

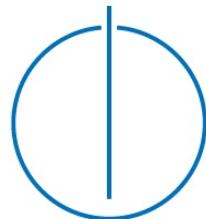
**Technische Universität  
München**

**Fakultät für Informatik**

**Seminararbeit in Informatik**

Die ersten Mikroprozessoren: ein Rechner, ein Chip

Michael Kratzer, Michael Kiener



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
<b>2 Der Intel 4004</b>	<b>4</b>
2.1 Die Geschichte und Entwicklung des 4004 . . . . .	5
2.2 Die Architektur des MCS-4 . . . . .	7
2.2.1 Der 4001 . . . . .	8
2.2.2 Der 4002 . . . . .	9
2.2.3 Der 4003 . . . . .	12
2.2.4 Der 4004 . . . . .	13
2.3 Der Befehlszyklus . . . . .	15
2.4 Der Befehlssatz . . . . .	15
<b>3 TMS-1000</b>	<b>18</b>
3.1 Geschichte . . . . .	19
3.2 Mikrocontroller . . . . .	19
3.2.1 Aufbau . . . . .	19
3.2.2 Abgrenzung zu Mikroprozessoren . . . . .	20
3.2.3 Architekturen . . . . .	20
3.3 TMS-1000 . . . . .	21
3.3.1 Allgemeine Daten . . . . .	21
3.3.2 Pins . . . . .	21
3.3.3 Aufbau & Funktionsweise . . . . .	22
3.3.4 Befehlssatz . . . . .	27
3.4 Verwendung Mikrocontroller . . . . .	28
<b>Abbildungsverzeichnis</b>	<b>30</b>
<b>Tabellenverzeichnis</b>	<b>30</b>
<b>Literaturverzeichnis</b>	<b>32</b>

# Kapitel 1

## Einleitung

Test Einleitung

# **Kapitel 2**

## **Der Intel 4004**

## 2.1 Die Geschichte und Entwicklung des 4004

Der 4004 ist der erste von Intel entwickelte Mikroprozessor und einer der Ersten überhaupt. Er gilt als einer der großen Meilensteine, die den Siegeszug der Computer einleitete. Schon Jahre vor seiner Entwicklung fingen Halbleiterchips an, die alten Elektronenröhren Rechner abzulösen. Durch die schnell voranschreitende Miniaturisierung von Transistoren ließen sich immer komplexere Logik Konstruktionen auf immer kleineren Chips umsetzen. Was vielversprechend begann, stellte sich allerdings bald als Problem heraus. Als die Komplexität und Spezialisierung einzelner Designs so stark zunahm, dass es nicht mehr kosteneffizient war manche Designs umzusetzen. Die Chips waren so spezialisiert geworden, dass die geringe Absatzmenge nicht die Entwicklungskosten rechtfertigte. Deshalb war es nur eine Frage der Zeit bis ein Universalrechner aus Silizium gebaut wurde. Deshalb beschäftigt sich der folgende Abschnitt mit der Geschichte des Intel 4004 und den Personen, die maßgeblich zu seiner Entwicklung beigetragen haben.

Als im Sommer 1969 Busicom, ein japanischer Hersteller von elektrischen Rechenmaschinen, das Unternehmen Intel damit beauftragte die Chips für ihre neue Reihe von Rechenmaschinen zu produzieren, war Intel gerade mal ein Jahr alt. Das Unternehmen war spezialisiert auf die Herstellung von Halbleiter Speicherchips und hatte zum Zeitpunkt des Auftrags nur zwölf Mitarbeiter. Intel war zu dieser Zeit, auf Grund von fehlenden Absatzzahlen ihrer Speicherchips, in einem finanziellen Engpass und sah sich gezwungen die Arbeit anzunehmen.

Busicom hatte schon einen großen Teil der Designarbeit getan und schickte drei Mitarbeiter zu Intel nach Kalifornien um die Arbeit zu vollenden und sie Intel zu übergeben. Unter ihnen befand sich auch Masatoshi Shima, der eine wichtige Rolle in der Entwicklung des Intel 4004 spielen sollte. Als Kontaktpersonen wurde von Intel Marcian Edward "Ted" Hoff, Jr. und Stanley SStan" Mazor abgestellt. Hoff, der in den Jahren bevor er zu Intel wechselte, in Stanford während seiner Forschung schon viele Arten von Computern studiert hatte, konnte sein persönliches Interesse nicht zurückhalten und studierte die Designpläne Busicoms. Der Plan von Busicom umfasste sieben Chips mit den speziellen Aufgaben: Programmkontrolle, Dezimalarithmetik,

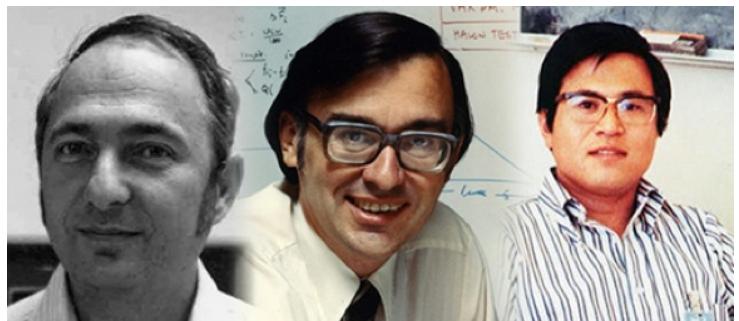


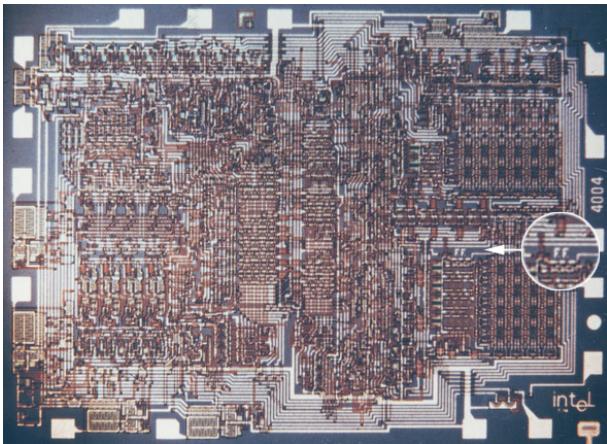
Abbildung 2.1: Mazor, Hoff und Shima

Timing, Read-Only-Memory, Schieberegister, Druckerkontrolle und Output Ports. Das ROM sollte zur Speicherung von Makroinstruktionen dienen, während die Schieberegister für den Datenspeicher genutzt werden sollten. Die Schieberegister wurden damals häufig verwendet, denn sie waren schnell für arithmetische Berechnungen und Ein- und Ausgabe Operationen. Problematisch hingegen waren das komplexe Timing und die langsamten Zugriffszeiten bei random access. Ein anderes Problem dieses ersten Designentwurfs war der komplexe Befehlssatz. Er bestand aus Makroinstruktionen, die viel fest verdrahtete Logik auf den Chips benötigten. Nachdem er das Design analysiert hatte, befand Hoff, dass es nur für Busicom Rechenmaschinen einsetzbar sein würde und damit für Intel nicht kosteneffizient sein würde. Auch hatte Intel zu diesem Zeitpunkt weder die nötigen Logikdesigner noch die Produktionsmöglichkeiten von Chips dieser Komplexität. Als Hoff der Führungsebene von Intel deshalb einen alternativen Vorschlag machte, waren sie sehr offen und ermutigten ihn sein Design weiterzuentwickeln. Eine der großen Änderungen, die Ted Hoff vorschlug, war das Aufbrechen von Makroinstruktionen in kleinere, elementare Mikroinstruktionen. Die alte Funktionalität ließe sich als Subroutine in einem Programm implementieren. Das erhöhte zwar den Speicherverbrauch für Programmcode, vereinfachte allerdings die Logik und war ein erster Schritt zu einem universal einsetzbaren Prozessor. Schon vorher sollten die Befehle aus einem ROM gelesen werden, die Schieberegister für wieder beschreibbaren Speicher wurden allerdings durch bei Intel entwickelte dynamische RAMs ersetzt. Diese hatten den selben Platzverbrauch wie Schieberegistern, waren bei zufälligen Speicherzugriffen aber weit aus schneller. Dadurch entstand auch, der Vorschlag drei Chips zu designen. Eine CPU für die Berechnungen, und zwei erweiterbare Speicher für Programm- und Datenspeicherung. Zusätzlich wurde ein weiterer Chip entworfen, der die Ein- und Ausgabe Möglichkeiten erweitern sollte. Das System dieser vier Chips wurde MCS-4 genannt. Eine Abschätzung des Designteams um Hoff, Mazor und Shima betrug 1900 Transistoren pro Chip, womit es bei weitem kostengünstiger als das währenddessen verbesserte Busicom Design war. Dieses enthielt nämlich immer noch 12 Chips mit jeweils 2000 Transistoren und 40 Pins. Die Einfachheit und die Flexibilität des neuen Chips waren Gründe, weshalb sich Busicom Ende 1969 dafür entschied mit



