$\overline{T2}$	$\overline{T3}$	$\overline{T4}$	$\overline{T5}$	$\overline{T6}$	$\overline{T7}$	
RAM	RAM	RAM	RAM	RAM	RAM	X	X	Normalbetrieb
IN	IN	IN	IN	IN	RAM	X	X	man. Befehlsausf.
IN	IN	IN	IN	IN	IN	IN	IN	Urladen

X = RAM oder INO/IN1 oder ALU, je nach Bedeutung des Befehls IN = INO oder IN1.

Die Schaltung selbst muß drei Enable-Signale liefern, die bei High-Pegel die Schaltergatter 74125 im hochohmigen Zustand halten und bei Low durchschalten. Dabei darf zu einer Zeit nur eines dieser Steuersignale Low-Pegel besitzen.

Wird keiner der beiden Mikrobefehle MIC19 oder MIC20 gegeben, so liegt die Information aus dem RAM auf dem Bus, werden (aus Versehen) beide Mikrobefehle gegeben, z.B. wenn ein anderer Befehl in der Mikroprogrammatrix realisiert wird, so ist MIC19 wirkungslos. Die Dioden in der Schaltung wirken wie UND-Gatter, am gemeinsamen Punkt liegt genau dann High-Signal, wenn an allen Dioden ebenfalls High-Signal vorliegt. Die 10 kOhm legen die jeweiligen High-Pegel fest.

In der Betriebsart Urladen muß während eines ganzen Zyklus die Information eines der Eingabekanäle auf dem Bus liegen. Dies erfolgt dadurch, daß $\overline{EN}$ (INO/IN1) über eine Diode durch $\overline{URL}$ Low-Pegel erhält, wenn $\overline{URL}$ Low ist. Die NAND-Gatter, deren Ausgänge die Signale $\overline{EN}$(RAM) und $\overline{EN}$(ALU) liefern, sind dadurch mit Sicherheit gesperrt, d.h. ihre Ausgänge sind High.

In den Betriebsarten Normalbetrieb und Urladen haben die Ausgänge der beiden ODER-Glieder High-Pegel während der ersten 6 Systemtakte (Fetch-Phase), da der gemeinsame Eingang ($X = \overline{T6} \wedge \overline{T7}$) High ist. Dadurch sind $\overline{EN}$(INO/IN1) und $\overline{EN}$(ALU) ebenfalls High und damit $\overline{EN}$(RAM) Low, d.h. der Arbeitsspeicher ist auf den Bus geschaltet. Dies gilt allerdings nur in der Betriebsart Normalbetrieb, da hier MB Low ist. Bei manueller Befehlsausführung ist der Eingang des dahintergeschalteten NAND's Low, wenn $\overline{T5}$ High ist, d.h. INO/IN1 liegt auf dem Bus. Während dem Ablauf der Systemtakte $\overline{T6}$ und $\overline{T7}$ hängen die Ausgänge der beiden ODER-Gatter nur von den Mikrobefehlen MIC19 und MIC20 ab. Sie legen fest, was auf dem Bus liegt. Der 500 pF Kondensator glättet kurze Umschaltpulse, die vom Ausgang des UND-Gatters beim Umschalten von $\overline{T6}$ und $\overline{T7}$ aufgrund von Laufzeitunterschieden auftreten können.

8.13 Die Befehlszählersteuerung

Die Steuerung des Befehlszählers hängt von allen Betriebsarten ab. Er kann als Zähler und Register arbeiten. Die Befehlszählersteuerung liefert zwei Signale T(PC) und $\overline{S}$(PC). Wie der Befehlszähler arbeiten muß, wird aus folgender Tabelle ersichtlich:

$\overline{T1}/\overline{T2}$	$\overline{T3}/\overline{T4}$	$\overline{T5}/\overline{T6}$	$\overline{T6}/\overline{T7}$	$\overline{T7}/\overline{T0}$	
1	MIC12	MIC13	Spr.?	0	Normalbetrieb
0	0	0	Spr.?	0	ma. Befehlsausf.
0	0	0	0	1	Urladen

Im Normalbetrieb muß er in jedem Fall beim Wechsel $\overline{T1}/\overline{T2}$ um eins erhöht werden, da jeder Befehl mindestens ein Wort lang ist. Ob dies auch bei den Übergängen $\overline{T3}/\overline{T4}$ und $\overline{T5}/\overline{T6}$ erfolgt, hängt davon ab, ob der Befehl zwei oder drei Worte lang ist. Falls ein Sprungbefehl vorliegt, hängt es von der Sprungbedingung ab, ob der Befehlszähler den Inhalt des Adreßteils des Befehls erhält oder nicht.

In der Betriebsart manuelle Befehlsausführung darf der Befehlszähler nie erhöht werden, da der Befehl ja nicht aus dem Speicher geholt wird. Ein Sprung jedoch muß ausgeführt werden können.

In der Betriebsart Urladen wird der Befehlszähler nach Abspeichern des zu ladenden Wortes um eins erhöht. Die Logik, die dieses Verhalten realisiert, ist im Gesamtschaltbild eingezeichnet.

Das NAND-Gatter, dessen Ausgang den Takt liefert, hat drei Eingänge:

- der linke Eingang liegt am Ausgang eines ODER-Gatters und ist genau dann Low, wenn die Betriebsart Urladen vorliegt und ein Wechsel von $\overline{T7}$ nach $\overline{T0}$ stattfindet. Er liefert somit den Takt in dieser Betriebsart. Die anderen Eingänge des NAND's sind alle High, da hier die Signale $\overline{URL}$, $\overline{MB}$ und URL eingehen.
- der mittlere Eingang liefert den Takt im Normalbetrieb. Ein NAND mit ebenfalls drei Eingängen, von denen zwei mit $\overline{URL}$ und $\overline{MB}$ belegt sind, sperrt das Signal bei einer anderen Betriebsart. Das Taktsignal selbst wird über ein weiteres Dreifach-NAND aus den drei Beiträgen der Systemtaktwechsel in Verbindung mit den Mikrobefehlen MIC12 und MIC13 erzeugt.
- der rechte Eingang liefert den Takt bei Vorliegen einer erfüllten Sprungbedingung. Der Systemtakt $\overline{T7}$ wird genau dann durch das erste ODER-Gatter durchgelassen, wenn die Sprungbedingung erfüllt ist und damit $\overline{S}$(PC) Low ist. Über zwei hintereinandergeschaltete ODER-Gatter ist dies dann der Fall, wenn die Betriebsart Urladen nicht vorliegt, wenn die Execute-Phase erreicht ist und die jeweiligen, durch die Mikrobefehle MIC16, MIC17 und MIC22 angewählten Sprungbedingungen erfüllt sind.

Der 1 nF Kondensator hat eine sehr wichtige Funktion: da Sprungbefehle alle eine Länge von drei Worten haben, wird der Befehlszähler beim Übergang $\overline{T5}/\overline{T6}$ getaktet, d. h. der Taktimpuls hat dort seine positive Flanke. Die negative Flanke liegt beim Wechsel $\overline{T6}/\overline{T7}$. Zu diesem Zeitpunkt soll der PC jedoch bei erfüllter Sprungbedingung gesetzt und damit getaktet werden. Der rechte Eingang des NAND's (Ausgang T(PC)) wird deshalb so verzögert, daß T(PC) sehr kurz Low wird.

