ELO- B-12 1332 MOPPEL
Übersicht und Angebot

212 Seiten

1 Grundlagen der Mikrocomputer-Technik

1.1.1 Begriffsbestimmungen: Mikroprozessor

- EIN MIKROPROZESSOR
  - IST EIN HOCHINTEGRIERTER LOGIKBAUSTEIN
  - DIENT ZUR AUSFÜHRUNG ARITHMETISCHER UND LOGISCHER OPERATIONEN MIT DATEN
  - IST ALLEIN NICHT ARBEITSFÄHIG
  - IST MEIST IM 40-POLIGEN DUAL-IN-LINE-GEHÄUSE UNTERGEBRACHT
  - IST KEIN RECHENBAUSTEIN

- ERFASSUNG UND VERARBEITUNG ALLER DATEN ERFOLGEN JEWEILS IN GRUPPEN PARALLELER BITS
  - DIE PARALLEL VERARBEITETEN DATENBITS NENNT MAN WORT
  - DIE WORTLÄNGE IN EINEM MIKROCOMPUTER-SYTEM LIEGT FEST

- EIN MIKROCOMPUTER
  - IST DER ARBEITSFÄHIGE AUFBAU UM EINEN MIKROPROZESSOR HERUM
  - BESITZT EINEN MIKROPROZESSOR ALS ZENTRALEINHEIT (CPU - CENTRAL PROCESSING UNIT)
  - IST GEKENNZEICHNET DURCH DIE DREITEILUNG
    + ZENTRALEINHEIT
    + SPEICHER
    + EIN/AUSGABE-BAUSTEINE
  - FÜHRT SEINE AUFGABEN GEMÄSS EINER FESTGELEGTEN FOLGE EINZELNER AKTIVITÄTEN DURCH
  - IST KEIN RECHNER

- EIN-CHIP-MIKROCOMPUTER VEREINEN DIE DREI FUNKTIONSGRUPPEN IN EINEM GEHÄUSE
  - IHRE KONFIGURATION IST IN DER REGEL NICHT ERWEITERBAR

TTL-Pegel

- TTL: TRANSISTOR/TRANSISTOR-LOGIK

  LOGIKFAMILIE EINZELNER KOMBINIERBARER BAUSTEINE
  MIT DEFINIERTER EIN/AUSGANGS-SCHNITTSTELLE

- LS-TTL: LOW-POWER-SCHOTTKY-TTL

  VERGLICHEN MIT STANDARD-TTL NUR 20...30% DER
  LEISTUNGSAUFNAHME BEI GLEICHER ARBEITSGESCHWINDIGKEIT

TTL-Pegel	Eingang		Ausgang	
	Standard	LS	Standard	LS
LOW	0,8 V -1,6 mA	0,8 V -360 µA	0,4 V 16 mA	0,5 V 8 mA
HIGH	2,0 V 40 µA	2,0 V 20 µA	2,4 V -400 µA	2,7 V -400 µA

- GATTERLAUFZEIT 10 ns

- TRI-STATE-LOGIK
  - Ø LOW
  - 1 HIGH
  - X HIGH IMPEDANCE
    DON'T CARE

Logische Grundfunktionen

- DIE LOGISCHEN GRUNDFUNKTIONEN VERKNÜPFEN ZWEI SIGNALE NACH LOGISCHEN REGELN

  - AND

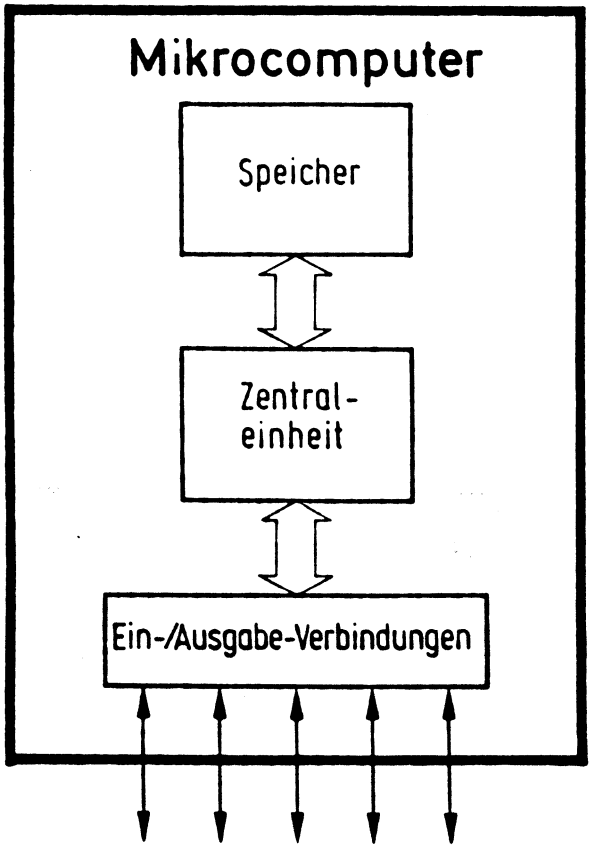
  - OR

  - EXOR

  - KOMPLEMENT

1.1.2 Mikrocomputer-Struktur

- KENNZEICHNEND DIE DREITEILUNG AUS
  - ZENTRALEINHEIT
  - SPEICHER
  - EIN/AUSGABE-VERBINDUNGEN



Aufgabenverteilung

- DIE ZENTRALEINHEIT ÜBERNIMMT DIE ZENTRALE VERWALTUNG UND ABLAUFSTEUERUNG SOWIE DIE EIGENTLICHE DATENVERARBEITUNG

- DER SPEICHER ÜBERNIMMT ZWEI AUFGABEN
  - DER PROGRAMMSPEICHER ENTHÄLT DEN ABLAUFPLAN, DER BEI SCHRITTWEISER ABARBEITUNG DEN COMPUTER IN DIE LAGE VERSETZT, EINE BESTIMMTE AUFGABE ZU ERFÜLLEN
  - DER DATENSPEICHER DIENT ZUR VORÜBERGEHENDEN AUFNAHME VERÄNDERLICHER INFORMATIONEN

- DER SPEICHER BESTEHT AUS EINER VIELZAHL EINZELNER SPEICHERSTELLEN
  - JEDE SPEICHERSTELLE HAT IHRE EIGENE HAUSNUMMER, DIE ADRESSE

- ÜBER DIE EIN/AUSGABE-VERBINDUNGEN WIRD DER INFORMATIONSAUSTAUSCH ZWISCHEN COMPUTER UND UMGEBENDEN SCHALTUNGEN ABGEWICKELT

- ZUR ANPASSUNG UND KOORDINATION DER EINZELNEN KOMPONENTEN SIND ERGÄNZENDE BAUSTEINE ERFORDERLICH

- ALLE GEGENSTÄNDLICH FASSBAREN BAUSTEINE BEZEICHNET MAN ALS HARDWARE

Mikrocomputer-Einsatz

- ZWEI EINSATZGEBIETE
  - STEUERUNG UMFANGREICHER FUNKTIONSABLÄUFE
    - o ZEITABLÄUFE NACH FESTEM SCHEMA
    - o PROZESS-STEUERUNGEN
  - EINLESEN, VERARBEITEN UND AUSGEBEN VON DATEN
    - o DATENVERKEHR MIT DER PERIPHERIE
    - o MODIFIKATION DER DATEN NACH ARITHMETISCHEN UND LOGISCHEN REGELN
    - o DATENABHÄNGIGE ENTSCHEIDUNGEN TREFFEN

- EINGREIFEN IN DIE ZU STEUERNDE PERIPHERIE
  - INFORMATIONEN UMSETZEN UND EINLESEN
  - VERARBEITEN DER DATEN MIT UNTERSCHIEDLICHEN REAKTIONEN IN ERSTER LINIE KEINE BERECHNUNGEN
  - STEUERSIGNALE/BEDIENERANWEISUNGEN AUSGEBEN

- DEN ANPASS-SCHALTUNGEN ZWISCHEN MIKROCOMPUTER UND PERIPHERIE KOMMT EINE ÜBERRAGENDE BEDEUTUNG ZU

Leistungsfähigkeit

- EIN MIKROCOMPUTER KANN JEDE NUR DENKBARE MANIPULATION MIT DATEN AUSFÜHREN
  - SOLANGE DIE AUFGABE EINDEUTIG DEFINIERBAR IST UND
  - SOLANGE AUSREICHEND ZEIT ZUR VERFÜGUNG STEHT

- DIE HOHE ARBEITSGESCHWINDIGKEIT IN VERBINDUNG MIT DEM GEORDNETEN ARBEITSABLAUF UND DER FÄHIGKEIT, ENTSCHEIDUNGEN ZU TREFFEN, MACHEN DIE EIGENTLICHE LEISTUNGSFÄHIGKEIT DES MIKROCOMPUTERS AUS
  - DIE ENTSCHEIDUNGEN SIND NICHT WILLKÜRLICH, SONDERN VORHER GENAU FESTGELEGT

- DIE GRENZEN DER LEISTUNGSFÄHIGKEIT WERDEN SCHNELL ERREICHT
  - EIN GROSSTEIL DER VERFÜGBAREN ZEIT GEHT MIT DEM DATENTRANSPORT VERLOREN
  - DIE EINZELNEN BEFEHLE SIND NUR WINZIGE MOSAIKSTEINCHEN EINER DURCHZUFÜHRENDEN OPERATION

- BEFEHLSAUSFÜHRUNGSZEITEN VON STANDARD-MIKROPROZESSOREN: $\sim 1\ \mu s$
  - TAKTFREQUENZ 5...10MAL SCHNELLER

Alternativen

- AUSWEGE BEI ÜBERLASTUNG
  - GRÖSSERE WORTLÄNGE
  - ERWEITERTER BEFEHLSSATZ
  - KUNDENSPEZIFISCHE SCHALTUNG

- GRÖSSERE WORTLÄNGE
  - BEI GLEICHER ARBEITSGESCHWINDIGKEIT DRASTISCH ERHÖHTER DATENDURCHSATZ

- KUNDENSPEZIFISCHE SCHALTUNG
  - AUFTRAGSENTWICKLUNG EINER LSI-SCHALTUNG NACH KUNDENWÜNSCHEN
  - DURCH ANGEPASSTE STRUKTUR OPTIMALE LÖSUNG
  - MINDESTSTÜCKZAHLEN HOCH
  - LIEFERZEITEN LANG

Bit-Slice-Prozessoren

- DIE FUNKTIONEN DER ZENTRALEINHEIT SIND NICHT BLOCKWEISE IN PARALLELEN BITS, SONDERN SCHEIBCHENWEISE ANGEORDNET
  - DIE SLICES SIND 2 ODER 4 BIT BREIT
  - DIE SLICES LASSEN SICH KASKADIEREN, UM BELIEBIGE WORTLÄNGEN AUFZUBAUEN

- BIT SLICES ERFORDERN (ERMÖGLICHEN) DIE ERWEITERUNG (ERSTELLUNG) DES BEFEHLSSATZES DURCH DEN ANWENDER
  - MIKROPROGRAMMIERBARKEIT

- BIT-SLICES SIND AUSSCHLIESSLICH IN BIPOLARER TECHNOLOGIE HERGESTELLT
  - GEGENÜBER MOS 10...100MAL SCHNELLER
  - GEGENÜBER MOS 100...1000MAL HÖHERE VERLUSTLEISTUNG

- DRASTISCH ERHÖHTER DATENDURCHSATZ
  - BELIEBIGE WORTLÄNGE
  - ANGEPASSTER BEFEHLSSATZ
  - SCHNELLERE LOGIK

- GERINGE ENTWICKLUNGSUNTERSTÜTZUNG

1.1.3 Entwicklungsgeschichte

- PIONIER UND WEGBEREITER WAR DIE FIRMA INTEL

1971: ERSTER MIKROPROZESSOR-VORLÄUFER 4004
RESULTIEREND AUS KUNDENSPEZIFISCHEM AUFTRAG

1972: NACHFOLGETYP 4040 (AUFWÄRTSKOMPATIBEL)

1971: ERSTER 8-BIT-MIKROPROZESSOR 8008

1973: VORSTELLUNG DES 8080 (AUFWÄRTSKOMPATIBEL)
WELTWEITER INDUSTRIESTANDARD

1975: TECHNOLOGISCHE VERBESSERUNG 8080A

1976: HARDWARE-OPTIMIERUNG 8085A
KONKURRENZ-FABRIKAT: Z-80 (AUFWÄRTSKOMPATIBEL)

1976: 16-BIT-PROZESSOR 8086 (≈ 8085)

1978: 16-BIT-PROZESSOR Z-8000 (≈ Z-80)

1979: 16-BIT-ALTERNATIVE 8088 (≈8085 / 8086)

- AUFWÄRTSKOMPATIBILITÄT
- BEISPIELLOSE INNOVATIONSRATE
- INTEL-MARKTVORSPRUNG

1.1.4 Markteinteilung: Wortlänge

- UNTERSCHEIDUNGSKRITERIEN
  - WORTLÄNGE
  - TECHNOLOGIE

- 4-BIT-TYPEN: BILLIGST-ANWENDUNGEN
  - ROCKWELL-FAMILIE PPS-4
  - TEXAS TMS 1000
  - INTEL 4004/4040

- 8-BIT-TYPEN: STANDARDFORMAT
  - INTEL 8080/8085
  - MOTOROLA 6800
  - FAIRCHILD F 8
  - SIGNETICS 2650
  - RCA 1802 * *CMOS-Typ
  - ZILOG Z-80

- 12-BIT-TYP: ANWENDUNGSTECHNISCH BEDEUTUNGSLOS
  - INTERSIL IM6100

- 16-BIT-TYPEN: OPTIMALER DATENDURCHSATZ
  - TEXAS TMS 9900 sehr gut
  - INTEL 8086
  - ITT CP1600

- BIT SLICES
  - AMD 2900
  - INTEL 3000
  - TEXAS SBP 0400

MOS-Technologie

- MOS
  - FELDEFFEKTTRANSISTOREN
  - GERINGE LEISTUNGSAUFNAHME 1‰ Bipol. (TTL)
  - HOHE AUSBEUTE, BILLIG
  - LANGSAM
  - EMPFINDLICH GEGEN STATISCHE AUFLADUNG

- P-KANAL MOS
  - HOHE AUSBEUTE, BILLIGST
  - EXOTISCHE VERSORGUNGSSPANNUNGEN
  - DYNAMISCHE LOGIK
  - BEFEHLSAUSFÜHRUNGSZEITEN 4...20 µs
  - BEISPIEL: INTEL 4004

P-Kanal MOS ≙ Zuerst N-kanal MOS

- N-KANAL MOS
  - STANDARD-TECHNOLOGIE
  - EINE VERSORGUNGSSPANNUNG MÖGLICH
  - DYNAMISCHE LOGIK
  - BEFEHLSAUSFÜHRUNGSZEITEN 1...5 µs
  - BEISPIEL: INTEL 8085

aufwendiger Takt Generator

- CMOS
  - NIEDRIGSTE LEISTUNGSAUFNAHME 10^{-6} · TTL
  - HÖCHSTE STÖRSICHERHEIT
  - STATISCHE ARBEITSWEISE
  - BEFEHLSAUSFÜHRUNGSZEITEN 1...10 µs
  - BEISPIEL: RCA 1802

Bipolare Technologie

- BIPOLAR
  - BIPOLARE TRANSISTOREN
  - SCHNELL
  - HOHE LEISTUNGSAUFNAHME
  - GENERELL BIT-SLICES
  - ROBUST
- TTL (LS-TTL)
  - GROSSE CHIPFLÄCHE
  - BEFEHLSAUSFÜHRUNGSZEITEN 100...300 ns
  - BEISPIEL: AMD 2900 (SCHOTTKY-TTL)
- ECL
  - UNGESÄTTIGTE LOGIK mit Diode Emitter-Basis realisiert
  - GERINGER STÖRABSTAND
  - SEHR HOHE LEISTUNGSAUFNAHME
  - BEFEHLSAUSFÜHRUNGSZEITEN ~ 30 ns
  - BEISPIEL: MOTOROLA 10800
- I^2L Sättigungsfrei
  - HÖCHSTE STÖRSICHERHEIT
  - SEHR GERINGE LEISTUNGSAUFNAHME
  - BILLIGER HERSTELLUNGSPROZESS
  - GERINGE CHIPFLÄCHE
  - BEFEHLSAUSFÜHRUNGSZEITEN 100...300 ns
  - BEISPIEL: TEXAS SBP 0400

8080A-Zentraleinheit

- DIE ZENTRALEINHEIT BESTEHT AUS DREI CHIPS
  - ≈ 1,3W – CPU 8080A (40PIN)
  - ≈ 0,7W – CONTROLLER 8228 (24PIN)
  - ≈ 1W – TAKTOSZILLATOR 8224 (16PIN)

  } untrennbar zusammengehörig

- DREI VERSORGUNGSSPANNUNGEN
  - +5 V BEI 375 mA
  - +12 V BEI 95 mA
  - -5 V BEI 1 mA

- BEFEHLSAUSFÜHRUNGSZEIT 2 µs
  - SPEICHERZUGRIFF 570 ns

- TAKTOSZILLATOR UND SYSTEMSTEUERBAUSTEIN KÖNNEN AUCH „DISKRET" AUFGEBAUT WERDEN

8085A-Zentraleinheit

- DIE ZENTRALEINHEIT BESTEHT AUS EINEM CHIP
  - TAKTOSZILLATOR INTEGRIERT
  - INTERNE ERZEUGUNG ALLER STEUERSIGNALE

- EINE VERSORGUNGSSPANNUNG
  - +5 V BEI 170 mA
  - INTERNE ERZEUGUNG DER NEGATIVEN SPANNUNG

- BEFEHLSAUSFÜHRUNGSZEIT 1,3 µs
  - SPEICHERZUGRIFF ~ 570 ns

Speicherbausteine müssen $t_{Zugriff} \leq 570$ ns haben

- IN MISCHSYSTEMEN IST ZUSÄTZLICH EIN 8-BIT-ADRESS-LATCH ERFORDERLICH

Verschiedene Hersteller

Aufgaben des Mikrocomputers

- EINGANGSLEITUNG AUF KONTINUITÄT ÜBERWACHEN
- ZEITMARKEN AUF MINDEST- UND HÖCHSTDAUER ÜBERPRÜFEN
- SYNCHRONISATIONSZEITPUNKT ERMITTELN
- PARALLELUMSETZUNG DER SERIELLEN EINGANGSINFORMATION
- AUSWERTUNG VON PRÜFBITS
- PLAUSIBILITÄTSÜBERPRÜFUNG DER EMPFANGENEN INFORMATION
- AUFBEREITUNG DER INFORMATION FÜR DIE ANZEIGE

- KONTINUIERLICHES HOCHZÄHLEN DER UHR IM SEKUNDENTAKT
  - ÜBERBRÜCKEN VON EMPFANGSSTÖRUNGEN
  - MINUTENÜBERTRAG
  - STUNDENÜBERTRAG
  - TAGESÜBERTRAG
  - WOCHENTAGSÜBERTRAG
  - MONATSÜBERTRAG
  - BERÜCKSICHTIGUNG VON SCHALTJAHREN

- ABFRAGE DER EINGESPEICHERTEN WECKZEITEN
  - AKUSTISCHER/ELEKTRISCHER ALARM
  - EINMALIGE/WIEDERHOLENDE AUSLÖSUNG

- ABFRAGE DES BEDIENFELDES
  - ALARMEINSTELLUNG
  - ZEITZONENWAHL
  - STOPPUHR

- UMRECHNUNG IN ANDERE ZEITZONEN

- STOPPUHR AUFWÄRTS/ABWÄRTS ZÄHLEN

1.2 Grundelemente der Software

- ENTSPRECHEND EINER FESTGELEGTEN FOLGE VON BEFEHLEN FÜHRT EIN COMPUTER OPERATIONEN MIT DATEN AUS
- DIE BEFEHLE ENTSTAMMEN EINEM STANDARD-BEFEHLSSATZ, DER VOM HALBLEITERHERSTELLER FESTGELEGT WORDEN IST
  - SIE WERDEN IM NORMALFALL SEQUENTIELL ABGEARBEITET
- UNTER OPERATION VERSTEHT MAN
  - VERKNÜPFUNG DER DATEN NACH LOGISCHEN REGELN
  - VERKNÜPFUNG DER DATEN NACH ARITHMETISCHEN REGELN
  - DATENTRANSPORT
- DIE FESTGELEGTE REIHENFOLGE VON BEFEHLEN NENNT MAN PROGRAMM
- FÜR JEDE, AUCH DIE GERINGSTE TÄTIGKEIT BRAUCHT DER COMPUTER EIN EIGENES PROGRAMM
- ALLE BEFEHLE UND DATEN BESITZEN DIE FESTE WORTLÄNGE, DIE DURCH DIE MIKROCOMPUTER-STRUKTUR VORGEGEBEN IST
  - REICHT DIESE WORTLÄNGE NICHT AUS, WERDEN MEHRERE WORTE ANEINANDERGEREIHT
- DIE ZUSAMMENFASSUNG ALLER PROGRAMME (DER GEGENSTÄNDLICH NICHT FASSBAREN BESTANDTEILE) BEZEICHNET MAN ALS SOFTWARE

1.2.1 Begriffsbestimmungen

- IM COMPUTER SELBST KÖNNEN DATEN UND BEFEHLE NUR IN FORM VON LOGIKPEGELN EXISTIEREN
- EINER SOLCHEN FOLGE VON LOW- UND HIGH-ZUSTÄNDEN KANN MAN NICHT ANSEHEN, WAS SICH DAHINTER VERBIRGT
  - DAS KANN EIN BEFEHL SEIN
  - DAS KANN EINE ZAHL SEIN
  - DAS KANN EINE STEUERINFORMATION SEIN
- UM DIE UNHANDLICH LANGEN UND UNÜBERSICHTLICHEN BITFOLGEN EINFACHER ERFASSEN ZU KÖNNEN, FÜHRT MAN EINE ABKÜRZUNG EIN
- BEI DER HEXADEZIMALEN ABKÜRZUNG VON BITFOLGEN TEILT MAN DIESE IN VIERERGRUPPEN EIN
- JEDE VIERERGRUPPE BESITZT 16 VERSCHIEDENE KOMBINATIONS-MÖGLICHKEITEN
- JEDE DIESER KOMBINATIONSMÖGLICHKEITEN KÜRZT MAN MIT EINEM EINSTELLIGEN HEX-DIGIT AB
  - FÜR DIE ERSTEN ZEHN KOMBINATIONSMÖGLICHKEITEN ØØØØ...1ØØ1 VERWENDET MAN DIE ZAHLZEICHEN Ø...9
- FÜR DIE RESTLICHEN SECHS KOMBINATIONSMÖGLICHKEITEN 1Ø1Ø..1111 VERWENDET MAN AUSHILFSWEISE DIE BUCHSTABEN A...F

Hexadezimalsystem

= Sedezimal

- BITMUSTER IN VIERERGRUPPEN EINTEILEN

- JEDE VIERERGRUPPE DURCH EIN HEX-DIGIT ABKÜRZEN

Bitmuster	HEX
Ø Ø Ø Ø	Ø
Ø Ø Ø 1	1
Ø Ø 1 Ø	2
Ø Ø 1 1	3
Ø 1 Ø Ø	4
Ø 1 Ø 1	5
Ø 1 1 Ø	6
Ø 1 1 1	7
1 Ø Ø Ø	8
1 Ø Ø 1	9
1 Ø 1 Ø	A
1 Ø 1 1	B
1 1 Ø Ø	C
1 1 Ø 1	D
1 1 1 Ø	E
1 1 1 1	F

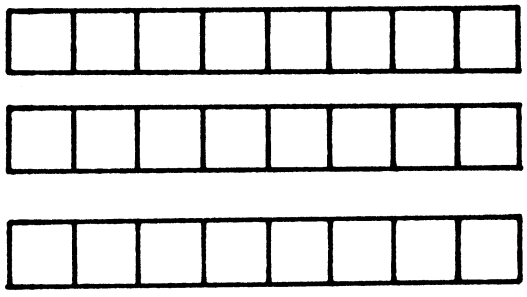
- HINTER JEDEM HEX-DIGIT VERBIRGT SICH EINE GRUPPE VON 4 BITS
- UNTER DER HEXADEZIMALZAHL ____ IST FOLGENDES BITMUSTER ZU VERSTEHEN:

- ALS KENNZEICHEN FÜR „HEXADEZIMAL" STELLT MAN DER ZAHL EIN „H" NACH: 3FH
  - ENTSPRECHENDES GILT FÜR DEZIMAL: 38D
  - ENTSPRECHENDES GILT FÜR BINÄR: 11Ø1B

Logische Grundfunktionen

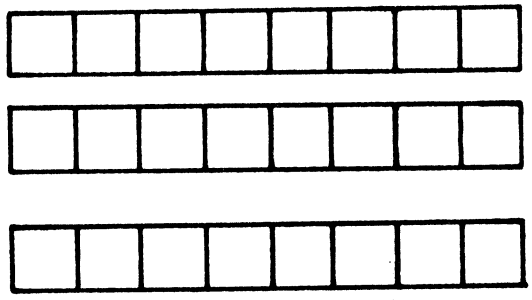
- DIE LOGISCHEN GRUNDFUNKTIONEN KANN MAN AUCH AUF DATENWORTE ANWENDEN
  - ES WERDEN JEWEILS DIE KORRESPONDIERENDEN BITS MITEINANDER VERKNÜPFT

- AND



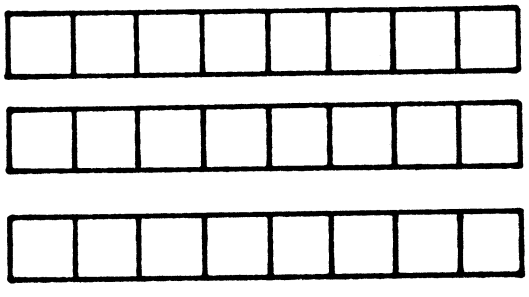
dient zum Löschen von Bits

- OR



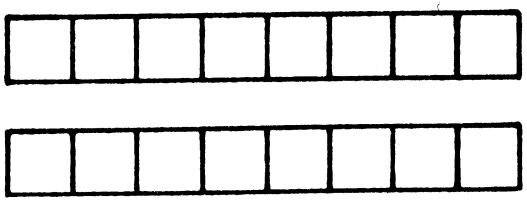
dient zum Setzen

- EXOR



dient Invertierung
XRA A dient zum Löschen von A

- KOMPLEMENT



Programmiersprachen

- DER COMPUTER SELBST IST NUR IN DER LAGE, LOGIKPEGEL ZU VERARBEITEN
  - DIESE FORM DER INFORMATIONSDARSTELLUNG NENNT MAN MASCHINENSPRACHE
  - DIE HEXADEZIMALE SCHREIBWEISE IST EINE FÜR DEN MENSCHEN LEICHTER ERFASSBARE FORM DER MASCHINENSPRACHE
- MNEMOTECHNISCHE ABKÜRZUNGEN BILDEN EINE ESELSBRÜCKE FÜR DEN PROGRAMMIERER
  - DIESE FORM DER INFORMATIONSDARSTELLUNG NENNT MAN ASSEMBLERSPRACHE
  - DIE ASSEMBLER-ABKÜRZUNG IST DIE SINNBILDLICHE DARSTELLUNG FÜR COMPUTER-BEFEHLE
  - EIN IM ASSEMBLER-CODE VORLIEGENDES PROGRAMM MUSS IN DIE MASCHINENSPRACHE UMGESETZT WERDEN, EHE ES DER COMPUTER VERSTEHT
  - FÜR DIESE UMSETZUNG GIBT ES FERTIGE PROGRAMME, DIE DIE BEZEICHNUNG „ASSEMBLER" TRAGEN
- HÖHERE PROGRAMMIERSPRACHEN VERWENDEN EINE AUSDRUCKSFORM, DIE DER MENSCHLICHEN SPRACHE NOCH NÄHERKOMMMT ALS DIE ASSEMBLER-SPRACHE

Beispiele

- MASCHINENSPRACHE UND HEXADEZIMALE ABKÜRZUNG

  Ø 1 Ø Ø 1 1 Ø 1 = 4 D

- ASSEMBLERSPRACHE

  CMA COMPLEMENT ACCUMULATOR

  BILDE DAS KOMPLEMENT DES AKKUMULATOR-INHALTS

- HÖHERE PROGRAMMIERSPRACHEN

  - FORTRAN

    ```
    2Ø  IF ALPHA=37 (9Ø,12Ø,21)
    ```

  - BASIC

    ```
    6ØØ  IF INSTR(F$,"*") THEN PRINT "STERN"
    62Ø  A=INSTR(F$,"*")
    ```

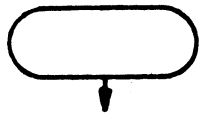
  -

1.2.2 Struktur des Befehlssatzes

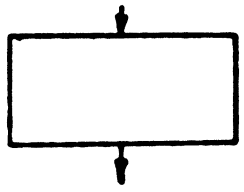
- EIN COMPUTER KANN
  - DATEN TRANSPORTIEREN
  - DATEN MANIPULIEREN
  - DATENABHÄNGIGE ENTSCHEIDUNGEN TREFFEN
- DEMENTSPRECHEND IST DER BEFEHLSSATZ AUFGEBAUT
- DATENTRANSPORTBEFEHLE
  - DATEN WERDEN VON EINER ZUR ANDEREN STELLE BEWEGT, OHNE DABEI MODIFIZIERT ZU WERDEN
  - DAS KANN EIN TRANSPORT ZWISCHEN CPU UND SPEICHER SEIN
  - DAS KANN DIE EIN- UND AUSGABE VON DATEN SEIN
- ARITHMETISCH/LOGISCHE BEFEHLE
  - DATEN WERDEN NACH ARITHMETISCHEN ODER LOGISCHEN REGELN MITEINANDER VERKNÜPFT
  - DAS KANN DER VERGLEICH ZWISCHEN ZWEI DATENWORTEN SEIN
  - DAS KANN DIE SUBTRAKTION ZWEIER DATENWORTE SEIN
- SPRUNGBEFEHLE
  - JE NACH EINTREFFEN ODER AUSBLEIBEN EINES BESTIMMTEN ERGEBNISSES VERZWEIGT DIE PROGRAMMAUSFÜHRUNG ZUR ANGEGEBENEN STELLE (BEDINGTE SPRUNGBEFEHLE)
  - DIE ENTSCHEIDUNG SETZT EINE ABFRAGE VORAUS
  - IM PROGRAMM WIRD EINE ABFRAGE ALS BEDINGTER SPRUNG FORMULIERT: SPRINGE BEIM EINTREFFEN EINES BESTIMMTEN ERGEBNISSES ZUR ANGEGEBENEN ADRESSE
  - IM ANDEREN FALL GEHT DIE PROGRAMMAUSFÜHRUNG SEQUENTIELL WEITER
  - UNBEDINGTE SPRUNGBEFEHLE WERDEN IN JEDEM FALL AUSGEFÜHRT, SIND ALSO NICHT AN EINE BESTIMMTE BEDINGUNG GEKNÜPFT

1.2.3 Programmstruktur

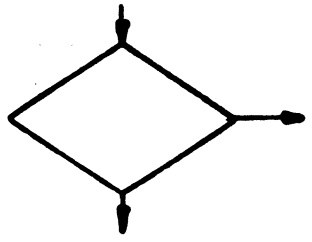
- ENTSPRECHEND DEM BLOCKSCHALTBILD IN DER ELEKTRONIK STELLT DAS FLUSSDIAGRAMM DIE GROBSTRUKTUR EINES PROGRAMMS DAR
- EINIGE SYMBOLE SIND ALLGEMEIN ÜBLICH
- PROGRAMMANFANG UND -ENDE



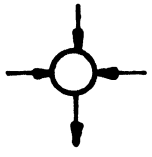
- JEDE AKTIVITÄT



- ABFRAGEN (BEDINGTE SPRÜNGE)



- SPRUNGZIELE



- VOR DER REALISIERUNG AUCH DES KLEINSTEN PROGRAMMS IST EIN FLUSSDIAGRAMM ZU ENTWERFEN
- ZUR PROGRAMMIERUNG GEHÖRT UNTRENNBAR DIE DOKUMENTATION

Programmbeispiel AMPEL

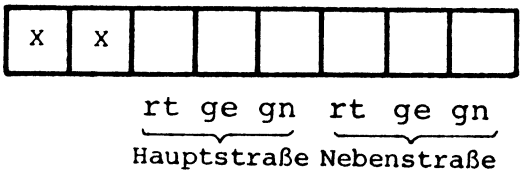
- DURCHLAUFEN VERSCHIEDENER AMPELPHASEN IN FORM EINER ENDLOSSCHLEIFE

- ZWISCHEN DEN EINZELNEN PHASEN VORGEGEBENE WARTEZEITEN EINHALTEN

- AKTIVIEREN DER LICHTZEICHEN ÜBER EINEN AUSGABEKANAL UND LEISTUNGSTREIBER

- RANDBEDINGUNGEN

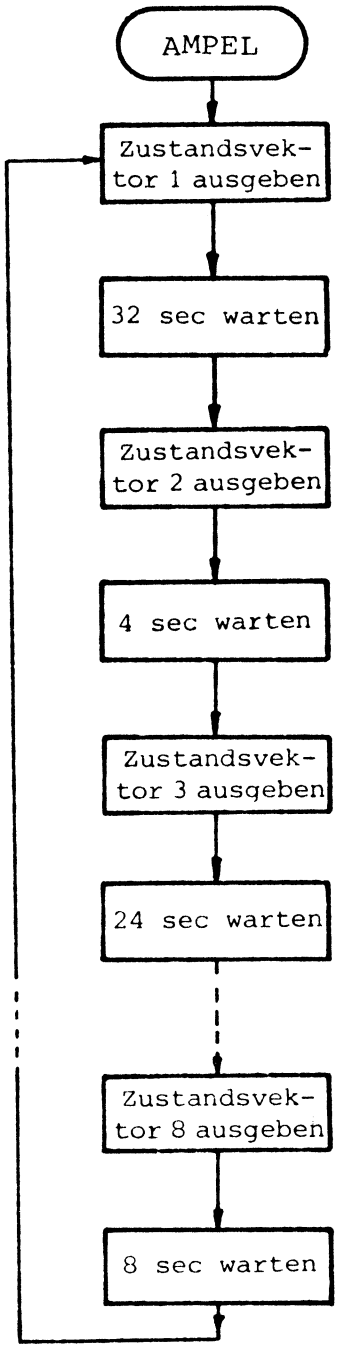
  - MIKROCOMPUTER-AUSGABEKANAL



  - ABLAUFSCHEMA

	Hauptstr.			Nebenstr.		
	rt	ge	gn	rt	ge	gn
Phase 1			•	•		
2		•		•		
3						
4						
5						
6						
7						
8						

Flußdiagramm AMPEL



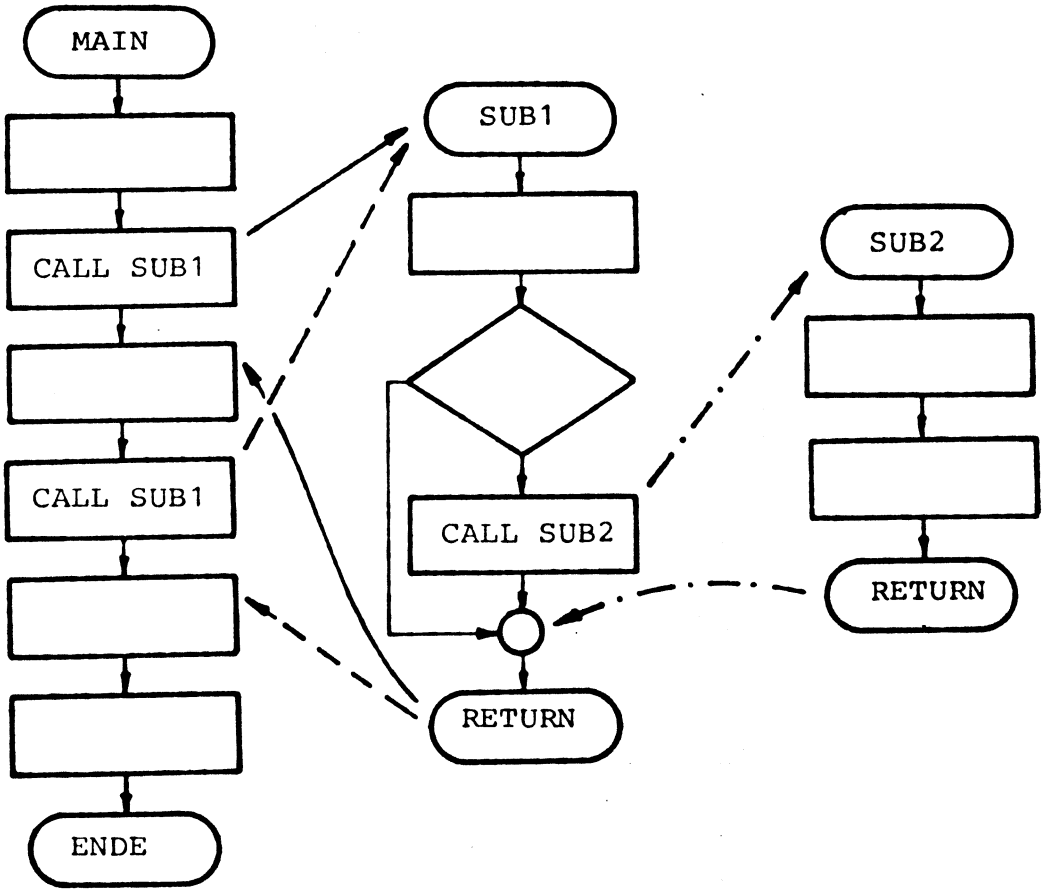
= 8×4 sec ⇒ Unterprogr.

= 6×4sec

1.2.4 Unterprogrammtechnik

- IM FLUSSDIAGRAMM AMPEL TAUCHT DIE AKTIVITÄT „WARTEN" MEHRFACH AUF
- MAN LÖST DIESEN TEIL ALS UNTERPROGRAMM HERAUS
  - UNTERPROGRAMM ONSEC LÄUFT GENAU EINE SEKUNDE LANG
  - ONSEC WIRD JE NACH ANFORDERUNG MEHRFACH AKTIVIERT
- DER COMPUTER GELANGT ÜBER EINEN SPEZIELLEN SPRUNGBEFEHL INS UNTERPROGRAMM
  - CALL SUBROUTINE IST DER UNTERPROGRAMM-AUFRUF
- AM ENDE DES UNTERPROGRAMMS STEHT IMMER DER RÜCKSPRUNG-BEFEHL INS HAUPTPROGRAMM
  - RETURN IST DER RÜCKSPRUNG AUS DEM UNTERPROGRAMM
- BEIM CALL-BEFEHL MERKT SICH DER COMPUTER AUTOMATISCH DIE STELLE DES HAUPTPROGRAMMS, DIE ER GERADE VERLÄSST
  - BEIM RETURN-BEFEHL SPRINGT ER DORTHIN ZURÜCK
- WO IMMER MÖGLICH, SIND UNTERPROGRAMME HERAUSZULÖSEN
  - EINSPARUNG VON SPEICHERPLATZ
  - ÜBERSICHTLICHKEIT DURCH MODULARE PROGRAMMSTRUKTUR
- UNTERPROGRAMME SOLLEN VIELSEITIG VERWENDBAR SEIN

Unterprogrammaufruf



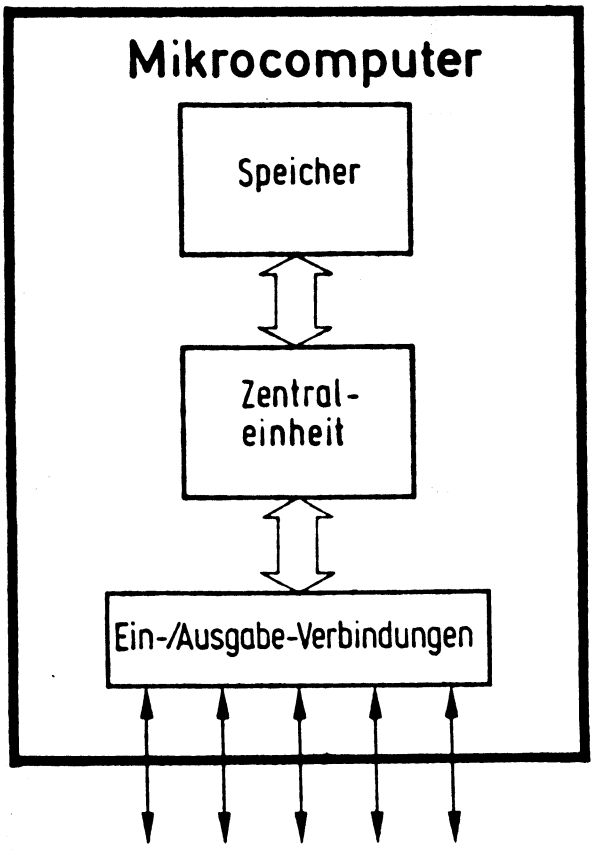
nested subroutines
bei 8080/8085 beliebig tief

- EIN UNTERPROGRAMM KANN EIN WEITERES UNTERPROGRAMM AUFRUFEN
  - VERSCHACHTELUNG VON UNTERPROGRAMMEN
  - BEIM 8080/8085 KANN DIE VERSCHACHTELUNG BELIEBIG TIEF SEIN
- DAS ZIEL FÜR DEN JEWEILIGEN RÜCKSPRUNG LEGT DIE CPU AUTOMATISCH IN EINEM DAFÜR VORGESEHENEN TEIL DES SPEICHERS AB

⇒ Strukturierte Programmierung

1.3 Grundbestandteile

- GRUNDBESTANDTEILE EINES MIKROCOMPUTER-SYSTEMS

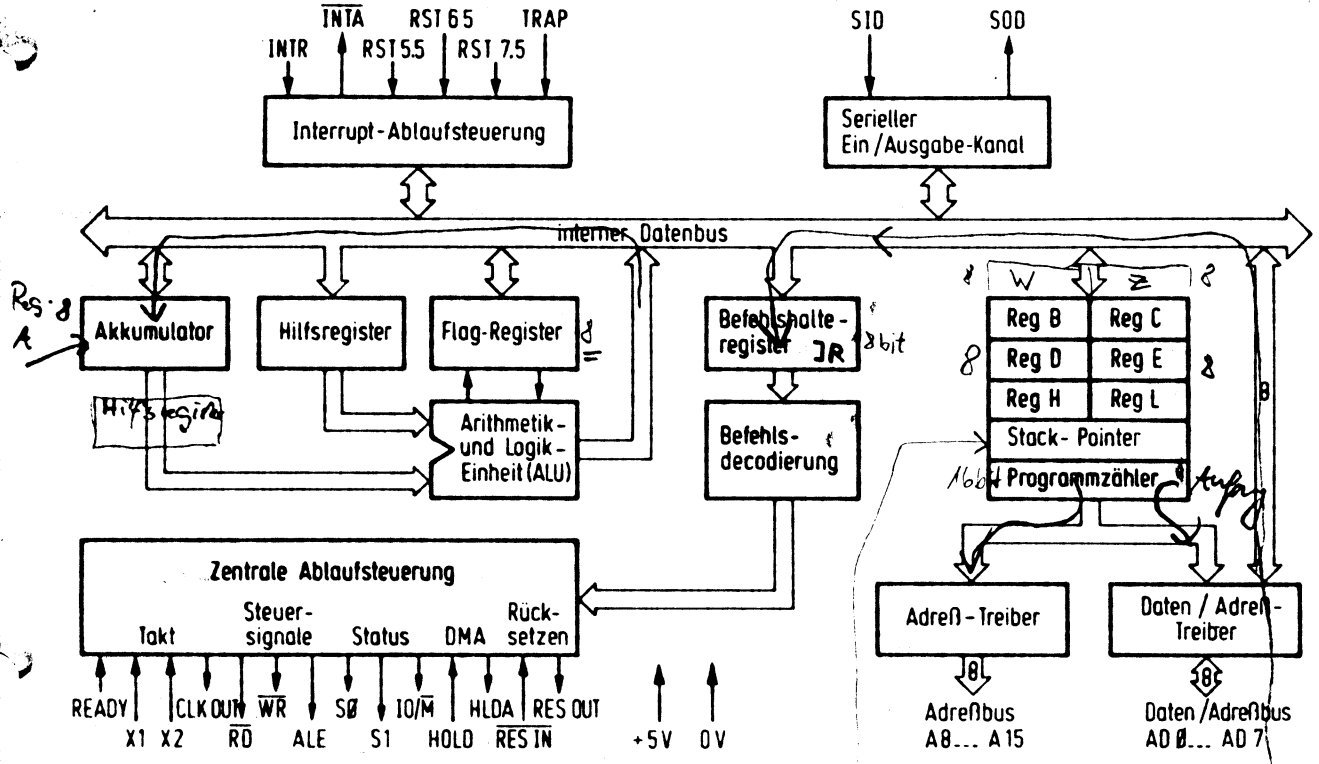


Busstruktur

- ALLE INFORMATIONEN, EGAL OB DATEN ODER BEFEHLE, HABEN IM 8080/8085-SYSTEM DIE FESTE WORTLÄNGE VON 8 BIT
  - REICHT DIESE WORTLÄNGE NICHT AUS, WERDEN MEHRERE WORTE ANEINANDERGEREIHT
- JEDES EINZELNE 8-BIT-WORT BELEGT EINE EIGENE SPEICHERSTELLE
- JEDE SPEICHERSTELLE, EGAL, WAS SIE ENTHÄLT, HAT IHRE EIGENE HAUSNUMMER, DIE ADRESSE
  - ADRESSEINGANG AM SPEICHER
  - INFORMATIONSAUSGANG AM SPEICHER
- DIE ADRESSE IST NICHTS WEITER ALS EINE BINÄRZAHL
  - IM 8080/8085-SYSTEM HAT SIE DIE FESTE LÄNGE VON 16 BIT
- DAS ANLEGEN EINER ADRESSE FÜHRT ZUM AUSLESEN DER ADRESSIERTEN SPEICHERSTELLE
  - INSOFERN SIND SPEICHERSTELLE UND ZUGEHÖRIGE ADRESSE UNTRENNBAR MITEINANDER VERKNÜPFT
  - ANSONSTEN BESTEHT KEINERLEI ZUSAMMENHANG ZWISCHEN ADRESSE UND INHALT EINER SPEICHERSTELLE
- ADRESSEN UND DATEN FÜHREN AUF GETRENNTEN SAMMELLEITUNGEN AN ALLE KOMPONENTEN DES MIKROCOMPUTER-SYSTEMS
- DER ADRESSBUS IST UNIDIREKTIONAL
  - NUR DIE CPU KANN ADRESSEN AUSSENDEN
- DER DATENBUS IST BIDIREKTIONAL
  - DIE CPU KANN DATEN AUSSENDEN („SCHREIBEN")
  - DIE CPU KANN DATEN EMPFANGEN („LESEN")
- ZWISCHENZEITLICH SCHALTET DIE CPU DIE BUSLEITUNGEN IN DEN HOCHOHMIGEN ZUSTAND