Abbildung 2.2: Federico Faggin

dem Vorschlag von Intel weiterzumachen anstatt das eigene Design weiter zu verfolgen. Für das Logik- und Chipdesign wurde Anfang 1970 Federico Faggin eingestellt. Er sollte zusammen mit Shima innerhalb von 6 Monaten eine getestete Logikschaltung entworfen und diese in einem integrierten Schaltkreis umgesetzt haben. Währenddessen würde



**Abbildung 2.3:** Intel 4004 mit Faggins Initialen

Shima an den Programmen für den neuen Prozessor arbeiten. Nach 9 Monaten harter Arbeit hatte Faggin es geschafft und es lagen die ersten funktionsfähigen Prototypen des Chips vor. Nach dem beheben einiger kleiner Fehler wurden die Chips an Busicom geliefert und wie geplant in den Rechenmaschinen eingesetzt. Einer der ersten Rechner, der das MCS-4 benutzte war der Busicom 141-PF. Er enthielt den 4004, 4 ROM-Chips mit dem Schon im Sommer 1971 kam Busicom allerdings erneut auf

Intel zu um die Produktionskosten der Chips zu reduzieren. Auf Grund von steigender Konkurrenz und fallenden Preisen war Busicom nicht mehr konkurrenzfähig und in finanziellen Problemen. Nach langen Diskussionen der Intel Führungsebene entschloss sich das Unternehmen auf Grund des vielfältigen Einsetzbarkeit des Prozessors die Rechte an dem Prozessor von Busicom für \$60.000 zurückzukaufen. So konnte Intel im Frühjahr 1971 die Chipreihe auf den Markt bringen und so der Öffentlichkeit zugänglich machen. Währenddessen lief schon die Entwicklung des Intel 8008, des ersten Intel 8-Bit Prozessors. Dieser profitierte klar von dem Design des 4004 und führte später zu Intels Erfolg.

## 2.2 Die Architektur des MCS-4

Unter dem MCS-4 veröffentlichte Intel in 1971 eine Sammlung von vier Chips. Die Chips waren durchnummiert von 4001 bis 4004 und stellten den ersten Universalcomputer auf Halbleiterbasis da. Das minimale System besteht aus nur einer CPU, dem 4004 und einem ROM. Es können allerdings bis zu 16 ROM und 16 RAM Chips angeschlossen werden. Verbunden werden diese Chips über einen 4 Bit breiten Bus. Der Prozessor kann zudem die anderen Chips durch Kontrollleitungen steuern. Als Schaltungskonzept wurde eine Mischung aus von Neumann und Harvard Architektur umgesetzt. Eigentlich

Konkurrenzarchitekturen vereint das MCS-4 Ansätze von beidem, in dem es wie in der Harvard Architektur üblich Programm- und Datenspeicher trennt, aber trotzdem beide Speicher über den selben Bus angeschlossen sind. Dadurch lassen sich Befehle und Daten nur sequentiell anstatt parallel laden. Die folgenden Abschnitte befassen sich mit dem Aufbau der einzelnen Chips.

### 2.2.1 Der 4001

Der 4001 ist ein programmierbarer Read-Only-Memory Chip. Er wird hauptsächlich dazu benutzt die Programmbefehle zu speichern. Dafür können im MCS-4 System bis zu 16 ROMs gleichzeitig angeschlossen werden. Pro Chip stehen dabei jeweils 2048 Bit zur Verfügung. Aufgeteilt sind diese in 256 Wörter mit jeweils 8 Bit. Der Chip besitzt 16 Pins über die Datenleitungen und Kontrollsingale angeschlossen werden. In 2.4 werden diese dargestellt. Die Pins  $D_0$  bis  $D_3$  sind die vier Datenleitungen, über die der 4 Bit breite Datenbus angeschlossen wird. Dieser Bus verbindet alle angeschlossenen 4001, 4002 und 4004

Chips. Über ihn werden die Speicheradressen vom Prozessor an das ROM geschickt. Daraufhin sendet das ROM die in der adressierten Speicherzelle enthaltene Instruktion zurück. Die Pins 5 und 12 werden mit den Versorgungsspannungen verbunden. Damit wird der Chip mit Energie versorgt.  $V_{SS}$  ist äquivalent zu GND, während an  $V_{DD}$  eine Spannung von  $-15V$  anliegt. An den Pins 6 und 7, bezeichnet mit  $\phi_1$  und  $\phi_2$ , werden zwei Taktsignale angeschlossen. Diese dienen der Synchronisation der Aktionen mehrerer Schaltkreise. In jedem Takt ist es für jeden Schaltkreis möglich eine Aktion auszuführen. Das SYNC-Signal, ist auch eine Art Takt und dient zur Synchronisation eines Befehlszyklus. Ein Zyklus besteht aus mehreren Takten der  $\phi_1$  und  $\phi_2$  Signale. Diese drei Signale sorgen für das Timing in allen Chips. Das CM-Signal auf Pin 11 wird dazu verwendet um einem adressierten Chip zu signalisieren, dass er in den nächsten zwei Zyklen auf den Bus schreiben darf. Das Reset-Signal löscht alle internen Register,

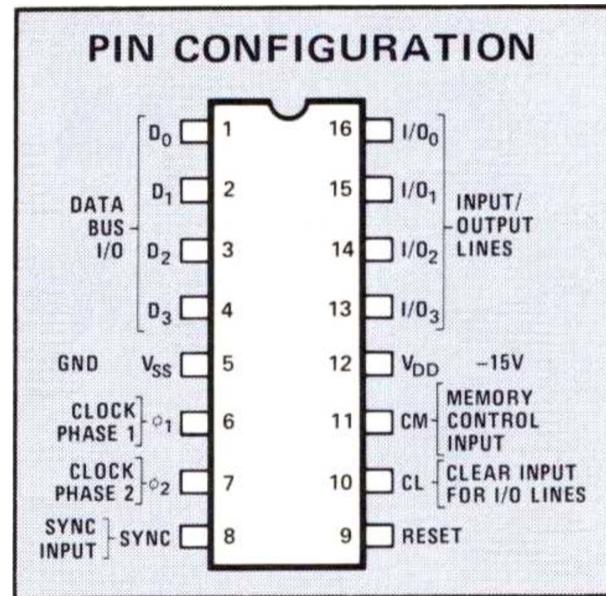


Abbildung 2.4: Pins des Intel 4001

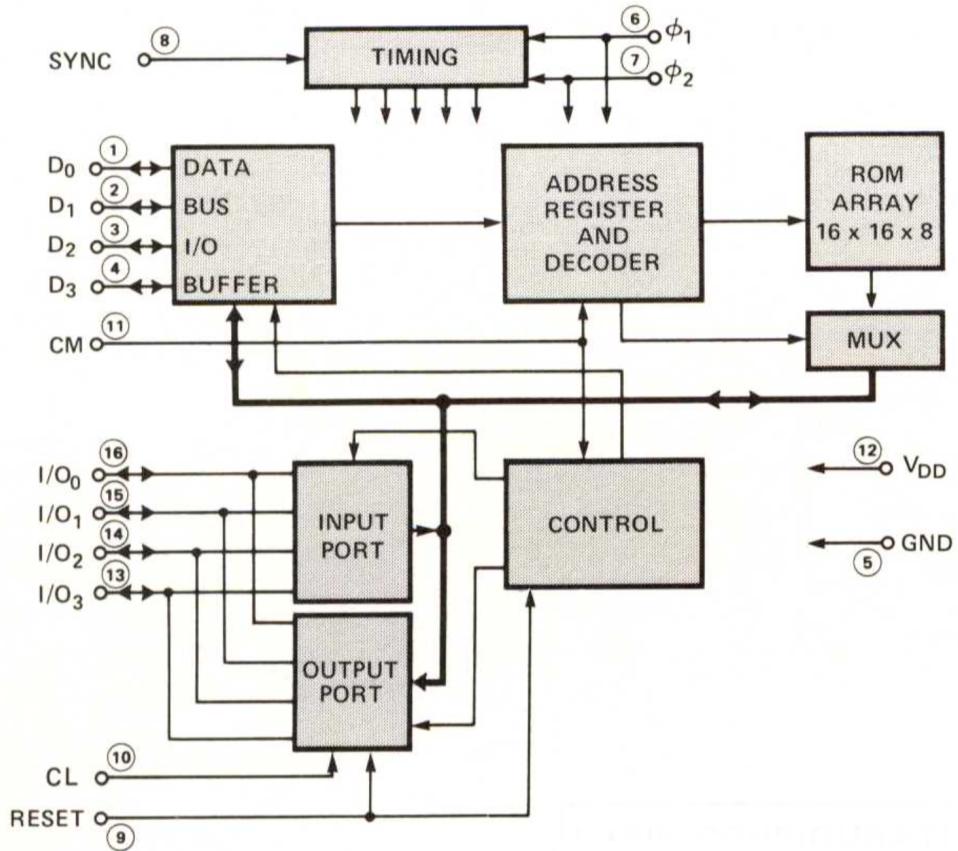


Abbildung 2.5: Layout des Intel 4001

mit der Ausnahme der Input und Output Register. Diese werden über das Clear-Signal gelöscht. Die Ein- und Ausgaberegister sind Teil der zweiten Funktionalität des Intel 4001. In seiner zweiten Funktion kann es auch als Schnittstelle zu Peripheriegeräten dienen. Dazu gibt es 4 weitere Leitungen an den Pins 13 - 16. In 2.5 ist zu sehen, dass es möglich ist jeden Pin individuell als Input oder als Output Port zu definieren. Über den internen Bus des 4001 werden Daten entweder direkt zu den I/O-Registern geliefert, oder wenn eine Adresse am Bus anliegt im Adressregister gespeichert. Nachdem die Adresse vollständig angekommen ist, wird sie dekodiert und die Instruktion aus dem ROM über einen Multiplexer wieder zurück auf den Datenbus geschrieben.