8.14 Liste der Steuersignale

Taktsignale:	T(OUT0)	T(OUT1)	T(CC)	T(ACC)
	W/R(RAM)	T(TR)	T(IR)	T(ADR0)
	T(ADR1)	T(PC)	T(RAR)	
	Systemtakte $\overline{T0}$, $\overline{T1}$, $\overline{T2}$, $\overline{T3}$, $\overline{T4}$, $\overline{T5}$, $\overline{T6}$, und $\overline{T7}$			
	Zentraltakt T			
	Hilfstakte X und Y (siehe Control-Logic)			
Bus-Steuersignale:	$\overline{EN}$(RAM)	$\overline{EN}$(ALU)	$\overline{EN}$(IN0/IN1)	
Reset-Signale:	$\overline{R}$(OUT0)	$\overline{R}$(OUT1)	$\overline{R}$(CC)	$\overline{R}$(ACC)
	$\overline{R}$(TR)	$\overline{R}$(IR)	$\overline{R}$(ADR0)	$\overline{R}$(ADR1)
	$\overline{R}$(PC)	$\overline{R}$(RAR)		
	zentrales Reset-Signal R und $\overline{R}$			
	Hilfsresetsignal $\overline{RES}$			
Mikrobefehle:	MIC0, MIC1, MIC2, ..., MIC21, MIC22			
	(siehe Mikrobefehlsliste)			
Signale zur Adressenselektion:	S(SEL1)	S(SEL2)		
ALU-Signale:	S0(ALU)	S1(ALU)	S2(ALU)	S3(ALU)
	(Funktionsauswahlsignale), Betriebssignal BA(ALU),			
	Übertragseingang C_n(ALU), Übertragsausgang C_{n+1}(ALU)			
CC-Signale:	$\overline{OV}$(CC)	$\overline{ZERO}$(CC)	externe Sprungbedingung $\overline{BY}$	
Schaltersignale:	Starttaste $\overline{ST}$ (externer Start)			
	manuelle Betriebsausführung MB und $\overline{MB}$,			
	Betriebsart Urladen URL und $\overline{URL}$			

9. Anschluß externer Geräte

Wie in der Einleitung schon angedeutet und wie aus der Beschreibung des Modellrechners hervorgeht, ist der Rechner weniger dafür geeignet, arithmetische Operationen mit vielstelligen Gleitkommazahlen durchzuführen, sondern er ist mehr als Modell eines Prozeßrechners aufzufassen. Seine Struktur ist daraufhin ausgebildet, daß er Daten besonders einfach von externen Geräten aufnehmen oder an sie abgeben kann.

Für die Ausgabe stehen zwei 4 Bit Register zur Verfügung, die über die Maschinenbefehle OUT0 und OUT1 vom Programm her angesprochen werden können. Die Daten werden in ihnen gespeichert und können über Buchsen abgegriffen werden. Man kann nun direkt diese insgesamt acht unabhängigen Bits dazu benutzen, irgendetwas zu steuern. Beispielweise ließen sich unmittelbar Leuchtdioden (z.B. Siebensegmentanzeigen) und über einen einfachen dahintergeschalteten Verstärkertransistor kleine Elektromagnete, Relais, Gleichstrommotoren, Lautsprecher oder Glühbirnen an- und ausschalten. Am Ende des Abschnittes sind Beispiele angegeben. Schaltet man nun mehrere zu steuernde Geräte an die beiden Register OUT0 und OUT1, muß man sich Gedanken um eine vernünftige Organisation der Geräte machen. Die gleichen Überlegungen sind notwendig, wenn man über den IN0 Eingabekanal Daten zum Rechner leiten will.

Im Folgenden wird ein Vorschlag gemacht. Der eine bezieht sich auf eine Ein-Ausgabeorganisation, die drei Geräte verwaltet, wobei jedes Gerät bis zu acht Bit vom Rechner aufnehmen und abgeben kann. Der zweite Vorschlag erweitert diese Organisation auf bis zu acht Geräte.

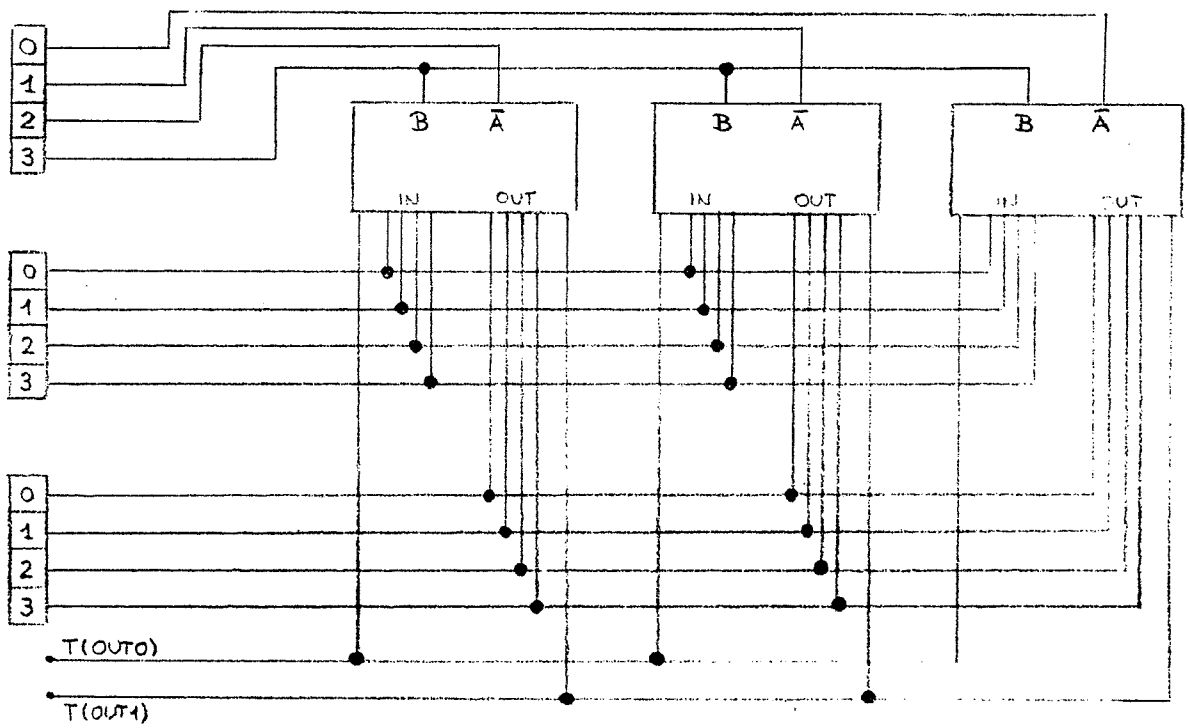
Grundidee dieser Ein-Ausgabeorganisation ist folgende:
Jedes Gerät erhält eine Adresse, d.h. eine Nummer, unter der es angesprochen und aktiviert wird. Diese Adresse wird in das OUT0-Register geschrieben, während die Daten, die ausgegeben werden sollen, ins OUT1-Register kommen. Im Programm sieht also die Ein- und Ausgabe von Daten wie folgt aus:

```
:
OUT0  n      Ausgabe der Geräteadresse n
OUT1         Ausgabe von Information an das Gerät Nr. n
INA          Eingabe von Information vom Gerät Nr. n
:
:
```

Da wir vier Bit im OUT0-Register haben, könnten wir vier Geräte ansprechen, wobei wir sogar die Möglichkeit haben, mehrere gleichzeitig zu aktivieren und Daten ihnen abzugeben. Wir wollen aber nun nur drei dieser Bits dazu verwenden, ein Gerät zu adressieren. Das vierte Bit heben wir uns auf, dem Gerät sagen zu können, was es mit der Information im OUT1-Register tun soll. Dieses Bit soll Befehlsbit (command bit) heißen.

Weiterhin wollen wir davon ausgehen, daß jedes Gerät fähig sein soll, die zu ihm transportierten Daten in einem eigenen Register zu speichern. Außerdem soll jedes Gerät auch die Möglichkeit haben, Daten abzugeben.

Somit sieht unsere Ein- und Ausgabeorganisation folgendermaßen aus:



Wie eingezeichnet, müssen auch die beiden Registertakte T(OUT0) und T(OUT1) herausgeführt werden, um dem Gerät mitzuteilen, wann es Daten aufnehmen bzw. abgeben soll.

9.1 Anzeigeeinheit CL 8145/2

Mit dieser zweifach 4-Bit Anzeigeeinheit können zwei 4-Bit Binärzahlen als 7-Segmentziffern dargestellt werden. Optisch entspricht die Anzeige dem hexadezimalen Wert der Binärziffer.

Die Anzeigeeinheit hat zwei eigene 4-Bit Register, die den anzuzeigenden Wert vom Rechner übernehmen, speichern, über Leuchtdioden binär anzeigen. Ein 7-Segmentdekoder übernimmt dann die Ansteuerung der einzelnen Segmente.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Das Gerät kann z.B. dazu dienen, 4-Bit Daten als Ergebnis einer Rechnung sowohl in binärer Form als auch als Hexadezimalziffer anzuzeigen.