1.3.1 Zentraleinheit 8085A

1 Chip 40 pin



keine Treiber erforderlich

verhaltet Rücksprungadressen

DAA

Treiber

DAA - Zusatzlogik, die Dezimalkorrektur ausführt
IR - Instruction Register

beim Befehl ausgeführt wird, wird Programmzähler +1

Interne Struktur

- ZENTRALE ABLAUFSTEUERUNG
  - STEUERUNG ALLER ZEITABLÄUFE
  - KOORDINATION DER SYSTEMKOMPONENTEN
  - VERANLASST DEN SICH ENDLOS WIEDERHOLENDEN ABLAUF EINES BEFEHLSZYKLUS'
- BEFEHLSDECODIERUNG MIT BEFEHLSHALTEREGISTER IR
  - AUS DEM BEFEHLSWORT IM BEFEHLSHALTEREGISTER WIRD DIE ERFORDERLICHE SEQUENZ VON STEUERSIGNALEN ERZEUGT, DIE DIE ZENTRALE ABLAUFSTEUERUNG ZUR AUSFÜHRUNG DES BEFEHLS VERANLASST
- 16-BIT-PROGRAMMZÄHLER PC

  HÄLT DIE ADRESSE DES NÄCHSTEN AUSZUFÜHRENDEN BEFEHLS
- BEFEHLSZYKLUS
  - HOLEN DES BEFEHLS

    DER PROGRAMMZÄHLERSTAND WIRD AUF DEN ADRESSBUS GESCHALTET DER SPEICHER GIBT DEN INHALT DER ADRESSIERTEN SPEICHERSTELLE AN DEN DATENBUS AUS, UND DER AUSGELESENE BEFEHL GELANGT INS BEFEHLSHALTEREGISTER
  - DECODIEREN DES BEFEHLS

    DIE BEFEHLSDECODIERUNG ENTSCHLÜSSELT DEN BEFEHL
    DIE ENTSPRECHENDEN STEUERSIGNALE GELANGEN AN DIE ZENTRALE ABLAUFSTEUERUNG; DER PROGRAMMZÄHLER WIRD AUTOMATISCH UM EINS ERHÖHT
  - AUSFÜHREN DES BEFEHLS

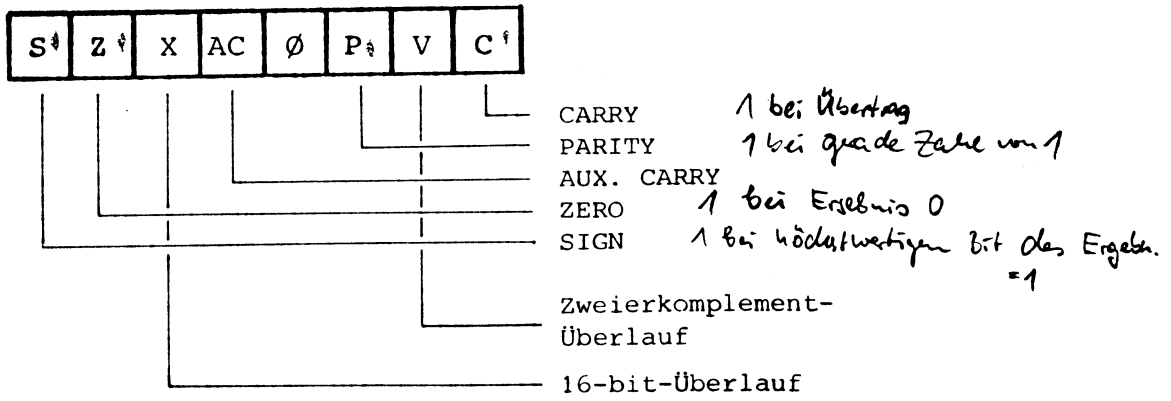
    DIE ZENTRALE ABLAUFSTEUERUNG GIBT AN DIE SYSTEMKOMPONENTEN DIE ERFORDERLICHEN STEUERSIGNALE AUS, UM DEN BEFEHL AUSZUFÜHREN

noch: Interne Struktur

- VIELZWECKREGISTER A...L
  - 16-BIT-SPEICHERSTELLEN MIT DIREKTEM ZUGRIFF
  - REG A (AKKUMULATOR)
  - REG B...REG L
- REGISTERPAARE
  - (B,C), (D,E), (H,L) KÖNNEN AUCH ALS 16-BIT-REGISTER-PAAR FUNGIEREN
- STACK POINTER SP
  16-BIT-ADRESSREGISTER FÜR DEN STACK-SPEICHERBEREICH
- ARITHMETIK- UND LOGIK-EINHEIT ALU
  - RECHENWERK DER CPU
  - EIN OPERAND STEHT IMMER IM AKKU
  - DAS ERGEBNIS GELANGT WIEDER IN DEN AKKU
  - HILFREGISTER DIENEN ZUR DATEN-ZWISCHENSPEICHERUNG
- DEZIMALKORREKTUR
  HARDWARE-ZUSATZLOGIK, DIE DEN BINÄREN AKKUMULATOR-INHALT IN ZWEI BCD-DIGITS UMSETZT die HEX anzeigen.
- NICHT EINGEZEICHNET IST DAS REGISTERPAAR W UND Z
  - FÜR DEN PROGRAMMIERER NICHT ZUGÄNGLICH
  - SETZT 16-BIT-ADRESSEN AUS ZWEI DATENWORTEN ZUSAMMEN

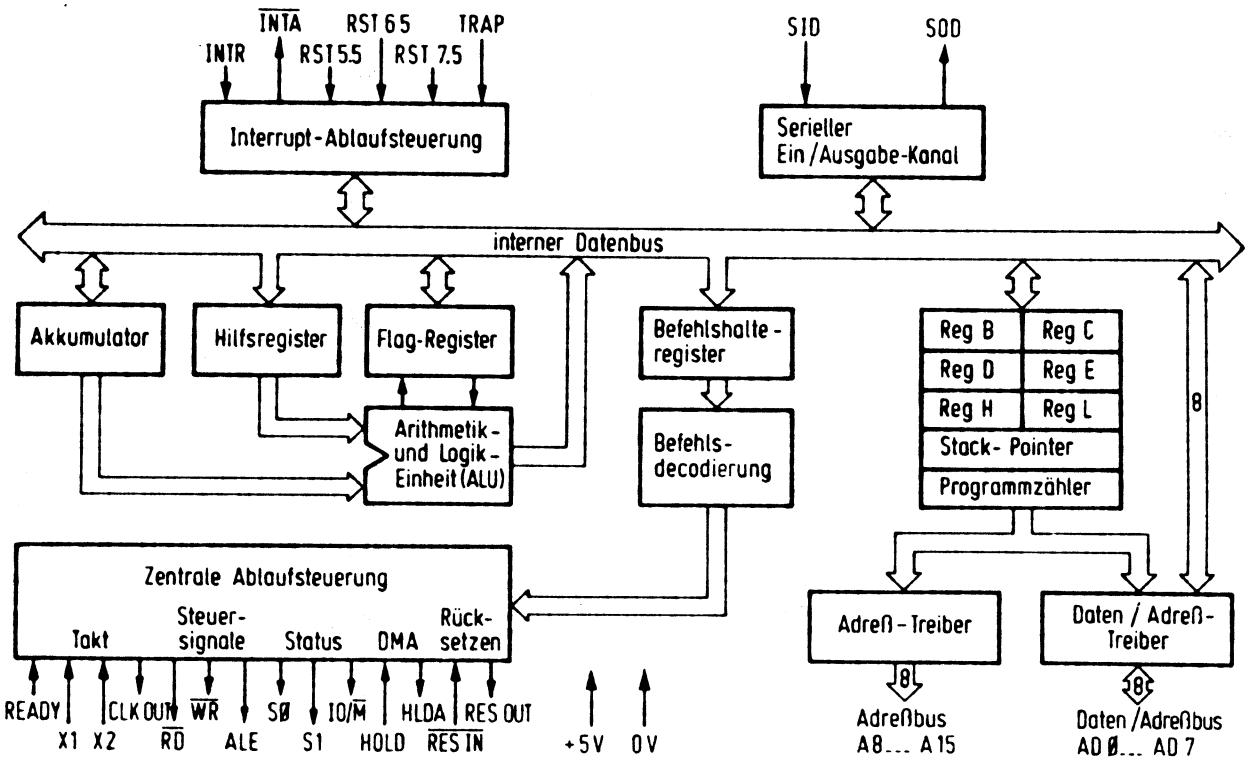
FLAG-Register

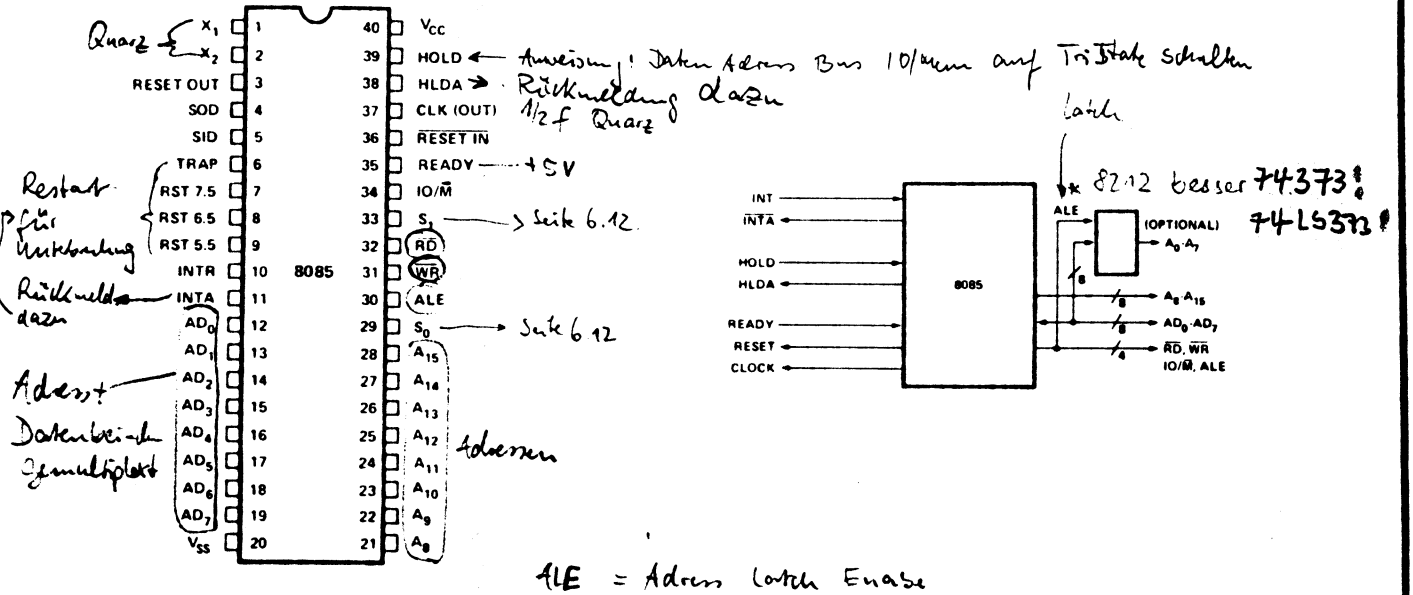
- FLAG-REGISTER — werden nach arithm. logischen Operation gesetzt
  - ERGEBNISANZEIGE NACH EINER ARITHMETISCH/LOGISCHEN OPERATION



- PROGRAMMABFRAGEN (BEDINGTE SPRÜNGE) STÜTZEN SICH AUSSCHLIESSLICH AUF DIE VIER ZUSTANDSBITS C, P, Z UND S
  - SPRINGE, WENN DAS ANGEGEBENE FLAG =1 IST
  - SPRINGE, WENN DAS ANGEGEBENE FLAG ≠1 IST
- AKKUMULATOR-INHALT UND INHALT DES FLAG-REGISTERS BILDEN ZUSAMMEN DAS PROZESSOR-STATUS-WORT PSW

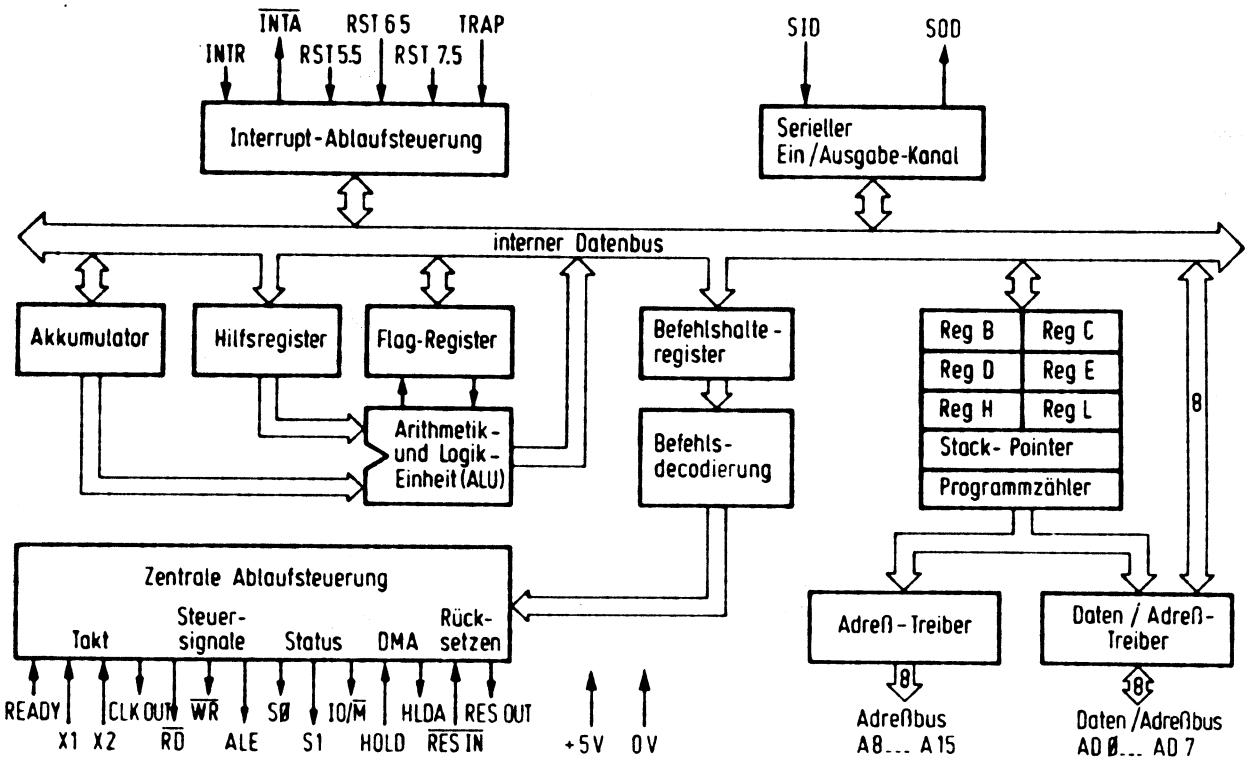
Anschlußbelegung 8085A

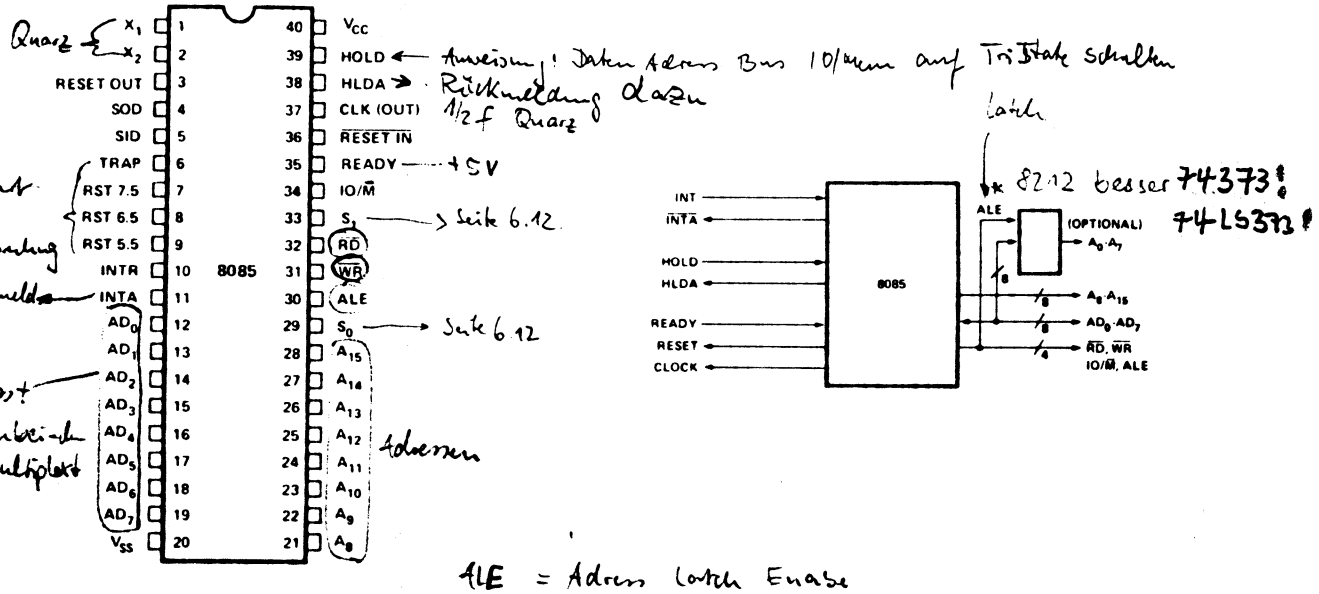




IO/M : Umschaltung ob Adressleitungen für IO (Peripherie) oder M Memory

Anschlußbelegung 8085A





ALE = Adress Latch Enable
(Umschaltung Adressen / Daten)

IO/M : Umschaltung ob Adressleitungen für IO (Peripherie) oder M Memory

- ADRESSBUS: DIE OBEREN ACHT ADRESSBITS AN A15...A8

  DIE UNTEREN ACHT ADRESSBITS AN AD7...ADØ
  STABIL BEI DER NEGATIVEN FLANKE VON ALE
  ZUSÄTZLICHES 8-BIT-LATCH ERFORDERLICH

- DATENBUS: DIE ACHT DATENBITS AN AD7...ADØ
  STABIL BEIM $\overline{RD}$- BZW. $\overline{WR}$-PULS

- STEUER-SIGNALE:

  ALE ADDRESS LATCH ENABLE
  DER MULTIPLEX-AD-BUS FÜHRT DIE UNTEREN ACHT ADRESSBITS

  IO/$\overline{M}$ I/O-/MEMORY-ADDRESS
  UNTERSCHEIDUNG ZWISCHEN SPEICHER- UND PORTADRESSE

  $\overline{RD}$ READ-PULS
  DATEN AUF DEM AD-BUS STABIL ZUM EINLESEN IN DIE CPU VOM SPEICHER ODER DEN E/A-BAUSTEINEN

  $\overline{WR}$ WRITE-PULS
  DATEN AUF DEM AD-BUS STABIL ZUM EINSCHREIBEN IN DEN SPEICHER ODER DIE E/A-BAUSTEINE

  SØ,1 DATA BUS STATUS
  ZUSTANDSINFORMATION ÜBER DEN AD-BUS

  READY BEREITMELDUNG AN DIE CPU, DASS DIE ADRESSIERTEN DATEN BEREITSTEHEN (STABIL SIND)

noch: Anschlußbelegung

- CPU ANHALTEN
  - HOLD — AUFFORDERUNG AN DIE CPU, DATEN- UND ADRESSBUS SOWIE $\overline{RD}$-, $\overline{WR}$- UND IO/$\overline{M}$-LEITUNG IN DEN HOCHOHMIGEN ZUSTAND ZU SCHALTEN
  - HLDA — HOLD ACKNOWLEDGE
    RÜCKMELDUNG DER CPU ÜBER DEN HOLD-ZUSTAND

- CPU UNTERBRECHEN
  - TRAP — NICHTMASKIERBARER INTERRUPT-EINGANG DER HÖCHSTEN PRIORITÄT
  - RST X — RESTART-INTERRUPTS
    MASKIERBARE INTERRUPT-EINGÄNGE MIT FESTER PRIORITÄTSSTRUKTUR
  - INTR — INTERRUPT REQUEST
    ALLGEMEINER INTERRUPT-EINGANG DER NIEDRIGSTEN PRIORITÄT
  - $\overline{INTA}$ — RÜCKMELDUNG DER CPU ÜBER DAS AKZEPTIEREN EINES INTERRUPTS

- HOLD HÄLT DIE CPU VORÜBERGEHEND AN (WÄHREND DES LAUFENDEN BEFEHLS) , UM EINER EXTERNEN STELLE DEN ZUGRIFF AUF DEN DATEN- UND ADRESSBUS ZU ERMÖGLICHEN

- BEIM INTERRUPT ERFOLGT (NACH BEENDIGUNG DES LAUFENDEN BEFEHLS) EIN SPRUNG IN EIN SPEZIELLES UNTERPROGRAMM, UM DIE UNTERBRECHENDE STELLE ZU BEDIENEN (INTERRUPT-SERVICE-ROUTINE)

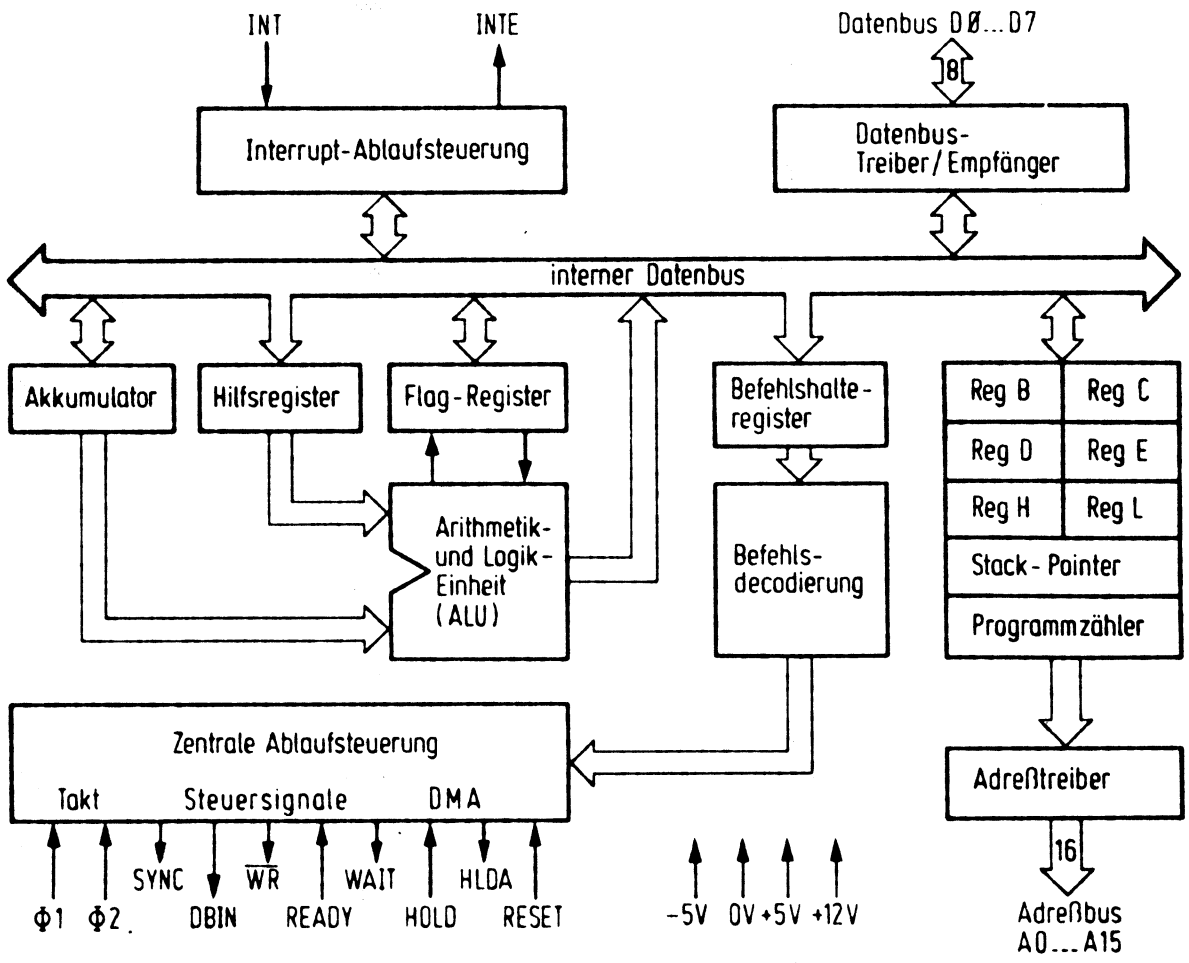
- TAKT
  - X1,X2 ANSCHLÜSSE FÜR QUARZ- ODER RC-BESCHALTUNG
  - CLK SYSTEMTAKT (TTL-PEGEL)

- EIN/AUSGANG
  - SID SERIAL INPUT DATA LINE
    SERIELLER DATENEINGANG (TTL-KOMPATIBEL)
  - SOD SERIAL OUTPUT DATA LINE
    SERIELLER DATENAUSGANG (TTL-PEGEL)

- INITIALISIEREN
  - $\overline{\text{RESIN}}$ RÜCKSETZEN DER CPU nach Einschalten vorgeschrieben
  - RESOUT RÜCKMELDUNG DER CPU ÜBER DEN RESET-ZUSTAND

- VERSORGUNG
  - V_{CC} = + 5 V
  - V_{SS} = 0 V

1.3.2 Zentraleinheit 8080A



- ZWEI ZUSÄTZLICHE BAUSTEINE ERFORDERLICH
  - EXTERNER TAKTOSZILLATOR 8224
  - SYSTEM-STEUERBAUSTEIN 8228
- DREI VERSORGUNGSSPANNUNGEN
- EIGENER 16-BIT-ADRESSBUS
- SEPARATER 8-BIT-DATENBUS
  - MULTIPLEX-BETRIEB MIT STATUS-INFORMATION
- UNTERSCHIEDE ZUM 8085
  - NUR EIN INTERRUPT-EINGANG
  - KEINE SERIELLER EIN/AUSGANG
  - 50% LANGSAMER
  - MODIFIZIERTES FLAG-VERHALTEN

1.3.3 Speicher

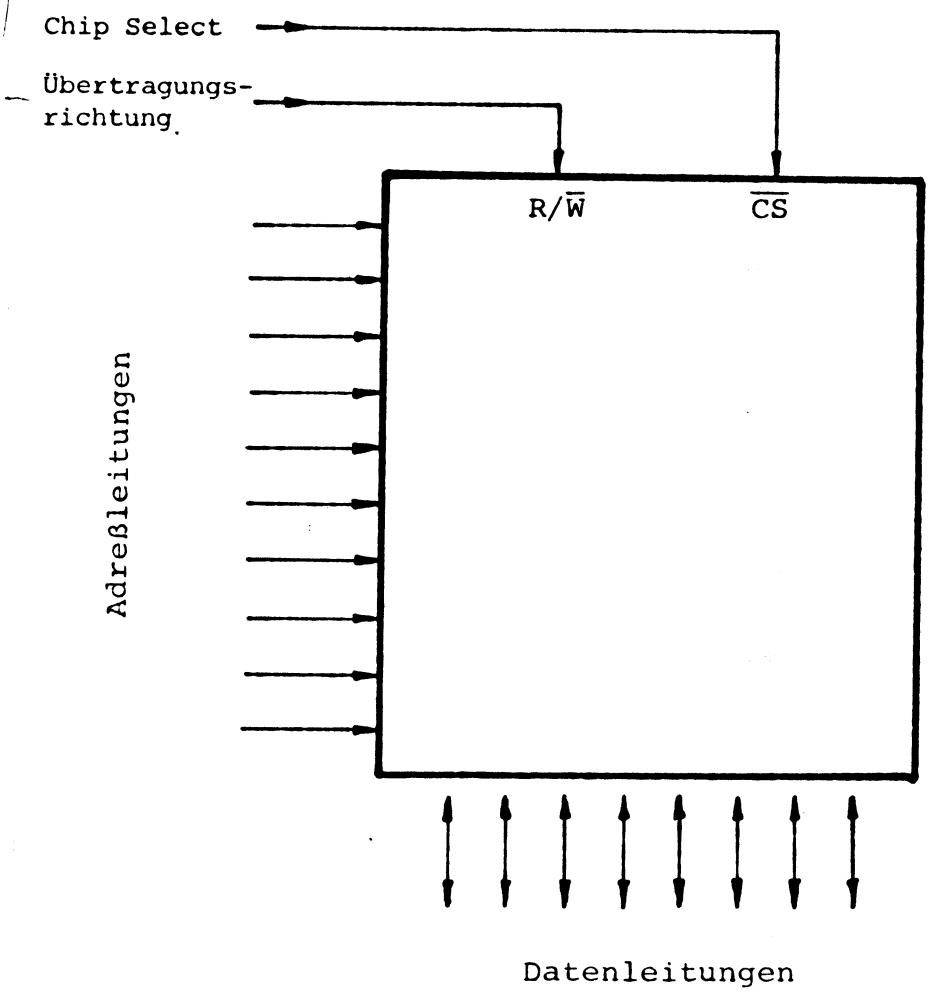
- JEDES EINZELNE 8-BIT-WORT BELEGT EINE EIGENE SPEICHERSTELLE
- JEDE SPEICHERSTELLE HAT IHRE EIGENE HAUSNUMMER, DIE ADRESSE
- DAS ANLEGEN EINER ADRESSE FÜHRT ZUM AUSLESEN DER ADRESSIERTEN SPEICHERSTELLE
  - ZUSAMMEN MIT DER ADRESSE GELANGT EIN LESESIGNAL AN DEN SPEICHER
- AUCH DAS EINSCHREIBEN IN EINE SPEICHERSTELLE IST MÖGLICH
  - ZUSAMMEN MIT DER ADRESSE GELANGT EIN SCHREIBSIGNAL AN DEN SPEICHER
- DIE ADRESSEINGÄNGE EINES JEDEN SPEICHERBAUSTEINS SIND AN DEN SYSTEM-ADRESSBUS ANGESCHLOSSEN
- DIE DATENEIN- UND -AUSGÄNGE EINES JEDEN SPEICHERBAUSTEINS
  - SIND AN DEN SYSTEM-DATENBUS ANGESCHLOSSEN
  - BEFINDEN SICH IM RUHEZUSTAND IM HOCHOHMIGEN ZUSTAND
- EIN SPEICHERBAUSTEIN WIRD ERST DANN ANGESPROCHEN, WENN SEIN CS-EINGANG AKTIVIERT WIRD
  - EINE ZUSÄTZLICHE HARDWARE-DECODIERUNG ERZEUGT AUS DER ADRESSE VERSCHIEDENE CS-SIGNALE

- DIE ZUGRIFFSZEIT IST NEBEN DER SPEICHERDICHTE DAS BESTIMMENDE MERKMAL FÜR EINEN SPEICHER
  - t_{acc}: ZEITSPANNE ZWISCHEN ANLEGEN DER ADRESSE UND BEREITSTEHEN DER DATEN (DATEN STABIL)
  - LANGSAME SPEICHER: t_{acc} = 1 µs
  - STANDARD-MOS-SPEICHER: t_{acc} = 450 ns
  - BIPOLARE ECL-SPEICHER: t_{acc} = 7...20 ns

- DER SPEICHER IST IN ALLER REGEL DER MIT ABSTAND TEUERSTE TEIL DES COMPUTERS

Speicheranwahl

Gatter erforderlich, da im 8080/8085 2 Leitungen /7400



Adreßbits

- DIE ANZAHL DER ADRESSLEITUNGEN RICHTET SICH NACH DER ZAHL DER INTERNEN SPEICHERSTELLEN
  - 256 SPEICHERSTELLEN ERFORDERN 8 ADRESSLEITUNGEN
  - DAMIT SIND DIE ADRESSEN VON ØØØØØØØØ...11111111 DARSTELLBAR (DEZIMAL Ø...255)
- EIN INTERNER DECODIERER WÄHLT IMMER NUR DIEJENIGE SPEICHERSTELLE AN, DEREN ADRESSE GERADE ANLIEGT
- DER DATENTRANSPORT IN DEN (AUS DEM) SPEICHER KANN NUR DANN ERFOLGEN, WENN DER CS-EINGANG AKTIVIERT IST

ROM

- INFORMATIONEN, DIE SICH WÄHREND DER PROGRAMMAUSFÜHRUNG NICHT ÄNDERN, LEGT MAN IN SOGENANNTEN NUR-LESE-SPEICHERN AB
  - READ ONLY MEMORY ROM
  - NIMMT PROGRAMME AUF
  - NIMMT KONSTANTEN AUF

- DER ROM-INHALT KANN VOM COMPUTER NUR AUSGELESEN WERDEN
  - EIN ROM HAT KEINEN WRITE-EINGANG

- DAS EINSCHREIBEN IN EIN ROM GESCHIEHT EINMALIG MIT EINEM SPEZIELLEN PROGRAMMIERGERÄT
  - DANACH BLEIBT DER ROM-INHALT PERMANENT ERHALTEN (AUCH NACH ABSCHALTEN DER VERSORGUNGSSPANNUNG)
  - DAS ROM IST EIN NICHTFLÜCHTIGER SPEICHER

Ausführungsformen

- MASKEN-ROMS

  DER INHALT WIRD NACH KUNDENWUNSCH VOM **HALBLEITERHERSTELLER** BEIM LETZTEN MASKIERUNGSVORGANG EINGEBRACHT
  - HOHE MASKENKOSTEN
  - MINDESTBESTELLMENGE ≈ 1000
  - LANGE LIEFERZEITEN
  - KEINE ÄNDERUNGSMÖGLICHKEITEN
  - SEHR PREISWERT IN STÜCKZAHLEN

- BIPOLARE PROMS (PROGRAMMABLE ROMS)

  DER ANWENDER ZERSTÖRT MIT EINEM SPEZIELLEN PROGRAMMIERGERÄT WINZIGE HALBLEITERSCHICHTEN, UM DIE INFORMATION EINZUBRINGEN
  - GERINGE PACKUNGSDICHTE
  - STROMINTENSIV
  - SEHR KURZE ZUGRIFFSZEITEN
  - KAUM ÄNDERUNGSMÖGLICHKEITEN

- EPROMS (ERASABLE PROMS)

  DER ANWENDER PROGRAMMIERT MIT EINEM SPEZIELLEN PROGRAMMIERGERÄT DEN INHALT; DURCH INTENSIVE **UV-BESTRAHLUNG** ERFOLGT LÖSCHEN DER EINGESPEICHERTEN INFORMATION
  - JEDERZEIT WIEDERVERWENDBAR
  - PREIS/LEISTUNGS-OPTIMUM
  - LANGE LÖSCHZEITEN ≈ 15 - 60 min
  - UMSTÄNDLICHES LÖSCHEN
  - PROBLEMLOSE ÄNDERUNGSMÖGLICHKEIT

  ! Licht gefährlich

- EAROMS (ELECTRICALLY ALTERABLE PROMS)

  DER INHALT IST ELEKTRISCH LÖSCH- UND NEUPROGRAMMIERBAR
  - LANGE LÖSCHZEITEN
  - TEUER
  - ANFÄLLIG
  - LANGSAM

? leicht anfällig

Standard-Programmspeicher

EPROMS

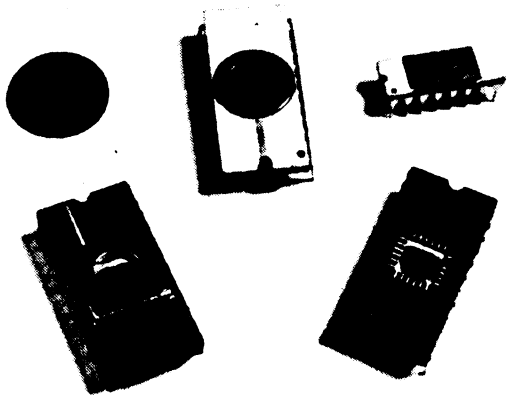
1974 **2708:** 1024 WORTE MIT JE 8 BIT ≈15-20DM
3 VERSORGUNGSSPANNUNGEN (±5 V, +12 V)

1976 **2758:** WIE 2708, ABER EINFACHE +5-V-VERSORGUNG

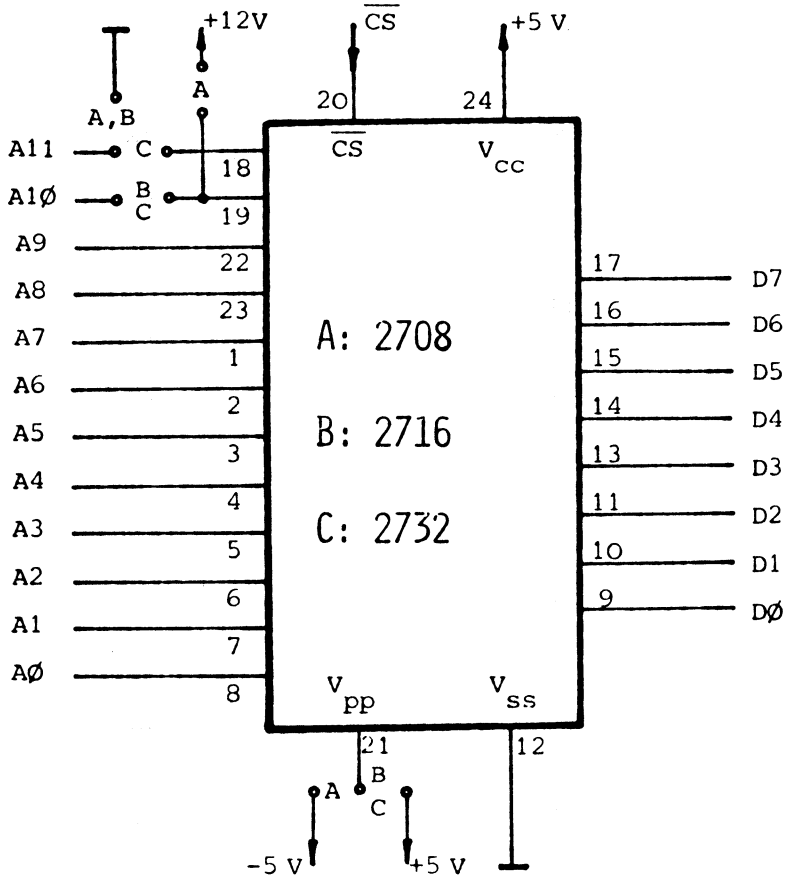
2716: 2048 WORTE MIT JE 8 BIT
EINFACHE +5-V-VERSORGUNG ≈ 60DM

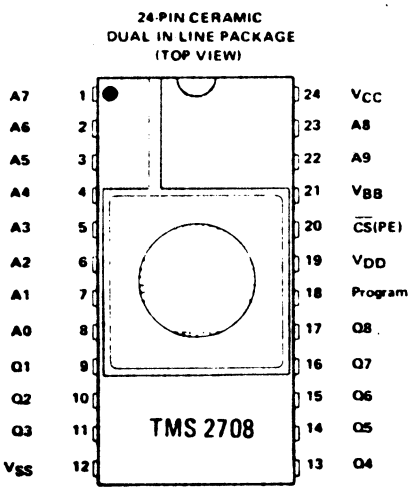
1980 **2732:** 4096 WORTE MIT JE 8 BIT mit 200ns $570 Preis Jan. 80
EINFACHE +5-V-VERSORGUNG

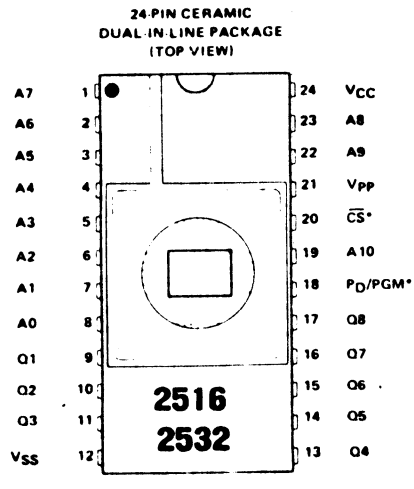
2764: 8192 x 8 Bit



Schaltungsdetail







PIN NOMENCLATURE	
AiNi	Address inputs
$\overline{CS}$	Chip Select
PD PGM,PD $\overline{PGM}$	Power Down/Program
QiNi	Input Output
V_{CC}	+5 V Power Supply
V_{PP}	+25 V Power Supply
V_{SS}	0 V Ground

RAM

- INFORMATIONEN, DIE SICH WÄHREND DER PROGRAMMAUSFÜHRUNG ÄNDERN, LEGT MAN IN SOGENANNTEN SCHREIB/LESE-SPEICHERN AB
  - READ/WRITE MEMORY RWM
  - GEBRÄUCHLICHER: RANDOM ACCESS MEMORY RAM
  - NIMMT ZWISCHENERGEBNISSE AUF
  - NIMMT VARIABLE DATEN AUF

- DER RAM-INHALT KANN VOM COMPUTER AUSGELESEN UND AUCH ÜBERSCHRIEBEN WERDEN
  - AM R/$\overline{W}$-EINGANG WIRD DIE ÜBERTRAGUNGSRICHTUNG ANGEGEBEN

- DIE IM RAM GESPEICHERTE INFORMATION IST FLÜCHTIG
  - DER RAM-INHALT GEHT NACH DEM ABSCHALTEN DER VERSORGUNGS-SPANNUNG VERLOREN
  - DER RAM-INHALT IST NACH DEM EINSCHALTEN DER VERSORGUNGS-SPANNUNG ZUFÄLLIG
  - PUFFERUNG DES INHALTS ÜBER KLEINE AKKUS MÖGLICH

Ausführungsformen

- STATISCHE RAMS
  - DIE INFORMATIONSSPEICHERUNG ERFOLGT IN FORM VON FLIPFLOPS
  - ZWEI AKTIVE ELEMENTE PRO BIT

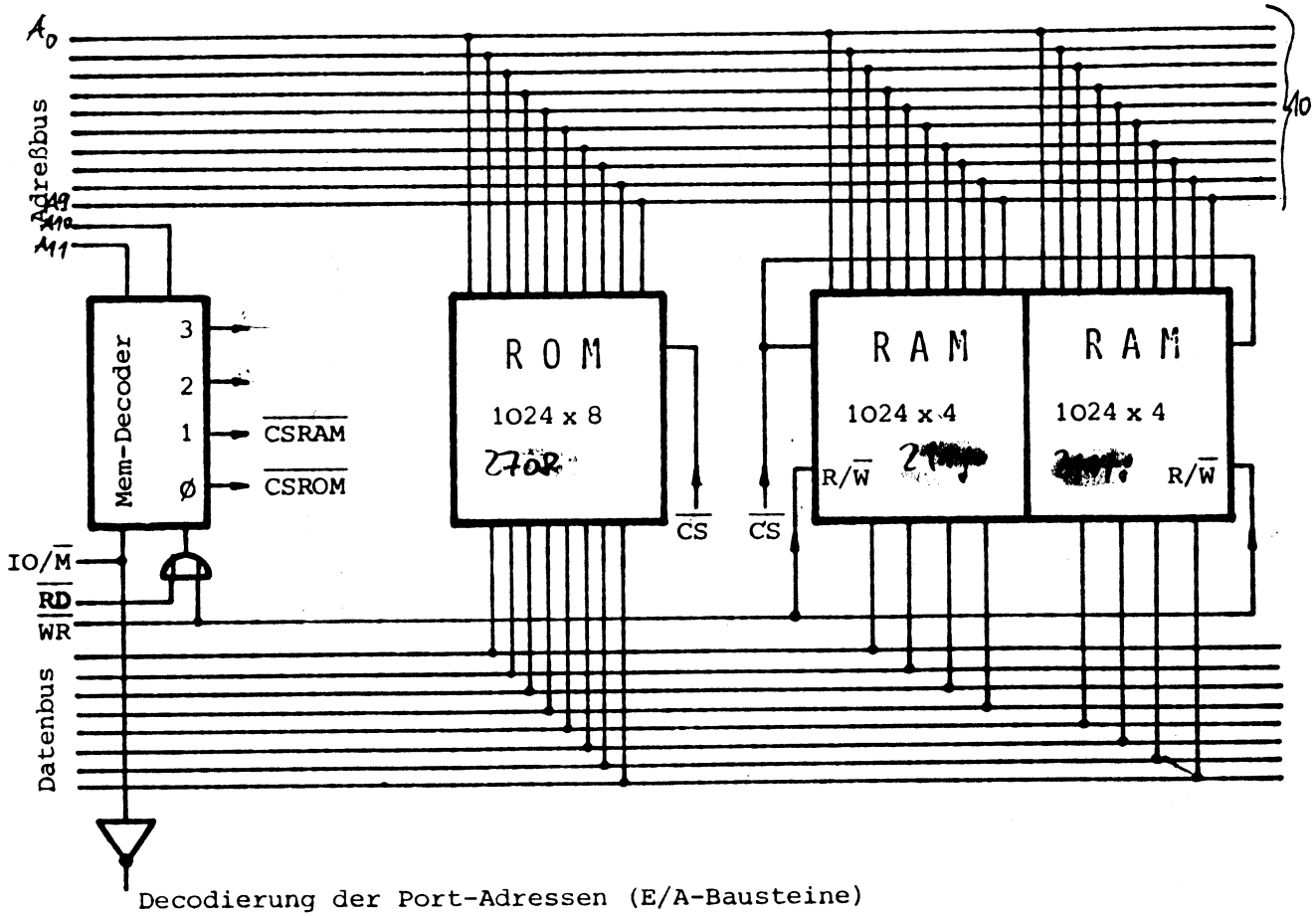
- DYNAMISCHE RAMS
  - DIE INFORMATIONSSPEICHERUNG ERFOLGT IN FORM VON LADUNGS-MENGEN
  - REFRESHING ERFORDERLICH
  - NUR EIN AKTIVES ELEMENT PRO BIT

2102	600ns	1972	256 x 4	
2112	450ns		256 x 4	
5101	≈450ns		256 x 4	Gehäuse ungünstig
2114		Toshiba CMOS		

Adreßdecodierung

- SÄMTLICHE SPEICHER- UND E/A-BAUSTEINE LIEGEN PARALLEL
  - MIT DEN ADRESSLEITUNGEN AM SYSTEM-ADRESSBUS
  - MIT DEN DATENLEITUNGEN AM SYSTEM-DATENBUS
- DIE DATENEIN- UND -AUSGÄNGE SIND NORMALERWEISE IM HOCHOHMIGEN ZUSTAND
  - NUR BEI AKTIVIERUNG DES CS-EINGANGS KANN EIN SPEICHER ODER E/A-BAUSTEIN ANGESPROCHEN WERDEN
- DIE CS-SIGNALE ERZEUGT EINE ZUSÄTZLICHE HARDWARE-DECODIERUNG
- BEISPIEL: SPEICHERSYSTEM MIT 1024 WORTEN ROM UND 1024 WORTEN RAM SOWIE RESERVE VON 2 X 1024 WORTEN AUFBAUEN
  - DIE ZEHN UNTEREN ADRESSBITS AØ...A9 FÜHREN AN ALLE SPEICHERBAUSTEINE
  - DIE ZWEI FOLGENDEN ADRESSBITS A1Ø UND A11 WÄHLEN EINEN VON VIER BLÖCKEN AN
- AUCH DER HARDWARE-DECODIERER BESITZT FREIGABE-EINGÄNGE
  - AKTIVIERUNG NUR BEIM SCHREIB- ODER LESESIGNAL $\overline{WR}$ BZW. $\overline{RD}$
  - IO/$\overline{M}$-SIGNAL UNTERSCHEIDET SPEICHER- UND PORTADRESSEN
- DIE IO/$\overline{M}$-LEITUNG WIRD VON DER ZENTRALEN ABLAUFSTEUERUNG VERWALTET
  - LOW BEI SPEICHERADRESSEN
  - HIGH BEI PORTADRESSEN
  - ZUSTAND ABHÄNGIG VOM JEWEILIGEN BEFEHL

Schaltungsrealisierung



Decodierung der Port-Adressen (E/A-Bausteine)