## 2.2.2 Der 4002

Der Intel 4002 ist das Gegenstück zu 4001. Während der eine die Programmdaten speichert, dient der 4002 als Datenspeicher. Im Vergleich zum 4001 besitzt dieser Chip weiter weniger Speicher. Nur 320 Bit stehen zur Verfügung aufgeteilt in 4 Register. Jedes dieser Register besteht aus 20 4-Bit Wörtern. Davon können 16 zum speichern von Daten

verwendet werden, während die letzten 4 als Registerstatusbits benutzt werden. Auch hier kann die Speichergröße wieder durch das Anschließen von bis zu 16 Chips erhöht werden. Wie der 4001 ist auch der RAM-Chip über den Datenbus mit den anderen Chips verbunden. Ebenso identisch sind die Timing Logik und Spannungsversorgung. Der erste Unterschied in der Pinbelegung ist der  $P_0$  Eingang. Der 4002 kommt in zwei verschiedenen Ausführungen: 4002-1 und 4002-2. Diese sind von der Logik identisch, haben jedoch eine andere Verdrahtung. Das führt dazu, dass pro CM-Signallinie des Prozessors nur vier 4002 Chips angeschlossen werden können. Pro Signallinie genau 2 von jeder Sorte. Damit ergeben sich folgende Adressen für die vier Chips an einer Signallinie:

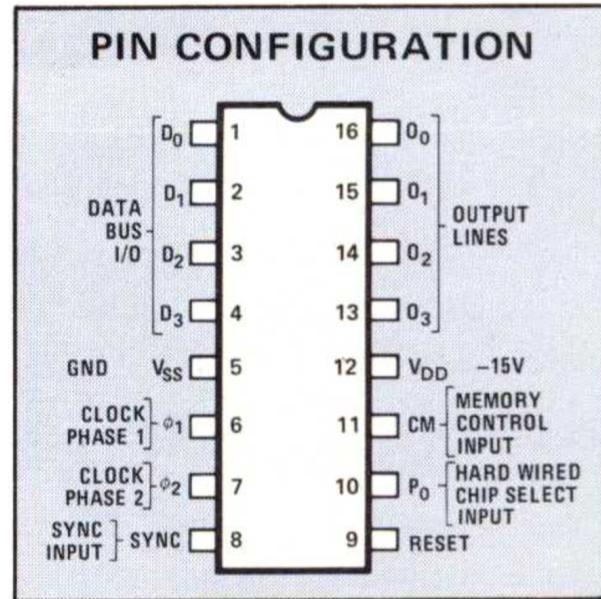


Abbildung 2.6: Pins des Intel 4002

Chipnummer	4002 Option	$P_0$	$D_3$	$D_2$
0	4002-1	GND	0	0
1	4002-1	$V_{DD}$	0	1
2	4002-2	GND	1	0
3	4002-2	$V_{DD}$	1	1

Tabelle 2.1: RAM-Chipauswahl

$P_0$  kann fest verdrahtet werden und ist eine zusätzliche Möglichkeit einen Chip auszuwählen. Wenn Daten aus dem RAM geladen werden sollen geschieht das in zwei Befehlszyklen. Im ersten wird die Adresse an den RAM Chip gesendet und im zweiten die Daten zurück an den Prozessor.

X2				X3			
$D_3$	$D_2$	$D_1$	$D_0$	$D_3$	$D_2$	$D_1$	$D_0$
Chip #				Register #			

Tabelle 2.2: RAM-Adressierung

Die Adressenübertragung findet in zwei Schritten statt, siehe 2.3. Im ersten Schritt X2 wird die Chipnummer und die Registernummer verschickt. Im zweiten die Adresse einer der 16 Speicherzellen innerhalb des Registers. Dabei sind D0 - D3 die vier Bits die gleichzeitig auf den 4 Datenbuslinien liegen. Befinden sich die gesuchten Daten nicht auf einem der 4 Chips der aktuell ausgewählten RAM-Bank, so muss vor der Adressenübertragung die Bank in einem extra Befehlszyklus gewechselt werden. Als RAM-Bank wird eine Kombination von bis zu vier RAM-Chips genannt, die an der selben CM-Signallinie liegen. Mit Hilfe des designate-command-line Befehls wird eine CM-Linie ausgewählt. Wie bei den 4001 Chips gilt auch hier, dass ein Chip nur Daten vom Bus empfängt, wenn seine CM-Linie aktiviert wird.

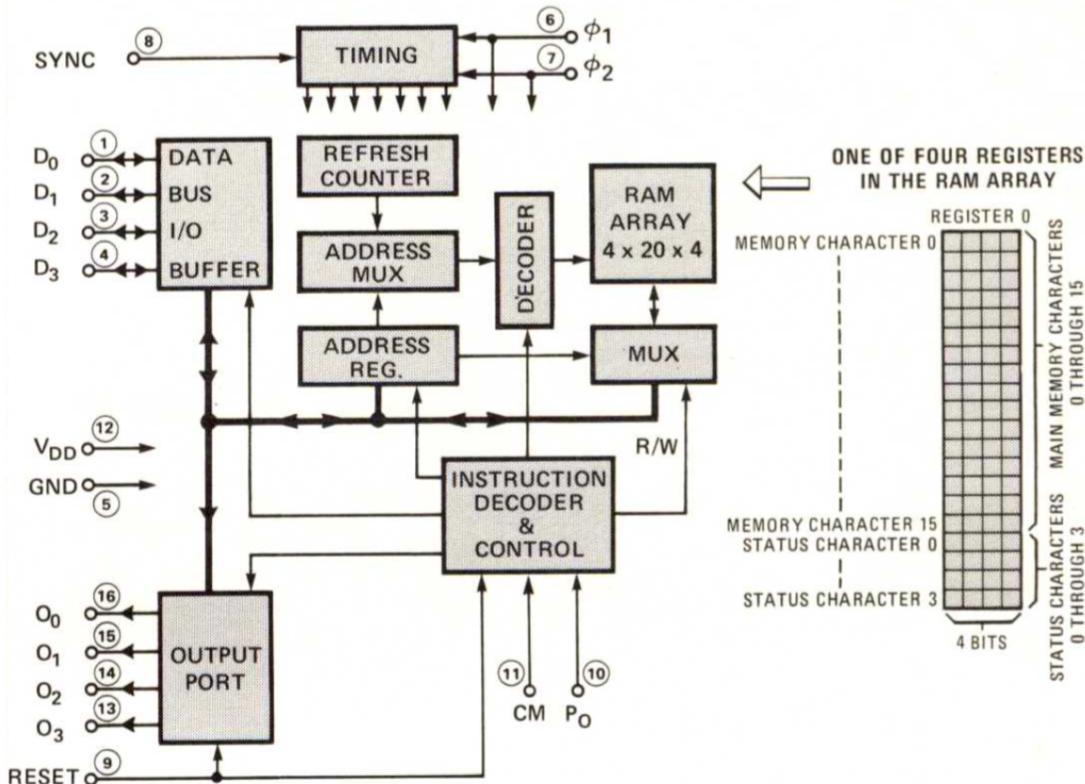


Abbildung 2.7: Layout des Intel 4002

Nachdem die Adresse angekommen ist, wird sie wie beim 4001 in einem Adressenregister zwischengespeichert und dekodiert. Danach wird im nächsten Befehlszyklus der Wert der entsprechenden Speicherzelle auf den Datenbus gelegt. Wie der Intel 4001 besitzt auch der 4002 4 Output Pins. Auch hier können Peripheriegeräte angeschlossen werden. Alle Register sowie der Speicher werden gelöscht wenn das Reset-Signal für mehr als 32 Befehlszyklen aktiviert wird.

### 2.2.3 Der 4003

Der dritte Chip aus der Reihe ist Intel 4003. Er besteht hauptsächlich aus einem 10 Bit Schieberegister. Der Chip wurde entworfen um die Ein- und Ausgabemöglichkeiten des Systems zu erweitern. Während der 4002 schon über Ausgabe und der 4001 sowohl über Aus- als auch Eingabe Möglichkeiten verfügt, gibt es immer noch Fälle in denen diese Pins nicht ausreichen. Deshalb können die 4003 Chips an diese Pins angeschlossen werden. Über den Serial-Input Pin wird das Schieberegister mit Daten befüllt. Über den CP-Pin wird das verschieben der Bits innerhalb des Registers gesteuert. Bei jeder Verschiebung wird das letzte Bit auf den seriellen Ausgang geschrieben. Die Daten aus dem Schieberegister können auch parallel ausgelesen werden. Dazu stehen 10 weitere Pins zur Verfügung.