Stromversorgung:	5 V $\pm$ 0.25 V, max. 400 mA
Aufbau, Maße:	durchkontaktierte Epoxyd-Europaplatine, 160 x 100 mm, Plexiglasgeschützt, 5 IC's, 8 LED's, 2 7-Segmentziffern mit 8 mm Ziffernhöhe, 10 2 mm Buchsen

9.2 Digital-Analogwandler CL 8145/3

Mit diesem zweifach 4-Bit Digital-Analogwandler kann man aus zwei 4-Bit Binärzahlen zwei Spannungen erzeugen, die proportional zum binären Wert dieser Zahlen sind. Der maximale Spannungsbereich kann über Potentiometer eingestellt werden und liegt zwischen 0 Volt und ca. 3.5 Volt. Durch die 4-Bit Auflösung ist dieser Spannungsbereich somit in 16 Stufen eingestellt. Es gilt:

$$U = n/16 \cdot U_{max}$$

wobei U_{max} durch die Potentiometerstellung und die Strombelastung des Analogausganges gegeben ist.

Die Digital-Analogwandlung erfolgt mittels eines Widerstandnetzwerkes aus vier Widerständen, die im Verhältnis 1:2:4:8 zueinander dimensioniert sind. Diese Widerstände sind Trimmpotentiometer, die bei Bedarf nachjustiert werden können bzw. verstellt werden können, um das Prinzip der Digital-Analogwandlung zu verstehen.

Der zweifach 4-Bit Digital-Analogwandler hat zwei eigene 4-Bit Register, die den jeweiligen zu konvertierenden binären Wert vom Rechner übernehmen und speichern.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Die beiden Analogausgänge können z.B. zur Ansteuerung der x- und y-Ablenkung eines Oszillographen oder eines XY-Schreibers dienen. Da beide Ausgänge hochohmig sind, können niederohmige Verbraucher nur über einen Verstärker angeschlossen werden. Damit können dann Motoren, Lampen, Spulen und andere Stromverbraucher mit dem vom Rechner bestimmten Analogsignal gesteuert werden.

Stromversorgung: 5 V $\pm$ 0.25 V, max. 150 mA

Aufbau, Maße: durchkontaktierte Epoxyd-Europaplatine, 160 x 100 mm, Plexiglasgeschützt, 3 IC's, 8 LED's, 12 2 mm Buchsen

9.3 Eingabetastatur CL 8145/5

Mit diesem Gerät kann ein 4-Bit Wort über Tasten dem Rechner übertragen werden. Die Eingabetastatur enthält einen eigenen 4-Bit Speicher, der die Tastenkombination speichert und dessen Inhalt bei Anforderung des Rechners abgerufen werden kann. Bei Betätigung einer Eingabetaste wird das zugehörige Bit invertiert. Diese Methode erlaubt eine schnelle Eingabe der 4-Bit Information.

Die Dateneingabe kann unter der Kontrolle des Rechners ablaufen. Dies geschieht folgendermaßen:

Die Tastatur kann über ein Flipflop gesperrt werden, d.h. ein Tastendruck bewirkt keine Änderung der gespeicherten Information. Vom Rechner her muß zunächst dieses Flipflop zurückgesetzt werden. Danach kann die Tastatur neue Information im Speicher unterbringen. Hat man das gewünschte 4-Bit Wort eingegeben und wird es über die Leuchtdioden angezeigt, wird bei Betätigung einer speziellen Übergabetaste dem Rechner über ein BY (Busy)-Signal mitgeteilt, daß nun gültige Information für ihn bereitliegt. Gleichzeitig wird mit diesem Tastendruck die Tastatur über das Flipflop gesperrt. Nachdem der Rechner das 4-Bit Wort abgeholt und verarbeitet hat, kann er die Tastatur wieder freigeben.

Durch diese unter Kontrolle des Rechners erfolgende Dateneingabe kann es nie vorkommen, daß Daten zu langsam, zu schnell eingegeben oder gar "verloren" gehen können.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Stromversorgung: 5 V $\pm$ 0.25 V, max. 150 mA

Aufbau, Maße: durchkontaktierte Epoxyd-Europaplatine, 160 x 100 mm, Plexiglasgeschützt, 5 IC's, 5 LED's, 11 2 mm Buchsen, 5 Tasten (entprellt)

9.4 Analog-Digitalwandler CL 8145/5

Dieser Analog-Digitalwandler erlaubt die Umwandlung einer Spannung zwischen 0 Volt und ca. 4 Volt in eine 4-Bit Binärzahl, die durch vier Leuchtdioden angezeigt wird. Der Spannungsbereich kann somit in 16 Stufen eingeteilt werden. Durch ein Potentiometer kann der Eingangsspannungsbereich auch weiter eingeschränkt werden, sodaß alle 16 Stufen für diesen eingeschränkten Spannungsbereich zur Verfügung stehen. Erweitert werden kann der Eingangsspannungsbereich über einen extern anzuschließenden Spannungsteiler. Der Eingang des Analog-Digitalwandlers ist überspannungsgeschützt, sodaß Eingangsspannungen von ca. 30 Volt keinen Schaden auf der Platine anrichten können.

Der Analog-Digitalwandler besteht aus einem Zähler, der mit einer Takt-

frequenz von ca. 16 kHz versorgt wird. An diesen 4-Bit Zähler ist ein Widerstandnetzwerk von im Verhältnis von 1:2:4:8 dimensionierten Widerständen angeschlossen, an dessen Ausgang eine 16-stufige Treppenspannung erzeugt wird. Die Widerstände sind Trimmpotentiometer, die bei Bedarf nachjustiert werden können, bzw. verstellt werden können, um das Funktionsprinzip zu verstehen. Diese Treppenspannung wird über einen Operationsverstärker mit der zu digitalisierenden Spannung verglichen, wobei bei Übereinstimmung der richtige Zählerstand in ein 4-Bit Register übernommen wird.

Die Schnittstelle zum Rechner (Daten und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Der Eingangswiderstand des Wandlers liegt bei ca. 100 kOhm.

Die benötigte Zeit zur Umwandlung einer Eingangsspannung liegt bei ca. 1 ms, d.h. die Spannung wird mit einer Frequenz von ca. 1 kHz abgefragt und digitalisiert.

Stromversorgung: 5 V $\pm$ 0.25 V

Aufbau, Maße: durchkontaktierte, glanzverzinnte Epoxyd-Europaplatine 160 x 100 mm, Unterseite Plexiglasgeschützt, 5 IC's, 4 LED's, 10 2 mm Buchsen

10. Schlußwort

Wenn der Leser nun an diese Stelle gekommen ist, wird er bemerkt haben, wie einfach im Grunde der Aufbau und die Funktionsweise eines Mikrocomputers ist, wenn man seine einzelnen Komponenten verstanden hat und gesehen hat, wie sie durch logische Gatter und Flipflops verwirklicht werden können. Er hat erkannt, daß die Flexibilität eines Rechners nur dadurch zustande kommt, daß sein Verhalten durch ein Programm, das in einem Speicher liegt und verändert werden kann, bestimmt wird. Wenn der Leser nun in die Situation kommt, daß er sich in einen anderen Mikrocomputer einzuarbeiten hat, wird es meist nicht mehr notwendig sein, sich in dieser Ausführlichkeit mit dem Ablauf einzelner Maschinenbefehle zu befassen. Wohl muß er sich die Gesamtheit der Maschinenbefehle genau ansehen, um Programme in Maschinensprache schreiben zu können. Wie mühsam dies sein kann, wenn man

Nebenbedingungen wie Rechenzeit, Speicherbedarf usw. beachten muß, hat er bei der Programmierung dieses Modellrechners schon mitbekommen. Oft den größeren Teil der Arbeit wird jedoch die Suche nach einem geeigneten Algorithmus, d.h. einem Verfahren beanspruchen, sein Problem zu lösen.

Zuletzt noch ein tröstendes Wort für den Leser, der verzweifelt vor dem Rechner sitzt und ihn noch nicht recht verstanden hat, oder für denjenigen, der ein Programm geschrieben hat, das nicht nach seinem Willen arbeitet. Beide mögen sich damit trösten, daß viele Tausend Leute tagtäglich sich mit dem Entwurf und der Programmierung von solchen Rechnern herumschlagen und laufend über Probleme und Problemchen stolpern, die diese Nullen und Einsen mit sich bringen. Dem Entwickler dieses Modellrechners ist es auch nicht anders ergangen, beliebig viel Zeit mußte aufgewandt werden, Detailprobleme zu lösen und Fehlerchen zu finden, um den Rechner in dieser abgerundeten Form präsentieren zu können. Immer jedoch wird er Kritik und Vorschlägen aus dem Kreis derjenigen, die sich mit diesem Modellrechner beschäftigen, aufgeschlossen sein.