E : Tri-State Latch

A : Latch

I/O Mapping

- I/O-MAPPED I/O
  DIE E/A-BAUSTEINE WERDEN ÜBER SPEZIELLE EIN/AUSGABE-BEFEHLE ANGESPROCHEN
  - DIESE BEFEHLE BRINGEN DIE IO/$\overline{M}$-LEITUNG AUF HIGH
  - SEPARATE CS-ERZEUGUNG FÜR DIE E/A-BAUSTEINE

- MEMORY MAPPED I/O
  DIE E/A-BAUSTEINE WERDEN WIE EINE SPEICHERSTELLE ANGESPROCHEN
  - DIE IO/$\overline{M}$-LEITUNG BLEIBT AUF LOW
  - E/A-BAUSTEINE UND SPEICHER HABEN DIESELBE CS-ERZEUGUNG

Günther, da Schneider

1.3.4 Datenein- und -ausgabe

- FÜR DEN DATENVERKEHR ZWISCHEN CPU UND PERIPHERIE GIBT ES DREI PRINZIPIELL VERSCHIEDENEN MÖGLICHKEITEN

1) - PROGRAMMGESTEUERT

2) - VOM INTERRUPT ABGELEITET

3) - DIREKTER SPEICHERZUGRIFF

1) • PROGRAMMGESTEUERTE EIN/AUSGABE

IM PROGRAMM WIRD STÄNDIG EINE BEFEHLSFOLGE DURCHLAUFEN, DIE DATEN EINLIEST BZW. AUSGIBT

- AUCH WENN KEINE NOTWENDIGKEIT BESTEHT, D.H. KEINE ÄNDERUNG VORLIEGT
- ERFORDERT EINE REIHE VON BEFEHLEN
- LANGSAMSTE EIN/AUSGABE-MÖGLICHKEIT
- HÖCHSTE CPU-BELASTUNG
- KEIN ZUSÄTZLICHER HARDWARE-AUFWAND

noch: Datenein- und -ausgabe

2) • VOM INTERRUPT ABGELEITET
DIE BEFEHLSFOLGE ZUM EINLESEN BZW. AUSGEBEN WIRD NUR DANN DURCHLAUFEN, WENN DIES ERFORDERLICH IST

- WENN DAS PERIPHERIEGERÄT NEUE DATEN BEREITHÄLT BZW. BENÖTIGT, LÖST ES EINEN INTERRUPT AUS
- DIE CPU UNTERBRICHT DAS LAUFENDE PROGRAMM UND SPRINGT IN EIN SPEZIELLES UNTERPROGRAMM, DIE INTERRUPT-SERVICE-ROUTINE
- NACH ABARBEITEN DES UNTERPROGRAMMS GEHT DIE PROGRAMMAUSFÜHRUNG AN DER ZUVOR VERLASSENEN STELLE WEITER
- GERINGERE CPU-BELASTUNG: NUR IM BEDARFSFALL
- HARDWARE-AUFWAND FÜR INTERRUPT-ERZEUGUNG

3) • DIREKTER SPEICHERZUGRIFF (DIRECT MEMORY ACCESS DMA)
DIE CPU WIRD KURZZEITIG ANGEHALTEN, SCHALTET DEN DATEN- UND ADRESSBUS SOWIE DIE LEITUNGEN $\overline{RD}$, $\overline{WR}$ UND IO/$\overline{M}$ IN DEN HOCHOHMIGEN ZUSTAND UND DIE PERIPHERE STELLE ÜBERNIMMT DIE SYSTEMSTEUERUNG

- DMA-INITIALISIERUNG: HIGH AM HOLD-EINGANG
- MIT DEM $\overline{HLDA}$-SIGNAL WIRD DIE GEWÜNSCHTE SPEICHERADRESSE ERZEUGT UND AUF DEN ADRESSBUS GESCHALTET
- DER DATENAUSTAUSCH ERFOLGT DIREKT UNTER UMGEHUNG DER CPU
- GERINGSTE CPU-BELASTUNG: NUR DIE SPEICHERZUGRIFFSZEIT PLUS $\overline{HLDA}$-VERZÖGERUNG
- HOHER HARDWARE-AUFWAND FÜR DIE ADRESS- UND $\overline{RD}$/$\overline{WR}$-ERZEUGUNG

HLDA Hold

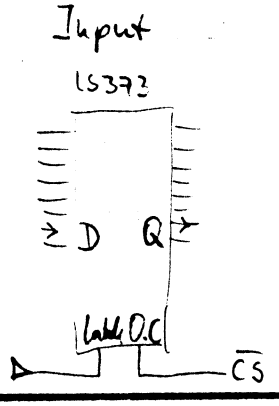
- ZUR DATENEIN- UND -AUSGABE GIBT ES KOMFORTABLE I/O-BAUSTEINE

- 8255 PROGRAMMABLE PERIPHERAL INTERFACE
  - ENTHÄLT DREI SEPARATE 8-BIT-E/A-KANÄLE
  - ÜBERTRAGUNGSRICHTUNG PROGRAMMIERBAR

- 8155 RAM MIT I/O-PORTS UND TIMER
  - DREI SEPARATE E/A-KANÄLE (2 X 8 BIT, 1 X 6 BIT)
  - RAM-SPEICHER MIT 256 WORTEN
  - DIREKT 8085-KOMPATIBEL (KEIN SEPARATES ADRESS-LATCH)

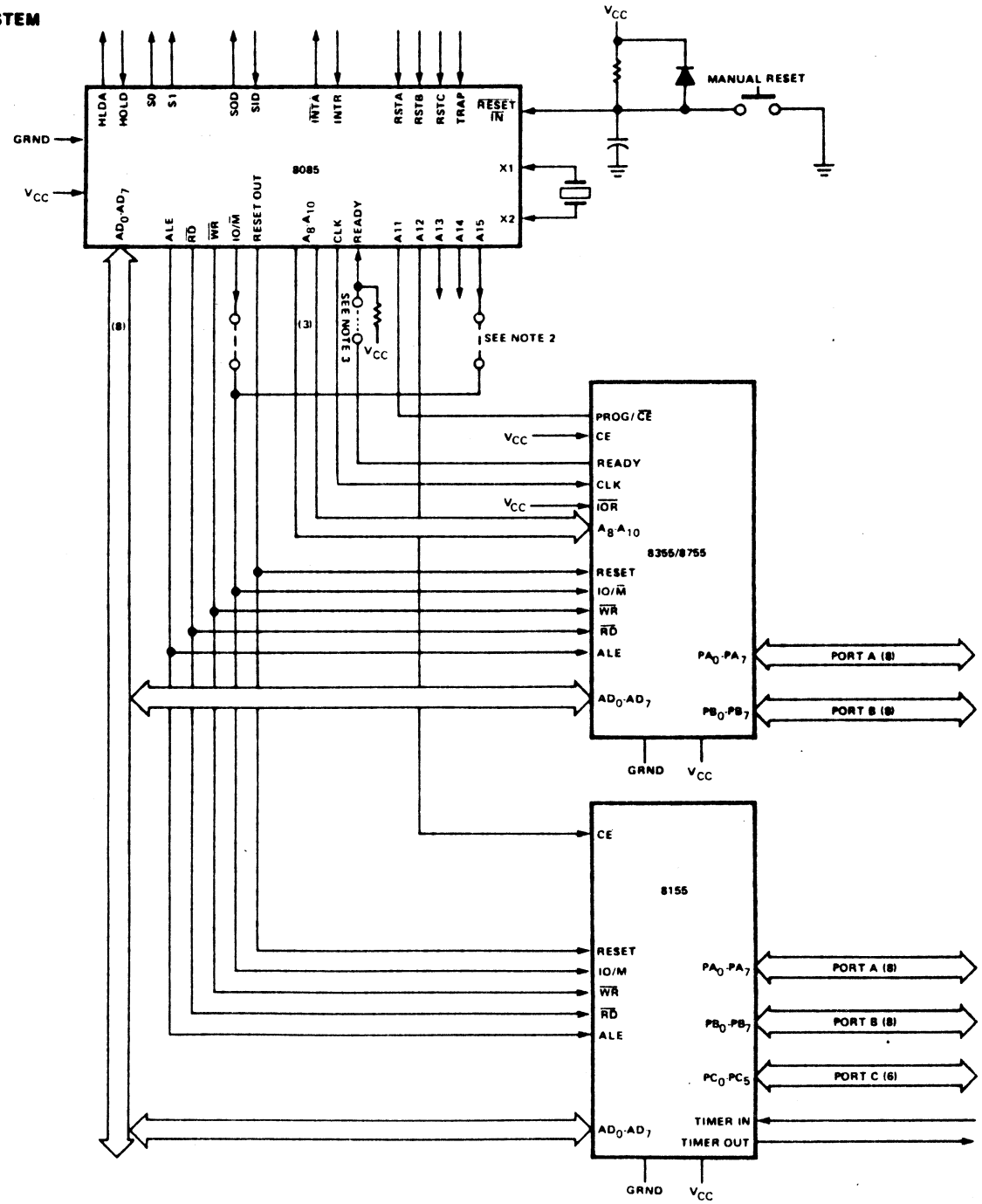
- EINFACHSTE AUSGABE: 8-BIT-LATCH
  - INFORMATION VOM DATENBUS ÜBERNEHMEN
  - ÜBERNAHME-IMPULS VON DER ADRESS-DECODIERUNG

- EINFACHSTE EINGABE: 8-BIT-TRI-STATE-BUFFER
  - INFORMATION AUF DEN DATENBUS SCHALTEN
  - AKTIVIERUNGSSIGNAL VON DER ADRESS-DECODIERUNG



Schaltungsdetail

MINIMUM 8085 SYSTEM



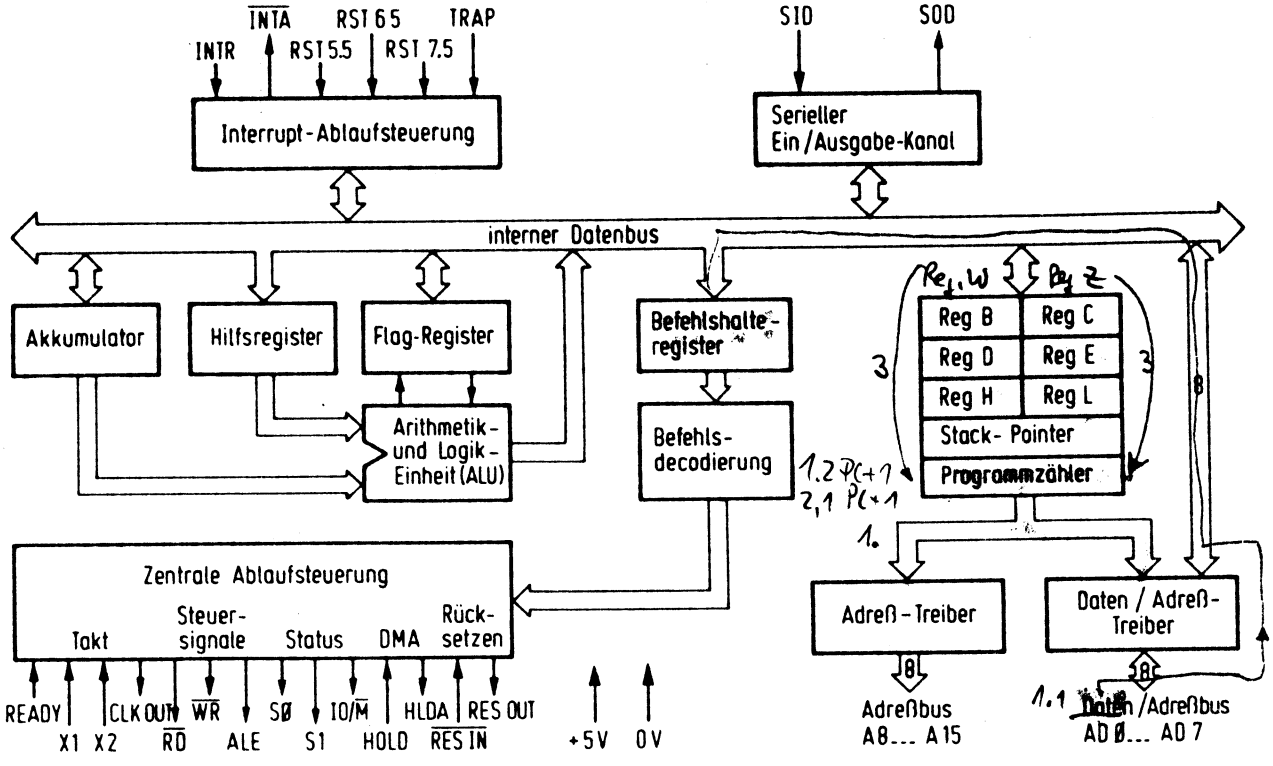
NOTE 1: TRAP, INTR, AND HOLD MUST BE GROUNDED IF THEY AREN'T USED.
NOTE 2: USE IO/M FOR STANDARD I/O MAPPING. USE A15 FOR MEMORY MAPPED I/O.
NOTE 3. CONNECTION IS NECESSARY ONLY IF ONE T_{WAIT} STATE IS DESIRED.

1.4 Befehlsausführung

- ALLE BEFEHLE UND DATEN BESITZEN DIE FESTE WORTLÄNGE, DIE DURCH DIE MIKROCOMPUTER-STRUKTUR VORGEGEBEN IST
  - REICHT DIESE WORTLÄNGE NICHT AUS, WERDEN MEHRERE WORTE ANEINANDERGEREIHT
- DER JMP-BEFEHL (SPRUNG-BEFEHL) IST EIN MEHRWORTBEFEHL
  - ER UMFASST DREI 8-BIT-WORTE
  - IM ERSTEN STEHT DER MASCHINENCODE FÜR JMP: C3
  - IM ZWEITEN UND DRITTEN WORT STEHT DIE ZIELADRESSE
- DER BEFEHL JMP ØØØØ BEWIRKT EINEN PROGRAMMSPRUNG ZUR ADRESSE ØØØØ
  - MASCHINENCODE: C3 ØØ ØØ
  - ER BELEGT DREI PLÄTZE IM SPEICHER
- BEFEHLSZYKLUS
  - HOLEN DES BEFEHLS
  - DECODIEREN DES BEFEHLS
  - AUSFÜHREN DES BEFEHLS
- DIE KOORDINATION ÜBERNIMMT DIE ZENTRALE ABLAUFSTEUERUNG

Holen des Befehls

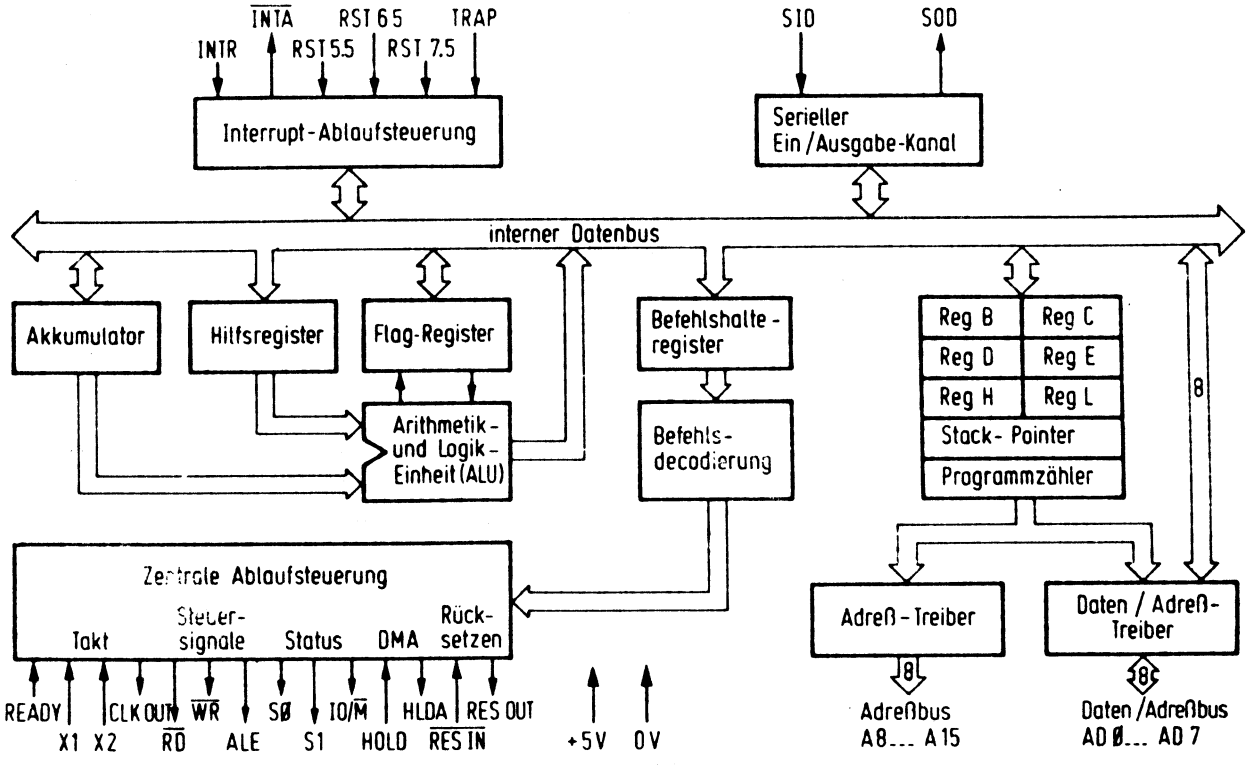
- DER PROGRAMMZÄHLERSTAND WIRD AUF DEN ADRESSBUS GESCHALTET
  - ER GELANGT AN ALLE ANGESCHLOSSENEN SYSTEMKOMPONENTEN
  - DIE HARDWARE-DECODIERUNG WÄHLT DEN PROGRAMMSPEICHER AN
  - DIE ADRESSIERTE SPEICHERSTELLE WIRD AUSGELESEN
  - DER INHALT GELANGT ÜBER DEN DATENBUS INS BEFEHLSHALTE-REGISTER IR IN DER CPU



Decodieren des Befehls

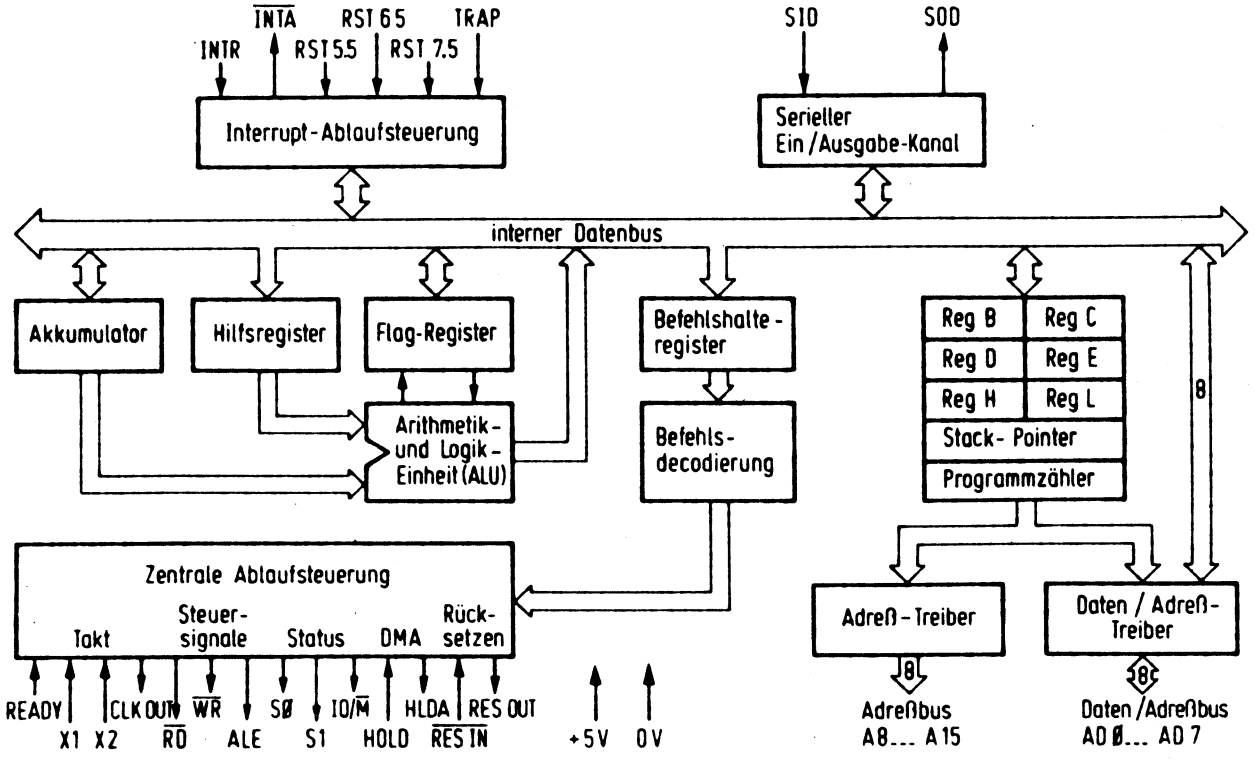
- DER BEFEHL IM BEFEHLSHALTEREGISTER WIRD VON DER BEFEHLSDECODIERUNG ENTSCHLÜSSELT; PARALLEL DAZU WIRD DER PROGRAMMZÄHLER UM EINS ERHÖHT
  - DIE CPU ERKENNT AUS DEM MUSTER „C3", DASS ES EIN DREIWORTBEFEHL IST
  - DER PROGRAMMZÄHLER WIRD ZWEI WEITERE MALE AUF DEN ADRESSBUS GESCHALTET, UM DIE RESTLICHEN BEIDEN WORTE DIESES BEFEHLS AUSZULESEN

- DIESMAL GELANGT DIE INFORMATION NICHT INS BEFEHLSHALTEREGISTER
  - DAS ZWEITE WORT DES BEFEHLS KOMMT INS HILFSREGISTER Z
  - DAS DRITTE WORT DES BEFEHLS KOMMT INS HILFSREGISTER W



Ausführen des Befehls

- DIE ZENTRALE ABLAUFSTEUERUNG SORGT FÜR DIE BEFEHLSAUSFÜHRUNG
  - DIE IN DEN HILFSREGISTERN W UND Z ZUSAMMENGESETZTE ADRESSE WIRD IN DEN PROGRAMMZÄHLER ÜBERSCHRIEBEN



2 Grundlagen der Programmierung

Begriffsbestimmungen

BYTE	EIN 8-BIT-WORT
NIBBLE	HALBES BYTE
BIT Ø	LEAST SIGNIFICANT BIT LSB
BIT 7	MOST SIGINIFICANT BIT MSB
1 K	BINÄRES VIELFACH $2^{10} = 1024$ 65 536 = 64 K!
Ø93Ø	ADRESSE EINER SPEICHERSTELLE (HEXADEZIMAL)
(Ø93Ø)	INHALT DER SPEICHERSTELLE Ø93Ø
((Ø93Ø))	INHALT DERJENIGEN SPEICHERSTELLE, DEREN ADRESSE IN Ø93Ø STEHT
A,B,L	REGISTER A, B, L
(A)	INHALT VON REG A
RP B,D,H	REGISTERPAAR B, D, H
((H,L))	INHALT DERJENIGEN SPEICHERSTELLE, DEREN ADRESSE IN RP H STEHT
PC	PROGRAMMZÄHLER
SP	STACK POINTER
SPH	STACK POINTER HIGH (DIE OBEREN ACHT BIT)
SPL	STACK POINTER LOW (DIE UNTEREN ACHT BIT)
PAGE	256-BYTE-SPEICHERBEREICH, IN DEM DIE OBEREN 8 ADRESSBITS KONSTANT BLEIBEN

2.1 UMS-Struktur

- FÜR JEDE, AUCH DIE GERINGSTE TÄTIGKEIT BRAUCHT DER COMPUTER EIN EIGENES PROGRAMM

- WENN NACH DEM EINSCHALTEN EINE DEFINIERTE ANZEIGE ERSCHEINT UND BEDIENER-EINGABEN ÜBER DIE TASTATUR MÖGLICH SIND, IST FÜR JEDEN DIESER VORGÄNGE EIN EIGENER PROGRAMMTEIL VERANTWORTLICH

- DERARTIGE BASISPROGRAMME DIENEN NUR ZUR ENTWICKLUNGSUNTERSTÜTZUNG

- MIT IHRER HILFE ERSTELLT, MODIFIZIERT UND TESTET DER PROGRAMMIERER SEINE ANWENDER-PROGRAMME

Software zur Entwicklungsunterstützung

- BETRIEBSSYSTEM

  PROGRAMMPAKET ZUR ENTWICKLUNGSUNTERSTÜTZUNG BEI DER ERSTELLUNG VON ANWENDERPROGRAMMEN; UMFASST EIN/AUSGABE-ROUTINEN, FEHLER-SUCHPROGRAMME, DOKUMENTATIONSHILFEN, PROM-PROGRAMMIEREINRICHTUNG INKL. ASSEMBLER UND EDITOR

- MONITOR (-PROGRAMM)

  ABGEMAGERTES BETRIEBSSYSTEM FÜR DIE PROGRAMMENTWICKLUNG AUF UNTERSTER EBENE

- ASSEMBLER

  ÜBERSETZER-PROGRAMM, DAS PROGRAMME AUS DEM ASSEMBLERCODE IN DIE MASCHINENSPRACHE UMSETZT UND AUS SYMBOLISCHEN SPRUNGZIELEN DIE ADRESSBERECHNUNG DURCHFÜHRT; DER ASSEMBLER LÄUFT AUF EINEM COMPUTER DERSELBEN ART, FÜR DEN ER DIE UMSETZUNG DURCHFÜHRT

- CROSS-ASSEMBLER

  UMSETZ-PROGRAMM, DAS AUF EINEM ANDEREN COMPUTER LÄUFT

- DISASSEMBLER

  UMSETZ-PROGRAMM, DAS EIN IN MASCHINENSPRACHE VORLIEGENDES PROGRAMM IN DEN ASSEMBLERCODE ÜBERSETZT

- EDITOR

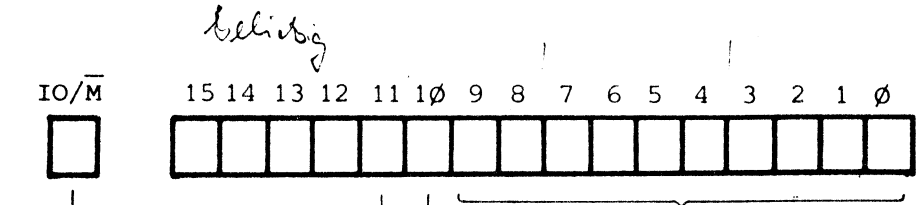
  SCHREIBMASCHINENFUNKTIONEN FÜR DIE EINGABE UND KORREKTUR VON ANWENDER-PROGRAMMEN

- COMPILER

  ÜBERSETZER-PROGRAMM, DAS PROGRAMME AUS EINER HÖHEREN PROGRAMMIERSPRACHE IN DIE MASCHINENSPRACHE ÜBERSETZT

2.1.1 Mikrocomputer-Hardware

- IN 8080/8085-SYSTEMEN IST JEDE SPEICHERADRESSE 16 BIT BREIT
  - LIEGT EINE SPEICHERADRESSE VOR, IST IO/$\overline{M}$ AUF LOW
- IN 8080/8085-SYSTEMEN IST JEDE PORTADRESSE 8 BIT BREIT
  - SIE ERSCHEINT DUPLIZIERT AUF DEM ADRESSBUS
  - LIEGT EINE PORTADRESSE VOR, IST IO/$\overline{M}$ AUF HIGH
- UMS-ADRESSRAUMBELEGUNG



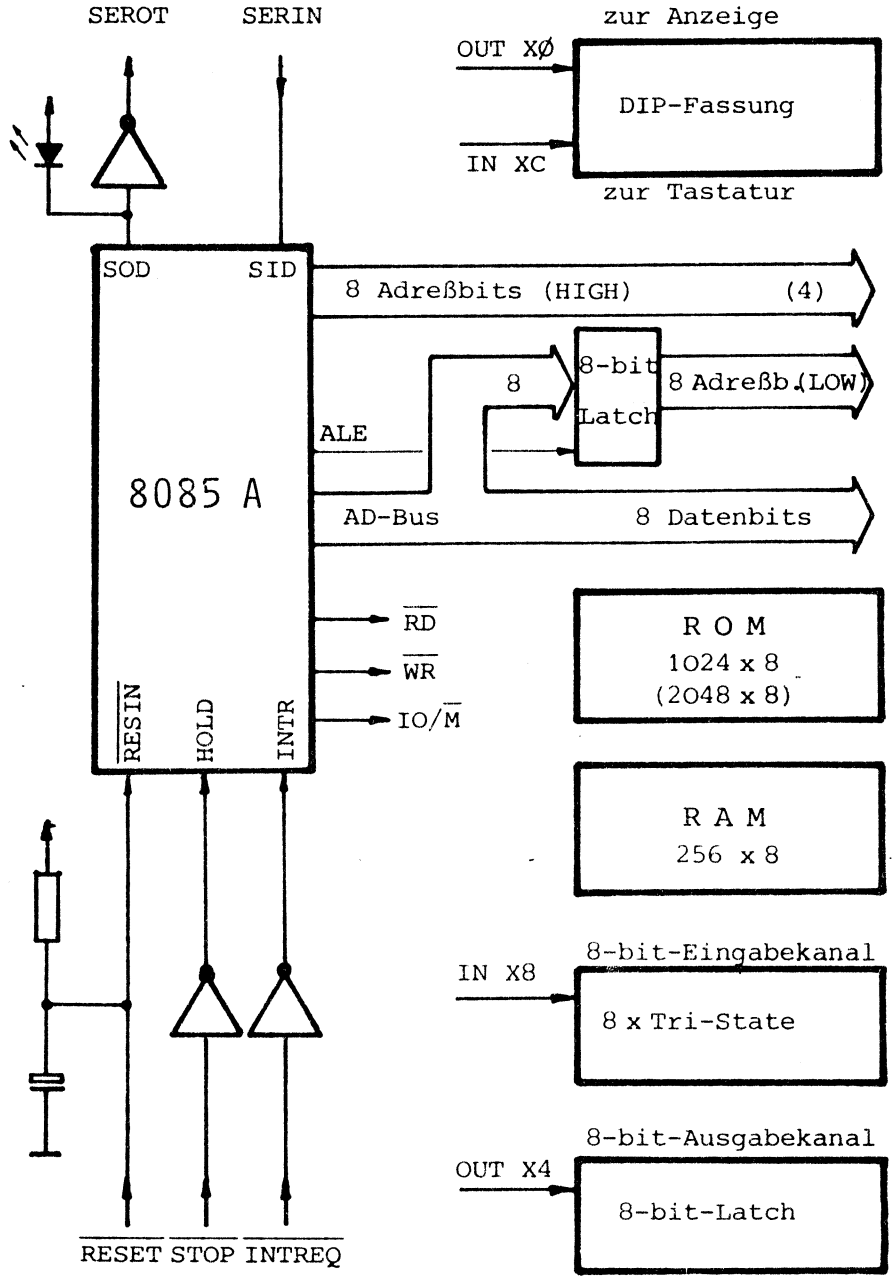
Mem.Adr.	0	0	0	ROM	1K	000 ... 3FF
	0	0	1	ROM (Reserve)	1K	400 ... 7FF
	0	1	0	RAM	1K	800 ... BFF
	0	1	1	externer Speicher	1K	C00 FFF
Portadr.	1	0	0	ext.Ausgabe-Kanal	1K	
	1	0	1	CPU-Ausgabe-Kanal	1K	
	1	1	0	CPU-Eingabe-Kanal	1K	
	1	1	1	ext.Eingabe-Kanal	1K	

2.43

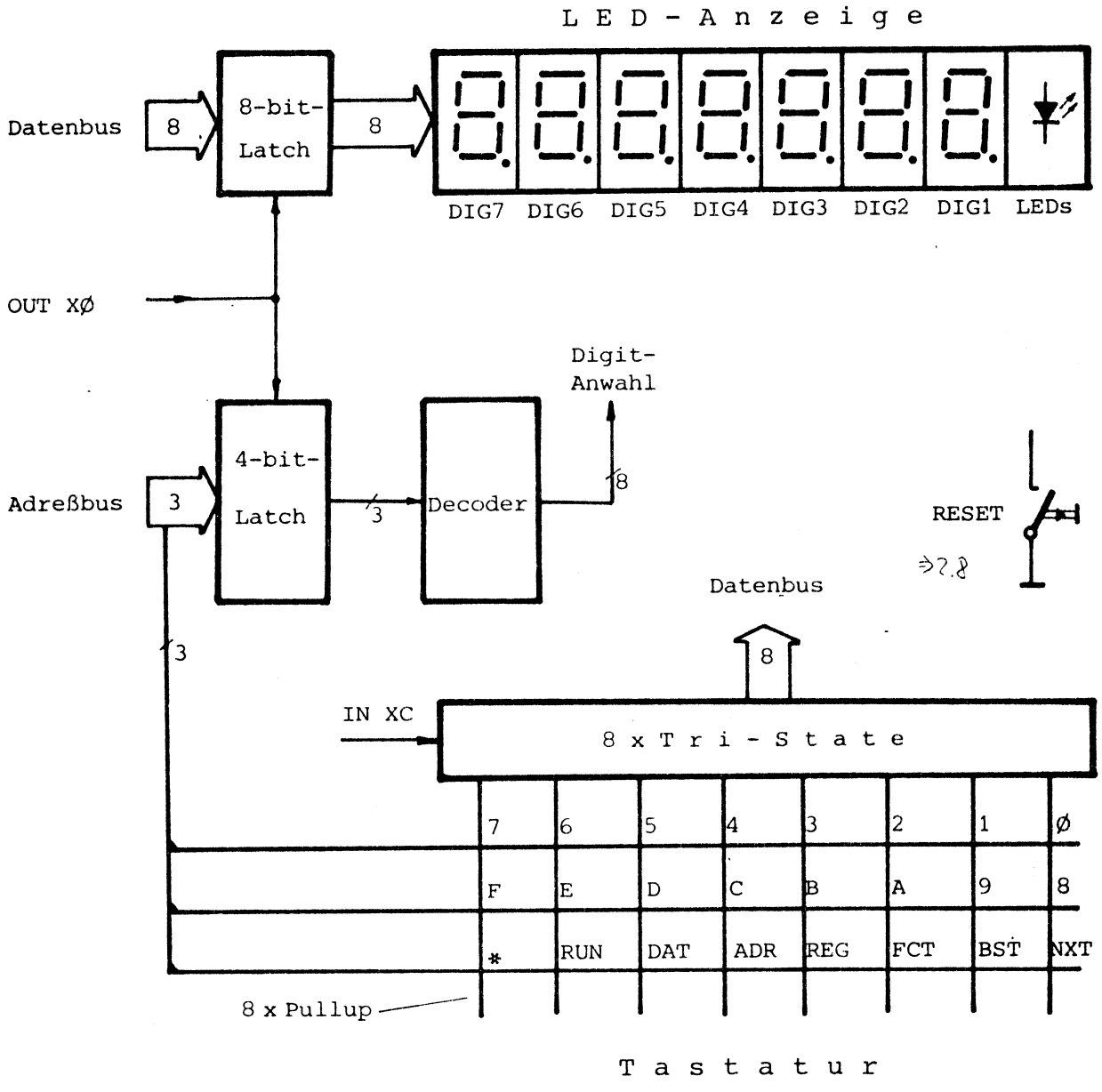
damit nach Rücksichten Betriebssystem angesprochen ist

möglichst alle Speicher gleiche Kapazität wählen, erleichtert wesentlich

Mikrocomputer 852



Tastatur/Anzeige 853



Initialisieren

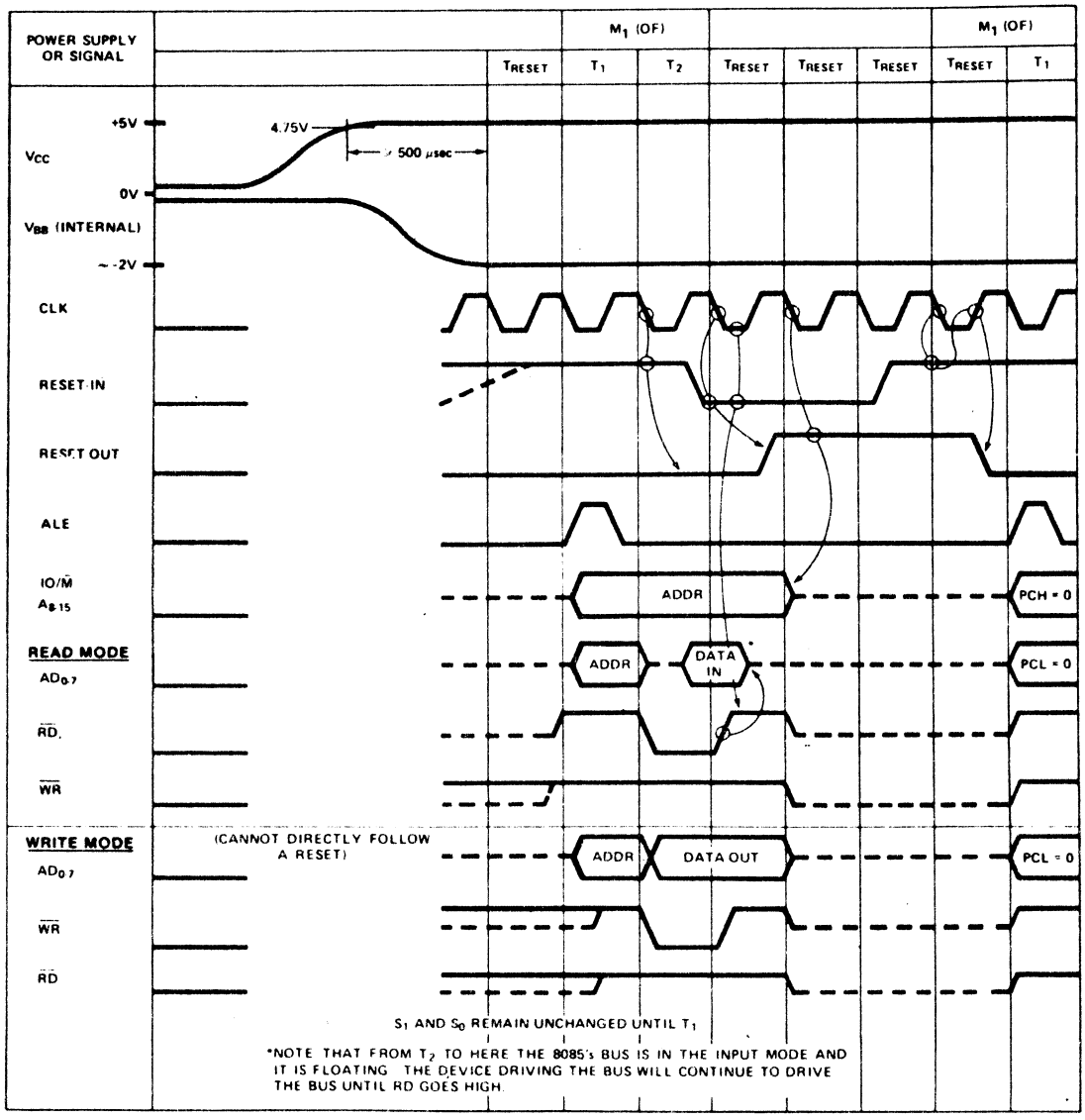
- RESET

  LOW-POTENTIAL AN $\overline{\text{RESIN}}$ BEWIRKT FOLGENDES

  - NULLSETZEN DES PROGRAMMZÄHLERS PC
  - LÖSCHEN DES BEFEHLSHALTEREGISTERS IR
  - RÜCKSETZEN DES SOD-LATCH'
  - RÜCKSETZEN DES HLDA-FLIPFLOPS
  - RÜCKSETZEN DES IE-FLIPFLOPS
  - SETZEN ALLER RST-MASKEN

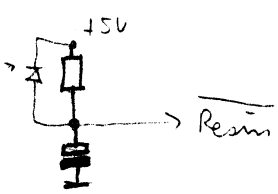
- $\overline{\text{RESIN}}$ HAT AM 8085 SCHMITT-TRIGGER-~~FUNKTION~~ Eingang

RESET-Timing



POWER ON AND $\overline{\text{RESET IN}}$ TIMING.

für $\geq 500\mu s$



bei Spannungseinbrüchen auf Versorgung → abschalten

2.1.2 UMS-Software

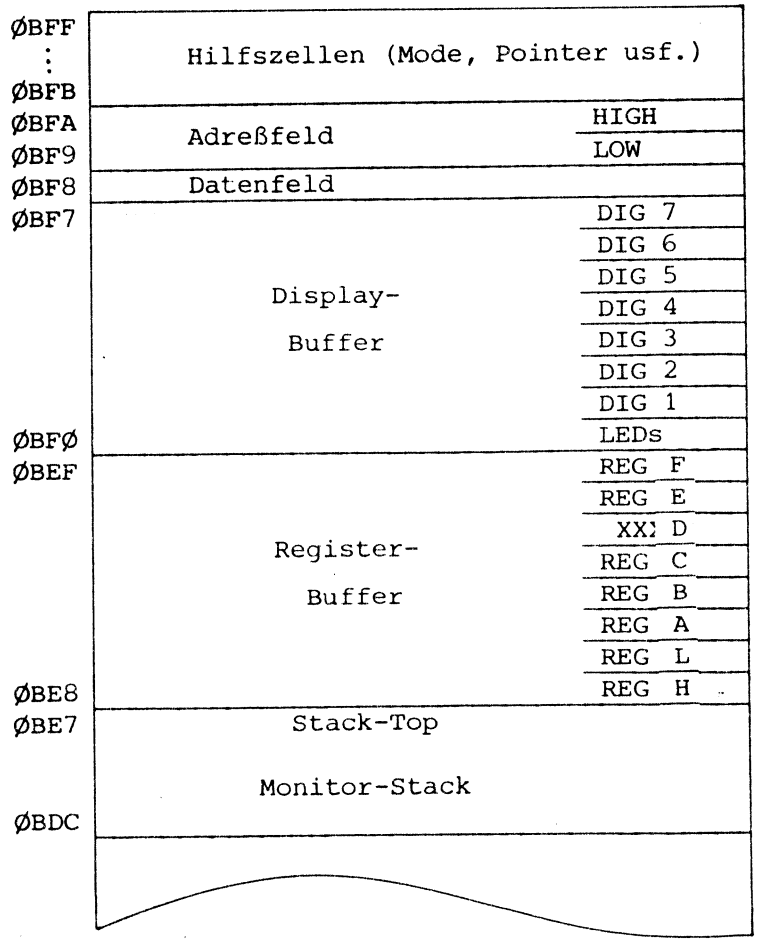
- DAS MONITOR-PROGRAMM IST FEST IM EPROM ABGESPEICHERT; ES KANN
  - SPEICHERINHALTE INSPIZIEREN
  - SPEICHERINHALTE MODIFIZIEREN
  - REGISTERINHALTE INSPIZIEREN
  - REGISTERINHALTE MODIFIZIEREN
  - ANWENDERPROGRAMME STARTEN
- DER MONITOR STEUERT DIE ANZEIGE AN
  - LINKSBÜNDIG ERSCHEINT EINE SPEICHERADRESSE
  - RECHTSBÜNDIG ERSCHEINT DER INHALT DER ADRESSIERTEN SPEICHERSTELLE
- DER MONITOR HAT DREI ANZEIGE-MODES
  - EINGABE UND ANZEIGE VON DATEN:
    DEZIMALPUNKT IM DATENFELD LEUCHTET
  - EINGABE UND ANZEIGE VON ADRESSEN:
    DEZIMALPUNKT IM ADRESSFELD LEUCHTET
  - EINGABE UND ANZEIGE VON REGISTERINHALTEN:
    ANSTELLE EINER ADRESSE ERSCHEINT DIE REGISTERBEZEICHNUNG
- NACH DEM RÜCKSETZEN GEHT DER MONITOR
  - IN DEN DATEN-MODE
  - ZUR ADRESSE Ø8ØØ (RAM-ANFANG)

- DER MONITOR FRAGT DIE TASTATUR AB
  - HEX-TASTEN Ø...9 UND A...F
  - BEFEHLSTASTEN
  - RESET FÜHRT DIREKT AN DEN CPU-RÜCKSETZ-EINGANG
- HEX: HEX-TASTEN DIENEN ZUR PARALLEN EINGABE VON VIER BITS, DIE INS DATEN- ODER ADRESSFELD NACHGESCHOBEN WERDEN
- NXT: DIE IM DATENFELD STEHENDE INFORMATION WIRD IN DIE LINKS ANGEZEIGTE SPEICHERSTELLE EINGESCHRIEBEN, UND DIE ADRESSE WIRD UM EINS ERHÖHT
- BST: ERNIEDRIGEN DER ADRESSE UM EINS
- DAT: DATEN-MODE ANWÄHLEN
- ADR: ADRESS-MODE WÄHLEN
- REG: REGISTER-MODE WÄHLEN
- RUN: PROGRAMM VON DER ANGEZEIGTEN ADRESSE AUS STARTEN UND VORHER REGISTER-INHALTE LADEN
- FCT: SONDERFUNKTIONEN FÜR MONITOR-ERWEITERUNGEN

Monitor-Unterprogramme

- DER MONITOR BESTEHT AUS EINER VIELZAHL MODULARER UNTER-PROGRAMME
  - DIESE KANN DER ANWENDER IN SEINE PROGRAMME EINBAUEN
- UNTERPROGRAMM-AUFRUF
  - ANGEGEBENE EINSPRUNGBEDINGUNG HERSTELLEN
  - IM ANWENDER-PROGRAMM CALL, GEFOLGT VON DER STARTADRESSE PROGRAMMIEREN
- UNTERPROGRAMM-RÜCKSPRUNG
  - AUTOMATISCH NACH ABARBEITEN DES UNTERPROGRAMMS
  - FORTSETZUNG IM ANWENDER-PROGRAMM AN DER ZUVOR VERLASSENEN STELLE
- FÜR SEINE AUFGABEN BRAUCHT DER MONITOR EINIGE RAM-ZELLEN
- STACK-ARBEITSWEISE
  - VOR DEM SPRUNG INS UNTERPROGRAMM RETTET DIE CPU DEN AUGENBLICKLICHEN PROGRAMMZÄHLER-STAND IN DEN STACK
  - BEIM RÜCKSPRUNG AUS DEM UNTERPROGRAMM HOLT DIE CPU DIE ZUVOR GERETTETE ADRESSE ZURÜCK IN DEN PROGRAMMZÄHLER
  - DER STACK-POINTER GIBT AN, BEI WELCHER ADRESSE IM ARBEITS-SPEICHER DER STACK BEGINNT
  - DER STACK POINTER MUSS ZU BEGINN SOFTWAREMÄSSIG GESETZT WERDEN