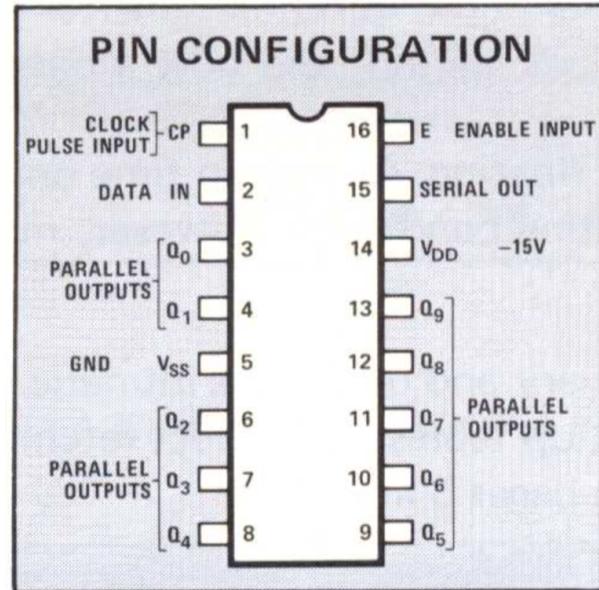


Abbildung 2.8: Pins des Intel 4003

Die Daten aus dem Schieberegister können auch parallel ausgelesen werden. Dazu stehen 10 weitere Pins zur Verfügung.

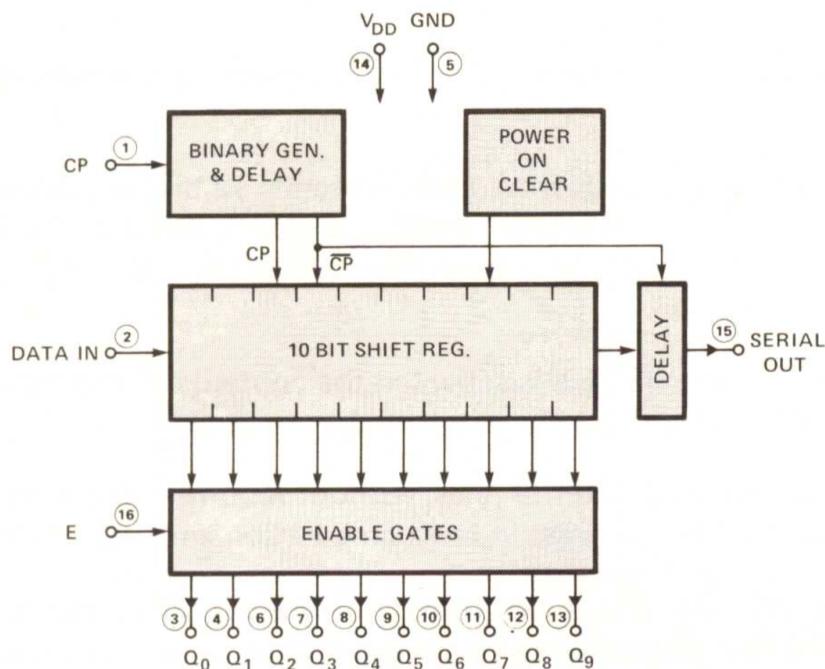


Abbildung 2.9: Layout des Intel 4003

Die parallele Ausgabe wird mit Hilfe des Enable-Signal gesteuert. Wenn es aktiviert wird, werden alle 10 Bits gleichzeitig über die Pins  $Q_0$  bis  $Q_9$  ausgegeben. Um mehr Geräte wie Tastaturen, Displays oder Drucker anschließen zu können, können mehrere dieser Chips hintereinander geschaltet werden, so dass ein vielfaches der 10 Pins erreicht werden können.

## 2.2.4 Der 4004

Der Intel 4004 ist das Kernstück des Systems. Der Chip ist eine CPU und wurde entworfen um mit den anderen Chips zusammenzuarbeiten und sie zu steuern. Er bearbeitet die Befehle, welche im 4001 ROM gespeichert sind. Dazu muss müssen die Befehle erst aus dem ROM geladen werden. Dazu wird die Adresse über den Datenbus an die ROMs verschickt und das CM-ROM Signal aktiviert. Wenn der aktuelle Befehl geladen wurde, wird er im Instruktionsregister gespeichert. Das Register besteht aus zwei 4 Bit Wörtern. Die ersten vier geladenen Bit enthalten den Opcode und werden im OPR Teil des Registers gespeichert. Die zweiten Vier enthalten den Modifier. Er wird im OPA Teil des Registers gespeichert und enthält Adressen oder Daten, auf denen der Befehl ausgeführt wird. Direkt an das Register ist der Dekodierer angeschlossen, der die Instruktion dekodiert um sie ausführen zu können. Der zweite große Bestandteil des 4004 ist das Indexregister. Das Indexregister ist ein interner Cache zur Speicherung von Zwischenergebnissen und Instruktionen. Dazu können die 64 Bit auf zwei verschiedene benutzt werden. Sie sind in 8 Reihen mit jeweils 8 Bit pro Reihe organisiert. Trotzdem können 16 verschiedene Zellen mit 4 Bit adressiert werden. Dadurch können sowohl 4 Bit Daten oder eine Reihe von 8 Bit als Instruktion oder Adresse ausgelesen werden. Der dritte große Teil der CPU ist die ALU (Arithmetic Logical Unit). Die ALU besteht aus einem 4 Bit Addierer und einem Akkumulator inklusive einer Carry Flip-Flop. Der erste Term für die Addition/Subtraktion kommt aus einem Register, das über den internen Datenbus gefüllt wird. Der zweite Term kommt aus dem Akkumulator. Das Ergebnis der Berechnung wird wieder im Akkumulator gespeichert und das Carry-Bit gesetzt, falls ein Überlauf auftritt. Die

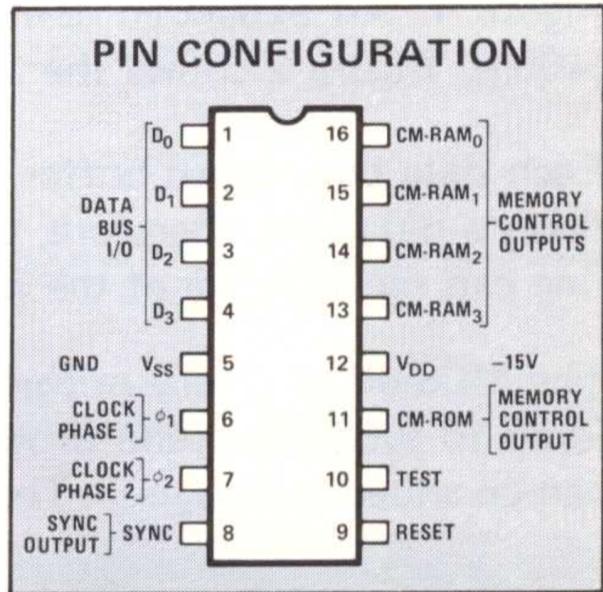


Abbildung 2.10: Pins des Intel 4004

Berechnungen werden in Binärarithmetik ausgeführt. Es ist aber auch möglich einen *Decimal Adjust* durchzuführen. Mit seiner Hilfe lassen sich Binärzahlen in so genannte Binary-Coded-Decimal Zahlen umwandeln. In dieser Art der Zahlendarstellung wird jede Dezimalstelle einer Zahl durch 4 Bit dargestellt. In diesen 4 Bit werden nur die Zahlen von 0-9 verwendet. Dazu wird nach der Addition zweier in BCD-Zahlen die Zahl 6 aufaddiert. Bei einer Addition die als Ergebnis eine Zahl über 9 liefert, kann dann nach dem *Decimal Adjust* die Zahl in zwei 4 Bit Teile aufgeteilt werden, die dann dem Ergebnis wieder als BCD-Zahlen entsprechen. Als Beispiel hier die Rechnung 9 + 8:

$$\begin{array}{r} 1001 \\ 9 \end{array} \quad + \quad \begin{array}{r} 1000 \\ 8 \end{array} = \quad \begin{array}{r} 10001 \\ 17 \end{array}$$

$$\begin{array}{r} 10001 \\ 17 \end{array} \quad + \quad \begin{array}{r} 0110 \\ 6 \end{array} = \quad 00010111 = 0001\ 0111$$

Das Adressenregister des Intel 4004 besteht aus 4 mal 12-Bit RAM. In einer dieser 12 Bit Speicherzellen liegt die aktuelle Programmadresse. Die zusätzlichen 3 Zellen werden benutzt um Adressen für Subroutinen zu speichern. Das bedeutet, dass bei der Programmierung des Prozessors bis zu 3 Subroutinen verschachtelt werden können.