11. Programmbeispiele

Testprogramm für die Zweifach-4-Bit Anzeigeeinheit CL 8145/2

Das Programm zählt synchron in beiden Hexadezimalstellen hoch. Anschluß der Anzeigeeinheit:

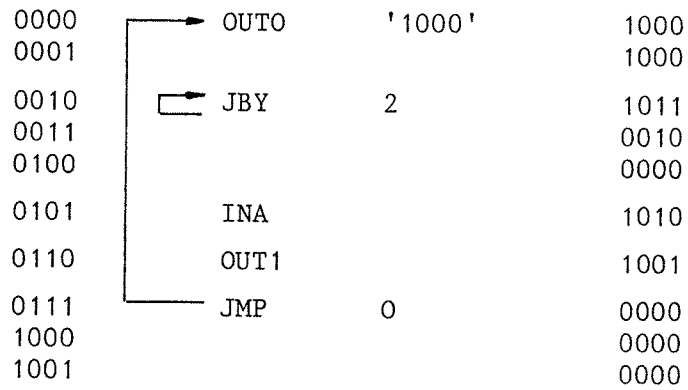
- Eingang B an Bit 3 des OUTO,
- Eingang $\overline{A}$ an Bit 0,1 oder 2 des OUTO,
- Dateneingänge an Datenausgänge des OUT1,
- Takteingang $\overline{T}$ an Taktausgang T des OUT1.

```
0000 ┌─► OUTO      '0000'     1000
0001 │                        0000
0010 │   OUT1                 1001
0011 │   OUTO      '1000'     1000
0100 │                        1000
0101 │   OUT1                 1001
0110 │   INCA   1             1101
0111 │   INCA                 0001
1000 │   WAIT                 1111
1001 └── JMP    0              0000
1010                           0000
1011                           0000
```

Testprogramm für die 4-Bit Eingabetastatur CL 8145/5

Das Programm zeigt die eingegebene Bitkombination im OUT1 an. Anschluß der Eingabetastatur:

- Eingang B an Bit 3 des OUTO,
- Eingang $\overline{A}$ an Bit 0,1 oder 2 des OUTO,
- Datenausgänge an Dateneingänge des INO
- Takteingang $\overline{T}$ an Taktausgang T des OUTO (!!!)
- Ausgang BY an Eingang BY des Rechners

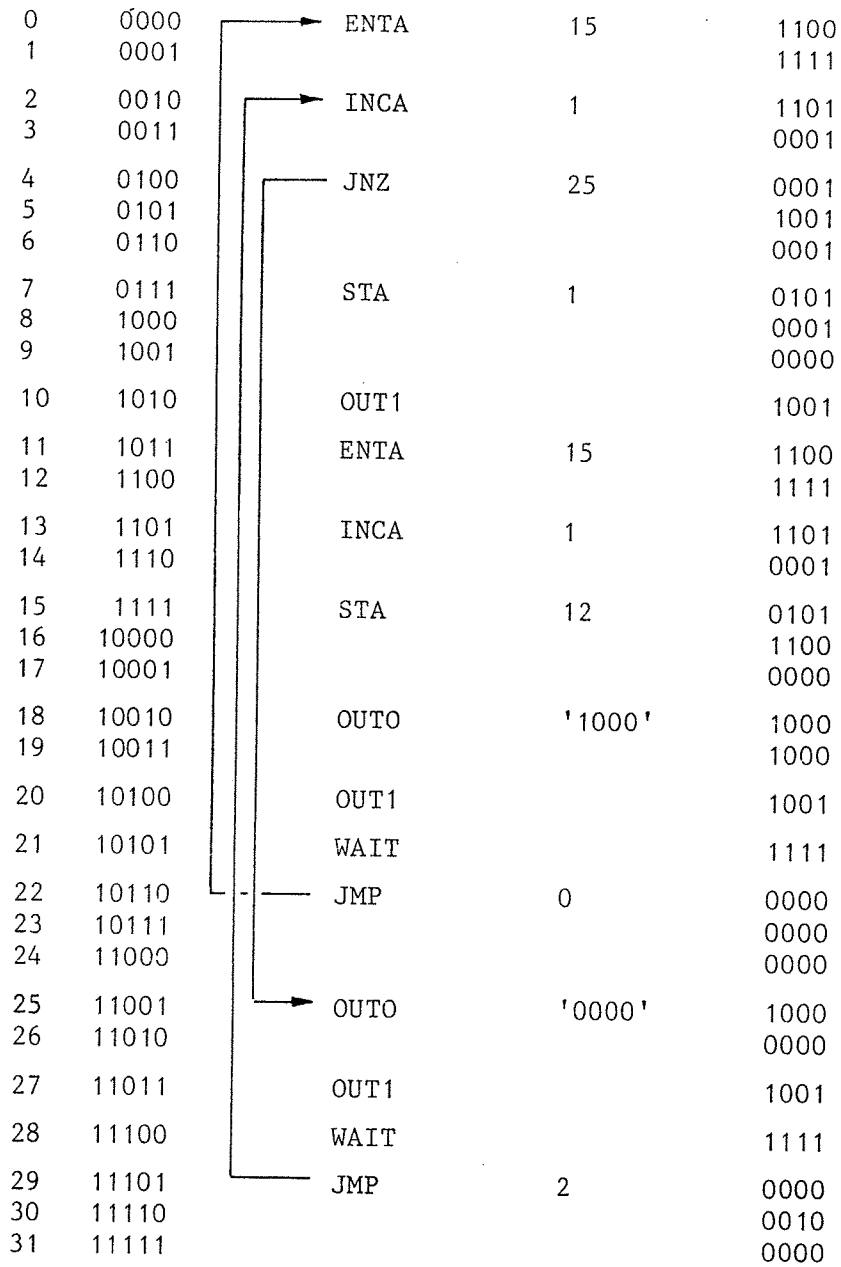


Hexadezimalzähler für 7-Segmentanzeige CL 8145/2 oder Digitalanalogwandler CL 8145/3

Anschluß der Platinen an den Rechner:

- Eingang $\overline{A}$ an Bit 0,1 oder 2 des OUTO,
- Eingang B an Bit 3 des OUTO,
- Eingang $\overline{T}$ an T(OUT1) und
- Dateneingänge an Ausgänge von OUT 1.

Das Programm erzeugt bei Anschluß der 7-Segmentanzeige eine hochzählende zweistellige Hexadezimalzahl, bei Anschluß des Digitalanalogwandlers ein Punktmuster von 16x16 Punkten (VO an x-Abl., V1 an y-Abl. eines Oszillographen). Durch das WAIT-Monoflop kann die Zählgeschwindigkeit, bzw. die Bildwiederholfrequenz eingestellt werden.



0	0000	ENTA	15	1100
1	0001			1111
2	0010	INCA	1	1101
3	0011			0001
4	0100	JNZ	25	0001
5	0101			1001
6	0110			0001
7	0111	STA	1	0101
8	1000			0001
9	1001			0000
10	1010	OUT1		1001
11	1011	ENTA	15	1100
12	1100			1111
13	1101	INCA	1	1101
14	1110			0001
15	1111	STA	12	0101
16	10000			1100
17	10001			0000
18	10010	OUTO	'1000'	1000
19	10011			1000
20	10100	OUT1		1001
21	10101	WAIT		1111
22	10110	JMP	0	0000
23	10111			0000
24	11000			0000
25	11001	OUTO	'0000'	1000
26	11010			0000
27	11011	OUT1		1001
28	11100	WAIT		1111
29	11101	JMP	2	0000
30	11110			0010
31	11111			0000

Speicherbedarf: 32 Worte, Startadresse: 0

Programm für doppeltlange Addition

Von der Hardware her lassen sich mit dem Mikrocomputer nur Daten einer Länge von 4 Bit verarbeiten. Da man für manche Anwendungen den Zahlenbereich für arithmetische Operationen erweitern will, kann man sich Unterprogramme zur Verarbeitung doppeltlanger Daten (8-Bit-Daten) zusammenstellen. Unterstützt wird dies durch die Möglichkeit, durch Verbinden der $\overline{OV}$-Buchse und der $\overline{BY}$-Buchse mit Hilfe des JBY-Befehles das Overflow-Flipflop abzufragen. Ein Unterprogramm, das es gestattet, zwei 8-Bit-Zahlen A und B zu addieren und das Ergebnis in RES abspeichert, ist im Folgenden angegeben. Die Zahl A belegt die Speicherplätze 160 (untere 4 Bit) und 161 (obere 4 Bit), die

Zahl B die Adressen 162 (untere 4 Bit) und 163 (obere 4 Bit) und das Ergebnis RES ist unter der Adresse 164 (untere 4 Bit) und 165 (obere 4 Bit) zu finden.