Memory Map



- ACHTUNG: BEI BESTÜCKUNG MIT 0,25-K-RAMS ADRESSÜBERSCHNEIDUNG

  ØBXX = ØAXX = Ø9XX = Ø8XX

2.2 Programmeingabe

Namen max. 5 Zeichen, 1 Ziffer muß Buchstabe

- DER CALL-BEFEHL DIENT ZUM AUFRUF VON UNTERPROGRAMMEN
  - IM ASSEMBLERCODE FOLGT DER SYMBOLISCHE NAME DES UNTER-PROGRAMMS: CALL SUB1
  - IM MASCHINENCODE FOLGT DIE STARTADRESSE DES UNTER-PROGRAMMS
- ADRESSEN SIND IM 8080/8085-SYSTEM IMMER 16 BIT BREIT .
  - SIE LASSEN SICH AUS ZWEI BYTES ZUSAMMENSETZEN
- DER CALL-BEFEHL BESTEHT DAHER AUS DREI BYTES
  - DAS ERSTE BYTE „CD" IST DER MASCHINENCODE FÜR „CALL"
  - DAS ZWEITE BYTE ENTHÄLT DIE UNTERE HÄLFTE DER ADRESSE
  - DAS DRITTE BYTE ENTHÄLT DIE OBERE HÄLFT DER ADRESSE
- BEISPIEL: IM ANWENDERPROGRAMM SOLL DAS UNTERPROGRAMM DSPSW AUFGERUFEN WERDEN; ES BEGINNT BEI ADRESSE Ø275
- UNTERPROGRAMM-AUFRUF IM
  - ASSEMBLERCODE: CALL DSPSW
  - MASCHINENCODE: CD 75 Ø2
- CALL IST EIN DREIWORTBEFEHL; ER BELEGT DREI AUFEINANDERFOLGENDE SPEICHERSTELLEN

- MONITOR-UNTERPROGRAMM DSPSW [Ø275] STELLT DAS PROZESSOR-STATUS-WORT PSW IN DER ANZEIGE DAR; (ACC) RECHTSBÜNDIG, (F) IN DER LED-ZEILE. RÜCKSPRUNG ERST NACH DRÜCKEN UND LOSLASSEN VON NXT

- DIESES UNTERPROGRAMM SOLL IN EIN ANWENDERPROGRAMM EINGEBAUT WERDEN

- DIE ANWENDERPROGRAMME BEGINNEN ZWECKMÄSSIGERWEISE BEI Ø8ØØ
  - DIESE ADRESSE ERSCHEINT AUTOMATISCH NACH RESET

- DAS EINSCHREIBEN IN DEN SPEICHER ERFOLGT ERST NACH DEM DRUCK AUF NXT

Adresse	Inhalt	Bedeutung
Ø 8 Ø Ø	C D	CALL DSPSW
Ø 8 Ø 1	7 5	
Ø 8 Ø 2	Ø 2	
Ø 8 Ø 3	X X	

- NACH DEM LADEN IMMER AUF RICHTIGKEIT ÜBERPRÜFEN

Programmierformulare

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
0800	CD	7502		EMMA	CALL DSPSW	1
0803	3C				INR A	2
	C3	0008			JMP EMMA	3
0807						4
						5
						6
						7
						8
						9
						10
						11
						12
						13
						14
						15
						16

- EIN ASSEMBLER-BEFEHL BELEGT IMMER EINE ZEILE
- LABEL SIND SYMBOLISCHE NAMEN, DENEN EIN ASSEMBLERPROGRAMM SPEICHERADRESSEN ZUWEIST
  - MAXIMAL FÜNF ALPHANUMERISCHE ZEICHEN
  - BEGINNEND MIT EINEM BUCHSTABEN

Programmbeispiel

- AKKUMULATOR-INHALT MODIFIZIEREN;
  PSW ANZEIGEN UND AUSWIRKUNGEN VERFOLGEN

* INR A (3C) INCREMENT ACCUMULATOR
ERHÖHE DEN AKKUMULATOR-INHALT UM EINS

* ADD A (87) ADD REGISTER A
ADDIERE ZUM INHALT DES AKKUMULATORS DEN
INHALT DES AKKUMULATORS; ERGEBNIS IM AKKU

* JMP ADDR (C3 YY XX) JUMP ADDRESS
SPRINGE ZUR ANGEGEBENEN ADRESSE;
ADRESS-ANGABE: ERST UNTERES, DANN OBERES BYTE

- STARTEN DES PROGRAMMS NACH DEM LADEN: RES, RUN

2.3 Stack

- DER STACK IST EIN TEIL DES ARBEITSSPEICHERS
  - ER NIMMT BEIM UNTERPROGRAMM-AUFRUF DIE RÜCKSPRUNGADRESSE AUF
  - ER KANN AUCH ZUR ZWISCHENSPEICHERUNG VON DATEN BENUTZT WERDEN
- DEN BEGINN DES STAPELSPEICHERS (STACK TOP) DEFINIERT DER PROGRAMMIERER SOFTWAREMÄSSIG DURCH LADEN DES STACK POINTERS SP

* LXI SP, ADDR (31 YY XX) LOAD STACK POINTER IMMEDIATE
STACK POINTER MIT ADRESSE XX YY LADEN

- JEDE STACK-AKTIVIERUNG BELEGT ZWEI BYTES IM ARBEITSSPEICHER
  - BEI CALL WERDEN DIE ZWEI BYTES DER RÜCKSPRUNGADRESSE EINGESCHRIEBEN
  - BEI RETURN WERDEN DIE BEIDEN BYTES WIEDER AUSGELESEN
  - BEI VERSCHACHTELTEN UNTERPROGRAMMEN DEHNT SICH DER STACK ENTSPRECHEND WEITER AUS
- EINSCHREIBEN IN DEN STACK
  - VOR DEM EINSCHREIBEN WIRD DER SP AUTOMATISCH UM EINS ERNIEDRIGT
  - NACH DEM EINSCHREIBEN VON ZWEI BYTES (RÜCKSPRUNGADRESSE) IST DER SP UM ZWEI ERNIEDRIGT WORDEN
- AUSLESEN AUS DEM STACK
  - NACH DEM AUSLESEN WIRD DER SP AUTOMATISCH UM EINS ERHÖHT
  - NACH DEM AUSLESEN VON ZWEI BYTES IST DER SP UM ZWEI ERHÖHT WORDEN

- AUTOMATISCHE AKTIVIERUNG BEI UNTERPROGRAMMEN
  - CALL: DER AUGENBLICKLICHE PC-STAND WIRD IN DEN STACK ÜBERSCHRIEBEN (PCH ZUERST)
  - RET: DER PROGRAMMZÄHLER WIRD MIT DEM INHALT DER VOM SP ADRESSIERTEN SPEICHERSTELLEN GELADEN

- AKTIVIERUNG PER PROGRAMM
  - PUSH RP: DER INHALT DES ANGEGEBENEN REGISTERPAARS WIRD IN DEN STACK ÜBERSCHRIEBEN
  - POP RP: DAS ANGEGEBENE REGISTERPAAR WIRD MIT DEM INHALT DER VOM SP ADRESSIERTEN SPEICHERSTELLEN GELADEN

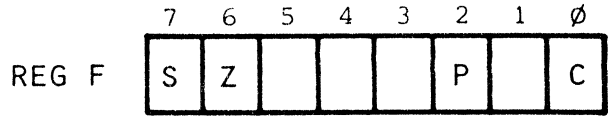
* PUSH PSW (F5) (ACC) UND (F) IN DEN STACK
* PUSH H (E5) (RP H,L) IN DEN STACK
* PUSH D (D5) (RP D,E) IN DEN STACK
* PUSH B (C5) (RP B,C) IN DEN STACK

* POP B (C1) RP B,C MIT DEM STACK TOP LADEN
* POP D (D1) RP D,E MIT DEM STACK TOP LADEN
* POP H (E1) RP H,L MIT DEM STACK TOP LADEN
* POP PSW (F1) ACC UND REG F MIT DEM STACK TOP LADEN

- ACHTUNG: AUF AUSGEWOGENHEIT BEI PUSH UND POP ACHTEN
- BEIM 8080/8085 BAUT SICH DER STACK VON OBEN NACH UNTEN AUF

2.4 FLAGs

- VIER ZUSTANDSSIGNALE IM REGISTER F KANN MAN SOFTWAREMÄSSIG ABFRAGEN
  - C (CARRY) BIT Ø
  - P (PARITY) BIT 2
  - Z (ZERO) BIT 6
  - S (SIGN) BIT 7



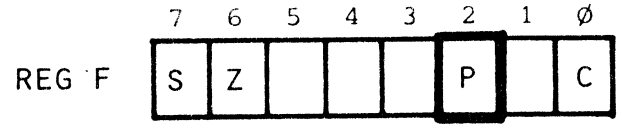
- NUR NACH EINER ARITHMETISCH/LOGISCHEN OPERATION WERDEN DIESE FLAGS GESETZT
  - BIS ZUR NÄCHSTEN ARITHMETISCH/LOGISCHEN OPERATION BLEIBEN SIE UNVERÄNDERT

- MIT DEM PROGRAMMBEISPIEL VON BLATT 2.19 SOLL DIE WIRKUNG AUF DIE FLAGS BEOBACHTET WERDEN

- DIESE VIER FLAGS WERDEN VON DER CPU BEI DER AUSFÜHRUNG BEDINGTER SPRUNGBEFEHLE ABGEFRAGT
  - SPRINGE, WENN C = 1
  - SPRINGE, WENN C ≠ 1=0
  - SPRINGE, WENN P = 1
  - SPRINGE, WENN P ≠ 1
  - SPRINGE, WENN Z = 1
  - SPRINGE, WENN Z ≠ 1
  - SPRINGE, WENN S = 1
  - SPRINGE, WENN S ≠ 1

PARITY

- PARITÄTSANZEIGE
- NACH EINER ARITHMETISCH/LOGISCHEN OPERATION IST DIE ANZAHL VON HIGH-BITS IM ERGEBNISWORT GERADE

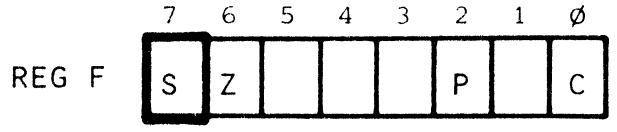


- ZÄHLBEFEHL LADEN (Ø8Ø3) = 3C
- PROGRAMM STARTEN
- MIT NXT SCHRITTWEISE WEITERSCHALTEN

Achtung:
Parity ist auch 1 bei : 00

SIGN

- VORZEICHENANZEIGE
- NACH EINER ARITHMETISCH/LOGISCHEN OPERATION IST DAS HÖCHSTWERTIGE BIT MSB IM ERGEBNISWORT AUF HIGH



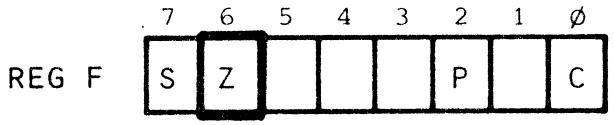
- ZÄHLBEFEHL LADEN (Ø8Ø3) = 3C
- REG A LADEN
- PROGRAMM STARTEN
- MIT NXT SCHRITTWEISE WEITERSCHALTEN

ZERO = 1, wenn

1. ● ERGEBNISANZEIGE NULL; NACH VERGLEICHSBEFEHLEN: (ACC) = XX

oder 2. ● NACH EINER ARITHMETISCH/LOGISCHEN OPERATION IST DAS ERGEBNIS-WORT ØØ ODER

oder 3. ● NACH EINEM DATENVERGLEICH SIND BEIDE BYTES GLEICH

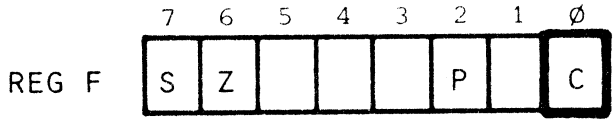


- ● ZÄHLBEFEHL LADEN (Ø8Ø3) = 3C
- ● REG A LADEN
- ● PROGRAMM STARTEN
- ● MIT NXT SCHRITTWEISE WEITERSCHALTEN

CARRY

- ÜBERTRAG NACH EINER ADDITION/SUBTRAKTION; NACH VERGLEICHS-BEFEHLEN: (ACC) < XX

- NACH EINER ADDITION/SUBTRAKTION IST EIN ÜBERLAUF ÜBER DAS MSB HINAUS ENTSTANDEN ODER
- BEI EINEM DATENVERGLEICH WAR DER AKKUMULATOR-INHALT KLEINER ALS DAS ZUM VERGLEICH HERANGEZOGENE DATENBYTE



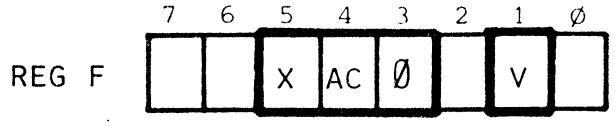
- ADDITIONSBEFEHL LADEN (Ø8Ø3) = 87
- REG A LADEN
- PROGRAMM STARTEN
- MIT NXT SCHRITTWEISE WEITERSCHALTEN; BEI NULLANZEIGE PROGRAMM NEU STARTEN

- DIREKTE CY-BEEINFLUSSUNG:

* STC (37) SET CARRY
* CMC (3F) COMPLEMENT CARRY

übrige FLAGs

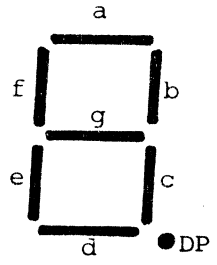
- IM REGISTER F SIND DREI WEITERE FLAGS UNTERGEBRACHT
  - V BIT 1
  - AC BIT 4
  - X BIT 5



- DAS AC-FLAG (AUXILIARY CARRY) KANN NICHT ABGEFRAGT WERDEN
  - ES GEHT BEI EINEM ÜBERTRAG VON BIT 3 NACH BIT 4 AUF HIGH
- DIE FLAGS V UND X SIND VOM HALBLEITERHERSTELLER OFFIZIELL NICHT DEFINIERT WORDEN
  - VORBEREITUNG FÜR DIE 16-BIT-ARITHMETIK
  - V: ZWEIERKOMPLEMENT-ÜBERLAUF
  - X: 16-BIT-ÜBERLAUF
- ES GIBT SOGAR BEFEHLSCODES, DIE DIESE FLAGS ABFRAGEN
  - DIESE BEFEHLSCODES SIND OFFIZIELL NICHT BESTÄTIGT WORDEN
  - DENNOCH ARBEITEN SIE MIT MASKENGLEICHEN 8085-TYPEN VERSCHIEDENER HERSTELLER EINWANDFREI

2.5 UMS-Anzeige

- JEDES SEGMENT DER ANZEIGE KANN SOFTWAREMÄSSIG EIN- UND AUSGESCHALTET WERDEN
  - EIN SEGMENT LEUCHTET, WENN DAS KORRESPONDIERENDE BIT IM ANSTEUERNDEN DATENWORT HIGH IST



DP	g	f	e	d	c	b	a

Adresse	DSP-Buffer	Anzeige	Zeichen
XBF7		DIG7	
XBF6		DIG6	
XBF5		DIG5	
XBF4		DIG4	
XBF3		DIG3	6D
XBF2		DIG2	77
XBF1		DIG1	3E
XBF0		LEDs	

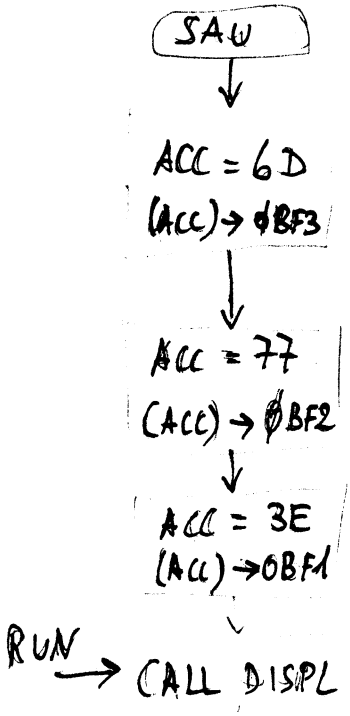
- MONITOR-UNTERPROGRAMM DISPL [011F] STEUERT MIT DEM INHALT DES DSP-BUFFERS 0BF0...0BF7 DIE ANZEIGE AN
- BEIM SPRUNG INS ANWENDERPROGRAMM LÖSCHT DER MONITOR AUTOMATISCH DEN DSP-BUFFER VON 0BF0...0BF7

Programmbeispiel

- SEGMENTMUSTER IN DER ANZEIGE ERZEUGEN

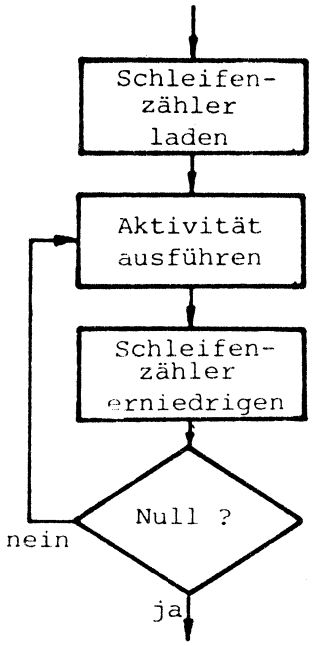
* MVI A, XX (3E XX) MOVE IMMEDIATE
  LADE REG A MIT DEM DATENBYTE XX

* STA ADDR (32 YY XX) STORE ABSOLUTE
  ÜBERSCHREIBE DEN AKKUMULATOR-INHALT IN DIE ANGEGEBENE ADRESSE
  ADRESSANGABE: ERST UNTERES, DANN OBERES BYTE



2.6 Schleifen und Abfragen

- UM EINE BESTIMMTE AKTIVITÄT MEHRFACH AUSZUFÜHREN, KLEIDET MAN SIE EINE ZÄHLSCHLEIFE EIN



* MVI B, XX (Ø6 XX) MOVE IMMEDIATE
LADE REG B MIT DEM DATENBYTE XX

* DCR B (Ø5) DECREMENT B
ERNIEDRIGE DEN INHALT VON REG B UM EINS

* JZ ADDR (CA YY XX) JUMP ON ZERO
SPRINGE ZUR ADRESSE XXYY, WENN DAS Z-FLAG = 1 IST

* JNZ ADDR (C2 YY XX) JUMP ON NO ZERO
SPRINGE ZUR ADRESSE XXYY, WENN DAS Z-FLAG ≠ 1 IST

- MONITOR-UNTERPROGRAMM DISPL [Ø11F] HAT EINE LAUFZEIT VON 8 ms BENUTZT DIE REGISTER A, D UND E
- MONITOR-UNTERPROGRAMM CLRDS [ØØ29] LÖSCHT DEN DSP-BUFFER VON ØBFØ...ØBF7

Adresse	Maschinencode			Label	Assemblercode	Zeile
	1.Byte	2.Byte	3.Byte			
						1
						2
						3
						4
						5
						6
						7
						8
						9
						10
						11
						12
						13
						14
						15
						16
						17
						18
						19
						20
						21
						22
						23
						24
						25

2.7 Datentransportbefehle

- DATEN WERDEN OHNE MODIFIKATION VON EINER STELLE IM SYSTEM ZUR ANDEREN TRANSPORTIERT
  - VON EINEM REGISTER ZUM ANDEREN
  - ZWISCHEN REGISTER UND SPEICHER
  - ZWISCHEN CPU UND PERIPHERIE
- DER CPU-INTERNE DATENTRANSPORT GESCHIEHT MIT DEM MOV-BEFEHLEN
  - MOV r,r: TRANSPORTIERE DATEN VOM QUELL- INS ZIELREGISTER
  - MOV r,XX: TRANSPORTIERE DIE DATEN XX INS ANGEGEBENE REGISTER
  - IM BEFEHL IST DAS ZIEL IMPLIZIT ANGEGEBEN
- BEIM DATENTRANSPORT AUS DER BZW. IN DIE CPU MUSS DAS ZIEL EXPLIZIT ANGEGEBEN WERDEN
  - BEIM SPEICHERVERKEHR ALS 16-BIT SPEICHERADRESSE

    STA Ø93Ø (32 3Ø Ø9): ÜBERSCHREIBE (ACC) NACH Ø93Ø
    LDA Ø93Ø (3A 3Ø Ø9): LADE DEN ACC MIT (Ø93Ø)

  - BEIM E/A-VERKEHR ALS 8-BIT-PORTADRESSE

    OUT 21 (D3 21): ÜBERSCHREIBE (ACC) IN DEN PORT 21
    IN 21 (DB 21): LADE DEN ACC MIT DEN DATEN VON PORT 21
- DIE SPEICHERADRESSE ERSCHEINT 16 BIT BREIT AUF DEM ADRESSBUS
- DIE 8-BIT-PORTADRESSE ERSCHEINT DUPLIZIERT AUF DEM ADRESSBUS

E/A-Verkehr

- EIN E/A-KANAL KANN HARDWAREMÄSSIG WIE EINE SPEICHERSTELLE ANGESCHLOSSEN SEIN
  - MEMORY MAPPED I/O
  - DANN KANN DER DATENTRANSPORT NUR PER SPEICHERBEFEHL ERFOLGEN

- IST EIN E/A-KANAL HARDWAREMÄSSIG ALS PERIPHERIEBAUSTEIN ANGESCHLOSSEN
  - I/O MAPPED I/O
  - KANN DER DATENTRANSPORT NUR PER E/A-BEFEHL ERFOLGEN

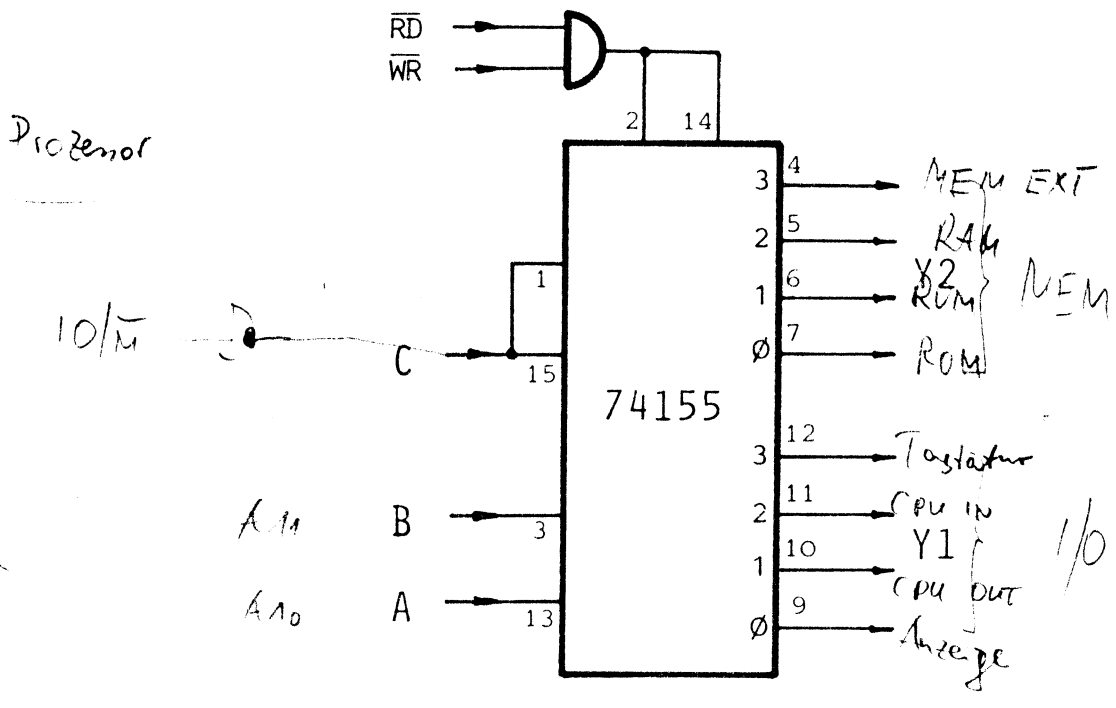
- DATENTRANSPORTBEFEHLE BEEINFLUSSEN DIE IO/$\overline{M}$-LEITUNG AM PROZESSOR
  - LOW BEI SPEICHERBEZOGENEN BEFEHLEN
  - HIGH BEI EIN/AUSGABE-BEFEHLEN

- SCHREIB- UND LESE-PULS $\overline{WR}$ BZW. $\overline{RD}$ WERDEN IN JEDEM FALL ERZEUGT

- BEIM UMS SIND BEIDE I/O-TECHNIKEN REALISIERT WORDEN

Adreßdecodierung

- EIN STANDARDBAUSTEIN ZUR ADRESSDECODIERUNG UND CS-ERZEUGUNG IST DER 74 155
  - ER HAT INTERN ZWEI EINS-AUS-VIER-DECODER
  - ER HAT AUSSERDEM ZWEI FREIGABELEITUNGEN



A UND B WÄHLEN EINEN DER VIER AUSGÄNGE Ø...3 AN

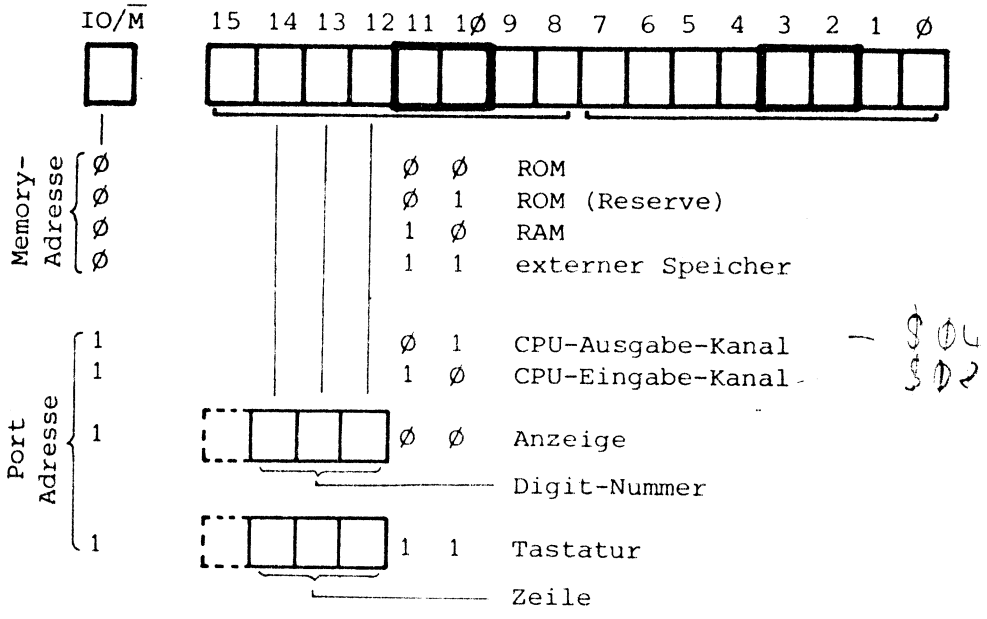
C WÄHLT GRUPPE Y1 ODER Y2 AN

ALLE AUSGÄNGE AKTIV LOW

BAUSTEIN-AKTIVIERUNG: LOW AN DEN FREIGABE-EINGÄNGEN

2.7.1 UMS-Portadressierung

- BEIM UMS LIEGEN DIE DECODIERER-EINGÄNGE A UND B AN DEN ADRESSBITS A1Ø UND A11
- EINGANG C WIRD VON DER IO/$\overline{M}$-LEITUNG UMGESCHALTET

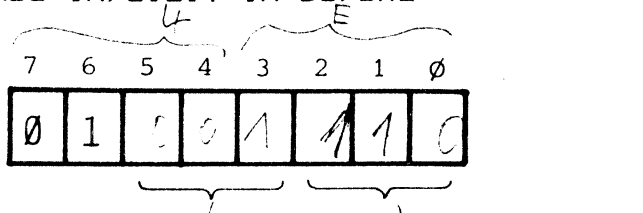


Digit X der Anzeige

- ANSTEUERUNG DER ANZEIGE PER OUT-BEFEHL: OUT XØ (D3 XØ)
  - UNTERES NIBBLE IMMER Ø
  - OBERES NIBBLE: DIGIT-NR.

MOV-Befehle

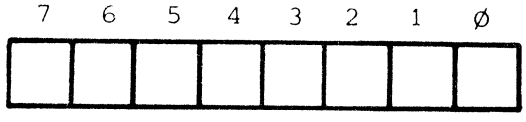
- DATENTRANSPORTBEFEHLE VON REGISTER ZU REGISTER
  - JEDES DATENREGISTER KANN ZIEL ODER QUELLE SEIN
  - ZIEL- UND QUELLANGABE IMPLIZIT IM BEFEHL
  - EINWORTBEFEHLE



- REGISTER-CODIERUNG:

Ø	Ø	Ø	-	REG B
Ø	Ø	1	-	REG C
Ø	1	Ø	-	REG D
Ø	1	1	-	REG E
1	Ø	Ø	-	REG H
1	Ø	1	-	REG L
1	1	Ø	-	m(SPEICHER)
1	1	1	-	REG A

- BEISPIEL:



2.7.2 Indirekte Adressierung

- BEIM DATENTRANSPORT ZWISCHEN CPU UND SPEICHER MUSS DIE 16-BIT-SPEICHERADRESSE EXPLIZIT ANGEGEBEN WERDEN
  - DREIWORTBEFEHL MIT DER ADRESSE IN BYTE 2 UND 3
- DIE BEFEHLE MOV r,m BZW. MOV m,r SIND DATENTRANSPORT-BEFEHLE
  - ZWISCHEN CPU UND SPEICHER
  - EINWORTBEFEHLE
- DIE 16-BIT SPEICHERADRESSE STEHT NICHT IM BEFEHL SELBST
  - DIE CPU HOLT SIE SICH AUS DEM RP H,L
  - RP H,L MUSS VORHER ENTSPRECHEND GELADEN WERDEN
  - DIESEN UMWEG BEI DER ADRESSFINDUNG NENNT MAN INDIREKTE ADRESSIERUNG
- DIE INDIREKTE ADRESSE KANN AUCH IM RP B ODER RP D STEHEN

* LDAX B (ØA) LOAD ACCUMULATOR INDIRECT

* LDAX D (1A) LOAD ACCUMULATOR INDIRECT

LADE DEN AKKUMULATOR MIT DEM INHALT DERJENIGEN SPEICHERSTELLE, DEREN ADRESSE IM ANGEGEBENEN REGISTERPAAR STEHT

* STAX B (Ø2) STORE ACCUMULATOR INDIRECT

* STAX D (12) STORE ACCUMULATOR INDIRECT

ÜBERSCHREIBE DEN AKKUMULATOR-INHALT IN DIEJENIGE SPEICHERSTELLE, DEREN ADRESSE IM ANGEGEBENEN REGISTERPAAR STEHT

* LXI RP, YYXX LOAD REGISTER PAIR IMMEDIATE

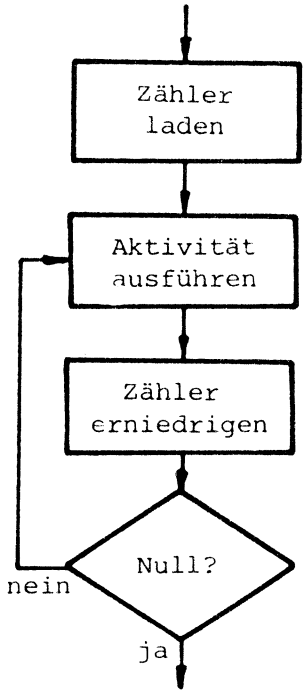
DREIWORTBEFEHL ZUM DIREKTEN LADEN EINES REGISTERPAARES
ADRESSANGABE: ERST UNTERES, DANN OBERES BYTE

2.8 Ausführungszeiten

- IN DER REGEL IST DER MIKROCOMPUTER-SYSTEMTAKT VOM QUARZ ABGELEITET
- DER SYSTEMTAKT WIRD DURCH TEILUNG DER OSZILLATORFREQUENZ GEWONNEN
  - 8080A: TEILUNG DURCH NEUN;
  - 8085A: TEILUNG DURCH ZWEI;
- DIE AUSFÜHRUNGSZEIT FÜR JEDEN BEFEHL IST GENAU DEFINIERT
  - SIE REICHT VON 4...18 TAKTZYKLEN T
- MANCHE BEFEHLE HABEN UNTERSCHIEDLICHE AUSFÜHRUNGSZEITEN
  - JNZ ADDR BEDINGTER SPRUNG BRAUCHT
    - 7 T BEI Z = 1
    - 10 T BEI Z ≠ 1
- DENNOCH KANN MAN MIT ENTSPRECHENDER SORGFALT DIE LAUFZEIT VON PROGRAMMEN ODER TEILEN DARAUS GENAU ANGEBEN
- ES IST SOGAR MÖGLICH, DEFINIERTE ZEITFUNKTIONEN PER PROGRAMM ZU REALISIEREN
  - IN DER PRAXIS HAT EIN MIKROCOMPUTER SINNVOLLERES ZU TUN ALS ZEIT ZU VERSCHWENDEN
  - FÜR ZEITFUNKTIONEN GIBT ES SPEZIELLE TIMER-BAUSTEINE

Software-Zeitfunktionen

- PRINZIP DER ZEITVERZÖGERUNG
  - BEFEHL(E) BEKANNTER AUSFÜHRUNGSZEIT WIEDERHOLEN
  - ZÄHLSCHLEIFE DEFINIERTER LAUFZEIT



- ERHÖHEN DER VERZÖGERUNGSZEIT
  - SCHLEIFEN ANEINANDERREIHEN
  - 16-BIT-ZÄHLER VERWENDEN
  - SCHLEIFEN VERSCHACHTELN

3 Angewandte Programmbeispiele

3.1 Externe Programmunterbrechungen

- UM VON AUSSERHALB DIE UNTERBRECHUNG DES LAUFENDEN PROGRAMMS ZU VERANLASSEN, STEHEN SOGENANNTE INTERRUPT-EINGÄNGE ZUR VERFÜGUNG (INTERRUPT REQUEST)
  - 8080A: 1 INTERRUPT-EINGANG
  - 8085A: 5 INTERRUPT-EINGÄNGE

- DIE CPU AKZEPTIERT NUR DANN INTERRUPTS, WENN DAS INTERNE INTERRUPT-ENABLE-FLIPFLOP (IE-FF) SOFTWAREMÄSSIG GESETZT WORDEN IST

* EI (FB) ENABLE INTERRUPT
AKZEPTIEREN VON INTERRUPTS VON DER AUSFÜHRUNG DES NÄCHSTEN BEFEHLS AN

* DI (F3) DISABLE INTERRUPT
VON DER DI-AUSFÜHRUNG AB WERDEN KEINE INTERRUPTS MEHR AKZEPTIERT

- EINEM INTERRUPT WIRD ERST DANN STATTGEGEBEN, WENN DER AUGENBLICKLICHE BEFEHL AUSGEFÜHRT WORDEN IST

- STATTGEBEN EINER UNTERBRECHUNGS-ANFORDERUNG HEISST
  - AUGENBLICKLICHEN PC-STAND IN DEN STACK LADEN
  - AUTOMATISCHES RÜCKSETZEN DES INTERRUPT-ENABLE-FLIPFLOPS
  - $\overline{\text{INTA}}$-SIGNAL AUSGEBEN UND GLEICHZEITIG DAMIT DIE INFORMATION VOM DATENBUS EINLESEN, DECODIEREN UND AUSFÜHREN (IN DER REGEL IST DAS EIN SPRUNGBEFEHL ZUR INTERRUPT-SERVICE-ROUTINE)

Interrupt Service Routine

- AM ENDE DER INTERRUPT-SERVICE-ROUTINE ISR STEHT IMMER DER RET-BEFEHL
  - DER VOR DEM EINSPRUNG IN DIE ISR GERETTETE PC-STAND WIRD AUS DEM STACK ZURÜCKGEHOLT
  - DIE PROGRAMMAUSFÜHRUNG WIRD AN DER ZUVOR VERLASSENEN STELLE FORTGESETZT

- EIN WEITERER INTERRUPT IST ERST DANN WIEDER MÖGLICH, WENN DAS IE-FF ERNEUT SOFTWAREMÄSSIG GESETZT WORDEN IST

- BEGINN JEDER ISR: RETTEN ALLER REGISTERINHALTE
  - PUSH PSW (F5)
  - PUSH H (E5)
  - PUSH D (D5)
  - PUSH B (C5)

- ENDE JEDER ISR: ZURÜCKHOLEN ALLER REGISTERINHALTE
  - POP B (C1)
  - POP D (D1)
  - POP H (E1)
  - POP PSW (F1)

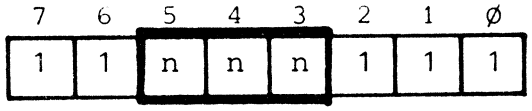
- VOR DEM RÜCKSPRUNG AUS DER ISR WEITERE INTERRUPTS WIEDER FREIGEBEN
  - EI (FB)

3.1.1 RESTART-Befehle

- ZUSAMMEN MIT DEM $\overline{\text{INTA}}$-SIGNAL SCHALTET DIE CPU DEN DATENBUS IN DEN HOCHOHMIGEN ZUSTAND
- DIE UNTERBRECHENDE STELLE SCHALTET NUN DENJENIGEN BEFEHL AUF DEN DATENBUS, DER DIE CPU ZUM SPRUNG IN DIE INTERRUPT-SERVICE-ROUTINE VERANLASST (INTERRUPT-VEKTOR)

* RST n RESTART

EINWORT-SPRUNGBEFEHLE MIT FESTEN SPRUNGZIELEN



Sprungziel:

Ø	Ø	Ø	-	RST Ø	-	(C7)	-	ØØØØ
Ø	Ø	1	-	RST 1	-	(CF)	-	ØØØ8
Ø	1	Ø	-	RST 2	-	(D7)	-	ØØ1Ø
Ø	1	1	-	RST 3	-	(DF)	-	ØØ18
1	Ø	Ø	-	RST 4	-	(E7)	-	ØØ2Ø
1	Ø	1	-	RST 5	-	(EF)	-	ØØ28
1	1	Ø	-	RST 6	-	(F7)	-	ØØ3Ø
1	1	1	-	RST 7	-	(FF)	-	ØØ38

- ANSTELLE DER RST-BEFEHLE KANN AUCH JEDER ANDERE BEFEHL AUF DEN DATENBUS GESCHALTET WERDEN
  - BEI MEHRWORTBEFEHLEN (Z.B. JMP) LIEST DIE CPU AUTOMATISCH ALLE ZUM BEFEHL GEHÖRENDEN BYTES EIN
- IN 8080-SYSTEMEN ERZEUGT DER 8228 DAS $\overline{\text{INTA}}$-SIGNAL
  - LEGT MAN DIESEN AUSGANG (PIN 23 AM 8228) ÜBER 1 kΩ AN +12 V , WIRD AUTOMATISCH EIN RST-7-BEFEHL ERZEUGT
- EIN INTERRUPT-CONTROLLER (8259)KANN MIT ACHT VERSCHIEDENEN ISR-SPRUNGADRESSEN GELADEN WERDEN

* HLT (76) HALT

SOFTWAREMÄSSIGES ANHALTEN DES PROZESSORS BIS ZUM NÄCHSTEN INTERRUPT ODER RESET

3.1.2 8085-Interrupt-Struktur

- DIE BISHER VORGESTELLTE INTERRUPT-STRUKTUR GILT SOWOHL FÜR DEN 8080A ALS AUCH FÜR DEN 8085A

- DER 8085 BESITZT AUSSERDEM VIER WEITERE INTERRUPT-EINGÄNGE MIT FEST ZUGEORDNETEN SPRUNGZIELEN
  - TRAP (PIN 6) - ØØ24
  - RST7.5 (PIN 7) - ØØ3C
  - RST6.5 (PIN 8) - ØØ34
  - RST5.5 (PIN 9) - ØØ2C

- DIESE INTERRUPT-EINGÄNGE BESITZEN EINE PRIORITÄTSSTRUKTUR
  - TRAP HAT DIE HÖCHSTE
  - RST5.5 DIE NIEDRIGSTE PRIORITÄT

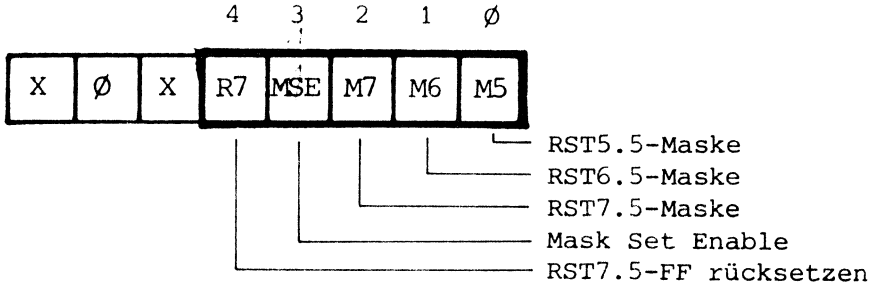
- DER INTR-EINGANG (PIN 10 AM 8085) LIEGT IN DER PRIORITÄTS-STRUKTUR UNTER RST5.5

- MIT AUSNAHME DES TRAP-INTERRUPTS SIND DIE 8085-INTERRUPTS MASKIERBAR (RESTART-INTERRUPT-MASKE)

3.1.3 RESTART-Interrupt-Maske

* SIM (3Ø) SET INTERRUPT MASK

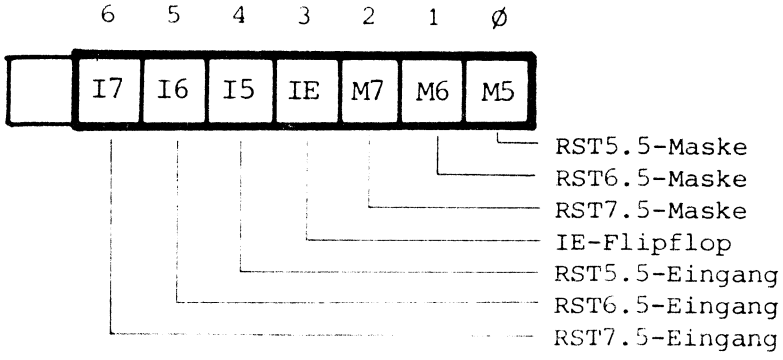
DIE AKKUMULATOR-BITS Ø...4 WERDEN INS INTERRUPT-MASKEN-REGISTER GELADEN



- DAS MASKIEREN EINES INTERRUPTS (MASKEN-BIT=1) SPERRT DEN BETREFFENDEN INTERRUPT
- DAS SETZEN ODER LÖSCHEN EINER MASKE ERFORDERT HIGH-POTENTIAL IN BIT 3
- HIGH-POTENTIAL IN BIT 4 SETZT DAS RST7.5-FF ZURÜCK, GLEICHGÜLTIG OB RST7.5 MASKIERT IST ODER NICHT
- DAS RESET-SIGNAL AN DER CPU SETZT DAS IE-FF SOWIE ALLE INTERRUPT-MASKEN UND SPERRT DAMIT SÄMTLICHE INTERRUPTS

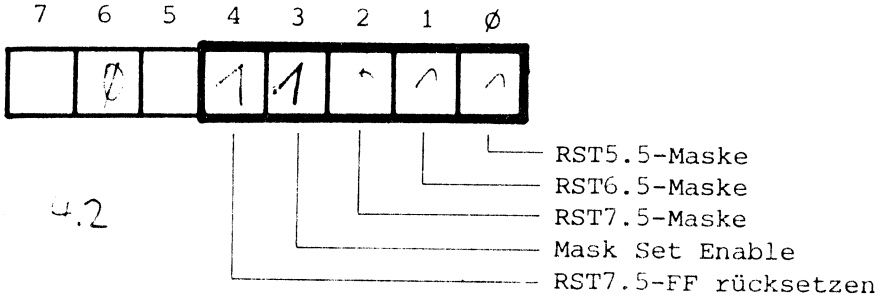
* RIM (2Ø) READ INTERRUPT MASK

ZUSTAND DER INTERRUPT-LEITUNGEN UND DIE INTERRUPT-MASKE WERDEN IN DIE AKKUMULATOR-BITS Ø...6 ÜBERTRAGEN



3.1.4 RST 7.5-Interrupt

- RST7.5-INTERRUPT ERMÖGLICHEN, RST5.5/6.5 SPERREN
  - INTERRUPT-MASKE



- DURCH DAS LADEN DER INTERRUPT-MASKE 3 INS MASKENREGISTER KANN DIE CPU RST7.5-INTERRUPTS AKZEPTIEREN
  - LADEN DES MASKEN-REGISTERS ÜBER DEN AKKUMULATOR: SIM (3Ø)
  - ZUSÄTZLICH SETZEN DES IE-FF ERFORDERLICH: EI (FB)

ISR-Programmbeispiel

- BEIM STATTGEBEN EINES RST7.5-INTERRUPTS SPRINGT DIE CPU ZUR FESTEN ZIELADRESSE ØØ3C
  - DORT IST IM UMS-MONITOR EIN SPRUNG NACH Ø83C VORGESEHEN

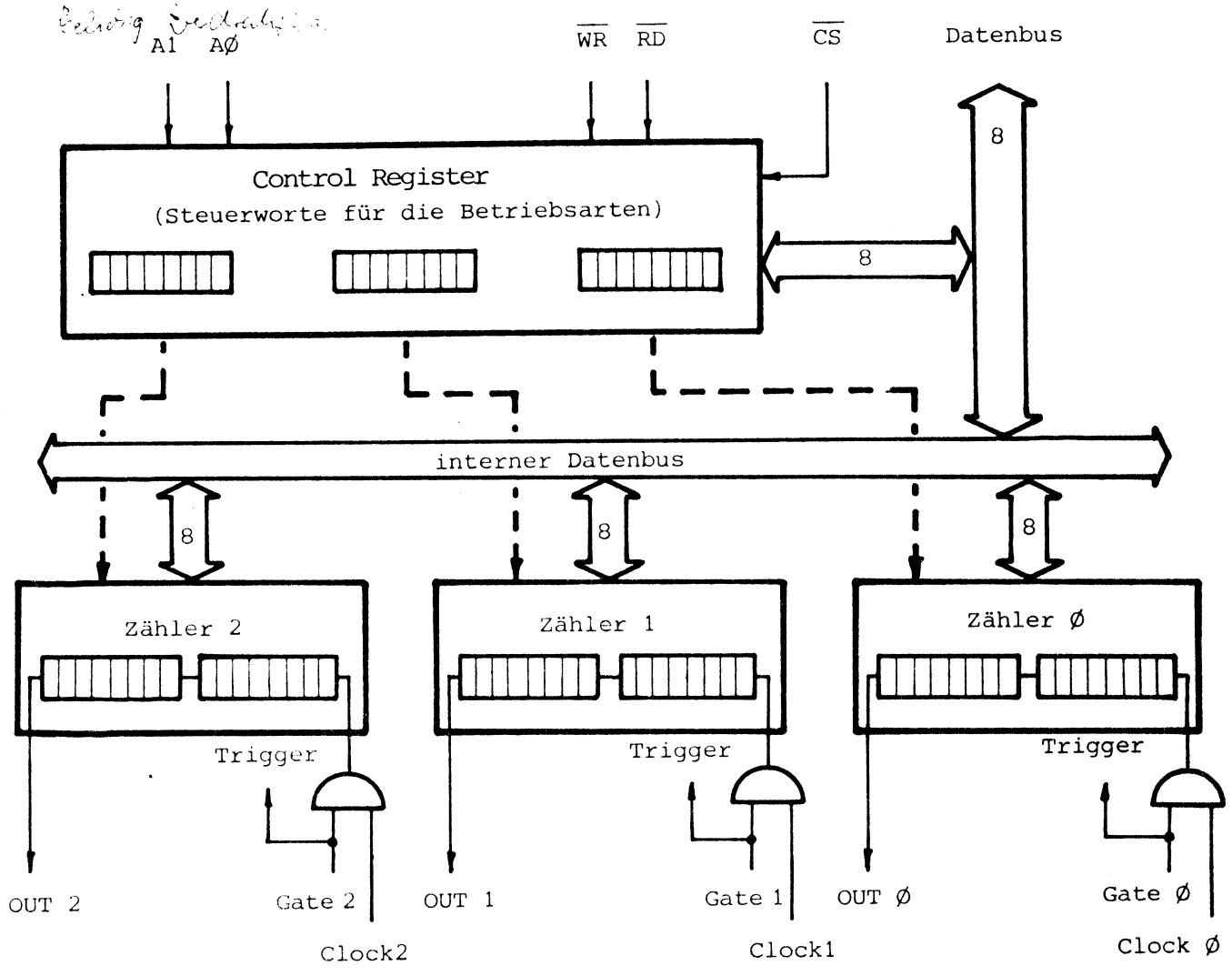
- AKUSTISCHE ANZEIGE EINER INTERRUPT-BEDIENUNG
  - LAUTSPRECHER ÜBER VORWIDERSTAND AN +5 V UND OFFENEN KOLLEKTOR-AUSGANG DER CPU ANSCHLIESSEN
  - ALS INTERRUPT-SERVICE-ROUTINE TONFREQUENZERZEUGUNG AUFRUFEN