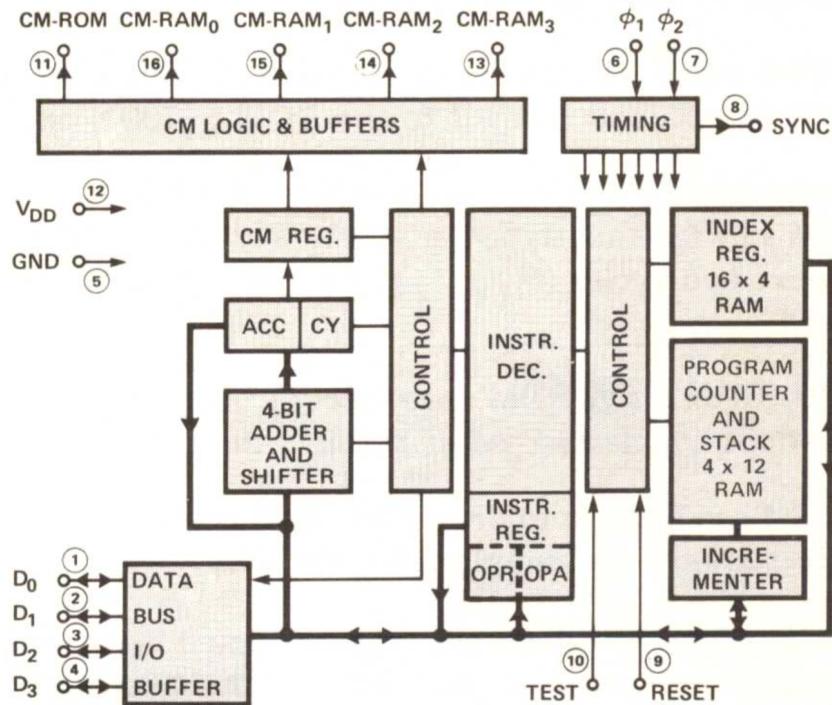


Abbildung 2.11: Layout des Intel 4004

## 2.3 Der Befehlszyklus

Der Befehlszyklus ist eine definierte Abfolge von Schritten, die dazu führen dass eine Instruktion ausgeführt wird. Jeder Befehlszyklus besteht aus 8 Takten. Ein Befehlszyklus beginnt mit einer steigenden Flanke des SYNC-Signals. In den ersten drei Takten  $A_1$  -  $A_3$  wird die auszuführende Instruktion im ROM adressiert. In  $A_1$  und  $A_2$  wird die Adresse auf dem Chip versendet und in  $A_3$  die Nummer des Chips auf dem sich die Speicherzelle befindet. Zusätzlich wird in  $A_3$  die CM-ROM Signallinie aktiviert. Die folgenden zwei Takte  $M_1$  und  $M_2$  werden dazu verwendet die 8 Bit Instruktion aus dem ROM in den Prozessor zu laden. In den letzten drei Takten  $X_1$  -  $X_3$  werden die Instruktionen ausgeführt. Zum Beispiel werden hier Adressen an RAM Chips geschickt, Berechnungen ausgeführt oder Daten von Ein- und Ausgabe Operationen gespeichert.

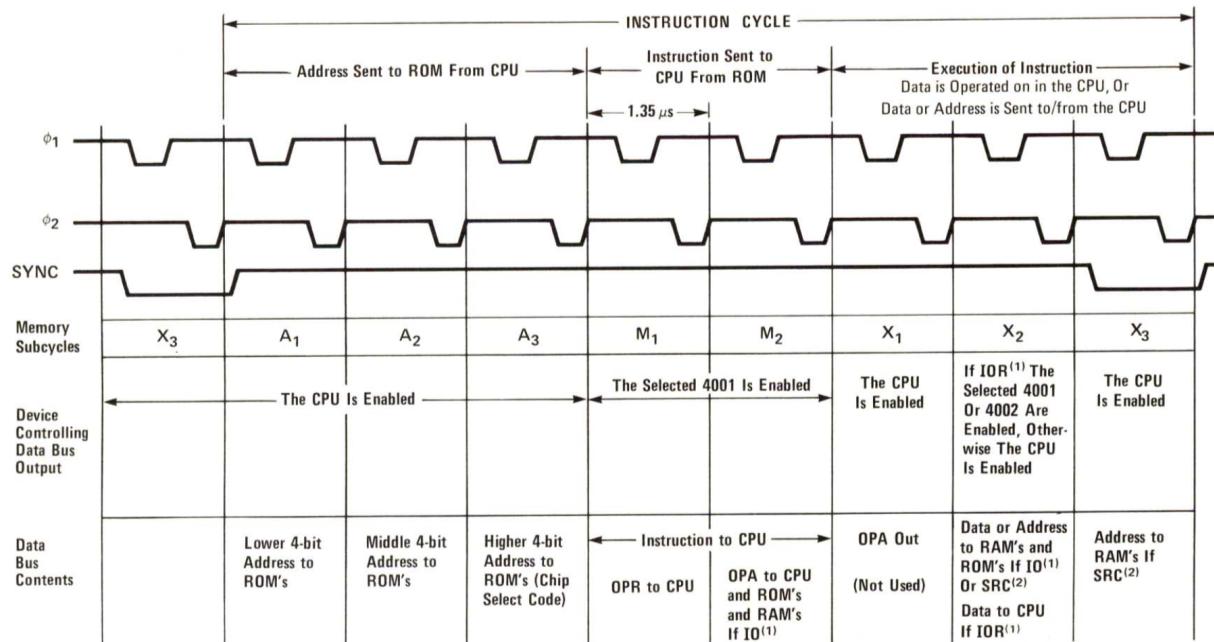


Abbildung 2.12: Der Befehlszyklus des MCS-4

## 2.4 Der Befehlssatz

Der Befehlssatz enthält alle Instruktionen, die der Intel 4004 ausführen kann. Ein Befehl ist normalerweise 8 Bit lang und besteht aus dem Opcode und dem Modifier. Befehle mit 8 Bit Länge haben drei mögliche Besetzungen für das Modifier Feld. Zum einen

kann es die Adresse einer der 16 Speicherzellen des Indexregisters enthalten. Das sind Befehle wie die arithmetischen Operationen, die den Inhalt dieser Adresse mit dem Wert des Akkumulators verrechnen. Die zweite Möglichkeit ist die Adresse eines 8 Bit Wortes innerhalb des Indexregisters. Das *Least-significant-bit* kann hier entweder auf 0 oder 1 gesetzt werden. Das wird als Erweiterung des Opcodes verwendet. Befehle die diese

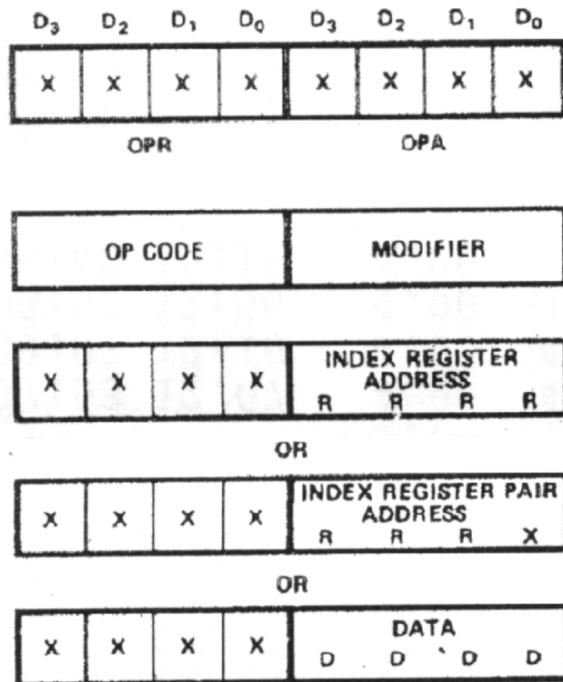


Abbildung 2.13: Befehle mit einem Wort Länge

Struktur verwenden sind SRC, welcher den Wert als Adresse verwendet und an ROM und RAM versendet, oder JIN, welcher zu dieser Adresse im Programmspeicher springt. Als letztes gibt es noch die Möglichkeit Daten zu übertragen. LDM lädt beispielsweise die zweiten 4 Bit der Instruktion in den Akkumulator. Fünf der insgesamt 45 Befehle bestehen aus 16 anstatt 8 Bit. Sie können deshalb nicht in einem Befehlszyklus bearbeitet werden.