Das Hauptprogramm, das dieses Unterprogramm aufruft, simuliert einen 8-Bit-Zähler, wobei sich die unteren 4 Bit in OUT0, die oberen 4 Bit in OUT1 befinden. Die Zählgeschwindigkeit kann durch das WAIT-Monoflop variiert werden.

Hauptprogramm:

0	0000	CALL	128	0010
1	0001			0000
2	0010			1000
3	0011	LDA	164	0100
4	0100			0100
5	0101			1010
6	0110	STA	160	0101
7	0111			0000
8	1000			1010
9	1001	STA	13	0101
10	1010			1101
11	1011			0000
12	1100	OUT0	0	1000
13	1101			0000
14	1110	LDA	165	0100
15	1111			0101
16	10000			1010
17	10001	STA	161	0101
18	10010			0001
19	10011			1010
20	10100	OUT1		1001
21	10101	WAIT		1111
22	10110	JMP	0	0000
23	10111			0000
24	11000			0000

Unterprogramm:

128	1000 0000	LDA	160	0100
129	1000 0001			0000
130	1000 0010			1010
131	1000 0011	ADDA	162	0110
132	1000 0100			0010
133	1000 0101			1010
134	1000 0110	JBY	145	1011
135	1000 0111			0001
136	1000 1000			1001
137	1000 1001	STA	164	0101
138	1000 1010			0100
139	1000 1011			1010
140	1000 1100	ENTA	0	1100
141	1000 1101			0000
142	1000 1110	JMP	150	0000
143	1000 1111			0110
144	1001 0000			1001
145	1001 0001	STA	164	0101
146	1001 0010			0100
147	1001 0011			1010
148	1001 0100	ENTA	1	1100
149	1001 0101			0001
150	1001 0110	ADDA	161	0110
151	1001 0111			0001
152	1001 1000			1010
153	1001 1001	ADDA	163	0110
154	1001 1010			0011
155	1001 1011			1010
156	1001 1100	STA	165	0101
157	1001 1101			0101
158	1001 1110			1010
159	1001 1111	RET		0011
160	1010 0000	Low A		0000
161	1010 0001	High A		0000
162	1010 0010	Low B		0001
163	1010 0011	High B		0000
164	1010 0100	Low RES		0000
165	1010 0101	High RES		0000

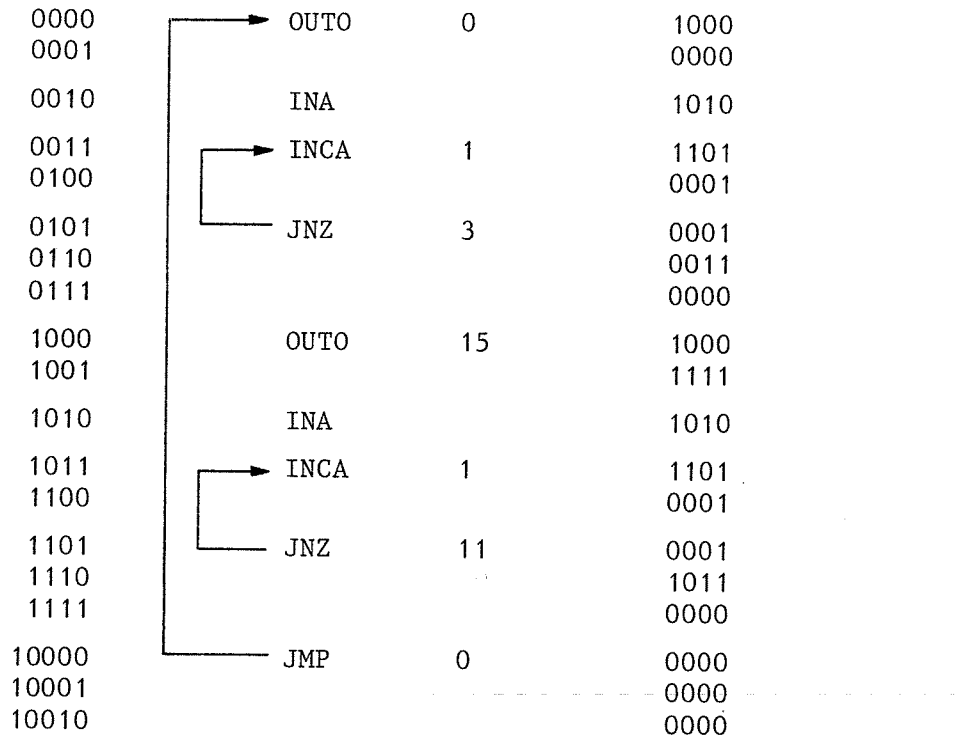
Startadresse Hauptprogramm: 0 Startadresse Unterprogramm: 128

Speicherbedarf Hauptprogramm: 25 Worte
Speicherbedarf Unterprogramm: 38 Worte

Anzahl Befehle Hauptprogramm: 10
Anzahl Befehle Unterprogramm: 12

Digitalprogrammierbarer Frequenzgenerator

Das Programm erzeugt einen Takt (Puls/Pausenverhältnis etwa 1:1), der am Ausgang des OUTO-Registers abgegriffen werden kann. Die Taktfrequenz wird digital an einen der beiden Eingabekanäle angelegt und kann mit diesem 4-Bit-Muster um einen Faktor acht variiert werden.



Bei Stellung des Eingabekanalwählschalters S3 auf IN1 kann an OUTO der Takt als Funktion des an IN1 angelegten vier-Bit-Musters abgegriffen werden, wobei sich die Taktfrequenz je nach Arbeitsgeschwindigkeit des Rechners bis in den Bereich von ca. 20 kHz erstreckt. Der Takt kann z.B. auf dem Oszillographen sichtbar gemacht werden. Schließt man einen nicht zu niederohmigen (> 300 Ohm) Ohrhörer an eines der Bit des OUTO-Registers, kann der Takt als Tonfrequenz hörbar gemacht werden. Der Ohrhörer wird zur Erzielung höchstmöglicher Lautstärke zwischen einem der Bits und der Versorgungsspannung als Last angeschlossen.