- MONITOR-UNTERPROGRAMM SOUND [Ø2A1] ERZEUGT TONFREQUENZ-SIGNAL AM SERIELLEN AUSGANGSKANAL DER CPU

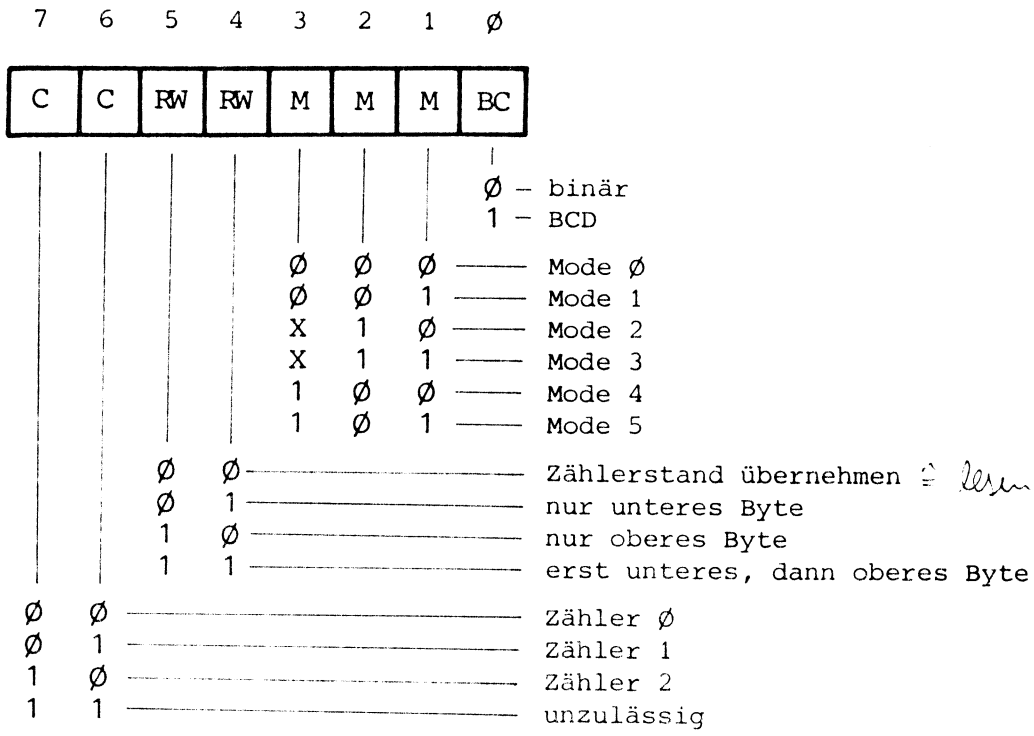
3.2 Timer-Baustein 8253

- ENTHÄLT DREI UNABHÄNGIGE 16-BIT-RÜCKWÄRTSZÄHLER
- JEDER ZÄHLER BESITZT
  - TAKTEINGANG CLK
  - FREIGABE-EINGANG GATE
  - AUSGANG OUT
- VERHALTEN DES ZÄHLER-AUSGANGS PROGRAMMIERBAR
  - IMPULSERZEUGUNG
  - TEILER
  - RECHTECKGENERATOR
  - TRIGGERSIGNAL-ERZEUGUNG
- BCD- ODER BINÄRES ZÄHLVERHALTEN
- DAS STEUERWORT IM CONTROL REGISTER BESTIMMT DAS ZÄHLVERHALTEN
- ANWAHL DER DREI ZÄHLER UND DES CONTROL REGISTERS ÜBER ZWEI ADRESSLEITUNGEN
- DER ZÄHLERSTAND KANN IM BETRIEB GELESEN WERDEN

Blockschaltbild



3.2.1 8253-Steuerwort



- LADEN DES STEUERWORTES INS CONTROL REGISTER
- JEDER 16-BIT-ZÄHLER KANN WIE FOLGT GELADEN/GELESEN WERDEN
  - NUR DAS UNTERE BYTE (OBERES AUTOMATISCH NULL)
  - NUR DAS OBERE BYTE (UNTERES AUTOMATISCH NULL)
  - NACHEINANDER ERST UNTERES, DANN OBERES BYTE

3.2.2 Timer-Modes

- MODE Ø: IMPULSERZEUGUNG (SOFTWARE-AUSLÖSUNG)

  LOW-IMPULS VON N TAKTPERIODEN
  BEGINN: LADEN DES ZÄHLERSTANDES (EINMALIG)

- MODE 1: IMPULSERZEUGUNG (HARDWARE-AUSLÖSUNG)

  LOW-IMPULS VON N TAKTPERIODEN
  BEGINN: POSITIVE FLANKE AM GATE (EINMALIG)

- MODE 2: TEILER DURCH N

  LOW-IMPULS VON EINER TAKTPERIODENDAUER
  ZYKLISCH ALLE N TAKTPERIODEN

- MODE 3: RECHTECKGENERATOR

  RECHTECKSIGNAL MIT $^{N}/_{2}$ BZW. $^{(N+1)}/_{2}$ -HIGH-ZEITEN

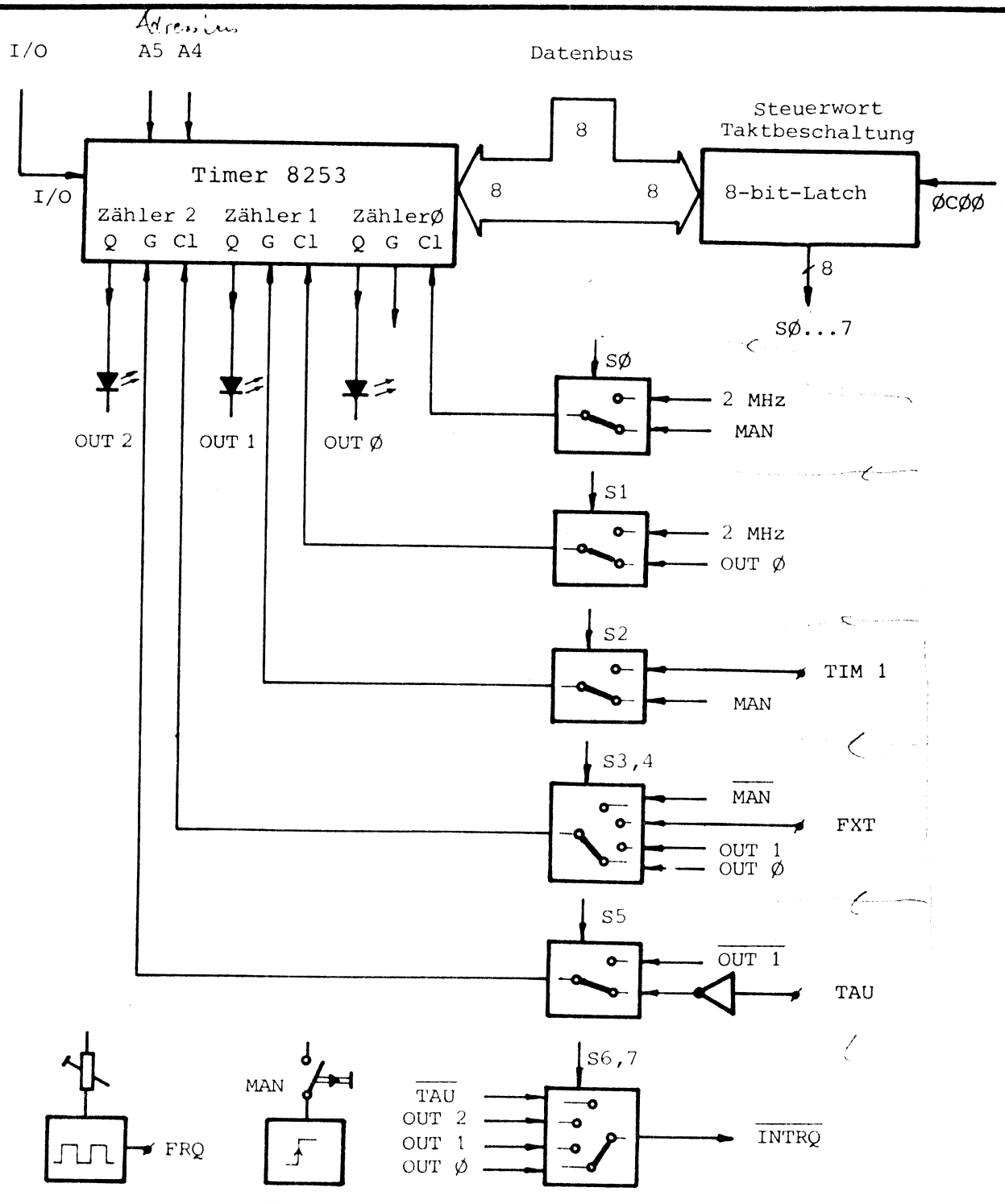
- MODE 4: TRIGGER-IMPULS (SOFTWARE-AUSLÖSUNG)

  LOW-IMPULS VON EINER TAKTPERIODENDAUER NACH N TAKTPERIODEN
  STARTZEITPUNKT: LADEN DES ZÄHLERSTANDES

- MODE 5: TRIGGER-IMPULS (HARDWARE-AUSLÖSUNG)

  LOW-IMPULS VON EINER TAKTPERIODENDAUER NACH N TAKTPERIODEN
  STARTZEITPUNKT: POSITIVE FLANKE AM GATE

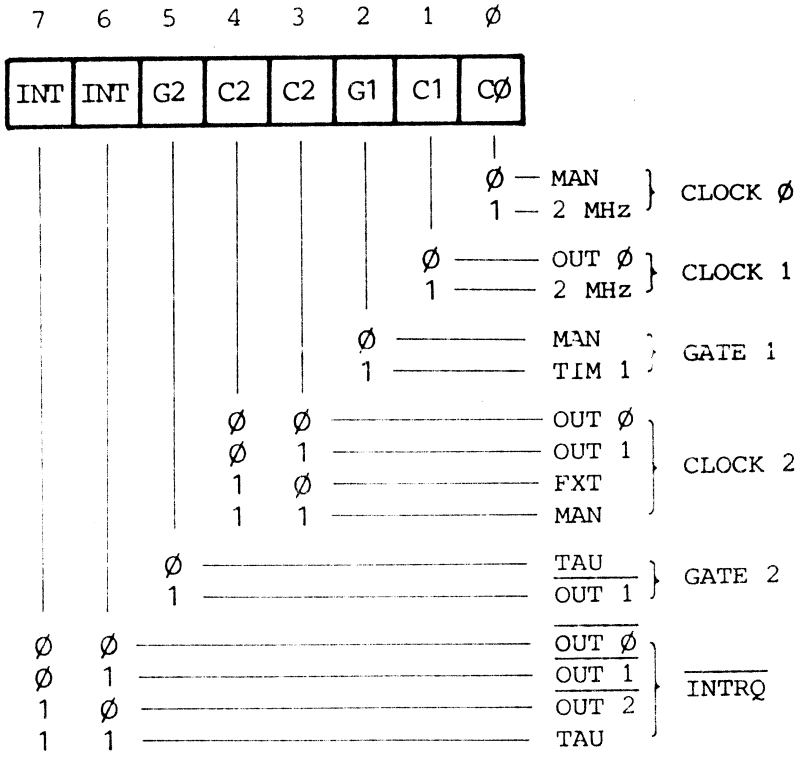
3.3 Timer-Karte 854



Organisation

- TIMER-ADRESSIERUNG ÜBER I/O MAPPED I/O
  - ZÄHLER Ø: PORTADRESSE Ø1
  - ZÄHLER 1: PORTADRESSE 11
  - ZÄHLER 2: PORTADRESSE 21
  - STEUERWORT: PORTADRESSE 31

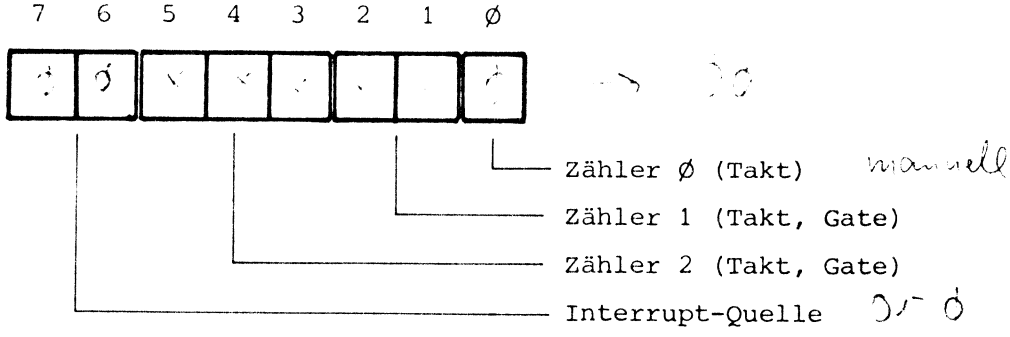
- TAKTBESCHALTUNG ÜBER MEMORY MAPPED I/O
  - ADRESSE ØCØØ



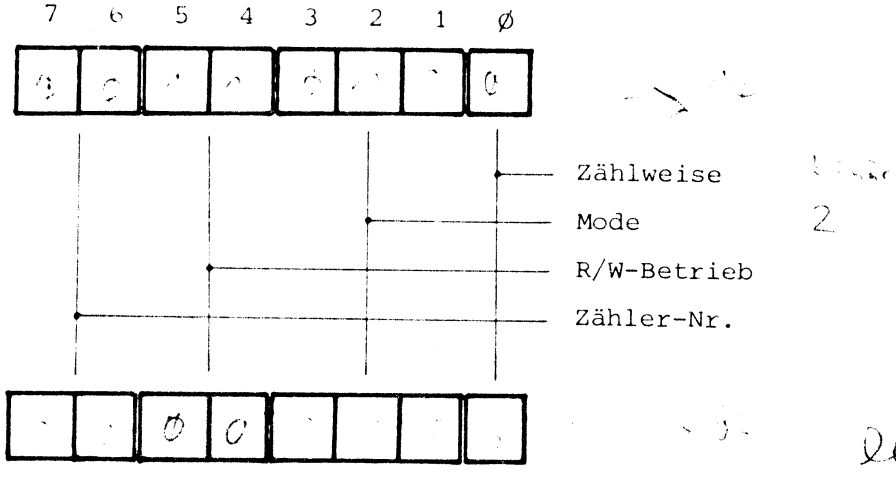
3.3.1 Ereigniszählung

- MANUELLE TAKTIMPULSE ZÄHLEN UND ZÄHLERSTAND ANZEIGEN
  - NUR 8-BIT-WORTLÄNGE (UNTERES BYTE)
  - BINÄRES ZÄHLEN (AUFWÄRTS)
  - DEFINIERTER ZÄHLBEGINN BEI ØØ

- STEUERWORT FÜR TAKTBESCHALTUNG FESTLEGEN
  - LADEN ÜBER STA ØCØØ



- STEUERWORT FÜR CONTROL REGISTER FESTLEGEN
  LADEN ÜBER OUT 31



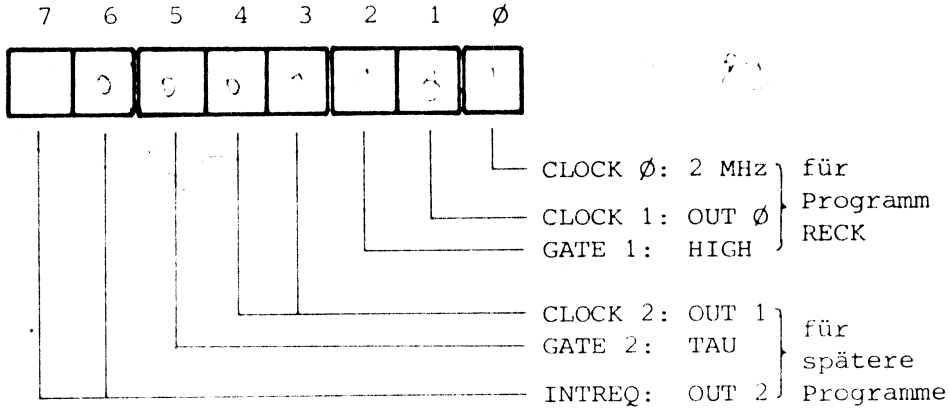
- MONITOR-UNTERPROGRAMM DISP2 [Ø372] STELLT DEN INHALT DER SPEICHERSTELLE ØBF8 IN DER ANZEIGE DAR

* CMA (2F) COMPLEMENT ACCUMULATOR
AKKUMULATOR-INHALT KOMPLEMENTIEREN

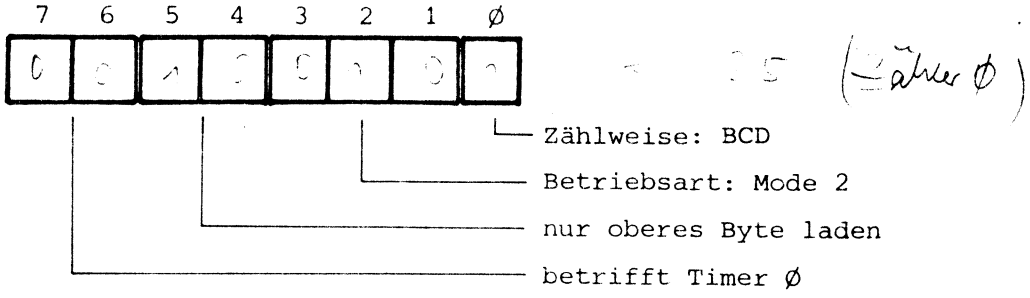
3.3.2 Teiler und Rechteckgenerator

- RECK: ERZEUGUNG EINES 1-Hz-RECHTECKSIGNALS MIT DEM TIMER
  - ZÄHLER Ø:TEILER DURCH 2000D
    0,5-µs-TAKTPERIODENDAUER ZU 1-ms-IMPULSEN VERLÄNGERN
  - ZÄHLER 1:RECHTECKGENERATOR, GETAKTET VON OUT Ø

- STEUERWORT FÜR TAKTBESCHALTUNG FESTLEGEN
  - LADEN ÜBER STA ØCØØ

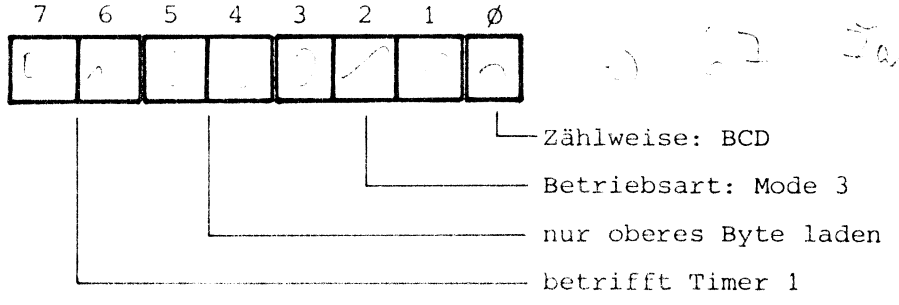


- ERSTES STEUERWORT FÜR CONTROL REGISTER FESTLEGEN
  - BETRIEBSART FÜR TIMER Ø
    LADEN ÜBER OUT 31
  - TEILER DURCH 2000D
    NUR OBERES BYTE LADEN



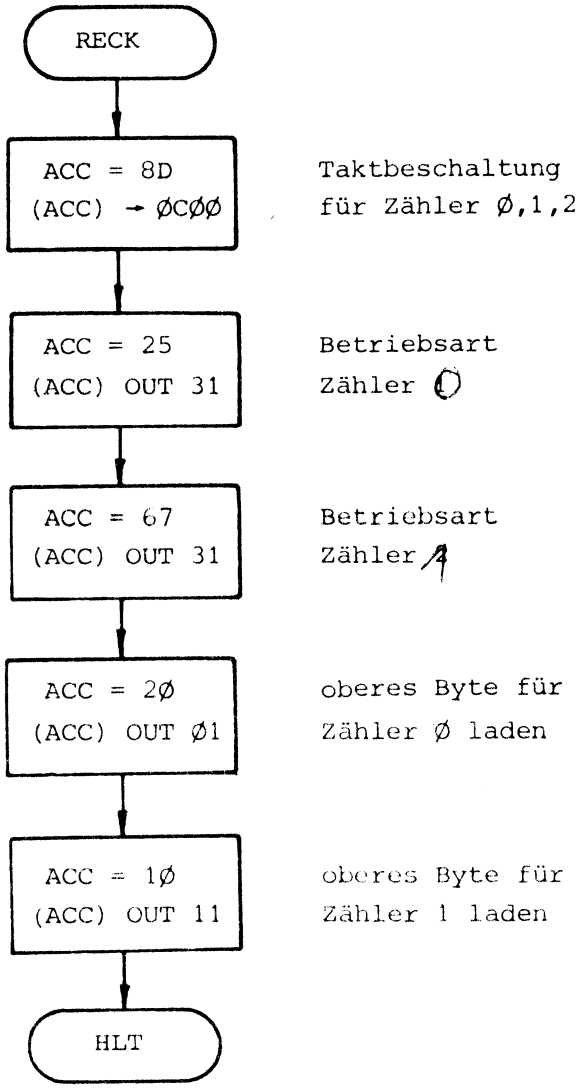
noch: Teiler und Rechteckgenerator

- ZWEITES STEUERWORT FÜR CONTROL REGISTER FESTLEGEN
  - BETRIEBSART FÜR TIMER 1
    LADEN ÜBER OUT 31
  - RECHTECKGENERATOR MIT 1000D 1-ms-TAKTIMPULSEN
    NUR OBERES BYTE LADEN



Flußdiagramm

- RECK: 1-Hz-RECHTECKSIGNAL AM ZÄHLERAUSGANG 1



- DIE LEUCHTDIODE AM ZÄHLERAUSGANG OUT 1 BLINKT IM 1-Hz-TAKT

3.3.3 Timer-Demonstration

- ZÄHLER 2 IN DEN VERSCHIEDENEN MODES BETREIBEN
  - STEUERWORT FÜR CONTROL REGISTER LADEN ÜBER OUT 31

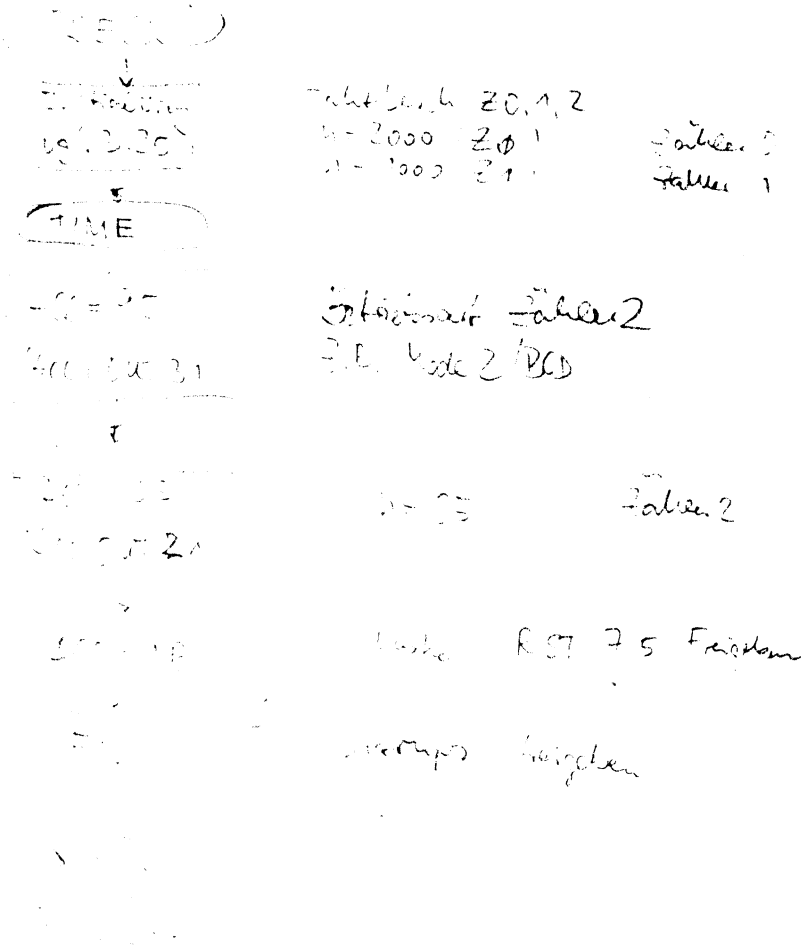
    MODE Ø: 9Ø FÜR BINÄRES ZÄHLEN (BCD: 91)
    MODE 1: 92 FÜR BINÄRES ZÄHLEN (BCD: 93)
    MODE 2: 94 FÜR BINÄRES ZÄHLEN (BCD: 95)
    MODE 3: 96 FÜR BINÄRES ZÄHLEN (BCD: 97)
    MODE 4: 98 FÜR BINÄRES ZÄHLEN (BCD: 99)
    MODE 5: 9A FÜR BINÄRES ZÄHLEN (BCD: 9B)

- ZAHLENWERT FÜR ZÄHLER 2 EINMALIG LADEN
  - NUR UNTERES BYTE
  - UNGERADE ZAHL

- TAKT FÜR ZÄHLER 2 VON OUT 1 ABLEITEN
  - .1-Hz-SIGNAL, GENERIERT VOM PROGRAMMTEIL RECK

- $\overline{\text{INTREQ}}$ VON OUT 2 ABLEITEN
  - INTERRUPT-SERVICE-ROUTINE PROGRAMMIEREN/AUFRUFEN

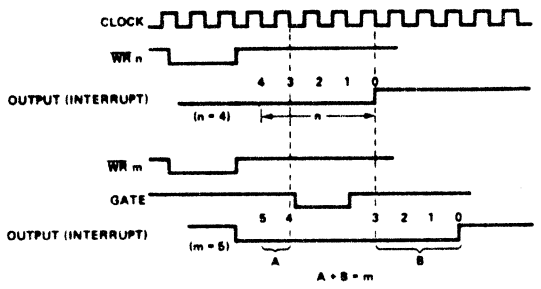
Flußdiagramm



- HARDWARE-TRIGGER-IMPULS: ANSCHLUSS „TAU" KURZZEITIG ERDEN
- LOW-PEGEL AN TAU STOPPT IN DEN MODES Ø, 2 UND 4 DAS HERUNTER-ZÄHLEN

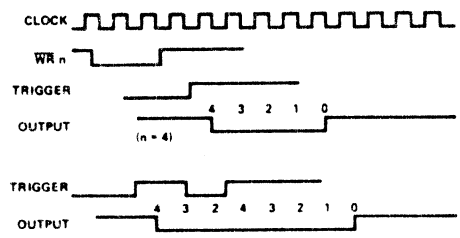
MODE Ø

MODE 0



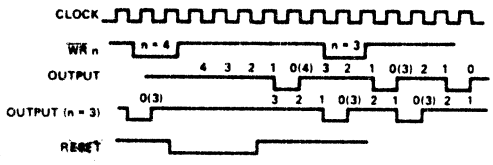
MODE 1

MODE 1



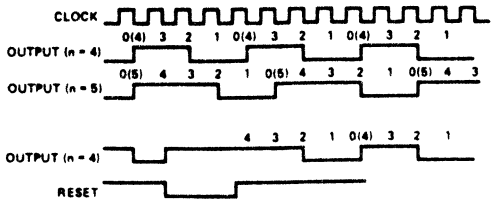
MODE 2

MODE 2



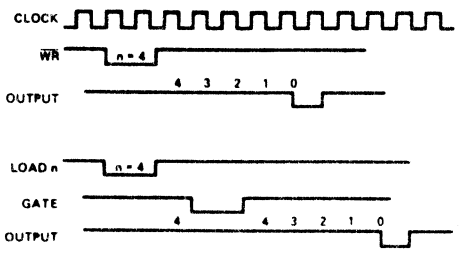
MODE 3

MODE 3



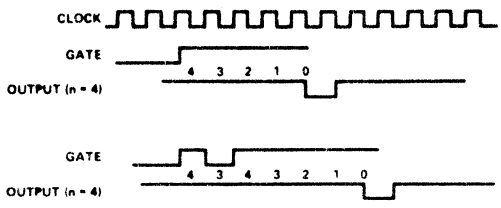
MODE 4

MODE 4



MODE 5

MODE 5



3.3.4 Frequenzmessung

- ZÄHLER 2 ZUR FREQUENZMESSUNG VERWENDEN
  - EREIGNISZÄHLUNG WÄHREND FESTER TORZEIT
  - CLOCK 2 ERHÄLT UNBEKANNTE FREQUENZ FXT, Z.B. VON FRQ
  - GATE 2 ERHÄLT FESTE TORZEIT, Z.B. 100 ms

- INTERRUPT AM ENDE DER TORZEIT
  - LIEST ERREICHTEN ZÄHLERSTAND
  - ZEIGT DAS ERGEBNIS AN
  - SETZT DEN ZÄHLER AUF DEN ANFANGSWERT

- AM ANSCHLUSS-STIFT FRQ STEHT VARIABLE FREQUENZ AN
  - EINSTELLBAR VON 0,3...3 kHz (3...0,3 ms)
  - TTL-PEGEL

3.3.5 Periodendauer-Messung

- ZÄHLER 2 ZUR PERIODENDAUER-MESSUNG VERWENDEN
  - TAKTPULSE FESTER FREQUENZ WÄHREND VARIABLER TORZEIT ZÄHLEN
  - CLOCK 2 ERHÄLT MESSFREQUENZ, Z.B. 1 kHz
  - GATE 2 ERHÄLT UNBEKANNTE PERIODENDAUER TAU, Z.B. VON FRQ

- INTERRUPT AM ENDE DER TORZEIT
  - LIEST ERREICHTEN ZÄHLERSTAND AN
  - ZEIGT DAS ERGEBNIS AN
  - SETZT DEN ZÄHLER AUF DEN ANFANGSWERT

3.4 Zählprogramme

- ZÄHLER AUFBAUEN, DER MIT FESTEM TAKT ZYKLISCH WEITERZÄHLT
  - ZÄHLERSTAND ANZEIGEN

- VERWENDETE PROGRAMMTECHNIKEN
  - HAUPTPROGRAMM MIT INTERRUPT-SERVICE-ROUTINE
  - ZEITSCHLEIFEN AUS TIMER- UND SOFTWARE-FUNKTIONEN
  - INDIREKTE ADRESSIERUNG (TABELLENUMSETZUNG)
  - MASKIEREN
  - EINZELNE BITS SETZEN, LÖSCHEN, INVERTIEREN
  - VERGLEICHSBEFEHLE
  - DEZIMALKORREKTUR

- SCHRITTWEISE INBETRIEBNAHME KOMPLEXER PROGRAMME
  - MODULARE STRUKTUR
  - HILFSPROGRAMME FÜR DIE INBETRIEBNAHME
  - PLATZHALTE-BEFEHLE

Randbedingungen

- HAUPTPROGRAMM ÜBERNIMMT ANZEIGE DES ZÄHLERSTANDES
- INTERRUPT-SERVICE-ROUTINE (ISR) ÜBERNIMMT WEITERZÄHLEN
- ZÄHLER Ø ERZEUGT ALLE 5 ms INTERRUPTS
  - ISR FÜHRT NUR ALLE 200 INTERRUPTS DAS WEITERZÄHLEN DURCH

* MOV r,m MOVE MEMORY

LADE DAS ZIELREGISTER r MIT DEM INHALT DERJENIGEN SPEICHERSTELLE m, DEREN ADRESSE IM RP H,L STEHT

* LDAX RP LOAD ACCUMULATOR INDIRECT

LADE DEN AKKUMULATOR MIT DEM INHALT DERJENIGEN SPEICHERSTELLE, DEREN ADRESSE IM ANGEGEBENEN REGISTERPAAR RP STEHT

* CPI XX (FE XX) COMPARE IMMEDIATE

VERGLEICHE DEN AKKUMULATOR-INHALT MIT DEM BYTE XX

(ACC) = XX: Z-FLAG = 1
(ACC) < XX: CY-FLAG = 1

* ANI XX (E6 XX) AND IMMEDIATE

UND-VERKNÜPFUNG DES AKKUMULATOR-INHALTS MIT DEM BYTE XX

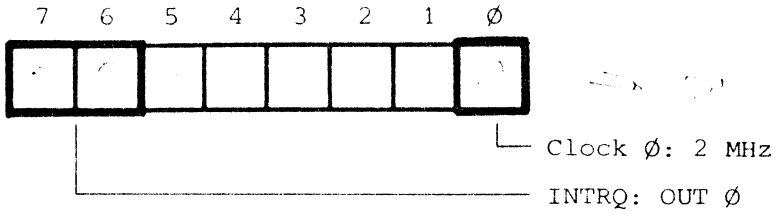
* NOP (ØØ) NO OPERATION

BEWIRKT NUR DAS WEITERZÄHLEN DES PROGRAMMZÄHLERS

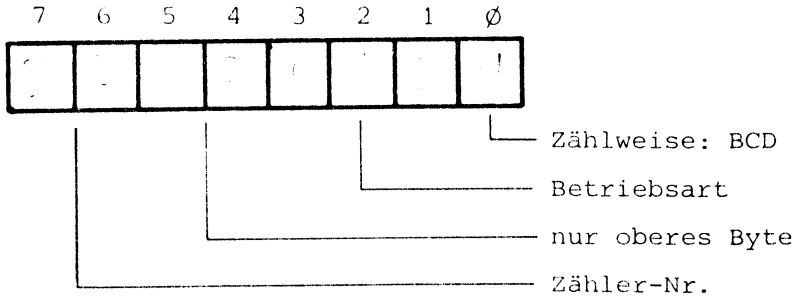
3.4.1 4-bit-Zähler

- ZÄHLEN VON Ø...F, TAKT 1 Hz

- STEUERWORT FÜR TIMER-TAKTBESCHALTUNG FESTLEGEN
  - LADEN ÜBER STA ØCØØ



- STEUERWORT FÜR CONTROL REGISTER FESTLEGEN
  - LADEN ÜBER OUT 31



3.4.2 Dezimalkorrektur

- DEZIMALES ZÄHLEN VON Ø...9, TAKT 1 Hz

* DAA (27) DECIMAL ADJUST ACCUMULATOR

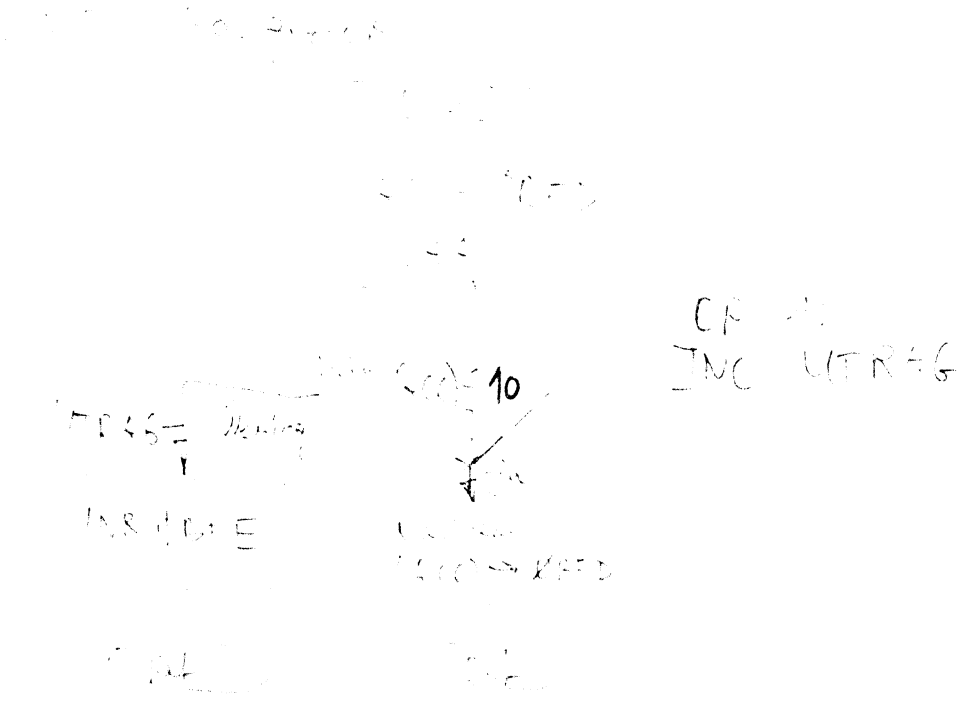
DEZIMALKORREKTUR DES AKKUMULATOR-INHALTS (UMSETZUNG IN ZWEI BCD-DIGITS)

MUSS UNMITTELBAR NACH DEM ZÄHLBEFEHL EINGESETZT WERDEN

3.4.3 Zählen mit Übertrag

- ZWEISTELLIG VON ØØ...99 ZÄHLEN, TAKT 1 Hz

- MODIFIKATIONEN
  - HAUPTPROGRAMM MUSS ZWEI ANZEIGE-STELLEN ANSPRECHEN
  - ISR-ZÄHLTAKT ERHÖHEN (SCHLEIFENZÄHLER KLEINER)



3.4.4 Bitmanipulationen

- ZUSÄTZLICH DEZIMALPUNKT EINBLENDEN

- SETZEN EINZELNER BITS
  - ODER-VERKNÜPFUNG MIT EINER MASKE, BEI DER DIE ZU SETZENDEN BITS HIGH SIND

* ORI XX (F6 XX) OR IMMEDIATE
ODER-VERKNÜPFUNG DES AKKUMULATOR-INHALTS MIT DEM BYTE XX

- LÖSCHEN EINZELNER BITS
  - UND-VERKNÜPFUNG MIT EINER MASKE, BEI DER DIE ZU LÖSCHENDEN BITS LOW SIND

* ANI XX (E6 XX) AND IMMEDIATE
UND-VERKNÜPFUNG DES AKKUMULATOR-INHALTS MIT DEM BYTE XX

- INVERTIEREN EINZELNER BITS
  - EXOR-VERKNÜPFUNG MIT EINER MASKE, BEI DER DIE ZU INVERTIERENDEN BITS HIGH SIND

* XRI XX (EE XX) EXCLUSIVE OR IMMEDIATE
EXOR-VERKNÜPFUNG DES AKKUMULATOR-INHALTS MIT DEM BYTE XX

- ACHTUNG: ORI, ANI UND XRI LÖSCHEN DAS CY-FLAG

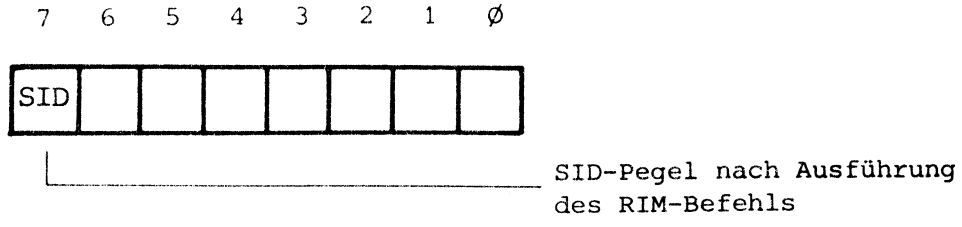
4 Serielle Datenübertragung

4.1 Serielle Ein/Ausgabe

- RESET LÖSCHT DAS SOD-LATCH
- DER RIM- BZW. SIM-BEFEHL DIENT AUSSERDEM ZUM LESEN/LADEN DER INTERRUPT-MASKE
- DER 8085 BESITZT JE EINEN TTL-KOMPATIBLEN EIN- UND AUSGANG
  - SID: SERIAL INPUT DATA LINE (PIN 5)
  - SOD: SERIAL OUTPUT DATA LINE (PIN 6)

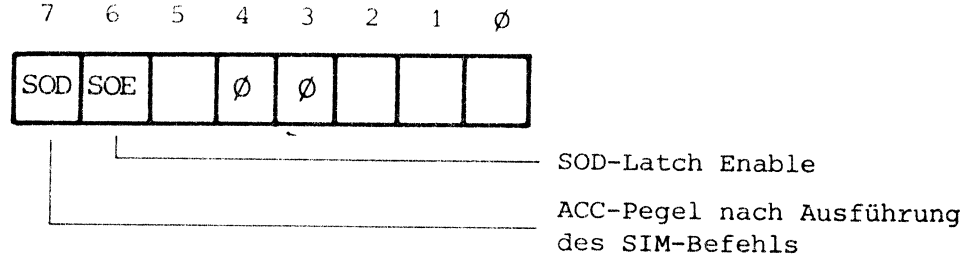
* RIM (2Ø) READ INTERRUPT MASK

DER SID-PEGEL WIRD INS AKKUMULATOR-BIT 7 ÜBERTRAGEN



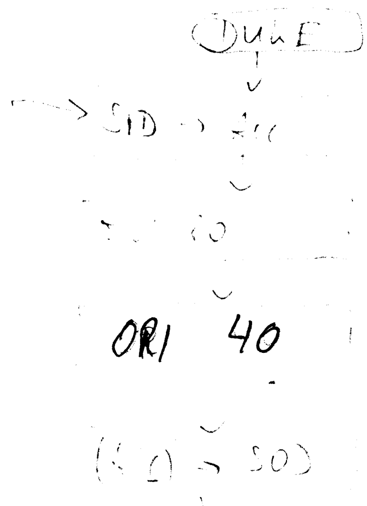
* SIM (3Ø) SET INTERRUPT MASK

DER PEGEL AUS DEM AKKUMULATOR-BIT 7 WIRD NACH SOD ÜBERTRAGEN, WENN BIT 6 DABEI AUF HIGH IST



4.1.1 Programmbeispiel

- PEGEL VON SID EINLESEN UND NACH SOD AUSGEBEN



Bit6 setzen

C800	20	START	RIM
	F680		ANI 80
	F640		ORI 40
			SIM
			JMP START

4.2 Rotationsbefehle

- ROTATIONSBEFEHLE (ROTATE) VERSCHIEBEN DEN AKKUMULATOR-INHALT ZYKLISCH UM EIN BIT LINKS BZW. RECHTS
  - UM EIN BIT AN EINE ANDERE POSITION ZU BRINGEN
  - UM EINE BINÄRZAHL MIT ZWEI ZU MULTIPLIZIEREN (BEIM LINKSSCHIEBEN)
  - UM EINE BINÄRZAHL DURCH ZWEI ZU DIVIDIEREN (BEIM RECHTSSCHIEBEN)

- ROTATIONSBEFEHLE (ROATATE) VERSCHIEBEN EIN DATENWORT ZYKLISCH UM EINE BITPOSITION
  - ES GEHT KEIN BIT „VERLOREN"

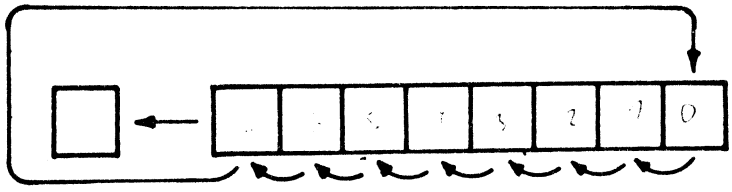
- SCHIEBEBEFEHLE (SHIFT) VERSCHIEBEN EIN DATENWORT ZYKLISCH UM EINE BITPOSITION
  - DIE BITS „FALLEN DABEI HERAUS", ES WERDEN NULLEN NACHGESCHOBEN

- IM 8080/8085-BEFEHLSSATZ GIBT ES NUR ROTATIONSBEFEHLE

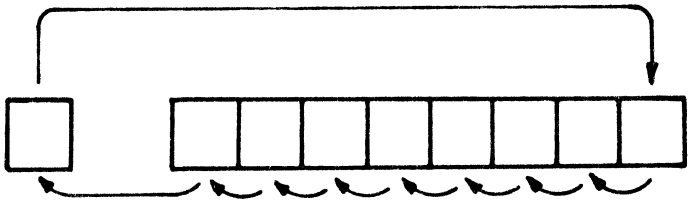
* RLC (Ø7) ROTATE LEFT
* RRC (ØF) ROTATE RIGHT
* RAL (17) ROTATE LEFT THROUGH CARRY
* RAR (1F) ROTATE RIGHT THROUGH CARRY

Einbeziehung des CY-Bits

- ROTATE LEFT
  - ZYKLISCHES VERSCHIEBEN DES AKKUMULATOR-INHALTS UM EINE BITPOSITION NACH LINKS; AUSSERDEM GELANGT BIT 7 INS CY-BIT



- ROTATE LEFT THROUGH CARRY
  - ZYKLISCHES VERSCHIEBEN DES AKKUMULATOR-INHALTS UM EINE BITPOSITION NACH LINKS, WOBEI DAS CY-BIT ALS NEUNTES BIT IN DEN KREISLAUF EINBEZOGEN WIRD