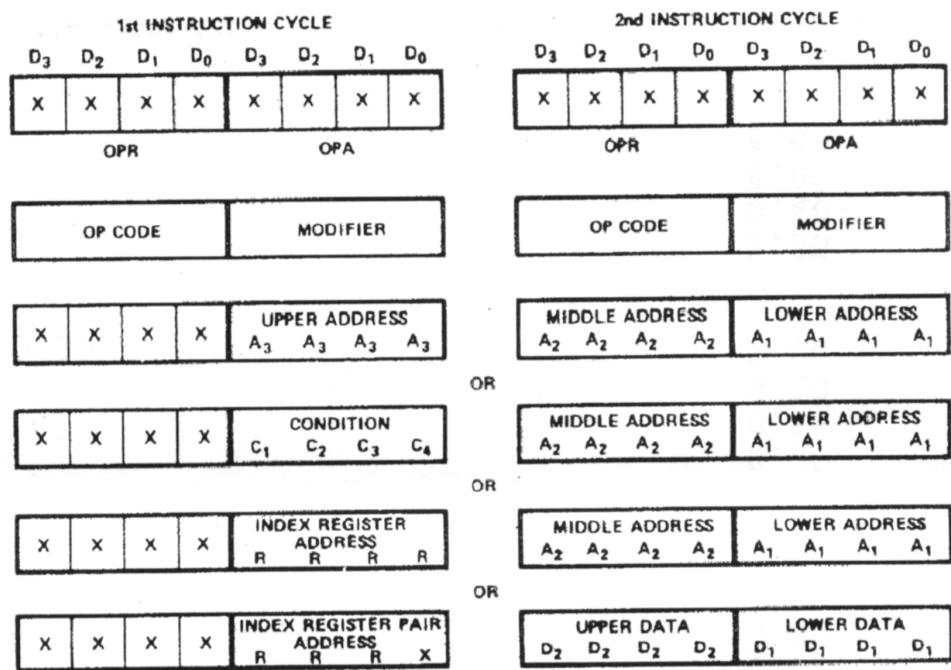


Abbildung 2.14: Befehle mit doppelter Wort Länge

# **Kapitel 3**

## **TMS-1000**

## 3.1 Geschichte

Während Intel es erstmals schaffte alle Bausteine eines Prozessors auf einem Mikrochip zu vereinigen, implementierte Texas Instruments parallel zusätzlich Peripherie auf dem Chip und schuf so den ersten Mikrocontroller, welcher auch System-on-a-Chip genannt wurde.

Texas Instruments wurde 1951 von Cecil Howard Green, Jon Erik Jonsson, Eugene McDermott und Henry Bates Peacock gegründet. Das Unternehmen war unter anderem für den ersten integrierten Schaltkreis bekannt. Die Konstruktion eines System-on-a-Chip gelang erstmal den Ingenieuren Gary Boone und Michael Cochran mit dem Controller TMS-1000 im Jahre 1971. Texas Instruments stellte verschiedene Versionen des TMS-1000 her, die in Größe des ROM und RAM Speichers variierten. Anfangs verwendete die Firma die Controller nur in eigenen Produkten, vor allem Taschenrechner, wie der SR-16, welcher 1972 auf den Markt kam. Die Firma ist aus diesem Grund und aufgrund der zahlreichen Nachfolger für ihre Taschenrechner bekannt.

Erst 3 Jahre nach der Erfindung, also 1974, war der TMS-1000 auf dem freien Markt erhältlich. Dies hatte jedoch zur Folge, dass der Markt durch Intel mit ihren Mikroprozessoren schon zum großen Teil eingenommen war. Trotzdem hatte der Controller aufgrund seines extrem niedrigen Preises, 2\$ pro Stück, großen Erfolg auf dem Markt. Der TMS-1000 half dadurch die moderne Elektronik für jedermann zugängig zu machen. Der Controller fand nicht nur in Taschenrechnern Verwendung, sondern auch in ersten Handheld-Geräten, Jukeboxen, Türklingeln, Uhren und vielen mehr. Bis zum heutigen Tag wurden ca. 100 Millionen Controller verkauft.

## 3.2 Mikrocontroller

### 3.2.1 Aufbau

Mikrocontroller werden nicht zu Unrecht System-on-a-chip genannt und der Aufbau eines Controllers kann daher sehr gut mit den Bestandteilen eines Computers verglichen werden. Im folgenden werden die Komponenten eines Computers und die Bestandteile eines Mikrocontrollers verglichen. Außerdem wird die Aufgabe der einzelnen Bauteile des Controllers beschrieben.

PC	Mikrocontroller	Aufgabe
Prozessor	CPU	Der Prozessor führt die arithmetischen und logischen Operationen aus
Arbeitsspeicher	RAM	RAM ist ein temporärer Speicher für Variablen. Verliert den Speicherinhalt nach dem Entfernen der Betriebsspannung
Festspeicher	ROM	Enthält das Programm
Takt	Takt	Gibt die Geschwindigkeit der Befehlsfolge an
Peripherie	I/O-Ports	Der Controller enthält einfache Ein- und Ausgänge für beispielsweise LEDs, Displays und Schalter

### 3.2.2 Abgrenzung zu Mikroprozessoren

Oftmals ist es schwierig eine klare Grenze zwischen Mikroprozessoren und Mikrocontrollern zu ziehen. Das liegt vor allem daran, dass nach einiger Zeit meist Mikrocontroller-Varianten von neuen Mikroprozessor-Architekturen erschienen sind. Die hauptsächliche Abgrenzung erfolgt durch die Ausstattung der Zusatzmodule des Chips. Während ein Mikrocontroller einen Prozessor inklusive Bausteinen wie Speicher, In- und Outputs, Timer, usw. vereint, konzentriert sich ein Mikroprozessor auf seine eigentliche Hauptaufgabe, die Rechengeschwindigkeit. Durch den gesparten Platz auf dem Chip wird diese wesentlich erhöht.

### 3.2.3 Architekturen

Viele der verbauten Mikrocontrollern verwenden 8-Bit-Prozessoren, deren Architektur auf die 1970er Jahre zurückzuführen ist. Allerdings gibt es auch 4-, 16- und 32-Bit Mikrocontroller. Die ersten Controller die gebaut wurden waren dabei 4-Bit-Controller und sind deswegen heutzutage noch stark vertreten. Den größten Marktanteil haben mittlerweile die 32-Bit-Controller da die meisten heutigen Mikrocontroller auf Prozessorkernen basieren, von denen 8- und 16-Bit Prozessoren fast nicht mehr hergestellt werden.

## 3.3 TMS-1000

### 3.3.1 Allgemeine Daten

Der TMS-1000 gehört zur Familie der 4-Bit Mikrocontroller. Die Größe des ROMs des Controllers beträgt 8.192 Bit, während die des RAMs 256 Bit beträgt. Die maximale Spannung, welche an der Clock und sowohl an den Eingang- und Ausgang-Pins angelegt werden darf, bemisst sich auf 20 Volt. Durch die intern verbauten Oszillatoren kann der Controller einen Takt von bis zu 0,4 MHz erreichen. Jede Instruktion benötigt 6 Oszillatoren-Zyklen um vollständig ausgeführt zu werden. Insgesamt verfügt der TMS-1000 über 43 Basisinstruktionen, 12 fixierte und 31 programmierbare.

### 3.3.2 Pins

Der TMS-1000 verfügt über insgesamt 28 Pins.

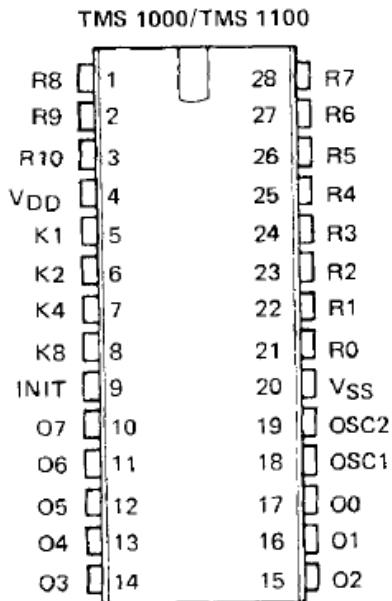


Abbildung 3.1: Skizze der Pins

Bezeichnung	Anzahl	Aufgabe
R-Output	11	Für die Ausgabe von Kontroll-Daten zuständig
O-Output	8	Gibt den Inhalt des Akkumulators und das Status-Logic-Bit aus
Oszillatoren	2	Zuständig für den Takt des Controllers
Power-Supply	2	Versorgt den Controller mit Spannung
K-Input	4	Eingang für 4-Bit zur Verwendung im Programm
INIT	1	Initialisiert oder setzt die Hardware zurück

### 3.3.3 Aufbau & Funktionsweise

Der folgende Abschnitt befasst sich mit dem Aufbau und der Funktionsweise der Hardware des Mikrocontrollers TMS-1000.