Startadresse: 0
Speicherbedarf: 19 Worte
Anzahl Befehle: 9

Testprogramm für den Analog-Digital-Wandler CL 8145/6

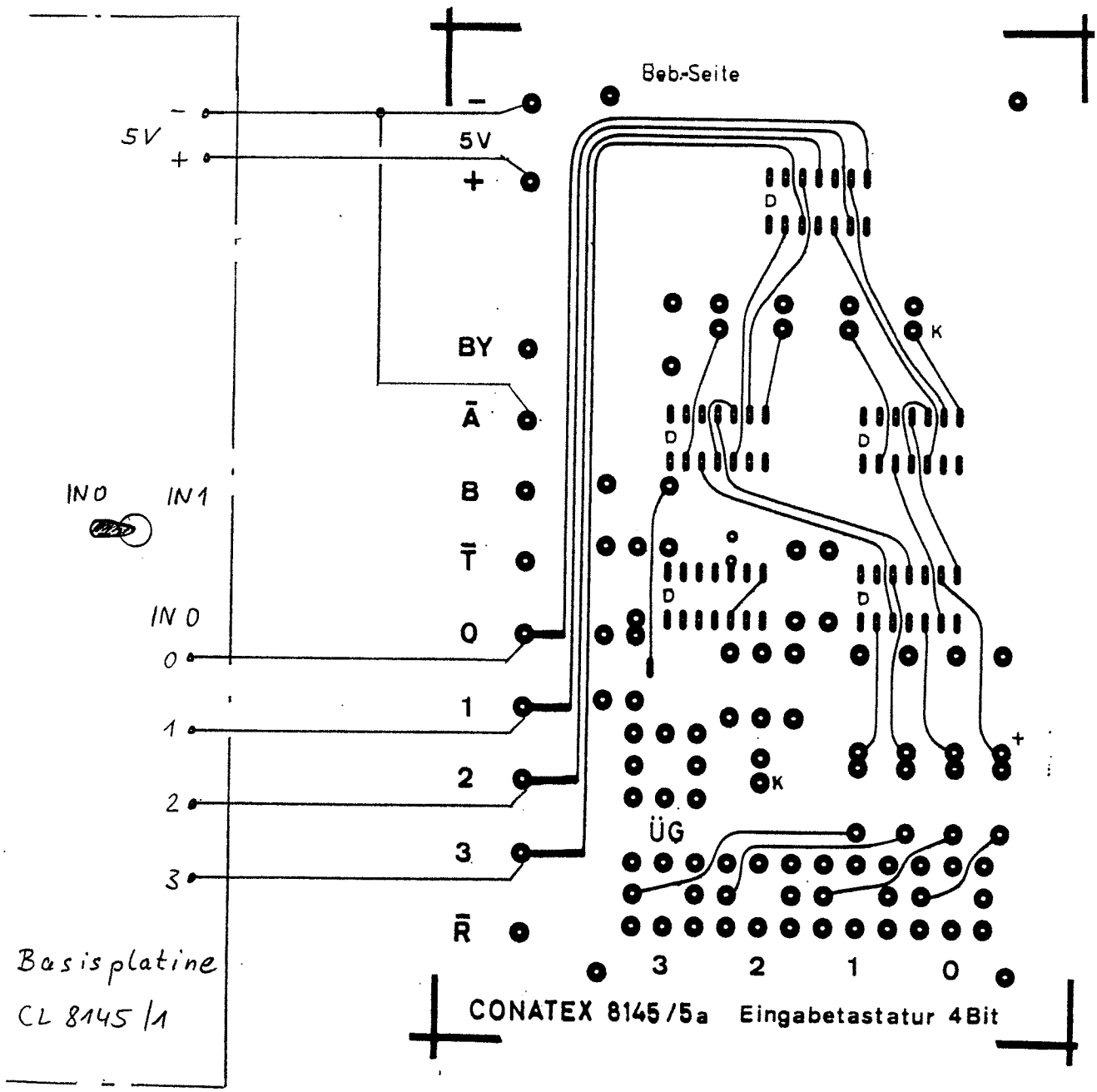
Das Programm zeigt die dem Analogwert entsprechende Bitkombination im OUT 1 an. Anschluß des A/D-Wandlers:

- Eingang $\overline{A}$ an ein Bit des OUT 0
- Datenausgänge an Dateneingänge des IN 0
- Schalter S 3 auf IN 0
- zu digitalisierende Spannung an V und -

0000	OUT 0	0	1000
0001			0000
0010	INA		1010
0011	OUT 1		1001
0100	JMP	0	0000
0101			0000
0110			0000

Beim Anschluß eines Oszillographen ist an Buchse M 1 die zu digitalisierende Spannung zu sehen, die auf V gegeben wird und an Buchse M 2 die Treppenspannung, die durch die drei Potentiometer eingestellt wird.

Verdrahtung zum "Urladen" eines Programmes über die Eingabetastatur



CO 8145 Mikrocomputer Lehrgerät

Kurzbeschreibung

1. Allgemein
2. Blockschaltbild 1 - 3
3. Technische Daten
4. Anzeigeeinheit
5. Digital-Analogwandler
6. Eingabetastatur

1. Allgemein

Es handelt sich bei diesem Lehrgerät um einen vollständigen Mikrocomputer, der auf einer großflächigen Platine aufgebaut ist. Im Gegensatz zu anderen Mikrocomputer Lehrgeräten wurde kein Mikroprozessor als integrierter Baustein verwendet, sondern sämtliche Funktionen sind mit Hilfe einzelner digitaler TTL-Schaltkreise realisiert. Dies hat für den Lernenden mehrere Vorteile:

- Der Inhalt von Befehlszähler, Registern und Flipflops sowie die Information auf dem Datenbus werden jederzeit über Leuchtdioden angezeigt.

- Die räumliche Anordnung der Schaltkreise auf der Platine ist so gewählt, daß sie optisch unmittelbar den Zusammenhang zum Blockschaltbild erkennen läßt. Dieser wird dadurch noch vertieft, daß die zum Verständnis wichtigen Datenleitungen auf der Platinenoberfläche angeordnet sind. So wird in direkter Weise die vom Verständnis her schwierige Lücke zwischen dem Blockschaltbild und dessen technischer Realisierung mittels einfacher Grundbausteine überbrückt.

- Durch die Zerlegung eines Maschinenbefehls in bis zu acht Teilschritte kann der zeitliche Ablauf eines einzigen Befehls erfaßt werden.

Diese Vorteile kann ein auf einem Mikroprozessor basierendes Lehrgerät prinzipiell nicht bieten, da hier alle komplexen Funktionen auf einem einzigen Halbleiterchip realisiert sind, in den man nicht "hineinschauen" kann.

Weitere Besonderheiten des vorgestellten Mikrocomputer Lehrgerätes sind:

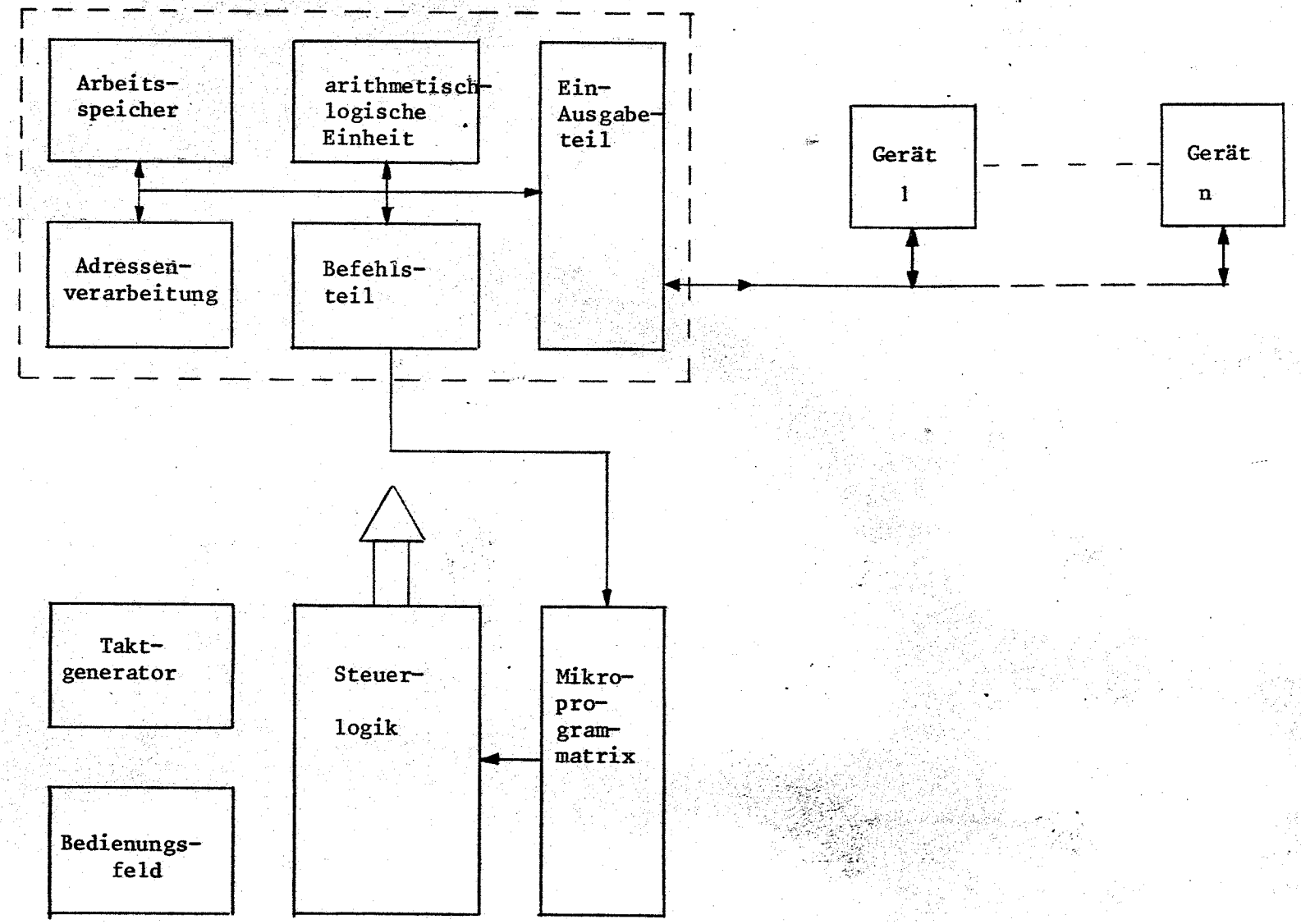
- Die Taktfrequenz ist bis zu einer Höchstgrenze kontinuierlich einstellbar, d.h. der Rechner kann vor allem auch so langsam arbeiten, daß man die Abarbeitung der einzelnen Befehle in einem Programm und deren Auswirkungen auf Registerinhalte u.s.w. anhand der zahlreichen Leuchtdioden Schritt für Schritt verfolgen kann.