- FÜR „ROTATE RIGHT" UND „ROTATE RIGHT THROUGH CARRY" GILT DAS ENTSPRECHENDE

4.2.1 Programmbeispiel

- DATENWORT 8Ø (NUR EIN BIT AUF HIGH) ZYKLISCH IM 1-Hz-RHYTHMUS VERSCHIEBEN UND AN DIE LED-ZEILE AUSGEBEN

  WECHSELWEISE RLC/RAL BZW. RRC/RAR EINSETZEN

  ZUSTAND DES CY-BITS ANZEIGEN

- MONITOR-UNTERPROGRAMM ONSEC [Ø3B1] HAT FESTE LAUFZEIT VON EINER SEKUNDE

4.2.2 Anzeige des CY-Bits

- ZUSTAND DES CY-BITS MIT DER SOD-LEUCHTDIODE ANZEIGEN

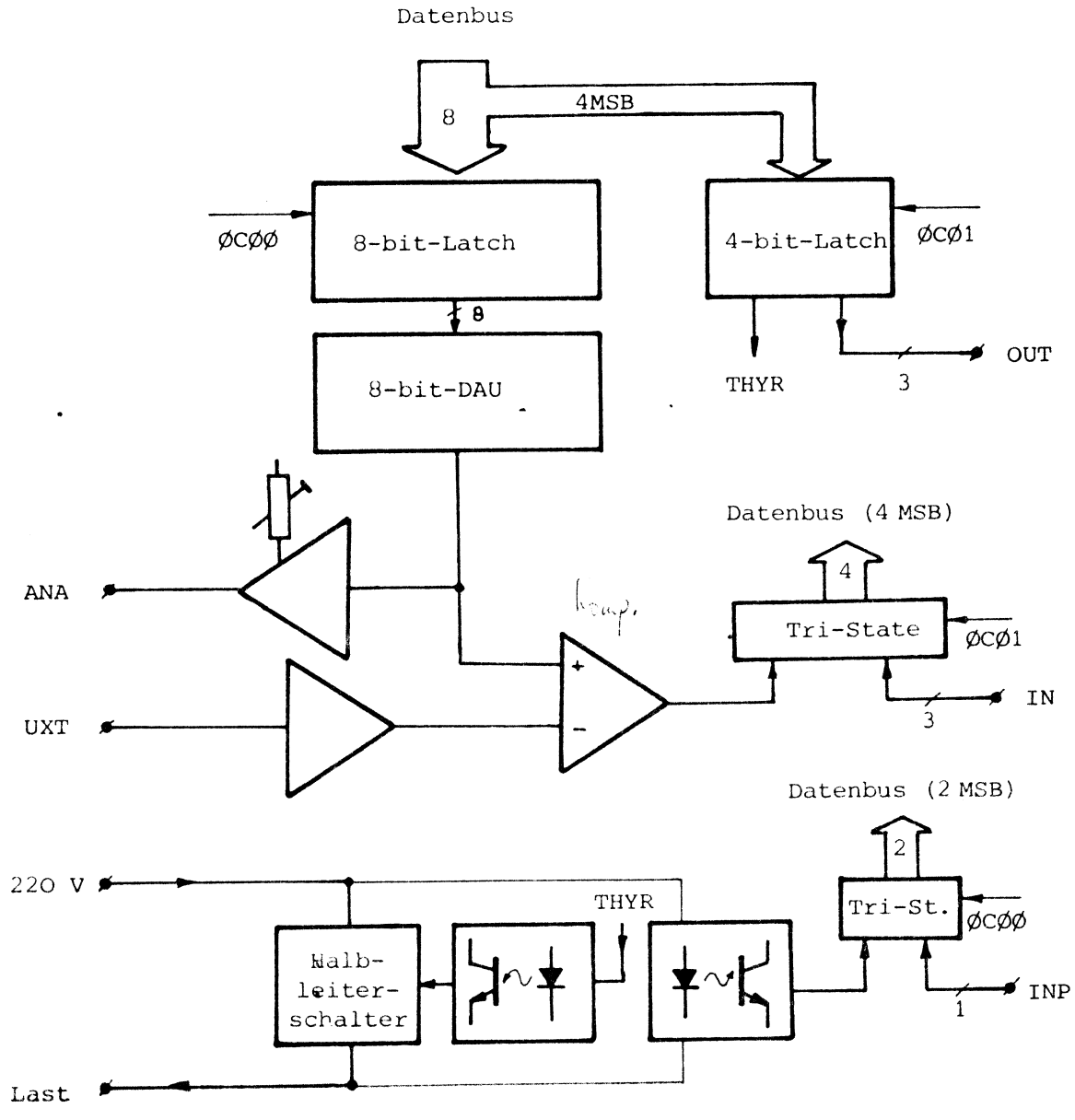
* STC (37) SET CARRY

* CMC (3F) COMPLEMENT CARRY

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
0200	3E	80		START	MVI A,80	1
02	D3	00		NEXT	AUT 00	2
04	CD	10	08		CALL CARRY	3
07	CD	B1	03		C=LL CUEC	4
0A	07				RLC	5
0B	C3	02	08		JMP NEXT	6
0E	E					7
						8
						9
						10
						11
0810	=			CARRY	PUSH PSW	12
	E	C0			MVI A,C0	13
	2	19	08		JMP LULL	14
	30					15
	=				POP PSW	16
	C9					17
	3E	40			MVI A,40	18
	C3	16	08		JMP LUS	19
	-					20
						21
						22
						23
						24
						25

5 Analoger Datenverkehr

5.1 Analog-Interface 855



EXTERNE VERSORGUNG MIT +12 V UND -5 V ERFORDERLICH

SEPARATES ANALOG-BEZUGSPOTENTIAL MÖGLICH

Organisation

- ADRESSIERUNG AUSSCHLIESSLICH ÜBER MEMORY MAPPED I/O

- DIGITAL/ANALOG-UMSETZER: 8 BIT, ADRESSE ØCØØ
  - AUSGANGSSPANNUNG EINSTELLBAR +2,5...+10,5 V
  - GEPUFFERTER ANALOGAUSGANG BIS MAXIMAL 100 mA

- KOMPARATOR: BIT 7, ADRESSE ØCØ1
  - GESCHÜTZTER UND GEPUFFERTER ANALOGEINGANG
  - UXT = -5 V...+12 V
  - KOMP = HIGH BEI ANA > ÜXT

- SCHMITT-TRIGGER-EINGÄNGE: BITS 4,5,6, ADRESSE ØCØ1
  - DREI LS-TTL-EINGÄNGE IN 4 ...IN 6
  - OBERE SCHWELLSPANNUNG 1,6 V
  - HYSTERESE 0,8 V (TEMPERATURKOMPENSIERT)

- THYRISTOR-ANSTEUERUNG: BIT 7, ADRESSE ØCØ1
  - LASTEN BIS 1 A
  - GALVANSICHE TRENNUNG ÜBER OPTOKOPPLER

- TTL-AUSGÄNGE: BITS 4,5,6, ADRESSE ØCØ1
  - DREI LS-TTL-AUSGÄNGE OUT 4 ...OUT 6

- PASENLAGE NETZSPANNUNG: BIT 7, ADRESSE ØCØØ
  - TTL-SIGNAL, GALVANISCH GETRENNT
  - ASYMMETRISCHES TASTVERHÄLTNIS

- EXTERNER SENSOR: BIT 6, ADRESSE ØCØØ
  - LICHTSCHRANKE/NÄHERUNGSSCHALTER

5.2 Digital/Analog-Umsetzung

- STANDARD 8-BIT-DAU: DAC0800/µA801
- TTL-ANSTEUERUNG
- ACHT SCHALTBARE STROMQUELLEN MIT BINÄREN WERTIGKEITEN
  1 : 2 : 4 : ... : 128
- STROM/SPANNUNGS-UMSETZUNG ÜBER OPAMP
- AUSGANGSAMPLITUDE EINSTELLBAR
  - ENDWERT +2,55 V FÜR BINÄRE STUFUNG
  - ENDWERT +10,0 V FÜR MAXIMALEN HUB
- ABGRIFF: ANA UND GND

5.2.2 Statische D/A-Umsetzung

- ENDWERT EICHEN
  - FF NACH ØCØØ AUSGEBEN, POTI ABGLEICHEN
- DIGITALWERT ÜBER TASTATUR EINGEBEN UND WIRKUNG AM VERBRAUCHER BEOBACHTEN
- MONITOR-UNTERPROGRAMM INPUT [Ø35A] LIEST BYTE VON DER TASTATUR EIN UND STELLT ES RECHTSBÜNDIG DAR; NACH NXT WIRD DAS BYTE IN DEN AKKUMULATOR ÜBERNOMMEN

5.3 Analog/Digital-Umsetzung

- DIGITALWERT AN DAU AUSGEBEN
- ANALOGSPANNUNG UXT UND DAU-WERT VERGLEICHEN
- KOMPARATOR-AUSGANG ABFRAGEN
- DIGITALWERT BEI BEDARF MODIFIZIEREN
- AUSGABE- UND ABFRAGE-ZYKLUS WIEDERHOLEN

5.3.1 Fensterkomparator

- VERGLEICH EINER EXTERNEN SPANNUNG UXT MIT ZWEI FESTEN GRENZWERTEN UND ERGEBNISANZEIGE

- ANALOGAUSGANG ANA BEI BEREICHSENDE ABGLEICHEN 5V
  - FF NACH ØCØØ AUSGEBEN, POTI ABGLEICHEN

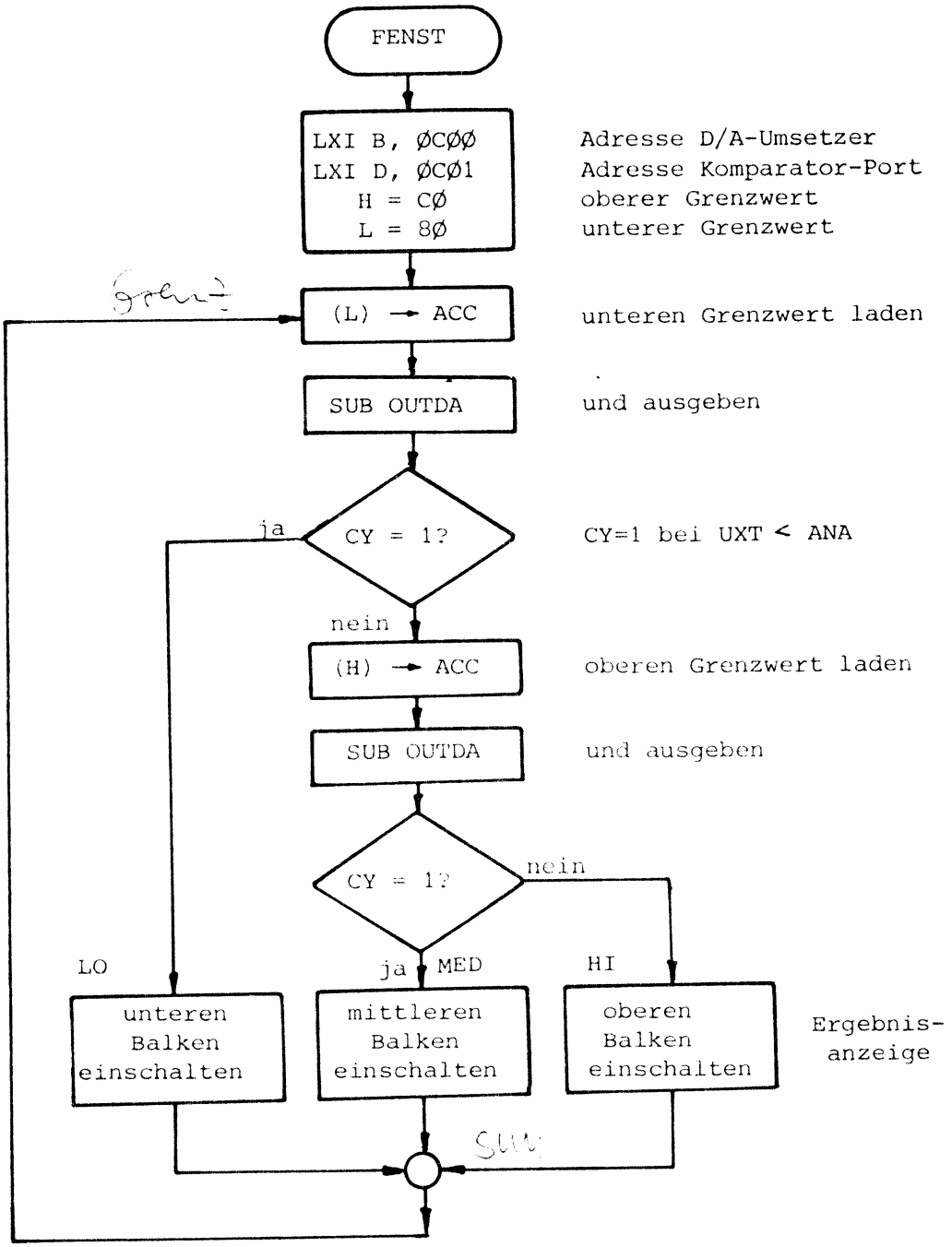
- GRENZWERTE FESTLEGEN
  - UNTERE GRENZE U_L = 2,5V = 80H
  - OBERE GRENZE U_H = 3,75V = C0H

- POTENTIOMETER-ANSCHLUSS
  - ENDEN AN „R_V" UND „GND" ANSCHLIESSEN
  - MITTELABGRIFF AN „UXT" LEGEN

- ERGEBNISANZEIGE
  - UNTERER BALKEN LEUCHTET, WENN $UXT < U_L$
  - MITTLERER BALKEN LEUCHTET, WENN $U_L < UXT < U_H$
  - OBERER BALKEN LEUCHTET, WENN $UXT > U_H$

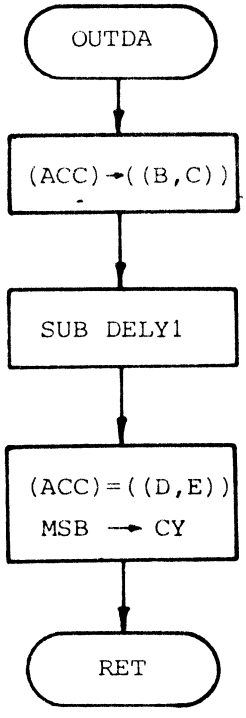
- ERGEBNISANZEIGE BEI VERSCHIEDENEN POTI-EINSTELLUNGEN BEOBACHTEN

Flußdiagramm FENST



Flußdiagramm OUTDA

- AUSGABE EINES DIGITALWERTES AN DEN DAU UND EINLESEN DES KOMPARATOR-AUSGANGSPEGELS INS CY-BIT



Grenzwert an DAU ausgeben

warten [1])

Komparator-Port einlesen

[1]) nach Ausgabe an den D/A-Umsetzer Einschwingzeit von DAU und Komparator abwarten

5.3.2 Digitalvoltmeter

- SCHRITTWEISE ANNÄHERUNG
  - SUCCESSIVE APPROXIMATION
  - DIGITALWERT VORGEBEN
  - GRÖSSER/KLEINER-ABFRAGE ÜBER KOMPARATOR

- HÖCHSTWERTIGES BIT SETZEN, WERT AN DAU AUSGEBEN
  - EINS INS ERGEBNISWORT NACHSCHIEBEN, WENN DAU-WERT KLEINER IST ALS UXT
  - NULL INS ERGEBNISWORT NACHSCHIEBEN, WENN DAU-WERT GRÖSSER IST ALS UXT

- MIT DEN NÄCHSTEN BITS ENTSPRECHEND FORTFAHREN, BIS DAS LSB ERREICHT IST

- FESTE UMSETZZEIT: ACHT VORGABE/ABFRAGE-ZYKLEN

Randbedingungen DVM

- DVM: UMSETZUNG EINER EXTERNEN ANALOGSPANNUNG UXT IN EIN 8-BIT-DIGITALWORT UND ERGEBNISANZEIGE

  KEINE VORZEICHEN- UND ÜBERLAUFERKENNUNG

- ANALOGAUSGANG ANA BEI BEREICHSENDE ABGLEICHEN
  - FF NACH ØCØØ AUSGEBEN, POTI ABGLEICHEN

- POTENTIOMETER ANSCHLIESSEN
  - ENDEN AN „R_V" UND „GND" ANSCHLIESSEN
  - MITTELABGRIFF AN „UXT" LEGEN

- VOLTMETER AN POTI-MITTELABGRIFF ANSCHLIESSEN UND ERGEBNIS-ANZEIGE BEI VERSCHIEDENEN POTI-EINSTELLUNGEN BEOBACHTEN

- MONITOR-UNTERPROGRAMM DISP2 [Ø372] STELLT DEN INHALT DER SPEICHERSTELLE ØBF8 IN DER ANZEIGE DAR

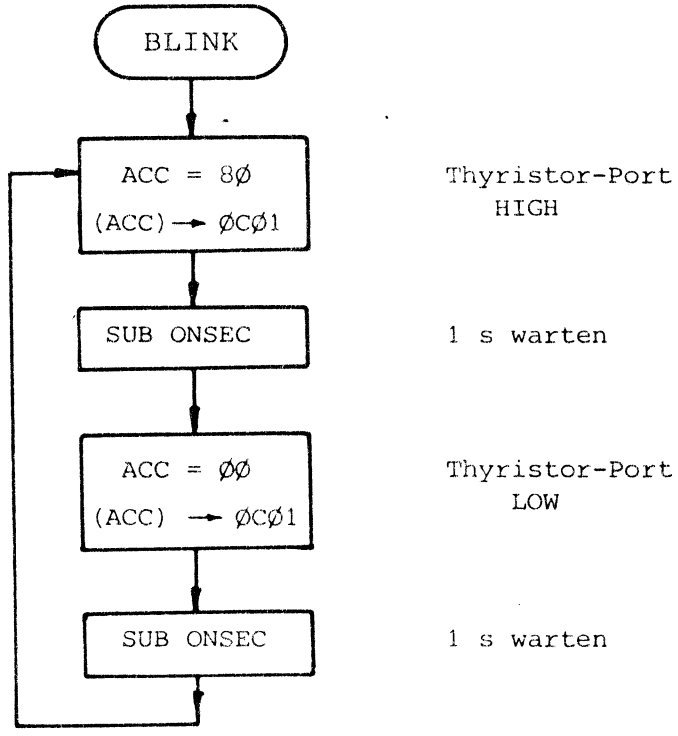
5.4 Leistungssteuerung

- IM NETZSTROMKREIS LIEGT EIN HALBLEITERSCHALTER
  - THYRISTOR MIT DIODENBRÜCKE
- ÜBER EIN TTL-SIGNAL LÄSST SICH DER HALBLEITERSCHALTER AKTIVIEREN
- GALVANISCHE TRENNUNG VOM NETZ ÜBER OPTOKOPPLER
- MAXIMALER LASTSTROM 1 A BEI 220 V

Vorsicht beim Umgang mit der Netzspannung!

5.4.1 220-V-Verbraucher schalten

- BLINK: EIN- UND AUSSCHALTEN DER 100-W-GLÜHBIRNE IM 1-s-TAKT
- MONITOR-UNTERPROGRAMM ONSEC [Ø3B1] ERZEUGT FESTE LAUFZEIT VON 1 s
- THYRISTOR ZÜNDEN DURCH HIGH-POTENTIAL AM THYRISTOR-PORT [ØCØ1, Bit 7]



- 220-V-NETZ AN KLEMMLEISTE „Netz" ANSCHLIESSEN (SCHUTZLEITER - PHASE - NULL)
- VERBRAUCHER (GLÜHBIRNE) AN KLEMMLEISTE „Last" ANSCHLIESSEN (SCHUTZLEITER - PHASE - NULL)

Programm BLINK

Adresse	Maschinencode 1.Byte	Maschinencode 2.Byte	Maschinencode 3.Byte	Label	Assemblercode	Zeile
0800	3E	80		BLINK	MVI, A 80	1
0802	32	01	0C		STA 0C01	2
0805	CD	B1	03		CALL ONSEC	3
0808	AF				XRA A	4
0809	32	01	0C		STA 0C01	5
080C	CD	B1	03		CALL ONSEC	6
080F	C3	00	08		JMP BLINK	7
0812						8
						9
						10
						11
						12
						13
						14
						15
						16
						17
						18
						19
						20
						21
						22
						23
						24
						25

Zeile	Kommentar
1	MSB auf HIGH
2	an Thyristor-Port ausgeben (Thyristor einschalten)
3	1 s warten
4	ACC löschen (MSB auf LOW)
5	an Thyristor-Port ausgeben (Thyristor ausschalten)
6	1 s warten
7	Sprung zum Anfang (Endlosschleife)
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	

5.4.2 Phasenanschnittsteuerung

- PHASENLAGE DER NETZSPANNUNG ÜBER OPTOKOPPLER ABNEHMEN
  - BIT 7 DES SENSOR-PORTS (ADR. ØCØØ) HAT 50-Hz-NETZFREQUENZ
  - ASYMMETRISCHES TASTVERHÄLTNIS VON CA. 7 ms : 20 ms
  - REALE NULLDURCHGÄNGE CA. 1,5 ms NACH DER NEGATIVEN FLANKE AM SENSOR-PORT, BIT 7

- DURCH VERSCHIEBEN DES ZÜNDZEITPUNKTES KANN DIE DEM VERBRAUCHER ZUGEFÜHRTE ENERGIE VERÄNDERT WERDEN

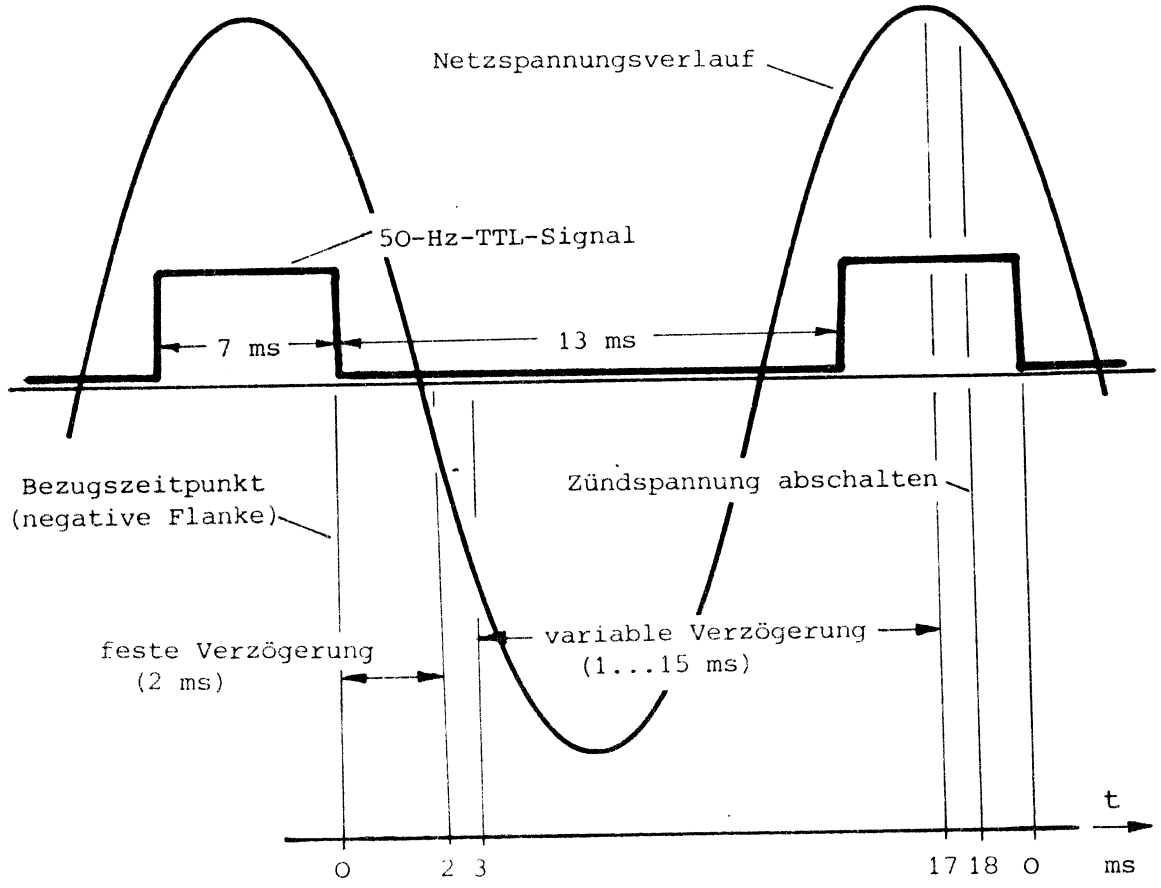
- ZÜND-VERZÖGERUNG IN 15 STUFEN Ø...F ÜBER DIE TASTATUR VORGEBEN (1...15 ms)

- MONITOR-UNTERPROGRAMM NIBIN [Ø34Ø]
  - LIEST EIN HEX-DIGIT NIBBLE EIN
  - STELLT DAS HEX-DIGIT IN DER ANZEIGE DAR
  - ÜBERNIMMT DAS HEX-DIGIT BEI NXT IN DEN AKKUMULATOR

- MONITOR-UNTERPROGRAMM DELY1 [Ø3C7] HAT FESTE LAUFZEIT VON 1 ms

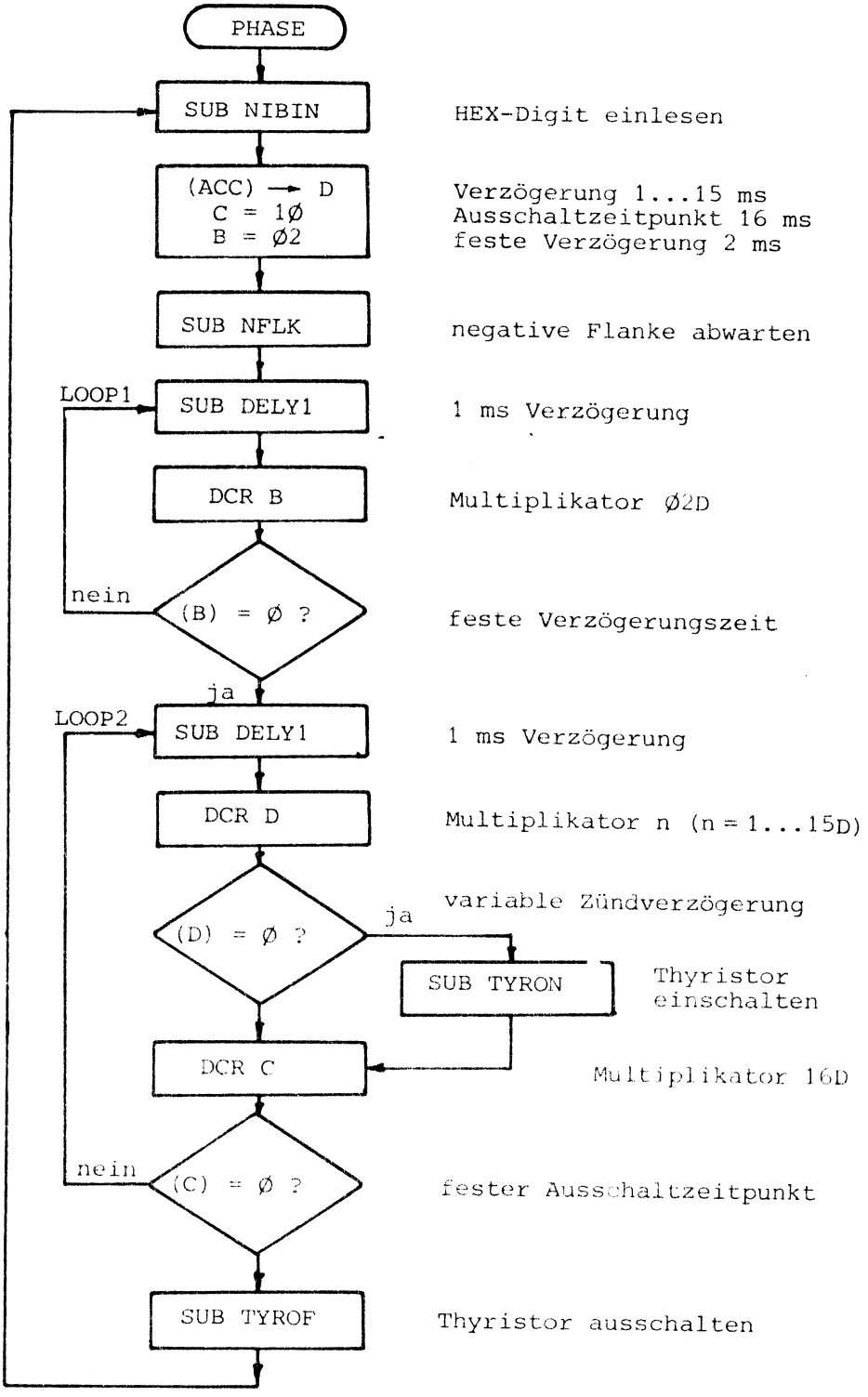
Vorsicht beim Umgang mit der Netzspannung!

Zeitdiagramm



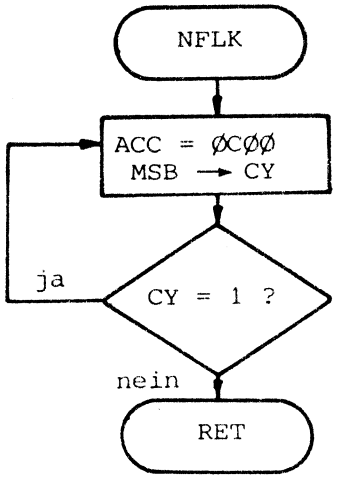
- THYRISTOR ZÜNDEN DURCH HIGH-POTENTIAL AM THYRISTOR-PORT [ØCØ1, Bit 7]
- VERLÖSCHEN DES THYRISTORS BEIM NÄCHSTEN NETZSPANNUNGS-NULL-DURCHGANG, NACHDEM LOW-POTENTIAL AM THYRISTOR-PORT ANLIEGT
- 220-V-NETZ AN KLEMMLEISTE „Netz" ANSCHLIESSEN (SCHUTZLEITER - PHASE - NULL)
- VERBRAUCHER (GLÜHBIRNE) AN KLEMMLEISTE „Last" ANSCHLIESSEN (SCHUTZLEITER - PHASE - NULL)

Flußdiagramm PHASE



Unterprogramme

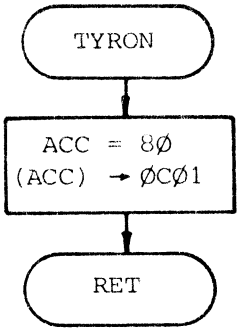
- **NFLK** WARTET AUF NEGATIVE FLANKE DES 50-Hz-SIGNALS



Sensor-Port einlesen
50-Hz-Pegel ins CY-Bit

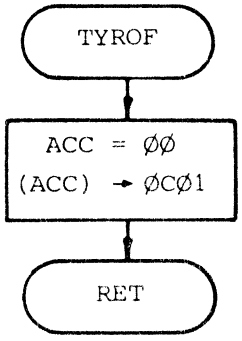
Schleife, bis der eingelesene Pegel LOW ist
(Rücksprung bei negativer Flanke)

- **TYRON** GIBT ZÜNDSPANNUNG AM THYRISTOR-PORT FREI



MSB = 1
an Thyristor-Port ausgeben

- **TYROF** SPERRT ZÜNDSPANNUNG AM THYRISTOR-PORT



MSB = Ø
an Thyristor-Port ausgeben

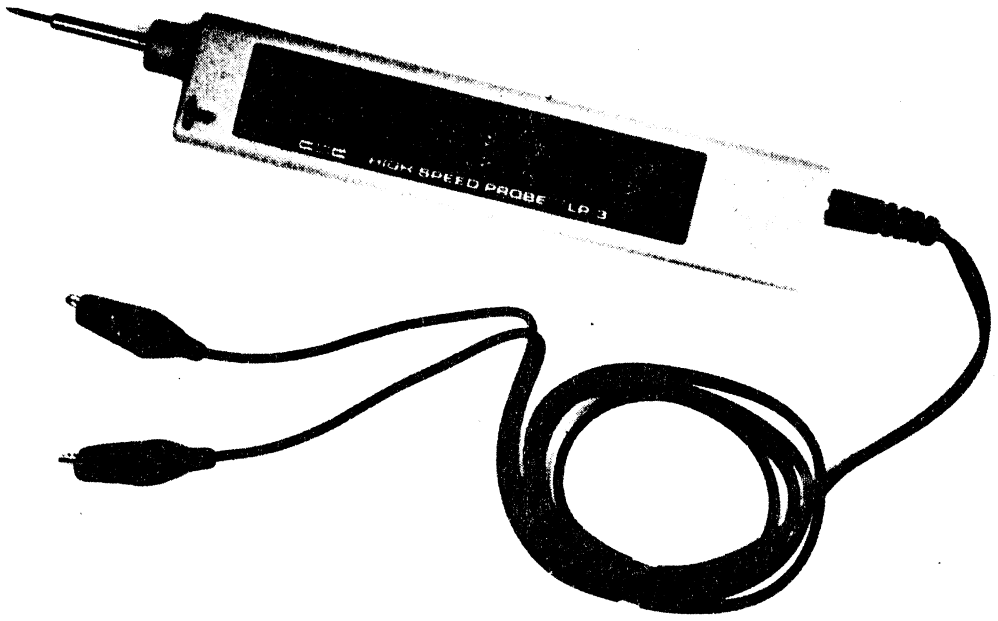
PHASE mit Unterprogrammen

Adresse	Maschinencode 1.Byte	2.Byte	3.Byte	Label	Assemblercode	Zeile
0800	CD	40	03	PHASE	CALL NIBIN	1
03	57				MOV D,A	2
04	01	10	02		LXI B,0210	3
07	CD	30	08		CALL NFLK	4
0A	CD	C7	03	LOOP1	CALL DELY1	5
0D	05				DCR B	6
0E	C2	0A	08		JNZ LOOP1	7
11	CD	C7	03	LOOP2	CALL DELY1	8
14	15				DCR D	9
15	CC	38	08		CZ TYRON	10
18	0D				DCR C	11
19	C2	11	08		JNZ LOOP2	12
1C	CD	3E	08		CALL TYROF	13
081F	C3	00	08		JMP PHASE	14
						15
0830	3A	00	0C	NFLK	LDA 0C00	16
33	17				RAL	17
34	D0			END	RNC	18
35	C3	30	08		JMP NFLK	19
0838	3E	80		TYRON	MVI A,80	20
3A	32	01	0C		STA 0C01	21
3D	C9				RET	22
083E	AF			TYROF	XRA A	23
3F	32	01	0C		STA 0C01	24
0842	C9				RET	25

Zeile	Kommentar
1	HEX-Digit einlesen (Verzögerungszeit 1...F ≙ 1...15 ms)
2	Verzögerungszeit nach D
3	(B) = Ø2 (feste Verzögerung); (C) = 1Ø (Ausschaltzeitpunkt 16 ms)
4	negative Flanke abwarten
5	}
6	} 2 ms warten (feste Verzögerungszeit)
7	}
8	} variable Verzögerungszeit
9	} (Multiplikator in Reg D) warten
10	nach Ablauf Thyristor einschalten
11	16 ms nach Einschalten Thyristor wieder
12	ausschalten (Multiplikator 16D = 1ØH in Reg C)
13	Unterprogramm zum Ausschalten des Thyristors
14	Sprung zum Anfang (Endlosschleife)
15	
16	Sensor-Port einlesen (5O-Hz-Netzfrequenz)
17	eingelesenen Pegel ins CY-Bit schieben
18	bei LOW-Pegel (CY=O) Rücksprung ins Hauptprogramm
19	Sprung zum Anfang
20	MSB auf HIGH
21	HIGH-Pegel an den Thyristor-Port ausgeben
22	Rücksprung
23	MSB auf LOW (ACC löschen)
24	LOW-Pegel an den Thyristor-Port ausgeben
25	Rücksprung

6 Hardware-Hilfsmittel

6.1 Logik-Prüfstift



- DER LOGIK-PRÜFSTIFT DIENT ZUR ZUSTANDSANZEIGE VON TTL- UND CMOS-PEGELN
  - ER BEZIEHT SEINE STROMVERSORGUNG VOM SYSTEM

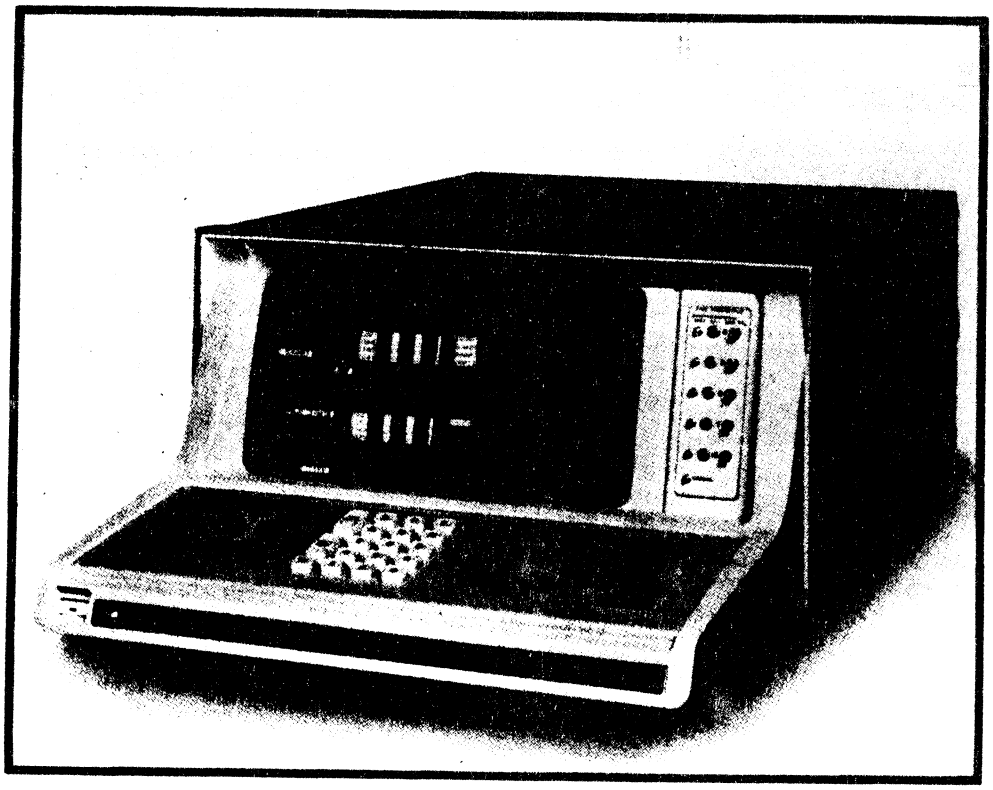
- PEGELANZEIGE ÜBER DREI LEDS
  - LOW
  - HIGH
  - PULSE
  - BEI OFFENEM EINGANG KEINE ANZEIGE

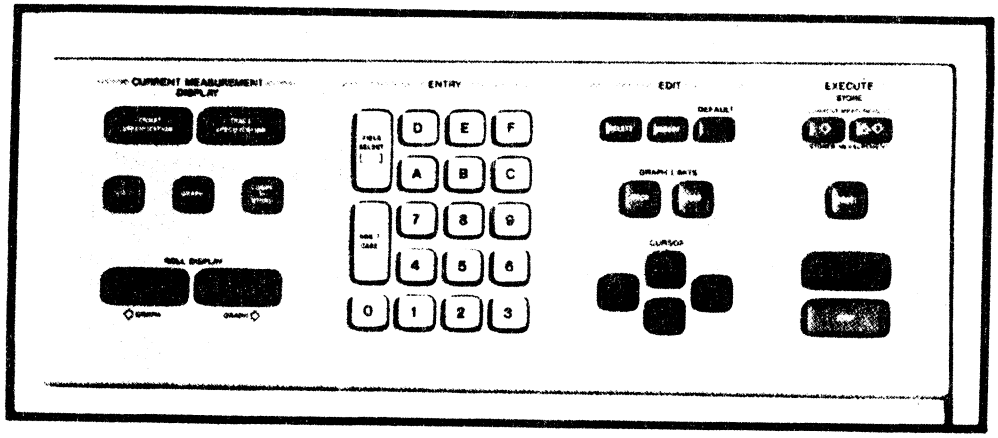
- ANGABE ÜBER DAS TASTVERHÄLTNIS
  - LOW-GRUNDPEGEL MIT HIGH-PULSEN:
    LOW-LED UND PULS-LED LEUCHTEN
  - HIGH-GRUNDPEGEL MIT LOW-PULSEN:
    HIGH-LED UND PULS-LED LEUCHTEN

- ANGABE ÜBER DIE PULSFREQUENZ
  - FREQUENZEN UNTER 1 MHz:
    HIGH-, LOW- UND PULS-LED LEUCHTEN
  - FREQUENZEN ÜBER 1 MHz:
    PULS-LED LEUCHTET ALLEIN

- EINMALIGE PULSE VON ≥ 20 ns WERDEN SICHTBAR VERLÄNGERT

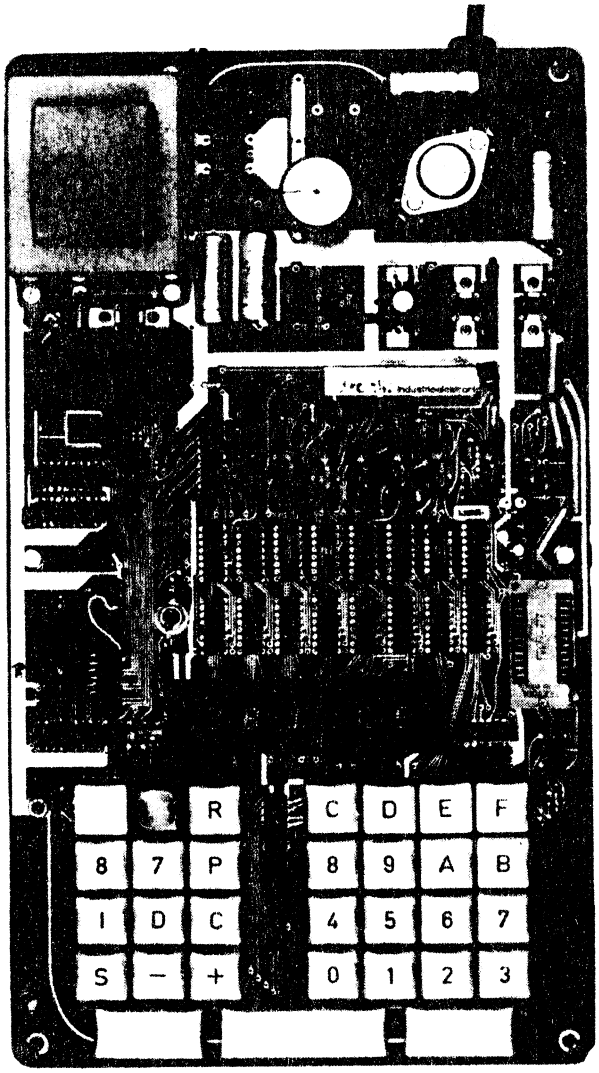
6.2 Logik-Analysator





- DER LOGIK-ANALYSATOR IST EIN MEHRKANAL-SPEICHEROSZILLOGRAF ZUR ERFASSUNG UND DARSTELLUNG VON LOGIKPEGELN
- GLEICHZEITIGE ERASSUNG/DARSTELLUNG PARALLELER LEITUNGEN
- MOMENTAUFNAHME DER DATENLEITUNGEN BEI POSITIVER ODER NEGATIVER FLANKE EINES CLOCK-SIGNALS
  ZUSÄTZLICHE VERKNÜPFUNG DES CLOCK MÖGLICH (QUALIFIER)
- VERSCHIEDENE TRIGGER-MODES (AUFZEICHNUNGSBEGINN)
  - VON EINEM EINGESTELLTEN BITMUSTER (WORT) AN
  - N TAKTPERIODEN NACH AUFTRETEN DES BITMUSTERS
  - BIS ZU EINEM EINGESTELLTEN BITMUSTER (WORT) HIN
  - N TAKTPERIODEN VOR AUFTRETEN DES BITMUSTERS
  - FREE RUN
- DARSTELLUNG ALS FOLGE VON Ø UND 1
  - EINTEILUNG IN VIERER-BITGRUPPEN (HEXADEZIMAL)
  - EINTEILUNG IN DREIER-BITGRUPPEN (OKTAL)
- DARSTELLUNG AUCH ALS HIGH- UND LOW-PEGEL (STRICHDARSTELLUNG)
- KOMFORTABLERE DARSTELLUNGEN DIREKT IM HEX- ODER ASSEMBLERCODE
  - PARALLELES ZEITDIAGRAMM
- AUFLÖSUNG BIS 10 ns ERHÄLTLICH

6.3 PROM-Simulator



- DER PROM-SIMULATOR ERSETZT WÄHREND DER TEST- UND INBETRIEB-NAHMEPHASE DEN PROGRAMMSPEICHER IM MIKROCOMPUTER
- ANSCHLUSS AN DEN MIKROCOMPUTER ÜBER ROM-KOMPATIBLEN STECKER
- DER SIMULATOR HAT EIN RAM DERSELBEN KAPAZITÄT UND ORGANISATION WIE DAS ZU SIMULIERENDE ROM/PROM
- EINFACHES LADEN/MODIFIZIEREN/INSPIZIEREN DES SIMULATOR-INHALTS
- NACH DEM PROGRAMM-TEST DIREKTES ÜBERSCHREIBEN DES SIMULATOR-INHALTS INS EPROM
- PERSONALITY-MODULES FÜR DIE GÄNGIGEN ROM/PROM-TYPEN
- DUPLIZIEREN VON PROMS/EPROMS

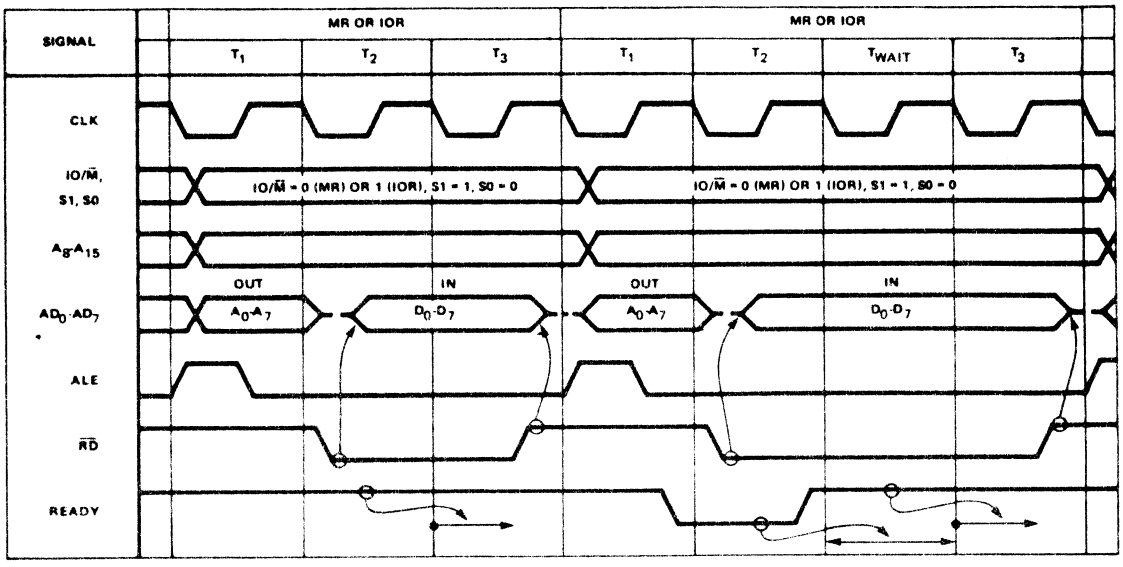
6.4 8085A-Befehlszyklus

- DER BEFEHLSZYKLUS UMFASST
  - HOLEN
  - DECODIEREN
  - AUSFÜHREN EINES BEFEHLS
- JEDER BEFEHLSZYKLUS BESTEHT AUS 1...5 MASCHINENZYKLEN
- MASCHINENZYKLUS: ZUGRIFF AUF EIN BYTE IM SPEICHER ODER PORT
  - BEFEHL MOV r,r: 1 MASCHINENZYKLUS
  - BEFEHL CALL: 5 MASCHINENZYKLEN
- JEDER MASCHINENZYKLUS IST ENTWEDER EINE READ- ODER WRITE-OPERATION
- JEDER MASCHINENZYKLUS IST ENTWEDER SPEICHER- ODER I/O-BEZOGEN
- JEDER MASCHINENZYKLUS HAT EINE FESTE ANZAHL VON TAKTZYKLEN T (T-STATES)
  - M1 (OF): 4 ODER 6 T-STATES
  - M2...5: IMMER 3 T-STATES

Zeitbedingungen

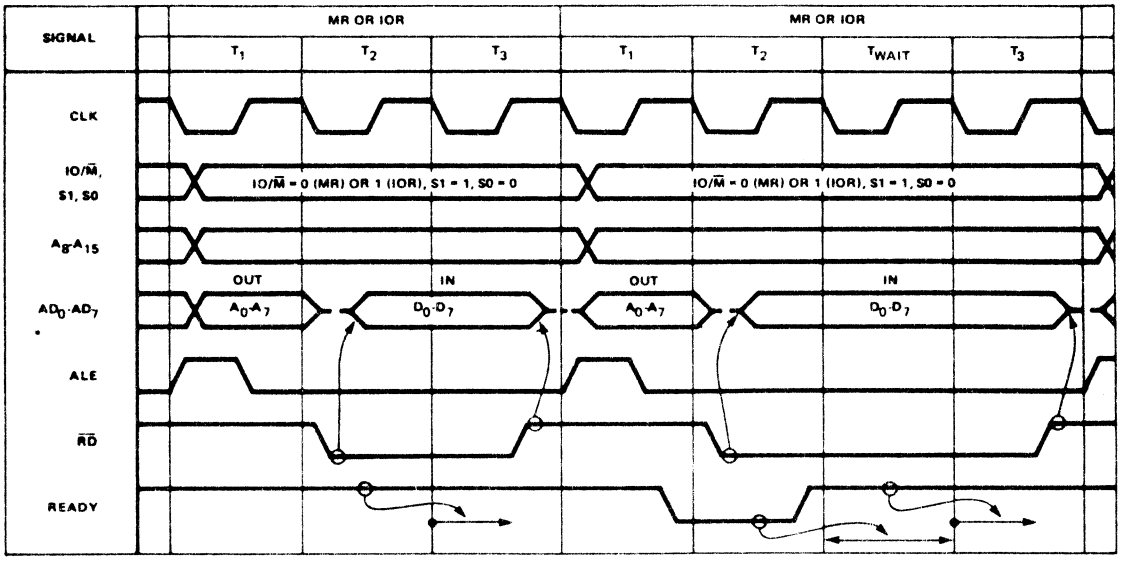
- MINIMALE TAKTPERIODENDAUER τ = 320 ns
- MAXIMALE TAKTFREQUENZ f_{cl} = 3,125 MHz
  - OSZILLATORFREQUENZ f_{osc} = 6,250 MHz
  - SPEICHERZUGRIFFSZEIT $t_{acc} \leq$ 575 ns
- ÜBLICHE 8085-OSZILLATORFREQUENZ f_{osc} = 6,144 MHz
  - TAKTFREQUENZ f_{cl} = 3,072 MHz
  - BINÄRE STUFUNG 1024 · 3 kHz
- BEFEHLSZYKLUSZEIT 4...18 τ
  - BEI f_{cl} = 3,125 MHz: 1,28...5,76 µs
- VERGLEICH 8080A
  - BEI f_{cl} = 2,083 MHz: 1,92...8,64 µs
- DER 8085 IST RUND 50% SCHNELLER (MOV r,r: 1,28 ⁄ 2,40 µs)
  - DIE SPEICHERZUGRIFFSZEIT IST GLEICH GEBLIEBEN

6.4.1 READ-Maschinenzyklus



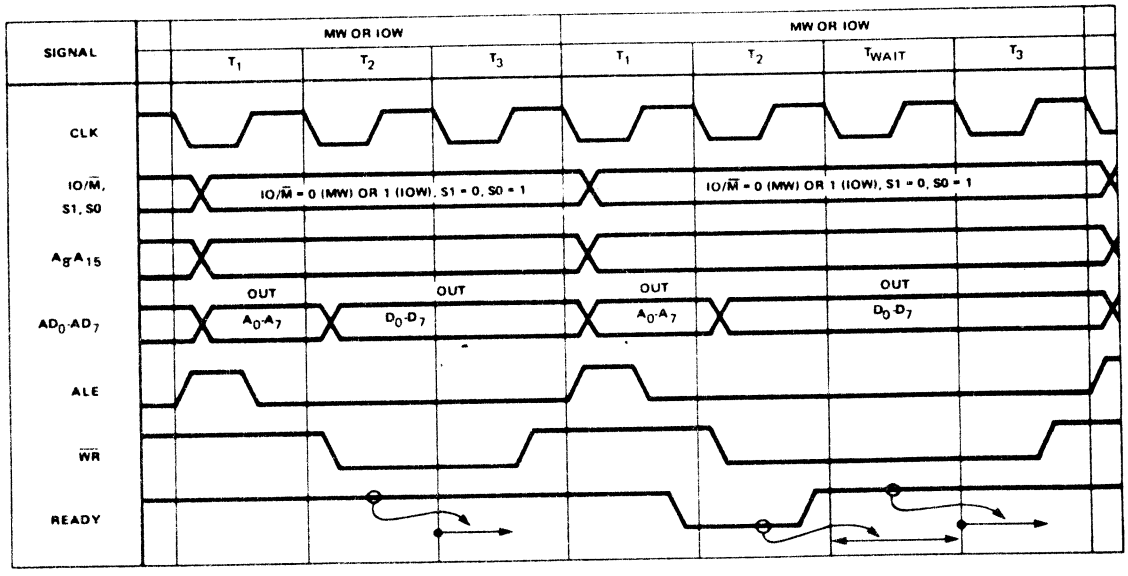
MEMORY READ (OR IO READ) MACHINE CYCLES.