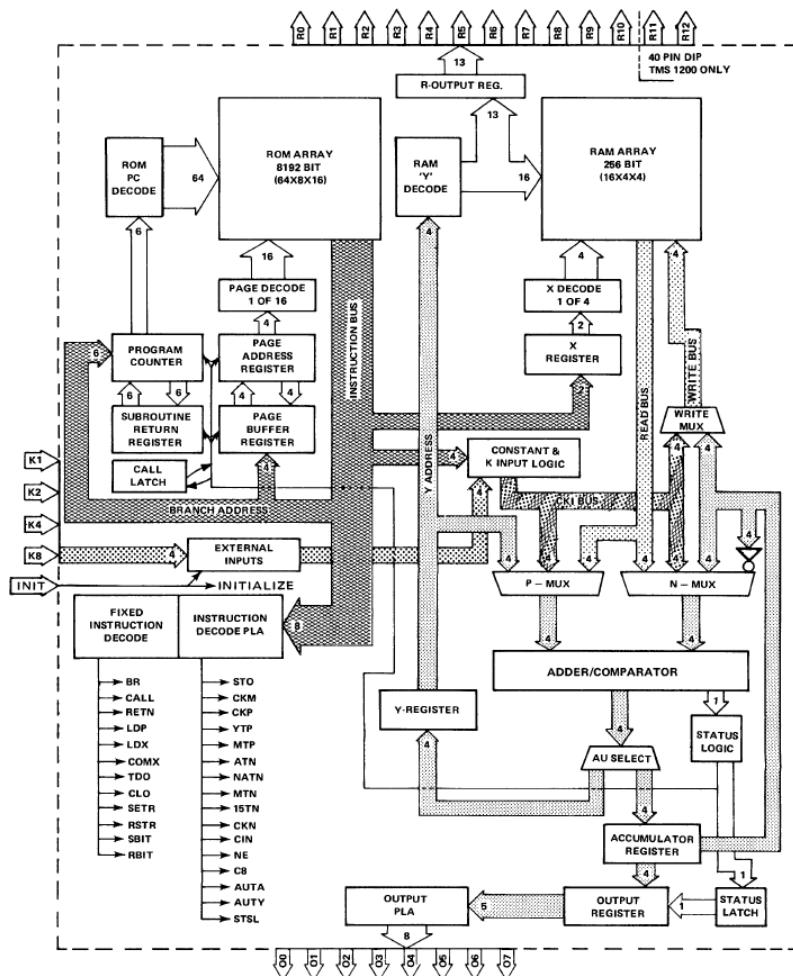


Abbildung 3.2: Schaltbild des TMS-1000

## ROM - Read Only Memory

Der Festspeicher besteht aus 16 Seiten, welche je 64 Wörter enthalten. Jedes Wort ist dabei 8 Bit groß. Es werden 4 verschiedene Register benutzt um den Speicher zu adressieren.

Das Page Adress Register(PA): Dieses Register enthält die aktuelle Seitenzahl, an dem sich das Programm zur Zeit befindet. Das Register ist 4-Bit groß, um alle Seiten von 0 bis 15 adressieren zu können.

Das Page Buffer Register(PB): Das PB-Register wird mit einer neuen Seitenadresse geladen und nach einer erfolgreichen Branch oder Subroutinen-Operation wird die Adresse in das PA-Register geladen. Aus diesem Grund entspricht die Größe des Registers der des PA-Registers

Das Program Counter Register(PC): Dieses Register enthält das aktuelle Wort der Seite, in dem sich das Programm zur Zeit befindet. Um alle 64 Wörter adressieren zu können enthält das Register 6 Bit.

Das Subroutine Return Register(SR): In dem SR-Register wird bei einer Call Operation das aktuelle Wort des Porgramms gespeichert, um bei einer Return Instruktion zum ursprünglichen Wort zurückkehren zu können.

## Branching & Subroutinen

Branching und Subroutinen sind ein essenzieller Teil der im Ablauf eines Mikrocontroller-Programms. Ein Branch ist mit einer herkömmlichen if-Abfrage aus einer beliebigen Programmiersprache gleichzusetzen. Der Branch wird erfolgreich ausgeführt, wenn das Status-Logic-Bit gesetzt ist. Das Bit ist zwar standardmäßig gleich 1, kann durch Rechenoperationen oder Vergleiche durch die Recheneinheit gleich 0 gesetzt werden. Nach einem Instruktionszyklus wird der Zustand wieder zurückgesetzt. Bei erfolgreicher Ausführung eines Branches wird das PB-Register in das PA-Register geladen.

Subroutinen sind sehr ähnlich zu Branch Instruktionen. Bei einer erfolgreichen Call-Instruktion springt das Programm an eine andere Adresse im ROM. Bei einer Return-Instruktion kehrt das Programm zu dem ursprünglichen Call-Statement zurück. Die Subroutine wird genau wie bei einem Branch nur ausgeführt, wenn das Status-Logic-Bit gesetzt ist. Zusätzlich muss jedoch das so genannte Call-Latch gesetzt sein. In diesem Fall wird der Inhalt des PA-Registers mit dem des PB-Registers

vertauscht. Die Adresse des PC-Registers wird in dem SR-Register zwischengespeichert. Bei der Return Instruktion werden die temporär gespeicherten Adressen wieder in das PA- und PC-Register geladen. Es ist nicht möglich eine Call Instruktion innerhalb einer Call Instruktion aufzurufen, ohne dass die korrekte Return-Adresse verloren geht.

### **RAM - Random Access Memory**

Der RAM-Speicher besteht aus 4 Dateien, welche jede 16 Wörter enthalten. Jedes Wort ist dabei 4 Bit groß. Der Speicher wird durch 2 Register X und Y adressiert. Das X-Register gibt dabei die Datei an, das Y-Register das Wort.

Der Input erfolgt durch den Write-Multiplexer. Sowohl das Akkumulator Register als auch die Constant and K Input Logic (CKI) können in den Speicher schreiben. Der Read-Bus übernimmt den Output des RAMs. Der Bus schreibt das Wort entweder in den P-Multiplexer oder den N-Multiplexer, welche beide in die Recheneinheit führen. Die Recheneinheit leitet die Daten entweder in das Y oder Akkumulator Register.

Dem Programmierer des Controllers steht es frei, wie er sich den Speicher einteilt. Typischerweise werden die ersten 7 Wörter jeder Seite für als Register verwendet. Der Rest wird durch beispielsweise Pointer, Event Counter oder Flags befüllt. Diese sollten sich wenn möglich in der gleichen Datei gespeichert sein, in der auch die Register liegen mit dem sie interagieren.

### **Constant & K Input Logic**

Die CKI Logik selektiert entweder die 4 K-Input Bit oder eine 4-Bit Konstante aus dem ROM und legt diese auf den CKI Daten Bus. Die Konstanten, welche aus dem ROM geliefert werden, werden für Befehle verwendet, welche Konstanten in ihrem Opcode enthalten. Zum Beispiel die Instruktion A8AAC. Diese Instruktion addiert die Zahl 8 zu dem Inhalt des Akkumulators und speichert das Ergebnis wiederum im Akkumulator. Der CKI Daten Bus kann die das 4-Bit Wort entweder in einen der beiden Adder leiten, die zur Recheneinheit führen, oder über den Write-Multiplexer in den RAM-Speicher.

### **Y-Register**

Das Y-Register adressiert zusammen mit dem X-Register den RAM-Speicher. Außerdem adressiert das Register das R-Output Register, welches die individuellen Latches setzt und

resettet. Das Y-Register wird zusätzlich als Arbeitsregister bezeichnet, da temporär ROM Wörter gespeichert werden können und sich das Register sehr gut als Counter eignet.

## R- und O-Output

Der TMS-1000 hat zwei verschiedene Arten an Output, R und O. Die R-Outputs werden für Kontroll Daten verwendet. Diese Kontroll Daten werden für externe Geräte verwendet oder Status-Logic-Outputs, wie zum Beispiel Overflow.

Das O-Output Register enthält den Inhalt des Akkumulators, sowie das Status-Logic-Bit. Durch ein PLA<sup>1</sup> werden die 5 Bit zu den 8 Output-Bit kodiert. Der Programmierer kann dabei selbst bestimmen wie er die Ein- und Ausgänge verbindet. Das O-Output Register wird generell für die Ausgabe von Daten benutzt im Gegensatz zum R-Output Register.

## Akkumulator & ALU - Aritmethic Logic Unit

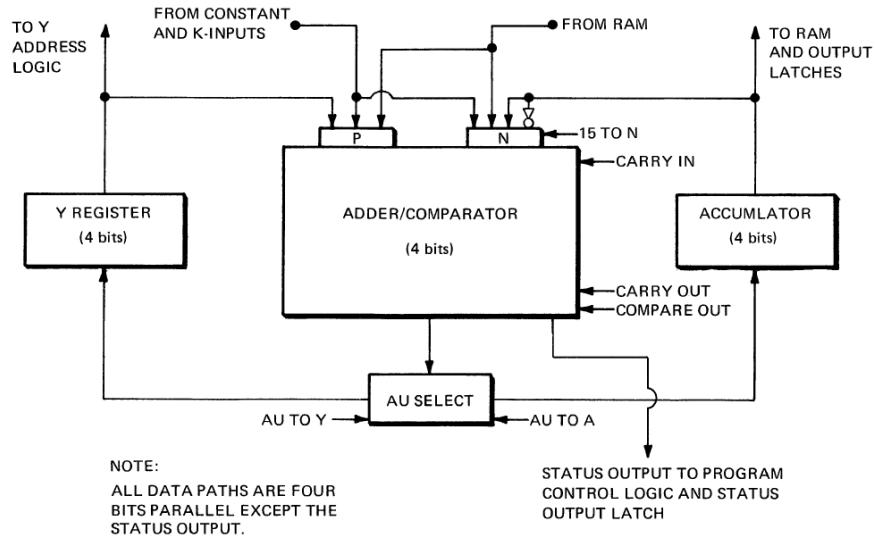
Das Akkumulator Register ist sehr wichtig, da es mit der ALU dem RAM-Speicher und dem O-Output Register interagiert. Hauptsächlich wird das Register für Additionen und Subtraktionen verwendet. Außerdem werden im Akkumulator Konstanten zwischengespeichert und alle Variablen, die in den RAM-Speicher geschrieben werden sollen, müssen den Akkumulator durchlaufen.