- Die Bedeutung der Maschinenbefehle wird durch eine Matrix festgelegt, die mit diskreten Dioden bestückt ist und an einer Stelle verändert werden kann. Mit ihrer Hilfe läßt sich der Begriff der Mikroprogrammierbarkeit an einem Lehrgerät verstehen.

- Eine übersichtliche Ein- und Ausgabeorganisation gestattet es, die ebenfalls angebotenen Zusatzgeräte (Digital-Analogwandler, 7-Segmentanzeige, Eingabetastatur u.a.m.) in einfacher Weise vom Programm her anzusprechen und die Einsatzmöglichkeiten eines Rechners zur Prozeßsteuerung anhand praktischer Beispiele zu erfassen.

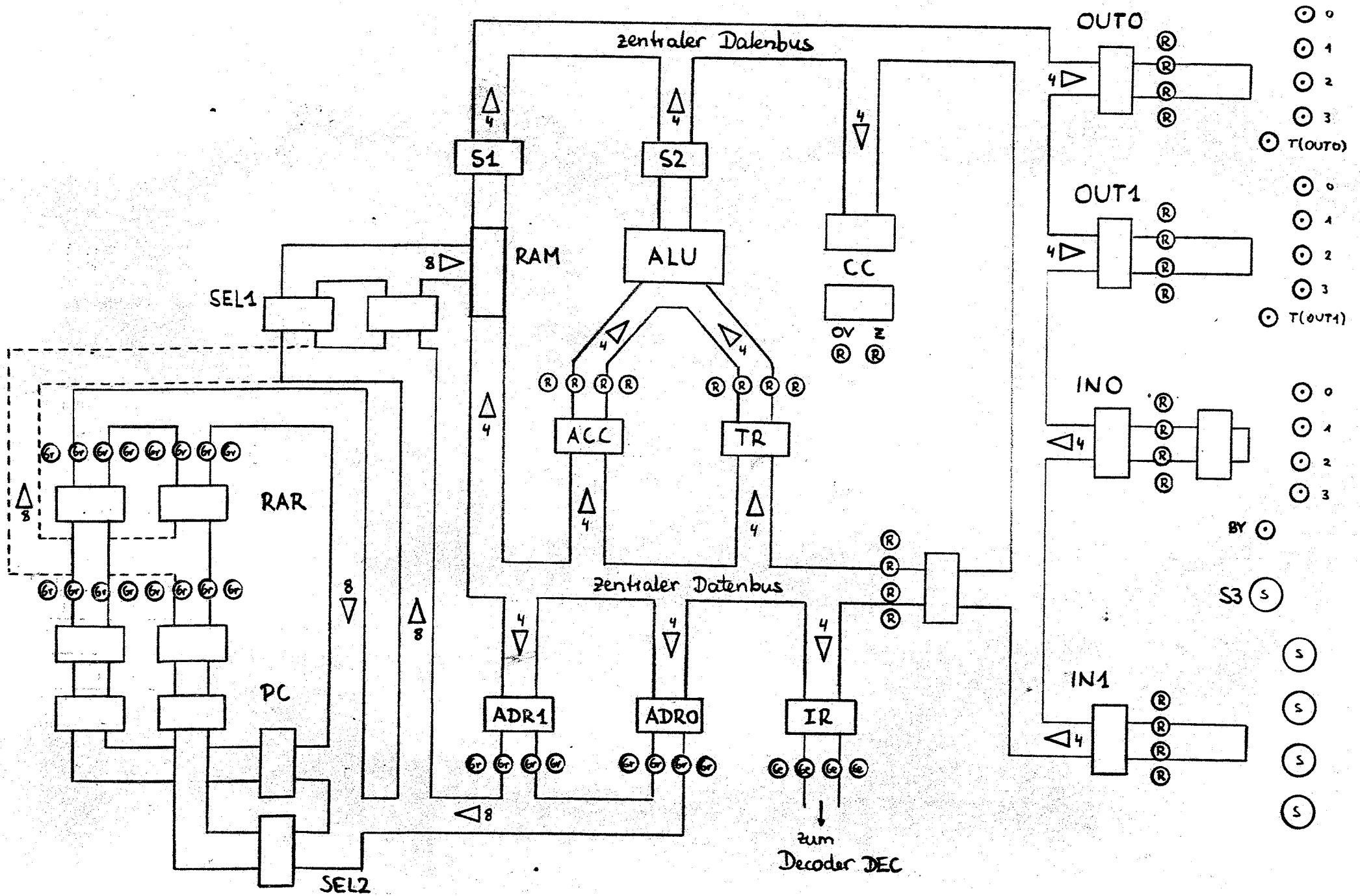
Dieses neue Mikrocomputer Lehrgerät erleichtert wesentlich das Verständnis des langen Weges zwischen Gattern als logischen Grundbausteinen und dem Mikrocomputer, seiner Programmierung und seinem praktischen Einsatz.

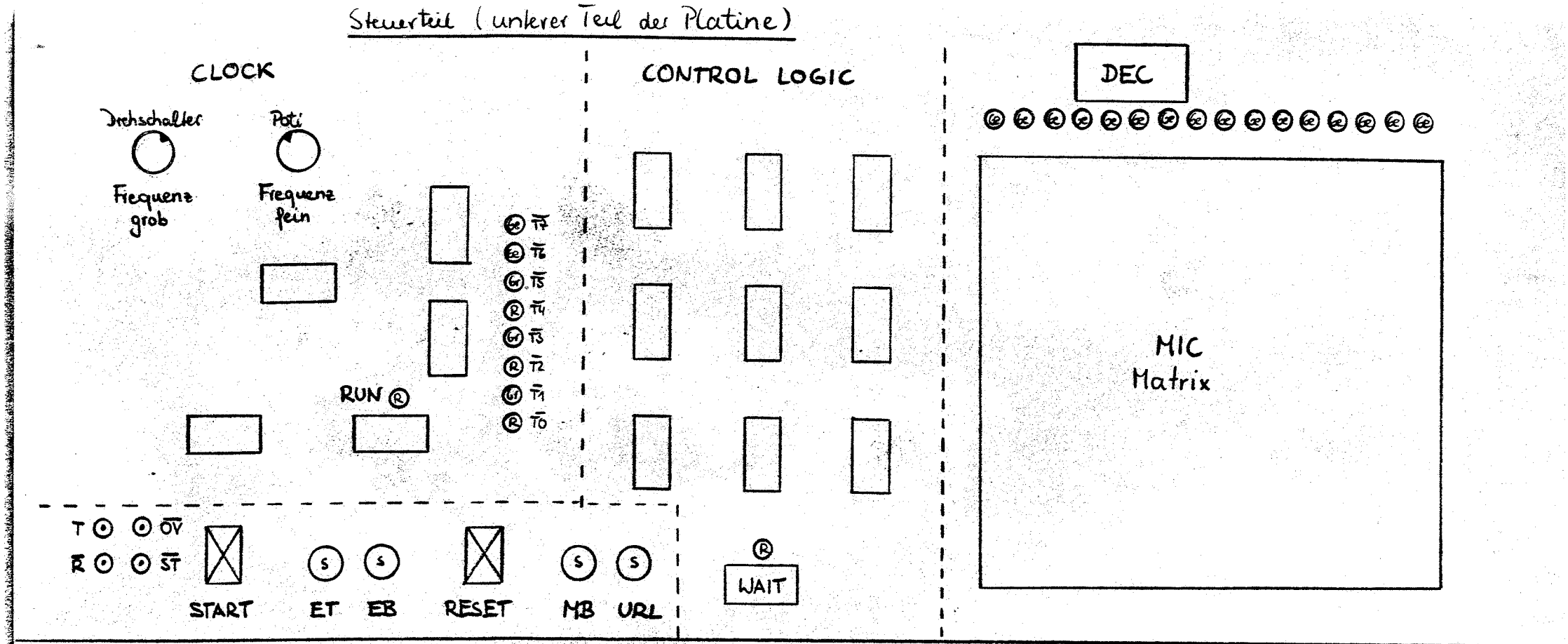
2. Blockschaltbild 1



Datenverarbeitungsteil (oberer Teil der Platine)