6.4.1 READ-Maschinenzyklus



MEMORY READ (OR IO READ) MACHINE CYCLES.

6.4.2 WRITE-Maschinenzyklus



MEMORY WRITE (OR IO WRITE) MACHINE CYCLES.

6.4.3 8085A-Maschinenzyklen

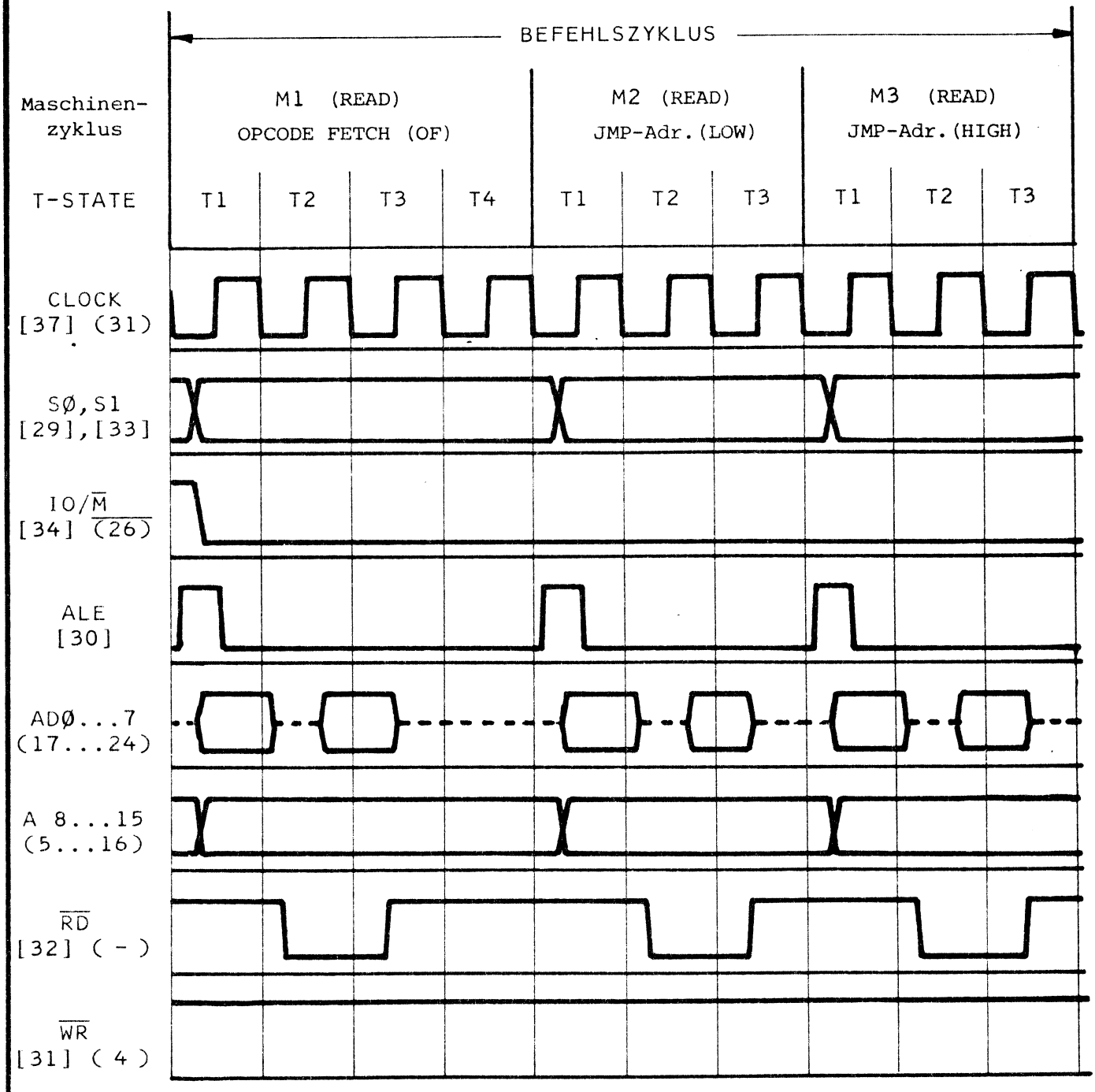
- STATUSINFORMATION AUF SØ,S1 UND IO/$\overline{M}$ BESTIMMEN DIE ART DES JEWEILIGEN M-ZYKLUS

IO/$\overline{M}$	
Ø	SPEICHERBEZOGEN
1	I/O-BEZOGEN

S1	SØ	
Ø	Ø	HALT-ZYKLUS
Ø	1	WRITE
1	Ø	READ
1	1	OPCODE FETCH (OF)

- JEDER TAKTZYKLUS HAT EINE BESTIMMTE BEDEUTUNG
  - T 1 : STATUSINFORMATION AUF SØ,S1 UND IO/$\overline{M}$ BESTIMMT DIE ART DES NEUEN M-ZYKLUS
    ADØ...AD7 HABEN UNTERE HÄLFTE DER SPEICHER/PORT-ADRESSE (ABLEGEN MIT ALE)
    A8...A15 HABEN OBERE HÄLFTE DER SPEICHER/PORT-ADRESSE
  - T 2 : SPEICHERZUGRIFFSZEIT
    ABFRAGE DER HOLD- UND READY-EINGÄNGE
    ABFRAGE NACH HALT-BEFEHL
  - T 3 : TRANSPORT DES ADRESSIERTEN BYTES IN DIE (AUS DER) CPU
  - T 4 : DECODIEREN DES OPCODES IM BEFEHLSHALTEREGISTER
  - T5,6: BEDINGUNGSABFRAGE
    DATENTRANSPORT IN DIE HILFSREGISTER (OPERANDEN)
    DATENTRANSPORT INS ZIELREGISTER (AUS DER ALU)

6.4.4 Ausführung des JMP-Befehls



7 Anhang

7.1 Sachverzeichnis

7.2 Befehlssatz (Kurzfassung)

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
MOV r1,r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	5
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RST	Restart	1	1	A	A	A	1	1	1	12
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (6/12) indicate instruction cycles dependent on condition flags.

Befehlssatz in numerischer Reihenfolge

Befehlsliste in Hex-Folge

Hex	Mnemonic		Hex	Mnemonic		Hex	Mnemonic		Hex	Mnemonic		Hex	Mnemonic		Hex	Mnemonic	
00	NOP		2B	DCX	H	56	MOV	D,M	81	ADD	C	AC	XRA	H	D7	RST	2
01	LXI	B,D16	2C	INR	L	57	MOV	D,A	82	ADD	D	AD	XRA	L	D8	RC	
02	STAX	B	2D	DCR	L	58	MOV	E,B	83	ADD	E	AE	XRA	M	D9	- - -	
03	INX	B	2E	MVI	L,D8	59	MOV	E,C	84	ADD	H	AF	XRA	A	DA	JC	Adr
04	INR	B	2F	CMA		5A	MOV	E,D	85	ADD	L	B0	ORA	B	DB	IN	D8
05	DCR	B	30	SIM		5B	MOV	E,E	86	ADD	M	B1	ORA	C	DC	CC	Adr
06	MVI	B,D8	31	LXI	SP,D16	5C	MOV	E,H	87	ADD	A	B2	ORA	D	DD	- - -	
07	RLC		32	STA	Adr	5D	MOV	E,L	88	ADC	B	B3	ORA	E	DE	SBI	D8
08	- - -		33	INX	SP	5E	MOV	E,M	89	ADC	C	B4	ORA	H	DF	RST	3
09	DAD	B	34	INR	M	5F	MOV	E,A	8A	ADC	D	B5	ORA	L	E0	RPO	
0A	LDAX	B	35	DCR	M	60	MOV	H,B	8B	ADC	E	B6	ORA	M	E1	POP	H
0B	DCX	B	36	MVI	M,D8	61	MOV	H,C	8C	ADC	H	B7	ORA	A	E2	JPO	Adr
0C	INR	C	37	STC		62	MOV	H,D	8D	ADC	L	B8	CMP	B	E3	XTHL	
0D	DCR	C	38	- - -		63	MOV	H,E	8E	ADC	M	B9	CMP	C	E4	CPO	Adr
0E	MVI	C,D8	39	DAD	SP	64	MOV	H,H	8F	ADC	A	BA	CMP	D	E5	PUSH	H
0F	RRC		3A	LDA	Adr	65	MOV	H,L	90	SUB	B	BB	CMP	E	E6	ANI	D8
10	- - -		3B	DCX	SP	66	MOV	H,M	91	SUB	C	BC	CMP	H	E7	RST	4
11	LXI	D,D16	3C	INR	A	67	MOV	H,A	92	SUB	D	BD	CMP	L	E8	RPE	
12	STAX	D	3D	DCR	A	68	MOV	L,B	93	SUB	E	BE	CMP	M	E9	PCHL	
13	INX	D	3E	MVI	A,D8	69	MOV	L,C	94	SUB	H	BF	CMP	A	EA	JPE	Adr
14	INR	D	3F	CMC		6A	MOV	L,D	95	SUB	L	C0	RNZ		EB	XCHG	
15	DCR	D	40	MOV	B,B	6B	MOV	L,E	96	SUB	M	C1	POP	B	EC	CPE	Adr
16	MVI	D,D8	41	MOV	B,C	6C	MOV	L,H	97	SUB	A	C2	JNZ	Adr	ED	- - -	
17	RAL		42	MOV	B,D	6D	MOV	L,L	98	SBB	B	C3	JMP	Adr	EE	XRI	D8
18	- - -		43	MOV	B,E	6E	MOV	L,M	99	SBB	C	C4	CNZ	Adr	EF	RST	5
19	DAD	D	44	MOV	B,H	6F	MOV	L,A	9A	SBB	D	C5	PUSH	B	F0	RP	
1A	LDAX	D	45	MOV	B,L	70	MOV	M,B	9B	SBB	E	C6	ADI	D8	F1	POP	PSW
1B	DCX	D	46	MOV	B,M	71	MOV	M,C	9C	SBB	H	C7	RST	0	F2	JP	Adr
1C	INR	E	47	MOV	B,A	72	MOV	M,D	9D	SBB	L	C8	RZ		F3	DI	
1D	DCR	E	48	MOV	C,B	73	MOV	M,E	9E	SBB	M	C9	RET	Adr	F4	CP	Adr
1E	MVI	E,D8	49	MOV	C,C	74	MOV	M,H	9F	SBB	A	CA	JZ		F5	PUSH	PSW
1F	RAR		4A	MOV	C,D	75	MOV	M,L	A0	ANA	B	CB	- - -		F6	ORI	D8
20	RIM		4B	MOV	C,E	76	HLT		A1	ANA	C	CC	CZ	Adr	F7	RST	6
21	LXI	H,D16	4C	MOV	C,H	77	MOV	M,A	A2	ANA	D	CD	CALL	Adr	F8	RM	
22	SHLD	Adr	4D	MOV	C,L	78	MOV	A,B	A3	ANA	E	CE	ACI	D8	F9	SPHL	
23	INX	H	4E	MOV	C,M	79	MOV	A,C	A4	ANA	H	CF	RST	1	FA	JM	Adr
24	INR	H	4F	MOV	C,A	7A	MOV	A,D	A5	ANA	L	D0	RNC		FB	EI	
25	DCR	H	50	MOV	D,B	7B	MOV	A,E	A6	ANA	M	D1	POP	D	FC	CM	Adr
26	MVI	H,D8	51	MOV	D,C	7C	MOV	A,H	A7	ANA	A	D2	JNC	Adr	FD	- - -	
27	DAA		52	MOV	D,D	7D	MOV	A,L	A8	XRA	B	D3	OUT	D8	FE	CPI	D8
28	- - -		53	MOV	D,E	7E	MOV	A,M	A9	XRA	C	D4	CNC	Adr	FF	RST	7
29	DAD	H	54	MOV	D,H	7F	MOV	A,A	AA	XRA	D	D5	PUSH	D			
2A	LHLD	Adr	55	MOV	D,L	80	ADD	B	AB	XRA	E	D6	SUI	D8			

D8 Eine Konstante oder ein logisch/arithmetischer Ausdruck der eine 8 Bit Datengröße darstellt

D16 Eine Konstante oder ein logisch/arithmetischer Ausdruck der eine 16 Bit Datengröße darstellt

Adr Eine 16 Bit Adresse

Umwandlungstabelle Hexadezimal - ASCII-Code

Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	\	70	p		
01	SOH	11	DC1 (X-ON)	21	!	31	1	41	A	51	Q	61	a	71	q		
02	STX	12	DC2 (TAPE)	22	"	32	2	42	B	52	R	62	b	72	r		
03	ETX	13	DC3 (X-OFF)	23	#	33	3	43	C	53	S	63	c	73	s		
04	EOT	14	DC4 (TAPE)	24	$	34	4	44	D	54	T	64	d	74	t		
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u		
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v		
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w		
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x		
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y		
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z		
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{		
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|		
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}	(ALT MODE)	
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^ (↑)	6E	n	7E	~		
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_ (←)	6F	o	7F	DEL (RUB OUT)		

7.3 Befehlssatz (Detailbeschreibung)

8085A INSTRUCTION SET*

A computer, no matter how sophisticated, can only do what it is "told" to do. One "tells" the computer what to do via a series of coded instructions referred to as a **Program**. The realm of the programmer is referred to as **Software**, in contrast to the **Hardware** that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's **Instruction Set**.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (e.g., OR the contents of two registers) and register operate instructions (e.g., increment a register) are included in the instruction set. A computer's instruction set will also have instructions that move data between registers, between a register and memory, and between a register and an I/O device. Most instruction sets also provide **Conditional Instructions**. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (i.e., a series of 1's and 0's), that is called **Machine Code**. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is **Assembly Language**. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the **Source Program**) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the **Object Code**). Each assembly language instruction is converted into one machine code instruction (1 or more bytes) by an **Assembler** program. Assembly languages are usually machine dependent (i.e., they are usually able to run on only one type of computer).

THE 8085A INSTRUCTION SET

The 8085A instruction set includes five different types of instructions:

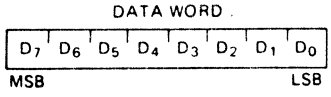
- **Data Transfer Group** – move data between registers or between memory and registers
- **Arithmetic Group** – add, subtract, increment or decrement data in registers or in memory
- **Logical Group** – AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- **Branch Group** – conditional and unconditional jump instructions, subroutine call instructions and return instructions
- **Stack, I/O and Machine Control Group** – includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

Instruction and Data Formats:

Memory for the 8085A is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

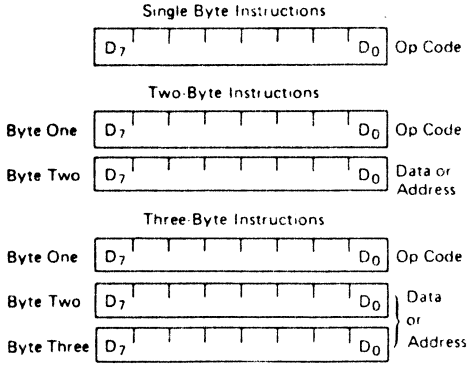
The 8085A can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8085A is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8085A, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8 bit number) is referred to as the **Most Significant Bit (MSB)**.

The 8085A program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



Addressing Modes:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8085A has four different modes for addressing data stored in memory or in registers:

- Direct – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- Register – The instruction specifies the register or register-pair in which the data is located.
- Register Indirect – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- Direct – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- Register Indirect – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

Condition Flags:

There are five condition flags associated with the execution of instructions on the 8085A. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1, "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8085A instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r1,r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD=destination, SSS=source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp	One of the register pairs: B represents the B,C pair with B as the high order register and C as the low-order register; D represents the D,E pair with D as the high order register and E as the low-order register; H represents the H,L pair with H as the high order register and L as the low-order register; SP represents the 16-bit stack pointer register.
RP	The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh	The first (high-order) register of a designated register pair.
rl	The second (low-order) register of a designated register pair.
PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).
r_m	Bit m of the register r (bits are number 7 through 0 from left to right).
Z,S,P,CY,AC	The condition flags: Zero, Sign, Parity, Carry, and Auxiliary Carry, respectively.
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
∀	Exclusive OR
∨	Inclusive OR
+	Addition
−	Two's complement subtraction
*	Multiplication
← →	"Is exchanged with"
—	The one's complement (e.g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

Description Format:

The following pages provide a detailed description of the instruction set of the 8085A. Each instruction is described in the following manner:

1. The MCS 85™ macro assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.
2. The name of the instruction is enclosed in parenthesis on the right side of the first line.
3. The next line(s) contain a symbolic description of the operation of the instruction.
4. This is followed by a narative description of the operation of the instruction.
5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page 4-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

Data Transfer Group:

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected** by any instruction in this group.

MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: none

MOV r, M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: none

MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),
(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

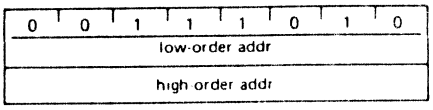
0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles: 3
States: 10
Addressing: immediate
Flags: none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

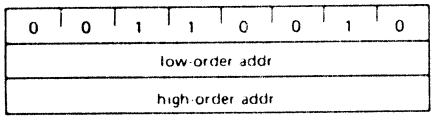


Cycles: 4
States: 13
Addressing: direct
Flags: none

STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



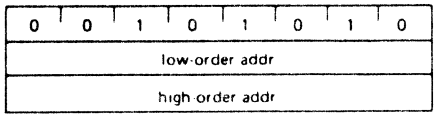
Cycles: 4
States: 13
Addressing: direct
Flags: none

LHLD addr (Load H and L direct)

(L) ← ((byte 3)(byte 2))

(H) ← ((byte 3)(byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



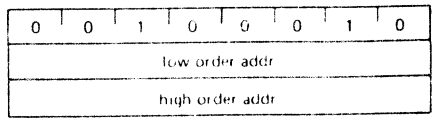
Cycles: 5
States: 16
Addressing: direct
Flags: none

SHLD addr (Store H and L direct)

((byte 3)(byte 2)) ← (L)

((byte 3)(byte 2) + 1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

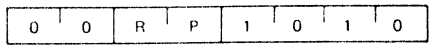


Cycles: 5
States: 16
Addressing: direct
Flags: none

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

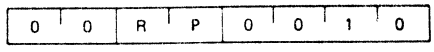


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

STAX rp (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.



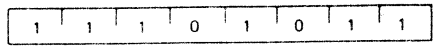
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

XCHG (Exchange H and L with D and E)

(H) ↔ (D)

(L) ↔ (E)

The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1
States: 4
Addressing: register
Flags: none

Arithmetic Group:

This group of instructions performs arithmetic operations on data in registers and memory.

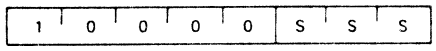
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

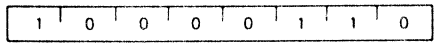


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADD M (Add memory)

(A) ← (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

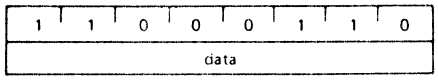


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ADI data (Add immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

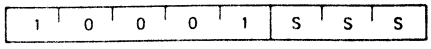


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ADC r (Add Register with carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

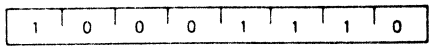


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADC M (Add memory with carry)

(A) ← (A) + ((H) (L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

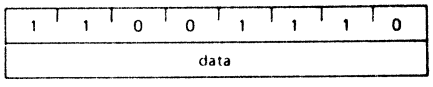


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ACI data (Add immediate with carry)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

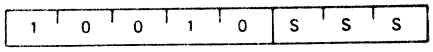


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SUB r (Subtract Register)

(A) ← (A) − (r)

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SUB M (Subtract memory)

(A) ← (A) – ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SUI data (Subtract immediate)

(A) ← (A) – (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	1	0	1	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SBB r (Subtract Register with borrow)

(A) ← (A) – (r) – (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SBB M (Subtract memory with borrow)

(A) ← (A) – ((H) (L)) – (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SBI data (Subtract immediate with borrow)

(A) ← (A) – (byte 2) – (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	1	0	1	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INR r (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags **except CY are affected.**

0	0	D	D	D	1	0	0

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except CY are** affected.

0	0	1	1	0	1	0	0

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement Register)

(r) ← (r) – 1

The content of register r is decremented by one. Note: All condition flags **except CY are affected.**

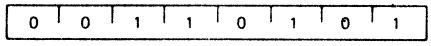
0	0	D	D	D	1	0	1

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) − 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except CY** are affected.

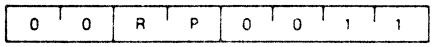


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

The content of the register pair rp is incremented by one. Note: **No condition flags are affected.**

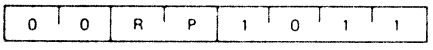


Cycles 1
States 6
Addressing register
Flags none

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

The content of the register pair rp is decremented by one. Note: **No condition flags are affected.**

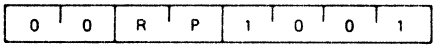


Cycles: 1
States 6
Addressing register
Flags none

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add, otherwise it is reset.



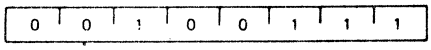
Cycles 3
States 10
Addressing: register
Flags: CY

DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 **or** if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, **or** if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.



Cycles: 1
States 4
Flags Z,S,P,CY,AC

Logical Group.

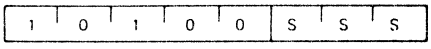
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (AND Register)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

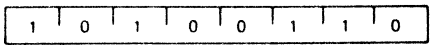


Cycles: 1
States 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (AND memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

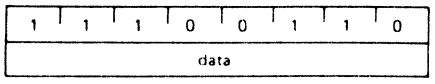


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANI data (AND immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

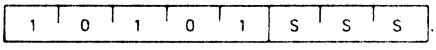


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR Register)

(A) ← (A) ∀ (r)

The content of register r is exclusive or'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

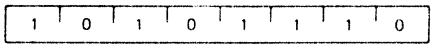


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

XRA M (Exclusive OR Memory)

(A) ← (A) ∀ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is exclusive OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

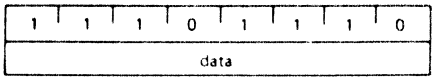


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XRI data (Exclusive OR immediate)

(A) ← (A) ∀ (byte 2)

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

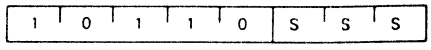


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ORA r (OR Register)

(A) ← (A) V (r)

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

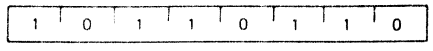


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR memory)

(A) ← (A) V ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

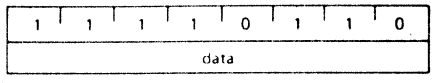


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate)

(A) ← (A) V (byte 2)

The content of the second byte of the instruction is inclusive OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

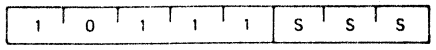


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

CMP r (Compare Register)

(A) − (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**

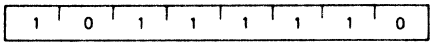


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

CMP M (Compare memory)

(A) − ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).

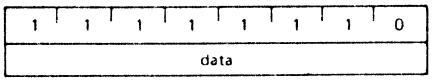


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

CPI data (Compare immediate)

(A) − (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).



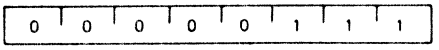
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**



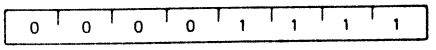
Cycles: 1
States: 4
Flags: CY

RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1})$; $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**



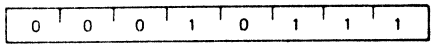
Cycles: 1
States: 4
Flags: CY

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**



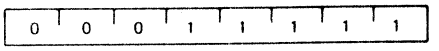
Cycles: 1
States: 4
Flags: CY

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$

$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**

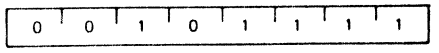


Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (\overline{A})$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**

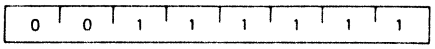


Cycles: 1
States: 4
Flags: none

CMC (Complement carry)

(CY) ← $\overline{(CY)}$

The CY flag is complemented. **No other flags are affected.**

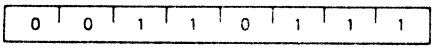


Cycles: 1
States: 4
Flags: CY

STC (Set carry)

(CY) ← 1

The CY flag is set to 1. **No other flags are affected.**



Cycles: 1
States: 4
Flags: CY

Branch Group:

This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

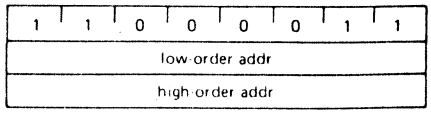
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ – not zero (Z = 0)	000
Z – zero (Z = 1)	001
NC – no carry (CY = 0)	010
C – carry (CY = 1)	011
PO – parity odd (P = 0)	100
PE – parity even (P = 1)	101
P – plus (S = 0)	110
M – minus (S = 1)	111

JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



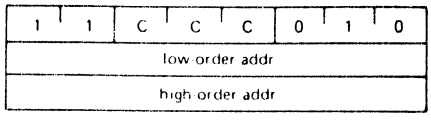
Cycles: 3
States: 10
Addressing: immediate
Flags: none

Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



Cycles: 2/3
States: 7/10
Addressing: immediate
Flags: none

CALL addr (Call)

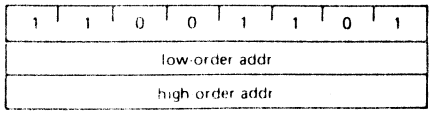
((SP) – 1) ← (PCH)

((SP) – 2) ← (PCL)

(SP) ← (SP) – 2

(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

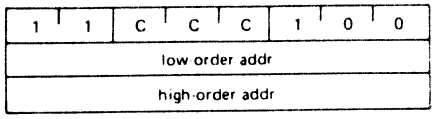


Cycles: 5
States: 18
Addressing: immediate/reg. indirect
Flags: none

Ccondition addr (Condition call)

If (CCC),
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

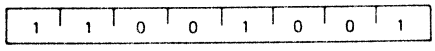


Cycles: 2/5
States: 9/18
Addressing: immediate/reg. indirect
Flags: none

RET (Return)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

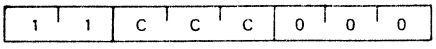


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

Rcondition (Conditional return)

If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially

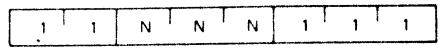


Cycles: 1/3
States: 6/12
Addressing: reg. indirect
Flags: none

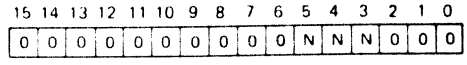
RST n (Restart)

((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← 8 • (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.



Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

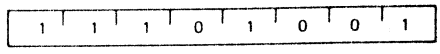


Program Counter After Restart

PCHL (Jump H and L indirect – move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1
States: 6
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group:

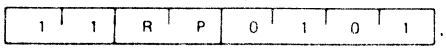
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**



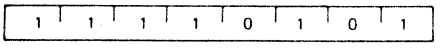
Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) − 1) ← (A)
((SP) − 2)$_0$ ← (CY), ((SP) − 2)$_1$ ← X
((SP) − 2)$_2$ ← (P), ((SP) − 2)$_3$ ← X
((SP) − 2)$_4$ ← (AC), ((SP) − 2)$_5$ ← X
((SP) − 2)$_6$ ← (Z), ((SP) − 2)$_7$ ← (S)
(SP) ← (SP) − 2

X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 3
States: 12
Addressing: reg. indirect
Flags: none

FLAG WORD

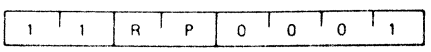
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

X: undefined

POP rp (Pop)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**

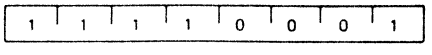


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

POP PSW (Pop processor status word)

(CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

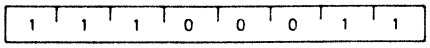


Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XTHL (Exchange stack top with H and L)

(L) ⟷ ((SP))
(H) ⟷ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

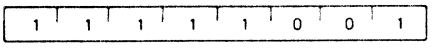


Cycles: 5
States: 16
Addressing: reg. indirect
Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

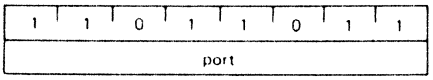


Cycles: 1
States: 6
Addressing: register
Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A

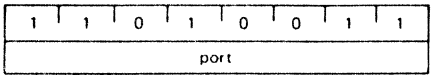


Cycles: 3
States: 10
Addressing: direct
Flags: none

OUT port (Output)

(data) ← (A)

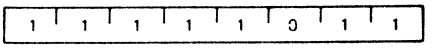
The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



Cycles: 3
States: 10
Addressing: direct
Flags: none

EI (Enable interrupts)

The interrupt system is enabled **following the execution of the next instruction.**

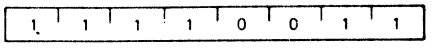


Cycles: 1
States: 4
Flags: none

NOTE: Interrupts are not recognized during the EI instruction.

DI (Disable interrupts)

The interrupt system is disabled **immediately following the execution of the DI instruction.**

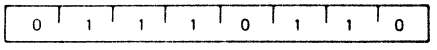


Cycles: 1
States: 4
Flags: none

NOTE: Interrupts are not recognized during the DI instruction.

HLT (Halt)

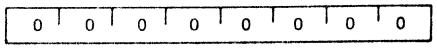
The processor is stopped. The registers and flags are unaffected.



Cycles: 1
States: 5
Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.



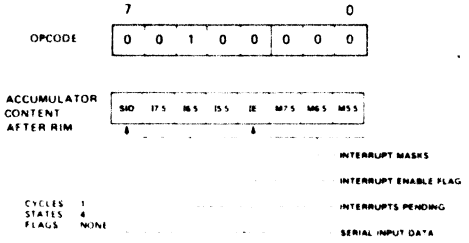
Cycles: 1
States: 4
Flags: none

RIM (Read Interrupt Mask)

After the execution of the RIM instruction, the accumulator is loaded with the restart interrupt masks, the Interrupt Enable Flag, any pending interrupts, and the contents of the serial input data line (SID).

When the first RIM instruction is executed following a TRAP interrupt, the IE bit in the accumulator reflects the *previous* interrupt enable status before the TRAP occurred. The first RIM after a TRAP also has the action of releasing the IE status bit such that all subsequent RIM's provide current interrupt enable status.

IMPORTANT: The RIM instruction must be executed following a TRAP interrupt in order to restore the correct Interrupt Enable Status.



SIM (Set Interrupt Masks)

During execution of the SIM instruction, the contents of the accumulator will be used in programming the restart interrupt masks. Bits 0–2 will set/reset the mask bit for RST 5.5, 6.5, 7.5 of the interrupt mask register, if bit 3 is 1 ("set"). Bit 3 is a "Mask Set Enable" control.

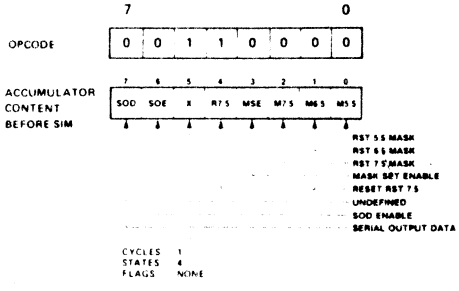
Setting the mask (i.e. mask bit = 1) **disables** the corresponding interrupt.

	Set	Reset
RST 5.5 MASK	if bit 0 = 1	if bit 0 = 0
RST 6.5 MASK	bit 1 = 1	bit 1 = 0
RST 7.5 MASK	bit 2 = 1	bit 2 = 0

The RST 7.5 (edge triggered) internal request flip flop will be reset if bit 4 of the accumulator = 1; regardless of whether RST 7.5 is masked or not.

A hardware RESET of the 8085A will set all RST MASKs, and reset/disable all interrupts.

SIM can also load the SOD output latch. Accumulator bit 7 is loaded into the SOD latch if bit 6 is set. The latch is unaffected if bit 6 is a zero. $\overline{\text{RESET IN}}$ input sets the SOD latch to zero.



8085-Maschinenbefehle

UMS-85®

DATENTRANSPORTBEFEHLE

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

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

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

LDA Adr 3A YY XX	STA Adr 32 YY XX	XTHL E3	
LHLD Adr 2A YY XX	SHLD Adr 22 YY XX	SPHL F9	
		XCHG EB	
		IN XX DB XX	
		OUT XX D3 XX	

Registerpaar rp	B	D
LDAX rp	0A	1A
STAX rp	02	12

ARITHMETISCH/LOGISCHE BEFEHLE

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

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

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

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

SPRUNGBEFEHLE

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

PCHL E9

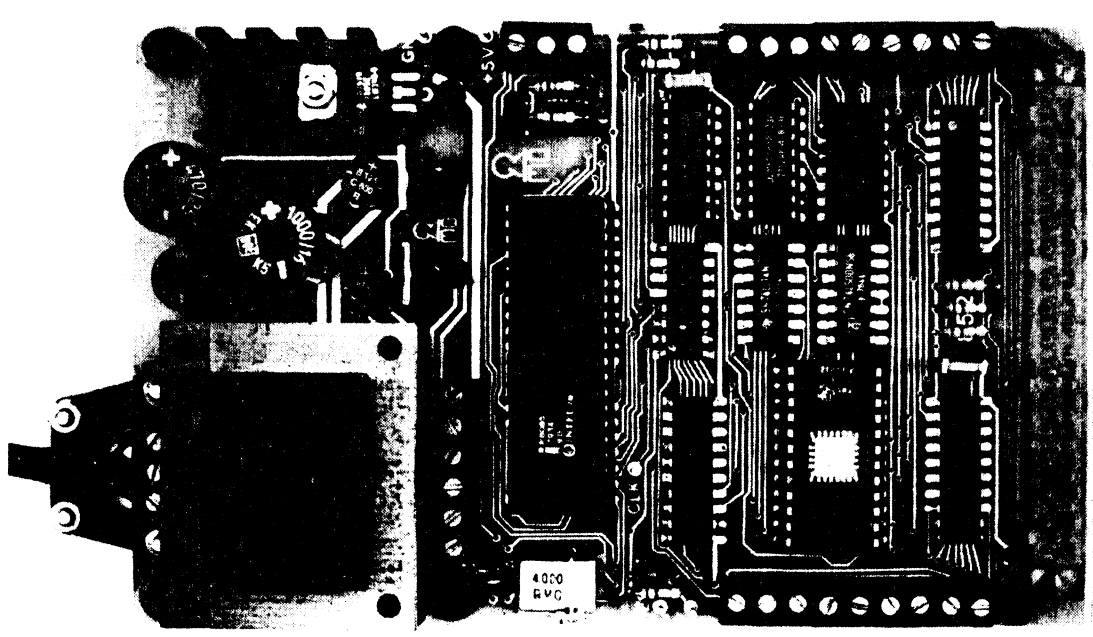
n	0	1	2	3	4	5	6	7
RST n Sprungziel	C7 ØØØØ	CF ØØØ8	D7 ØØ1Ø	DF ØØ18	E7 ØØ2Ø	EF ØØ28	F7 ØØ3Ø	FF ØØ38

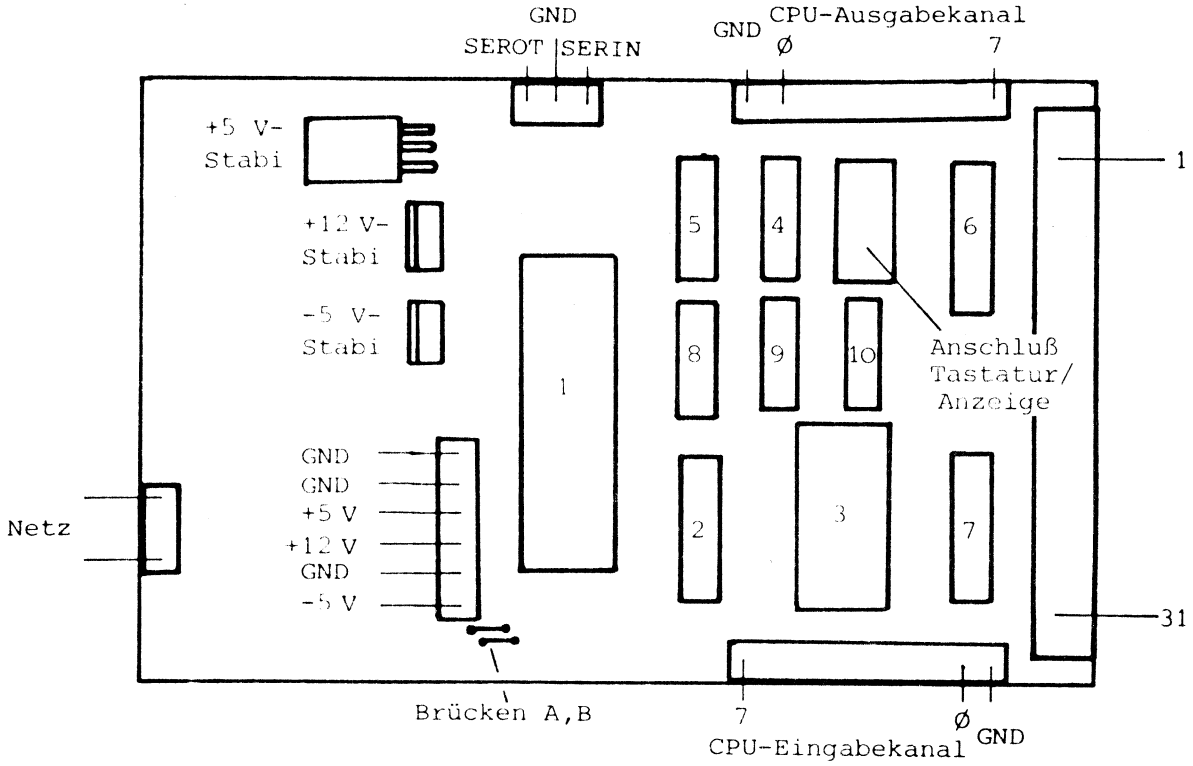
ÜBRIGE

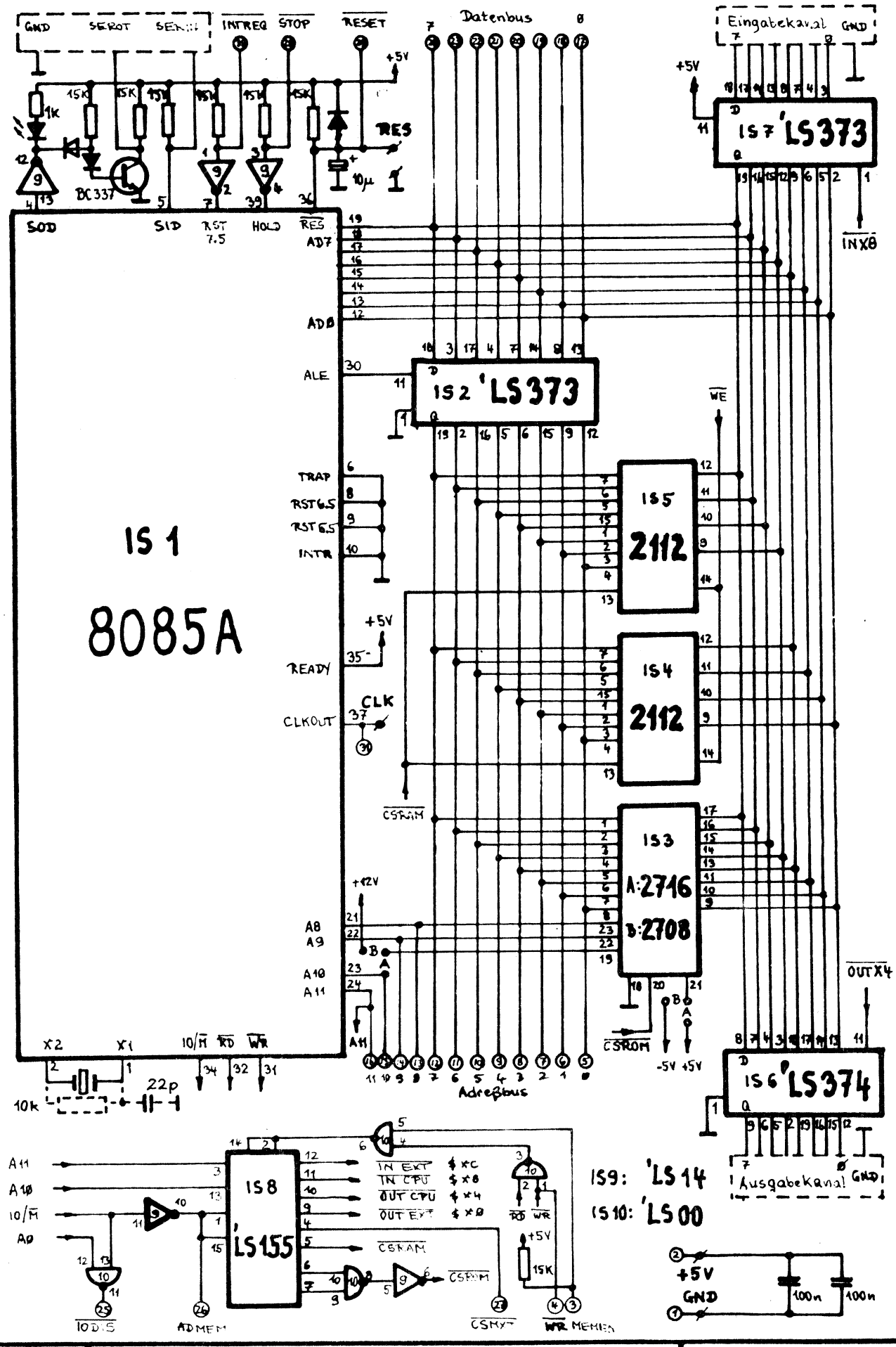
NOP 00	STC 37	EI FB	RIM 20
HLT 76	CMC 3F	DI F3	SIM 30