Die ALU ist ein 4-Bit Adder und Komparator. Er kann addieren, subtrahieren und logische Vergleiche durchführen. Seine Inputs bekommt die Einheit durch 2 4-Bit Multiplexer, die ihren Inhalt durch das Y-Register, die CKI, den RAM-Speicher oder den Akkumulator bekommen. Das Ergebnis wird je nach Instruktion entweder im Y-Register oder Akkumulator gespeichert. Die Subtraktion wird durch die two's complement arithmetic ausgeführt. Bei dieser Arithmetik wird der Subtrahend invertiert und auf den Minuend addiert.

Die ALU setzt außerdem das Status-Logic-Bit. Das Bit kann nicht nur bei logischen Vergleichen gesetzt werden, sondern kann auch bei arithmetischen Operationen als Carry gesetzt werden für beispielsweise Überträge.

---

<sup>1</sup>Programmable Logic Array



**Abbildung 3.3:** Eine Skizze der arithmetisch logischen Einheit

### Instruction-Decoders

Der TMS-1000 enthält zwei logische Blöcke, die jeweils die 8-Bit Instruktionen in die verschiedenen Mikroinstruktionen dekodiert. Der erste Dekodierer, der „Fixed instruction decoder“, kann nicht modifiziert werden und aktiviert 12 fixierte Mikroinstruktionen. Die restlichen 31 Basis Instruktionen sind die programmierbaren Instruktionen. Diese werden durch den zweiten Dekodierer, den „Programmable instruction PLA“, definiert. Standardmäßig sind diese für den Programmierer vordefiniert. Sie können aber auch durch das umprogrammieren des PLAs umdefiniert werden. Dabei stehen dem Dekodierer 16 Mikroinstruktionen zur Verfügung, die beliebig kombiniert werden können.

### 3.3.4 Befehlssatz

#### Befehlsformate

Die 43 Basisinstruktionen werden durch jewils eine der 4 Instruktionsformate beschrieben. Jedes Format enthält dabei 8-Bit, welche wie folgt aufgeteilt sind:

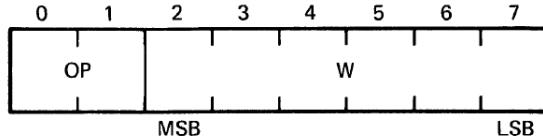


Abbildung 3.4: Instruktionsformat 1

Das erste Format besteht aus 2 Bit Op-Code und einem 6-Bit ROM-Wort. Dieses Format wird für Branch und Subroutinen-Control benutzt. Das ROM-Wort beschreibt die zu ladene Adresse.

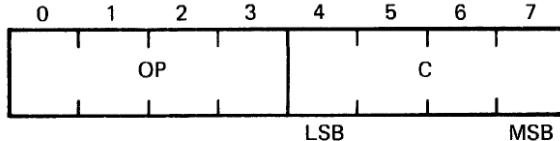


Abbildung 3.5: Instruktionsformat 2

Das zweite Format besteht aus 4-Bit Op-Code und einer 4-Bit Konstante. Es wird benutzt, um Konstanten in ein Register oder in den RAM-Speicher zu schreiben. Bei diesem Format ist das Most-significant-Bit und das Least-significant-Bit vertauscht.

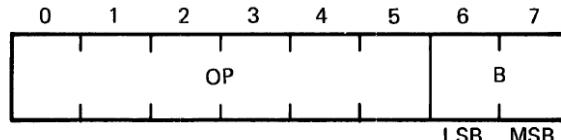
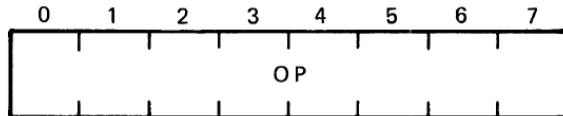


Abbildung 3.6: Instruktionsformat 3

Das dritte Format besteht aus 6-Bit Op-Code und einem 2-Bit großen Operanden. Der Operand adresiert ein bestimmtes Bit aus einem RAM-Wort, welches wiederum durch das X und Y Register adressiert wird.



**Abbildung 3.7:** Instruktionsformat 4

Das vierte und letzte Format besteht ausschließlich aus Op-Code. Dieses Format wird für Instruktionen benutzt, die immer dasselbe erledigen und deswegen keine Konstanten benötigen.

### Instruktionsbeispiele

Das erste Beispiel ist eine arithmetische Instruktion, AMAAC. Die Instruktion addiert ein Wort aus dem RAM-Speicher auf den Inhalt des Akkumulators. Das Ergebnis wird im Akkumulator gespeichert. Die Instruktion aktiviert 4 der programmierbaren Mikroinstruktionen. MTP lädt ein Wort aus dem Speicher in den P-Multiplexer, ATN den Akkumulator in den N-Multiplexer. C8 setzt das Status-Logic-Bit bei einem Übertrag. AUTA lässt das Ergebnis der Addition im Akkumulator speichern.

Das zweite Beispiel, ALEC, ist ein arithmetisches Vergleich. Die Instruktion, welche durch das zweite Format beschrieben wird, vergleicht den Inhalt des Akkumulators mit einer Konstante, die mit der Instruktion übergeben wird. Wenn der Akkumulator kleiner oder gleich groß wie die Konstante ist, wird das Status-Logic-Bit gesetzt. Auch bei dieser Instruktion, werden 4 der programmierbaren Mikroinstruktionen aktiviert. CKP lädt die Konstante in den P-Multiplexer, NATN den invertierten Inhalt des Akkumulators in den N-Multiplexer, da der Vergleich über eine Subtraktion erfolgt. CIN addiert 1 auf die Summe, um die Subtraktion zu vervollständigen. C8 setzt erneut das Status-Logic-Bit bei einem Übertrag.

## 3.4 Verwendung Mikrocontroller

Mikrocontroller spielen eine wichtige Rolle in vielen elektronischen Geräten. Sie finden Verwendung in eingebetteten Systemen und sind im Alltag oft unbemerkt. In Leistung und Ausstattung sind sie meist auf eine Anwendung angepasst, um maximale Effizienz gewährleisten zu können.

Mikrocontroller werden nicht nur in der Unterhaltungselektronik wie CD- und

DVD-Spielen verwendet, sondern auch für Gebrauchsgegenstände wie Waschmaschinen oder Kühlschränke. Die hohe Verwendung der Mikrocontroller lässt sich zum einen auf die extrem billigen Preise zurückführen, zum anderen auf die extrem hohe Zuverlässigkeit. Laut Texas Instruments fällt ein TMS-1000 im Durchschnitt nur ein einziges mal innerhalb von 240 Jahren aus.

Der TMS-1000 ist vor allem für seine Verwendung in den von Texas Instruments produzierten Taschenrechnern bekannt. Jedoch wurde der Controller auch unter anderem in Türklingeln und diversen Spielen verwendet. Dazu gehört zum Beispiel das elektronische Hand-Held „Speak & Spell“, welches 1978 von Texas Instruments auf den Markt gebracht wurde.



Abbildung 3.8: Das Hand-Held Speak & Spell

# Abbildungsverzeichnis

2.1	Mazor, Hoff und Shima . . . . .	5
2.2	Federico Faggin . . . . .	6
2.3	Intel 4004 mit Faggins Initialen . . . . .	7
2.4	Pins des Intel 4001 . . . . .	8
2.5	Layout des Intel 4001 . . . . .	9
2.6	Pins des Intel 4002 . . . . .	10
2.7	Layout des Intel 4002 . . . . .	11
2.8	Pins des Intel 4003 . . . . .	12
2.9	Layout des Intel 4003 . . . . .	12
2.10	Pins des Intel 4004 . . . . .	13
2.11	Layout des Intel 4004 . . . . .	14
2.12	Der Befehlszyklus des MCS-4 . . . . .	15
2.13	Befehle mit einem Wort Länge . . . . .	16
2.14	Befehle mit doppelter Wort Länge . . . . .	17
3.1	Skizze der Pins . . . . .	21
3.2	Schaltbild des TMS-1000 . . . . .	22
3.3	Eine Skizze der arithmetisch logischen Einheit . . . . .	26
3.4	Instruktionsformat 1 . . . . .	27
3.5	Instruktionsformat 2 . . . . .	27
3.6	Instruktionsformat 3 . . . . .	27
3.7	Instruktionsformat 4 . . . . .	28
3.8	Das Hand-Held Speak & Spell . . . . .	29

# Tabellenverzeichnis

2.1	RAM-Chipauswahl	10
2.2	RAM-Adressierung	10

# Literaturverzeichnis

- [1] R. N. Noyce and M. E. Hoff, “A history of microprocessor development at intel,” *IEEE Micro*, vol. 1, pp. 8–21, Feb 1981.
- [2] F. Faggin, M. E. Hoff, S. Mazor, and M. Shima, “The history of the 4004,” *IEEE Micro*, vol. 16, pp. 10–20, Dec 1996.
- [3] W. Aspray, “The intel 4004 microprocessor: what constituted invention?,” *IEEE Annals of the History of Computing*, vol. 19, pp. 4–15, Jul 1997.
- [4] A. Ruge, “Rekonstruktion und visualisierung von systemen mit intel 4004 prozessor,” 2003.