2. Blockschaltbild 2





integrierter Schaltkreis

Leuchtdioden (5mm Ø)

Buchse (2mm)

Schalter

Taster

Daten- oder Adreßbus (der gestrichelte Teil liegt auf der Platinenunterseite)

8

Richtung des Datenflusses

Breite des Bus (4 Bit, 8 Bit)

Maßstab ca.: 1:2

3. Technische Daten

Typ:	digitaler, mikroprogrammierter 4-Bit Mikrocomputer, nur aus TTL-Schaltkreisen auf einer Platine (45 x 40 cm) aufgebaut
Arbeitsspeicher:	RAM, 256 Worte à 4 Bit
Befehlssatz:	16 Befehle, ein, zwei oder drei Worte lang; Bedeutung durch Diodenmatrix definiert, ein Befehl durch Ein- und Auslöten von Dioden veränderbar
Register:	4-Bit Akkumulator mit 4-Bit Hilfsregister und 4-Bit arithmetisch-logischer Recheneinheit, sowie Zero- und Overflow-Flipflop; zwei 4-Bit Ausgaberegister, ein 4-Bit Eingabekanal, ein 4-Bit Schalterregister zur manuellen Befehls- und Dateneingabe, 4-Bit Befehlsregister mit 8-Bit Adreßteil, 8-Bit Befehlszähler, 8-Bit Rücksprungadressenregister, ein Monoflop zum kurzzeitigen Stop des Rechners
Betriebsarten:	Taktgenerator ermöglicht Betriebsarten Dauerlauf, Einzeltakt und Einzelbefehl; Taktfrequenz zwischen ca. 0.1 Hz bis ca. 1 MHz kontinuierlich einstellbar, bei 1 MHz beträgt die Ausführungszeit eines Maschinenbefehls 8 µs; Rechner kann in den Betriebsarten Urladen, manuelle Befehlsausführung und Normalbetrieb arbeiten.

Aufbau, Maße:	durchkontaktierte Epoxyd-Platine, Maße 400 x 450 mm, Leiterbahnen verzinnt, Platine auf beiden Seiten durch Plexiglasplatten geschützt; 42 integrierte TTL-Bausteine, 1 RAM, 84 verschiedenfarbige 5 mm - Leuchtdioden zur Anzeige von Registerinhalten u.s.w., 9 Schalter, 2 Taster, zwei 4 mm Buchsen zur Stromversorgung, 21 2 mm Buchsen für Daten und Steuersignale
Stromversorgung:	5 V ± 0.25 V, ca. 1.6 A, einfacher Überspannungs- und Verpolungsschutz auf Platine vorhanden, Standard-TTL-Pegel für alle Ein- und Ausgänge
Literatur:	Anleitung mit ca. 100 Seiten mit vollständiger Beschreibung und Programmbeispielen
Zusatzgeräte:	2 x 4-Bit Anzeigeeinheit (7-Segmentziffern), 2 x 4-Bit Digital-Analogwandler (0.. 3.5 V), 4-Bit Eingabetastatur, Verteilerplatine Interface zur Datenspeicherung auf Tonband oder Kassettenrekorder (z.Z. in Entwicklung) 4-Bit Analog-Digitalwandler (z.Z. in Entwicklung)

4. Anzeigeeinheit

Mit dieser zweifach 4-Bit Anzeigeeinheit können zwei 4-Bit Binärzahlen als 7-Segmentziffern dargestellt werden. Optisch entspricht die Anzeige dem hexadezimalen Wert der Binärziffer.

Die Anzeigeeinheit hat zwei eigene 4-Bit Register, die den anzuzeigenden Wert vom Rechner übernehmen, speichern, über Leuchtdioden binär anzeigen. Ein 7-Segmentdekoder übernimmt dann die Ansteuerung der einzelnen Segmente.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Das Gerät kann z.B. dazu dienen, 4-Bit Daten als Ergebnis einer Rechnung sowohl in binärer Form als auch als Hexadezimalziffer anzuzeigen.

Stromversorgung:	5 V $\pm$ 0.25 V, max. 400 mA
Aufbau, Maße:	durchkontaktierte Epoxyd-Europaplatine, 160 x 100 mm, Plexiglasgeschützt, 5 IC's, 8 LED's, 2 7-Segmentziffern mit 8 mm Ziffernhöhe, 10 2 mm Buchsen

5. Digital-Analogwandler

Mit diesem zweifach 4-Bit Digital-Analogwandler kann man aus zwei 4-Bit Binärzahlen zwei Spannungen erzeugen, die proportional zum binären Wert dieser Zahlen sind. Der maximale Spannungsbereich kann über Potentiometer eingestellt werden und liegt zwischen 0 Volt und ca. 3.5 Volt. Durch die 4-Bit Auflösung ist dieser Spannungsbereich somit in 16 Stufen eingeteilt. Es gilt:

$$U = n/16 \cdot U_{max} ,$$

wobei U_{max} durch die Potentiometerstellung und die Strombelastung des Analogausganges gegeben ist.

Die Digital-Analogwandlung erfolgt mittels eines Widerstandnetzwerkes aus vier Widerständen, die im Verhältnis 1:2:4:8 zueinander dimensioniert sind. Diese Widerstände sind Trimmpotentiometer, die bei Bedarf nachjustiert werden können bzw. verstellt werden können, um das Prinzip der Digital-Analogwandlung zu verstehen.

Der zweifach 4-Bit Digital-Analogwandler hat zwei eigene 4-Bit Register, die den jeweiligen zu konvertierenden binären Wert vom Rechner übernehmen und speichern.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Die beiden Analogausgänge können z.B. zur Ansteuerung der x- und y-Ablenkung eines Oszillographen oder eines XY-Schreibers dienen. Da beide Ausgänge hochohmig sind, können niederohmige Verbraucher nur über einen Verstärker angeschlossen werden. Damit können dann Motoren, Lampen, Spulen und andere Stromverbraucher mit dem vom Rechner bestimmten Analogsignal gesteuert werden.

Stromversorgung: 5 V ± 0.25 V, max. 150 mA

Aufbau, Maße: durchkontaktierte Epoxyd-Europaplatine,
160 x 100 mm, Plexiglasgeschützt,
3 IC's, 8 LED's, 12 2 mm Buchsen

6. Eingabetastatur

Mit diesem Gerät kann ein 4-Bit Wort über Tasten dem Rechner übertragen werden. Die Eingabetastatur enthält einen eigenen 4-Bit Speicher, der die Tastenkombination speichert und dessen Inhalt bei Anforderung des Rechners abgerufen werden kann. Bei Betätigung einer Eingabetaste wird das zugehörige Bit invertiert. Diese Methode erlaubt eine schnelle Eingabe der 4-Bit Information.

Die Dateneingabe kann unter der Kontrolle des Rechners ablaufen. Dies geschieht folgendermaßen:

Die Tastatur kann über ein Flipflop gesperrt werden, d.h. ein Tastendruck bewirkt keine Änderung der gespeicherten Information. Vom Rechner her muß zunächst dieses Flipflop zurückgesetzt werden. Danach kann die Tastatur neue Information im Speicher unterbringen. Hat man das gewünschte 4-Bit Wort eingegeben und wird es über die Leuchtdioden angezeigt, wird bei Betätigung einer speziellen Übergabetaste dem Rechner über ein BY (Busy)-Signal mitgeteilt, daß nun gültige Information für ihn bereitliegt. Gleichzeitig wird mit diesem Tastendruck die Tastatur über das Flipflop gesperrt. Nachdem der Rechner das 4-Bit Wort abgeholt und verarbeitet hat, kann er die Tastatur wieder freigeben.
Durch diese unter Kontrolle des Rechners erfolgende Dateneingabe kann es nie vorkommen, daß Daten zu langsam, zu schnell eingegeben oder gar "verloren" gehen können.

Die Schnittstelle zum Rechner (Daten- und Steuersignale) ist so ausgebildet, daß das Gerät unmittelbar an den Rechner angeschlossen werden kann.

Stromversorgung:	5 V ± 0.25 V, max. 150 mA
Aufbau,Maße:	durchkontaktierte Epoxyd-Europaplatine, 160 x 100 mm, Plexiglasgeschützt, 5 IC's, 5 LED's, 11 2 mm Buchsen, 5 Tasten (entprellt)