8085-Maschinenbefehle

7.4.1 Bestückungsplan

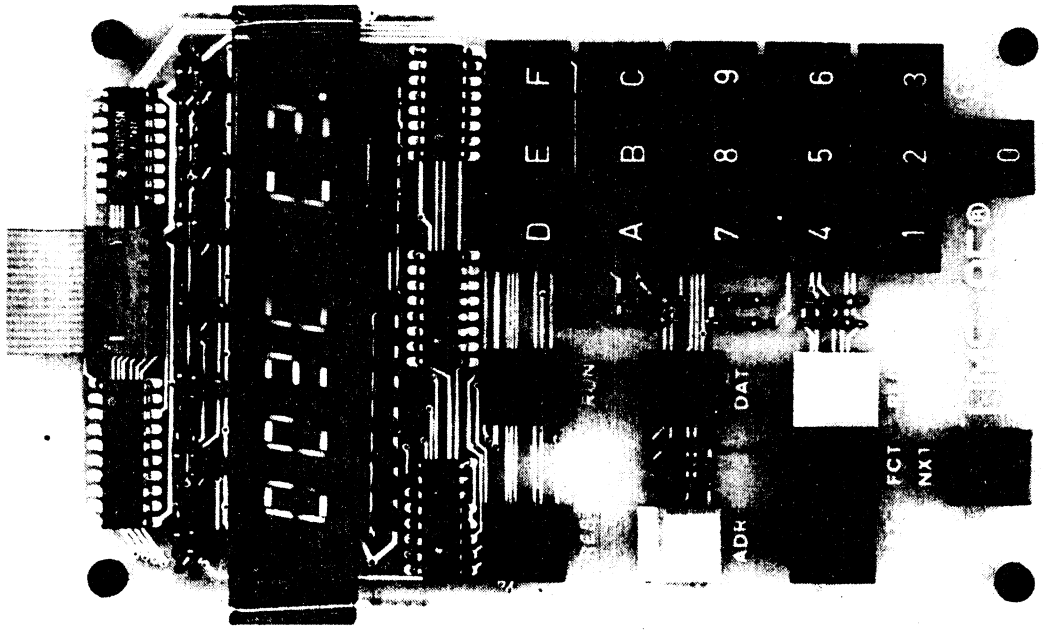


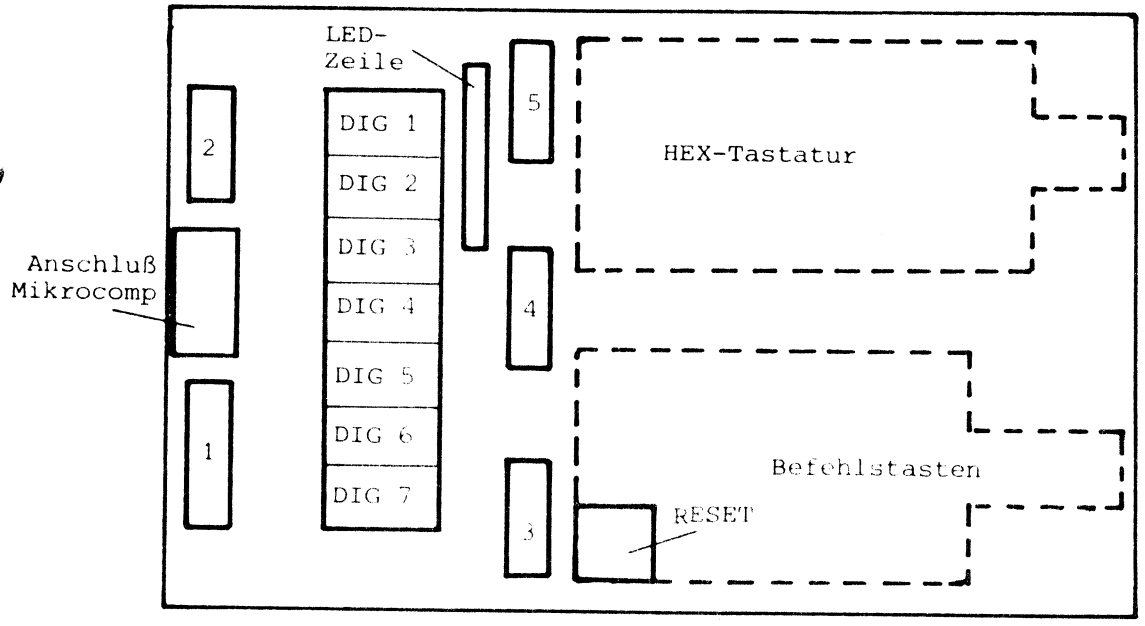


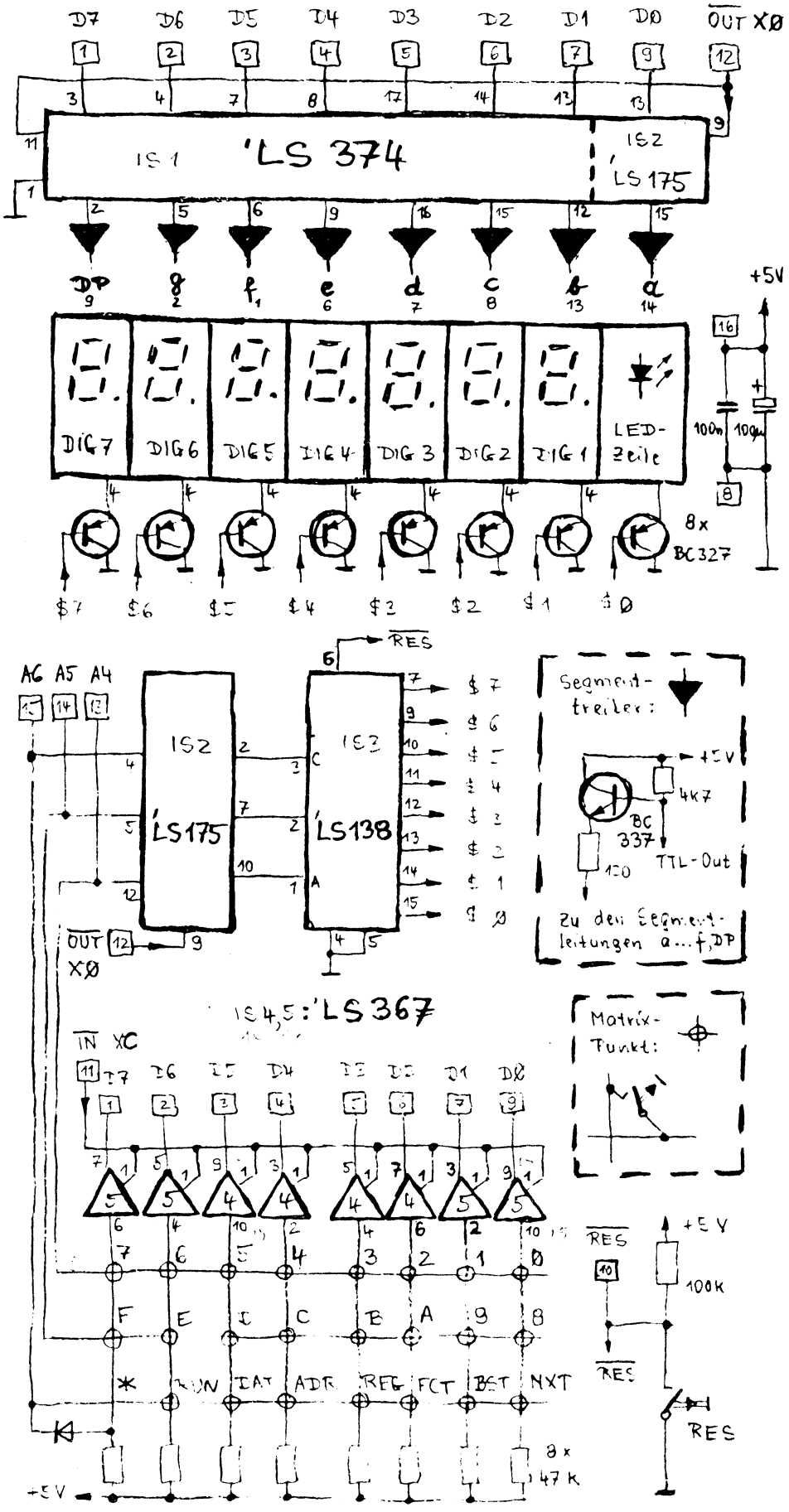


30.11.79	**Mikrocomputer 852** Stromlauf	8523 7.23

7.4.2 Bestückungsplan

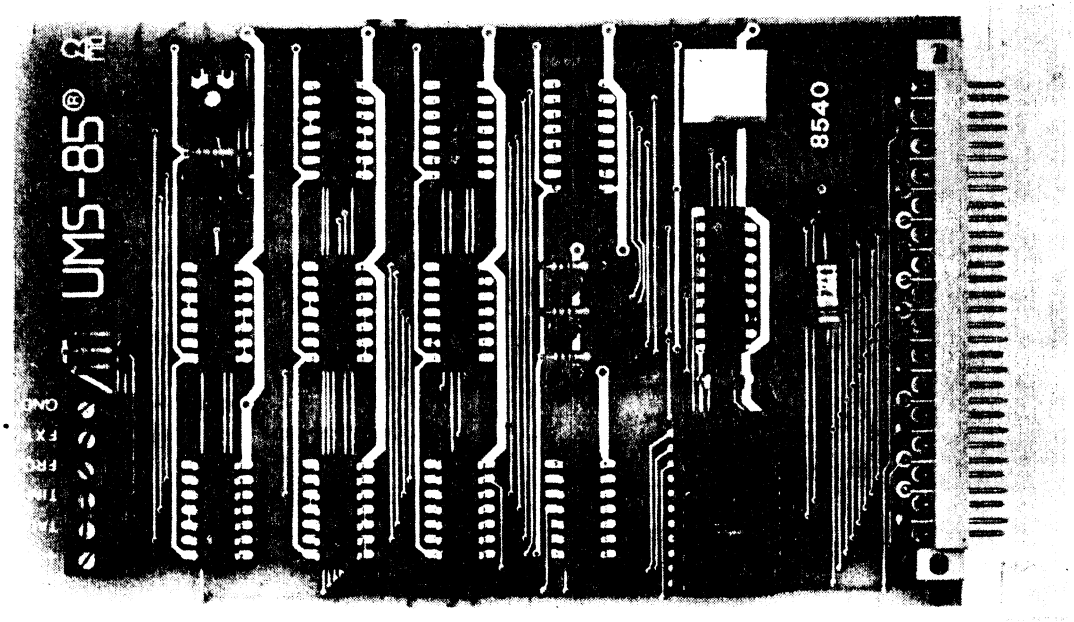


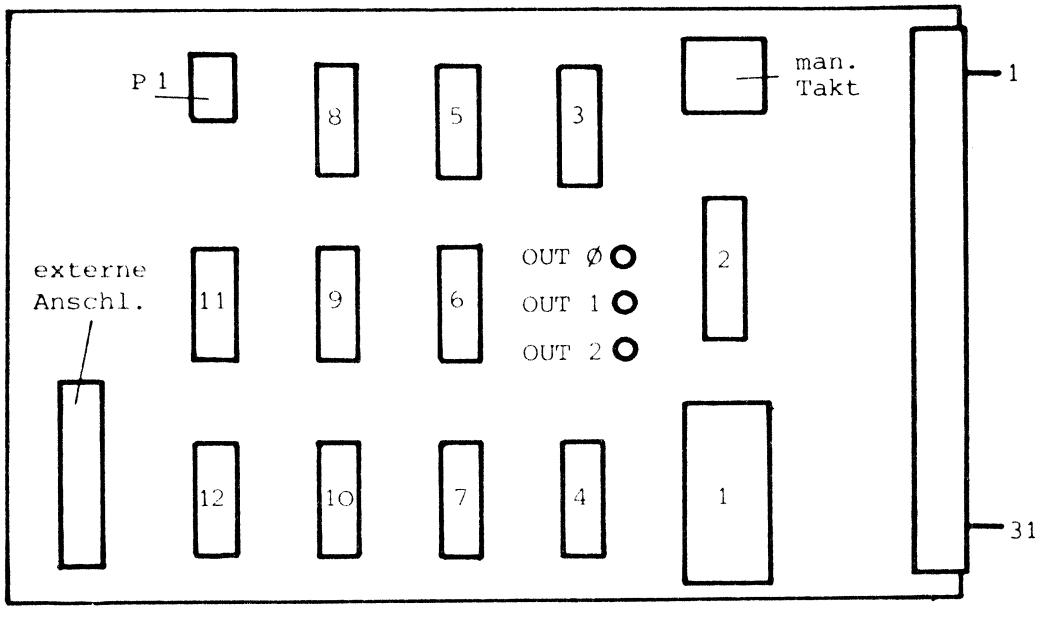


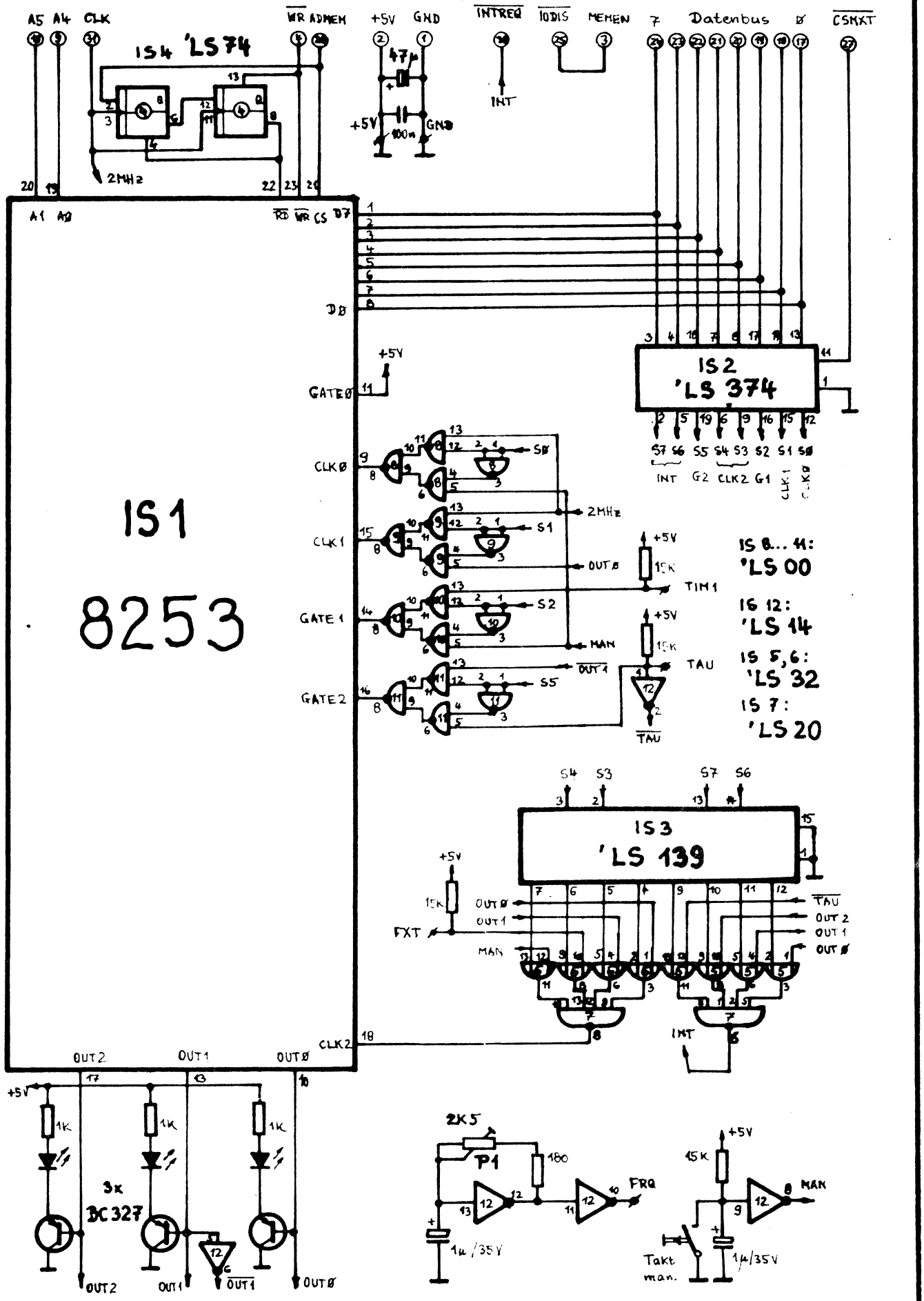


MÖ 25.11.79	**Tastatur/Anzeige 853** Stromlauf	8531 7.25

7.4.3 Bestückungsplan

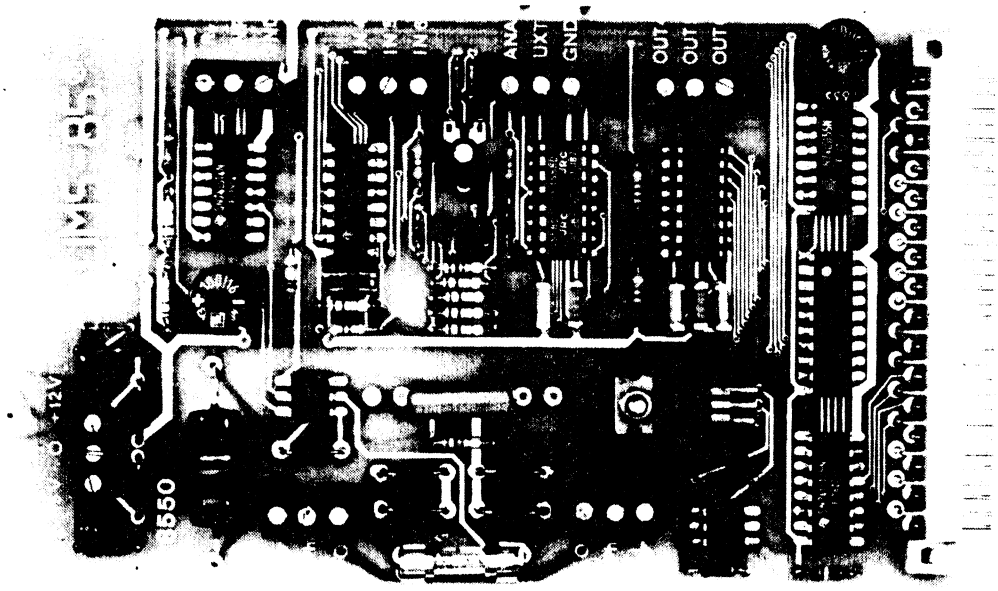


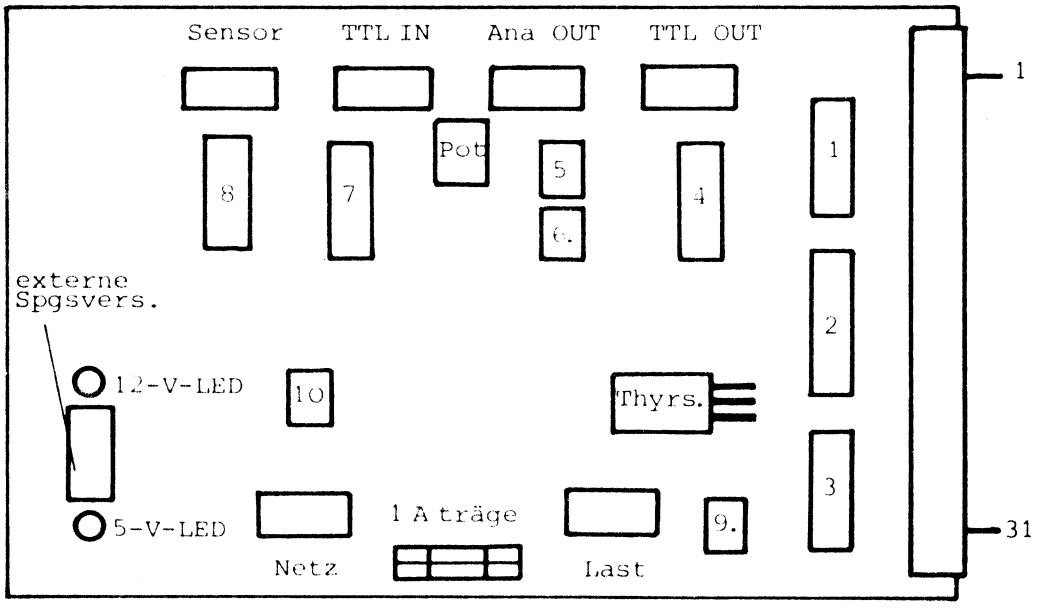


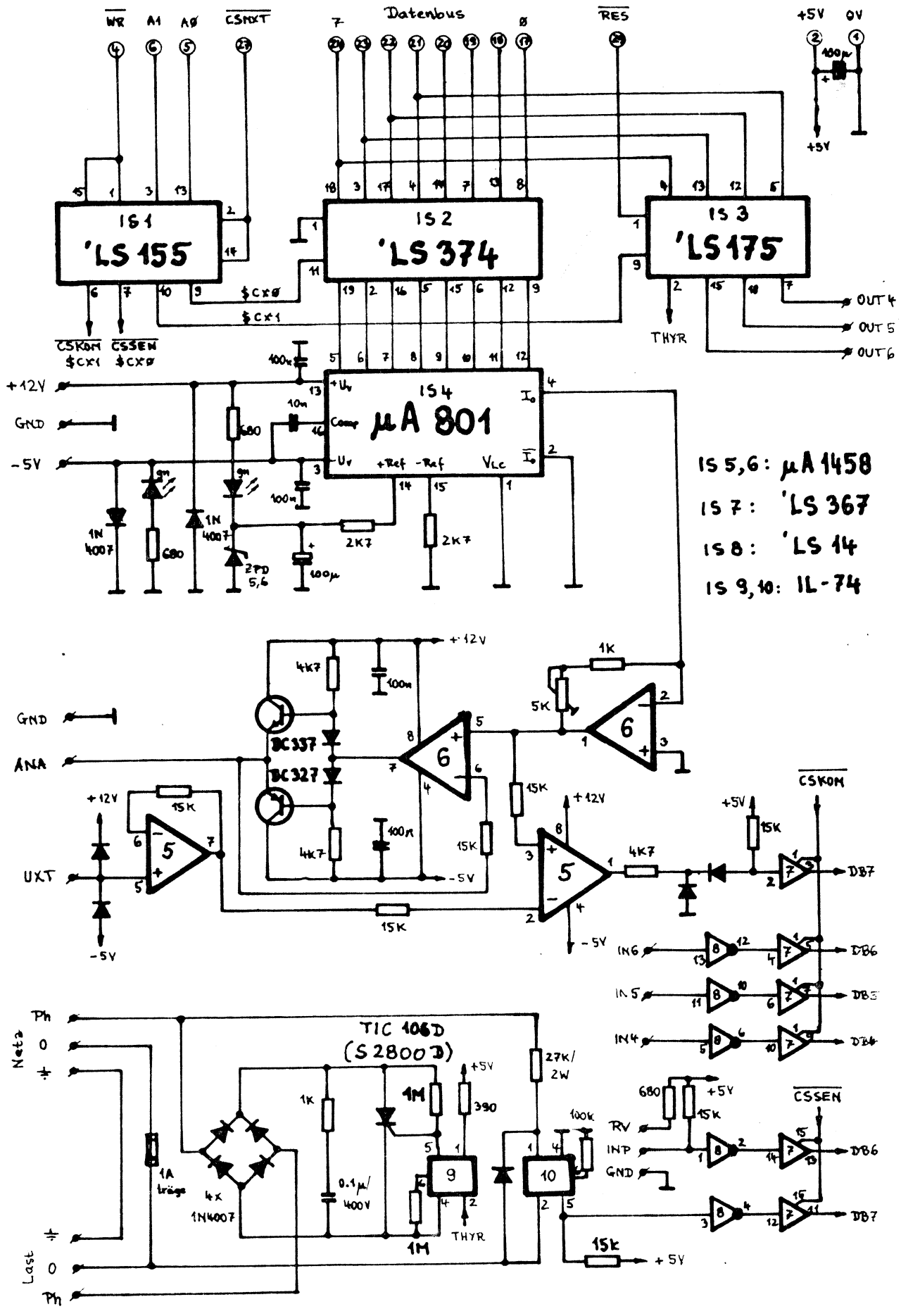


29.11.79	**Timer-Karte 854** Stromlauf	8540 7.27

7.4.4 Bestückungsplan







3. 12. 79	**Analog-Interface 855** Stromlauf	8550 7.29

7.5 Monitor-Unterprogramme

[Ø35A]	INPUT	Byte von der Tastatur einlesen, darstellen und nach NXT in den ACC überschreiben
[Ø34Ø]	NIBIN	Nibble von der Tastatur einlesen, darstellen und nach NXT in den ACC überschreiben
[Ø275]	DSPSW	PSW darstellen; Rücksprung nach Drücken u. Losl. v. NXT
[Ø3A2]	NEXT	fragt NXT ab; ja: CY=1, dann (ØBF8)→ØBFE überschreiben
[Ø11F]	DISPL	überschreibt DSP-Buffer ØBFØ...ØBF7 in die Anzeige (8ms)
[ØØ29]	CLRDS	löscht DSP-Buffer ØBEØ...ØBE7
[ØØBC]	CONVT	Siebensegmentumsetzung:(ØBF8)→ØBF1,2; (ØBF9)→ØBF4,5; (ØBFA)→ØBF6,7
[ØØD5]	BYTE	Siebensegmentumsetzung des (ACC); (D,E):Zieladresse
[ØØE2]	SSGCV	SSG-Umsetzung des unteren ACC-Nibbles;(D,E):Zieladresse
[Ø3FØ]	SSGTB	Siebensegmenttabelle; Balkenmuster für Ø...F
[ØØB2]	SPLIT	ACC=((D,E)); unteres Nibble nach oben und mit (B) verknüpfen; Ergebnis in A und B
[ØØ61]	HHKEY	HEX-Tastatur abfragen; ja:Tasten-Nr. in A und CY=1
[Ø165]	CCKEY	CMD-Tastatur abfragen; ja:Tasten-Nr. in A und CY=1
[Ø1Ø8]	RELSH	auf Loslassen einer HEX-Taste warten und Anzeige aktivieren
[Ø193]	RELSC	auf Loslassen einer CMD-Taste warten und Anzeige aktivieren
[Ø2A1]	SOUND	Tonfrequenzerzeugung an SEROT
[Ø2A4]	MUSIC	Musikerzeugung an SEROT; (H,L):Anfang Notentabelle (H,L)=Ø2EØ: Anfang „Wilhelm Tell"
[Ø3B1]	ONSEC	Software-Zeitverzögerung 1 s
[Ø3BD]	DELAY	Software-Zeitverzögerung 100 ms
[Ø3C7]	DELY1	Software-Zeitverzögerung 1 ms

UMS-85®

Universelles Mikrocomputer-System auf 8085-Basis

Ausführliche Baubeschreibung mit Anwendungsbeispielen ab ELO 1/'81

- Basierend auf dem Industriestandard 8085
  (kompatibel mit dem Mikroprozessor 8080)
- Programmspeicher wahlweise 1 K oder 2 KBytes bestückbar
- Leistungsfähiges Monitor-Programm im EPROM
- Eingebautes Netzteil
- Arbeitsspeicher mit 0,25 KBytes RAM bestückt,
  extern auf 1,25 KBytes erweiterbar
- 8-Bit-Eingabe-Kanal (parallel)
- 8-Bit-Ausgabe-Kanal (parallel)
- Serieller Eingabe-Kanal
- Serieller Ausgabe-Kanal
- Interrupt-Möglichkeit
  sowohl flanken- als auch zustandsgetriggert
- Anschlußmöglichkeit für zusätzliche Interface-Karten
- Europakarten mit Bus-Struktur

- Bausatzpreis: DM 299,00

  kpl. wie oben beschrieben
  Mikrocomputer mit Netzteil, Tastatur und
  Siebensegmentanzeige, einschließlich Versandkosten und Mehrwertsteuer

- Fertiggerät: DM 368,00

  kpl. montiert und getestet,betriebsbereit

- Bestellung: drei Möglichkeiten

  ▶ Verrechnungsscheck einsenden
  Einschreiben nicht erforderlich; Einlösung erfolgt selbstverständlich erst nach Auslieferung der Ware. Risikolos, da Sie den Scheck verfolgen können

  ▶ Banküberweisung des Betrages
  Konto-Nr. 58 09 00, Bremer Landesbank (BLZ 290 500 00) Absender vermerken!

  ▶ Nachnahme-Bestellung; Mehrkosten DM 2,70

UMS-85®

PREISLISTE UMS-85-KOMPONENTEN

		Bausatz DM	Fertiggerät DM
1	Mikrocomputer UMS-85 einschließlich Netzteil ±5 V/+12 V	155,00	188,00
2	Mikrocomputer UMS-85 ohne Netzteile und Trafo	146,00	179,00
3	Tastatur/Anzeige-Platine	144,00	180,00
4	Timer-Platine drei 16-Bit-Zähler/Zeitgeber, Betriebsarten und Taktbeschaltung universell programmierbar	136,00	149,00
5	Analog-Interface D/A- und A/D-Wandlung, Thyristor für Phasenanschnittsteuerung, E/A-Kanäle	153,00	188,00
6	Matrixdrucker mit Interface 21 Zeichen, Normalpapier, 12-V-Anschluß	399,00	592,00
7	Netzteil 5...15 V/2,5 A Trafo nach VDE, geeignet zum Betrieb des Druckers	99,00	a.A.
8	Bus- und Speichererweiterung zusätzlich 1 KBytes RAM und Aufnahme der Karten 3, 4 und 5	167,00	192,00
9	Zubehör-Satz zum Experimentieren Lautsprecher, Vielfach-Instrument, Meßschnüre, Abgleichschraubendreher u.a.m.	----	49,00
10	Kursunterlagen UMS-85 280 Seiten mit Anleitungen zur Selbstprogrammierung; ausführliche Vorstellung des Befehlssatzes, Schaltbilder	----	a.A.
11	Cassetten-Interface (in Kürze lieferbar)	59,00	75,00

Komplettbausätze einschließlich aller Bauteile

Baugruppen komplett montiert und geprüft, ohne Gehäuse

Preise inkl. Mehrwertsteuer zzgl. Versandkostenanteil

Stand 1/81

UMS-85®

S O N D E R A N G E B O T E

LIEFERUNG NUR SOLANGE VORRAT REICHT; PREISE INKL. MWST.

Super-LED-Anzeigen HP-5082-7300 — DM 44,50
interner 4-Bit-Speicher, Decoder und Treiber
exzellente Lesbarkeit durch Punktmatrix-Darstellung (vgl. ELO-Serie „Einfache Logik")

CMOS-RAM 2K x 8 — DM 139,00
kompatibel zum EPROM 2716; dadurch anstelle dessen einsetzbar zur Speicherung von Daten, die batteriegepuffert werden sollen (I_V=1µA!)

Siebensegment-Decoder/Treiber 9368 — DM 7,50; ab 5 St. DM 6,95
Ansteuerung von Siebensegmentanzeigen ohne Vorwiderstände! Interner 4-Bit-Speicher, Decoder und Stromtreiber für gemeinsame Katode

Elko 4700 µF/16 V — DM 1,95
Überbestände 1. Wahl; liegende Bauform

Druckwerk, 40stellig — DM 249,00
ähnlich dem in ELO 9+10/81 beschriebenen; verwendet Normalpapier und Farbband, daher ausgezeichnete Lesbarkeit; inkl. Controller

Trafo M65, 7,5 V/ 5 A — DM 35,00
Industriequalität, nach VDE ausgeführt; für Dauerbetrieb ausgelegt; inkl. Montageleiste und Stehbolzen mit Innengewinde

Plastikgehäuse 20 x 50 x 100 mm³ — DM 7,50
geeignet zur Aufnahme kleiner Elektronikschaltungen; Farbe hell/dunkelgrau (Deckel/Unterteil); inkl. Montageschrauben

8poliges Flachbandkabel, farbig — je m DM 1,00
acht farbig gekennzeichnete Leiter 0,25 mm²

Crimp-Kontakte mit 8poliger Fassung — DM 1,50
zur Herstellung von Verlängerungen/Steckbuchsen

Crimp-Zange zur Verarbeitung von Crimp-Kont. — DM 19,80
Werkzeug für die Verarbeitung der oben angebotenen Crimp-Kontakte; absolut sichere Verbindung ohne Löten!

Oszillograf — DM 275,00
Hobby-Oszillograf mit heller 7-cm-Röhre, Ablenkfrequenz max. 100 kHz, Abschwächung 1/10/100-fach; 220-V-Anschluß

Zusätzlich zu der ausführlichen Baubeschreibung in ELO 1/1981 beachten Sie bitte folgende Punkte:

1. Die beiden schwarzen 16poligen Fassungen für die Verbindung des Flachbandkabels verwenden (Anzeigen-Platine oben Mitte und CPU-Platine unter den RAMs).

   Die IC-Stecker des Flachbandkabels halten besser in den schwarzen Fassungen.

2. Auf der Anzeigen-Platine bitte die Germanium-Diode (AA112 o.ä.) nachlöten, wie unten skizziert.

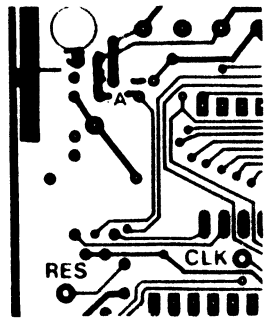
   Diese Diode ist für die Betriebsart „Register-Mode" erforderlich.

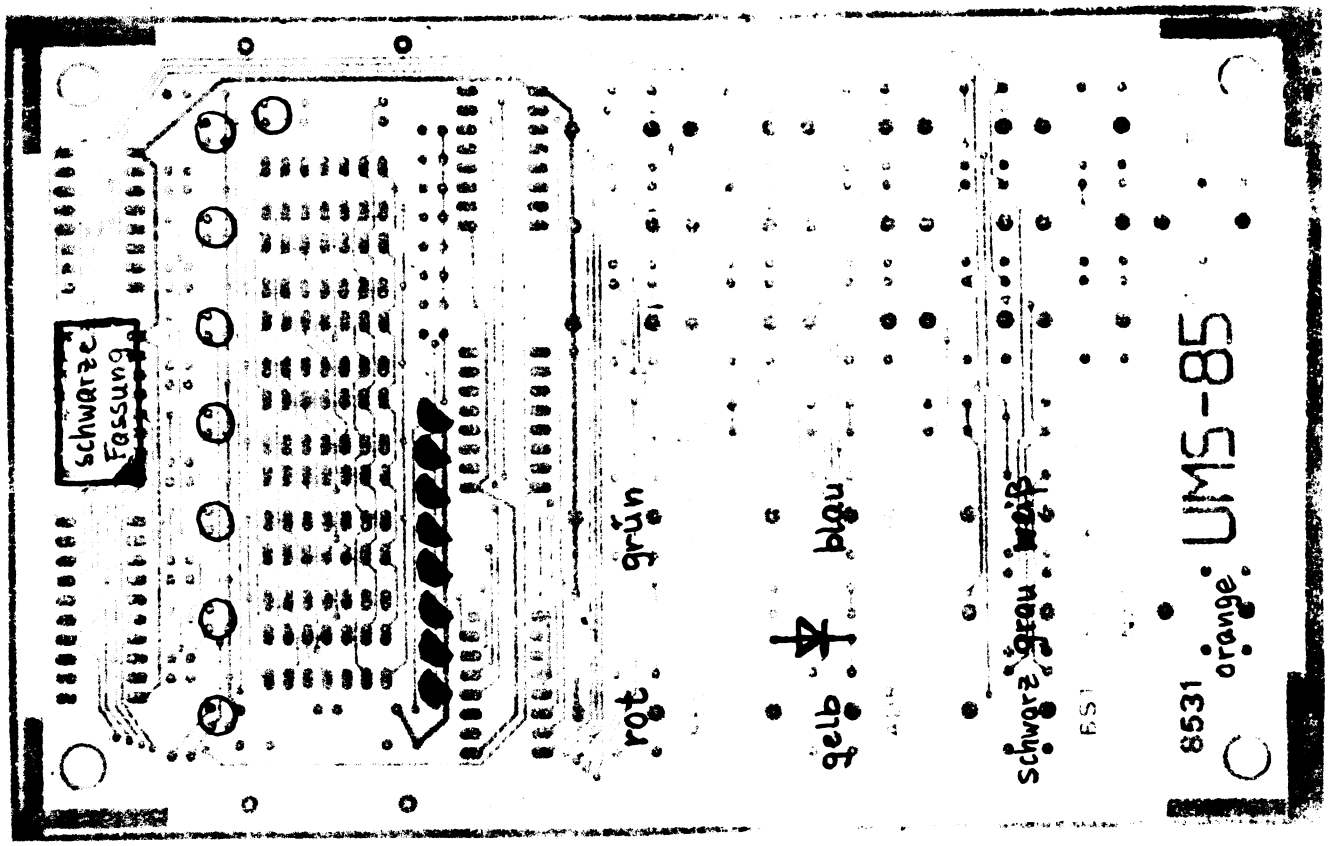
3. Die npn-Transistoren haben die runden Köpfe (BC208); Sie können sie also problemlos von den pnp-Typen (BC327) unterscheiden.

4. Die Tastenfarben erkennen Sie aus der Skizze unten; sie sind zwar willkürlich, aber die ELO-Beiträge beziehen sich auf die unten angegebene Verteilung.

5. Zwei Brücken „B" einlöten (für 2708; beim 2716-EPROM zwei Brücken „A" einlöten)

6. Die 14poligen Fassungen für die Anzeigen verwenden.





30.12.80	Ergänzung zur ELO-Bauanleitung	

1	1	CPU 8085
2	1	EPROM 2708 bzw. 2716
3	2	RAM 2112
4	1	74LS00
5	1	74LS04 oder 74LS14
6	1	74LS138
7	1	74LS155
8	1	74LS175
9	2	74LS367
10	2	74LS373
11	2	74LS374
12	7	Anzeigen MAN74A o.ä.
13	1	Stabi +5 V (LM340T5.0 oder 7805UCT)
14	1	Stabi -5 V (7905AUC)
15	1	Stabi +12 V (LM340 T 12 oder 7812UCT)
16	2	Brückengleichrichter B40C1000 o.ä.
17	9	npn-Transistoren BC208 o.ä.
18	8	pnp-Transistoren BC327 o.ä.
19	9	LEDs, rot, Ø3 mm CQY 54
20	2	Si-Dioden 1N4148 o.ä.
21	2	Ge-Dioden AA116 o.ä.
22	1	Quarz HC-18/U 4,000 MHz
23	1	Becherelko 1000 uF
24	1	dito 470 uF
25	1	dito 100 uF
26	1	Elko ca. 47 uF
27	4	Tantalperlen
28	3	MKT-Kondensatoren 100 nF, RM7,5
29	8	Widerstand SBB 120 Ohm
30	1	dito 1 kOhm
31	8	dito 4,7 kOhm
32	7	dito 15 kOhm
33	8	dito 47 kOhm
34	1	dito 100 kOhm
35	1	Fassung 40polig
36	1	dito 24polig
37	2	dito 16polig
38	7	dito 14polig
39	2	Texas-Fassung 16polig 084 1602
40	11	Gehäusefüße
41	4	Zugentlastung
42	1	Isolierscheibe für Netzanschluß
43	2	Lötstützpunkt Ø1,3 mm (für „+5 V" und „GND")
44	1	Isolierscheibe für Quarz
45	5	Zylinderkopfschraube M3
46	5	Mutter M3
47	24	REK-Schaltteil (Tastenunterteil)
48	24	REK-Kappe (Tastaenoberteil)
49	10	Anschlußklemme dreipolig 953GDS
50	1	Kühlkörper K 18.1-6
51	1	Buchsenleiste 31polig 48.1630
52	1	Filterscheibe rot
53	1	Platine 8523
54	1	Platine 8531
55	1	Nettzzuleitung mit Eurostecker
56	1	Trafo
57	1	Kabelgarnitur 16polig, 22 cm

7.2.1981

Mikrocomputer-System UMS-85

Stückliste Mikrocomputer und Tastatur/Anzeige

7.5 Monitor-Unterprogramme

[035A]	INPUT	Byte von der Tastatur einlesen, darstellen und nach NXT in den ACC überschreiben
[0340]	NIBIN	Nibble von der Tastatur einlesen, darstellen und nach NXT in den ACC überschreiben
[0275]	DSPSW	PSW darstellen; Rücksprung nach Drücken u. Losl. v. NXT
[03A2]	NEXT	fragt NXT ab; ja: CY=1, dann (ØBF8)→ØBFE überschreiben
[011F]	DISPL	überschreibt DSP-Buffer ØBFØ...ØBF7 in die Anzeige (8ms)
[0029]	CLRDS	löscht DSP-Buffer ØBEØ...ØBE7
[00BC]	CONVT	Siebensegmentumsetzung: (ØBF8)→ØBF1,2; (ØBF9)→ØBF4,5; (ØBFA)→ØBF6,7
[00D5]	BYTE	Siebensegmentumsetzung des (ACC); (D,E):Zieladresse
[00E2]	SSGCV	SSG-Umsetzung des unteren ACC-Nibbles; (D,E):Zieladresse
[03FØ]	SSGTB	Siebensegmenttabelle; Balkenmuster für Ø...F
[00B2]	SPLIT	ACC=((D,E)); unteres Nibble nach oben und mit (B) verknüpfen; Ergebnis in A und B
[0061]	HKEY	HEX-Tastatur abfragen; ja:Tasten-Nr. in A und CY 1
[0165]	CKEY	CMD-Tastatur abfragen; ja:Tasten-Nr. in A und CY 1
[0108]	RELSH	auf Loslassen einer HEX-Taste warten und Anzeige aktivieren
[0193]	RELSC	auf Loslassen einer CMD-Taste warten und Anzeige aktivieren
[02A1]	SOUND	Tonfrequenzerzeugung an SEROT
[02A4]	MUSIC	Musikerzeugung an SEROT; (H,L):Anfang Notentabelle (H,L)=Ø2EØ: Anfang „Wilhelm Tell"
[03E1]	ONSEC	Software-Zeitverzögerung 1 s
[03E5]	DELAY	Software-Zeitverzögerung 100 ms
[03C7]	DELY1	Software-Zeitverzögerung 1 ms

obere Adreßbyte | untere Adreßbyte
0800 ← Startadresse
zuerst | danach

Funktionen des 2-K-Monitors

- FCT-0: SEROT (Cassetten-Ausgabe)

  nach 0BFB/C die Startadresse und nach 0BFD/E die Stopadresse des zu übertragenden Datenblocks eingeben

- FCT-1: SERIN (Cassetten-Eingabe)

  nach 0BFB/C die Startadresse und nach 0BFD/E die Stopadresse des zu übertragenden Datenblocks eingeben; SERIN im 10-s-Vorspannbereich starten (hoher Dauerton). Fehlerfreies Einlesen: Anzeige 0800 XX; andernfalls Fehlermeldung „Err"

- FCT-2: LIST (Speicherabzug ausdrucken)

  nach 0BFB/C die Startadresse des Datenblocks und nach 0BFD die Anzahl der gewünschten Druckzeilen eingeben (pro Druckzeile Ausdruck von acht Bytes)

- FCT-3: SCHAM (Schaltautomat)

  nach 0BFB/C die Startadresse des Datenbuffers und nach 0BFD die Schrittzahl (hexadezimal) eingeben

  - automatisches Weiterschalten im Sekundentakt: SERIN=HIGH (offen)
  - Weiterschalten bei externem Trigger: SERIN=LOW; zum Fortschalten ein beliebiges Bit des CPU-Eingabe-Kanals kurzzeitig ab Masse legen

- FCT-4: DVM (Digitalvoltmeter)

  Betrieb des Analog-Interfaces als Digitalvoltmeter (Anzeige 0,0...9,9 V) Eingangsspannung an UXT (max.10 V), Ausgang zuvor bei „FF" auf 10 V eichen (vgl. ELO 5+6/81)

- FCT-5: PHONE (automatischer Rufnummerngeber)

  Rufnummer eintasten; zum Starten „D" (=Durchwahl) drücken; zum Löschen „E" (=Erase) drücken. Relais-Anschlüsse siehe ELO 7+8/81

- FCT-6: DIGUR (Digitaluhr)

  nach 0BFB/C Stunden (01 - 05 = 15 h) und nach 0BFD/E Minuten eingeben; Sekunden werden automatisch auf 00 gesetzt

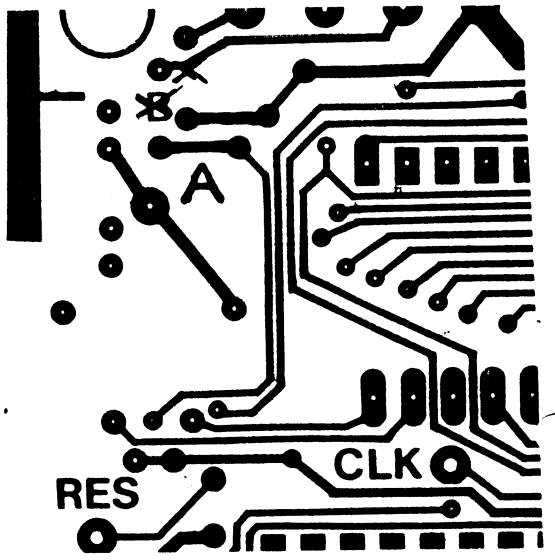
- FCT-7: SPUHR (sprechende Uhr)

  nach 0BFB/C Stunden (01 - 05 = 15 h) und nach 0BFD/E Minuten eingeben; Sekunden werden automatisch auf 00 gesetzt; zum Betrieb entsprechende Interface-Karte erforderlich. Nach 083C ff. eingeben C3 - 10 - 06

  Hinweis: Das Weiterzählen der Uhr erfolgt automatisch per Interrupt (CPU-Belastung unter 3%), so daß nebenbei ein eigenes Hauptprogramm bearbeitet werden kann.

  Dies muß beginnen mit 3E - 1B - 30 - FB (Interrupts freigeben) und muß in 083C die Sequenz JMP 0610 (C3 - 10 . 06) vorfinden; in diesem Fall das Hauptprogramm über RUN starten und nicht über FCT-7-NXT. Die Uhrzeit kann bei Bedarf aus 0BE8...0BEF (Stunden-Zehner...Sekunden-Einer) ausgelesen werden.

Wichtige Hinweise zum Betrieb des 2-K-Monitors



- Das 2-K-Monitor-Programm ist in einem 2716-EPROM enthalten, bei dessen Einsatz die beiden Brücken auf der CPU-Karte unbedingt in Stellung „A" zu verdrahten sind; Umwechseln nur bei abgeschalteter Versorgungsspannung!

- Der 2-K-Monitor stellt die im Datenfeld erscheinende Information (rechtsbündig in der Anzeige) zusätzlich als Bitmuster in der LED-Zeile dar; Sie sehen so unmittelbar den Zusammenhang zwischen hexadezimaler Anzeige und korrespondierendem Bitmuster.

- Die Randbedingungen für die einzelnen Programmteile werden vor dem Starten der Programmteile in die Speicherstellen 0BFB ff. geladen; bei der Eingabe wird stets die höchstwertige Information zuerst eingegeben, also z.B. erst das obere Adreßbyte, gefolgt von NXT und dem unteren Adreßbyte, wiederum gefolgt von NXT. Diese Eingabeform erleichtert Ihnen die Handhabung ganz wesentlich.

- Zum Starten der einzelnen Programmteile drücken Sie einfach NXT (also nicht RUN); zuvor muß die Eingabe FCT (=Function), gefolgt von der betreffenden Funktionsnummer, erfolgt sein.

BUS- UND SPEICHERERWEITERUNG

Bei der Bus- und Speichererweiterung handelt es sich um eine Europakarte, die mit ihrer 31poligen Stiftleiste in die Buchsenleiste der Mikrocomputer-Platine eingesteckt wird. Auf der Karte ist ein RAM-Bereich von 1 KBytes vorhanden, der zusammen mit dem vorhandenen Arbeitsspeicher einen RAM-Bereich von 1,25 KBytes ergibt.

Außerdem sind auf der Erweiterungskarte drei weitere 31polige Buchsenleisten vorhanden, um Interface-Karten aus dem UMS-85-Programm aufzunehmen (Timer, Drucker und Analog-Interface).

Wenn die Erweiterungskarte ohne ein zusätzliches Interface betrieben wird (sie also nur zur Speichererweiterung dient), kann die Stromversorgung von der Mikrocomputer-Platine erfolgen. Bei Betrieb einer oder mehrerer Interface-Karten ist eine zusätzliche Stromversorgung (z.B. aus dem UMS-85-Netzteil) vorzusehen. Die Erweiterungskarte besitzt dazu einen Festspannungsregler mit Kühlkörper.

Bausatzpreis:	DM 167,00 (inkl. MwSt)
Fertig montiert:	DM 192,00 (inkl. MwSt)

Jeweils zuzüglich Versandkostenanteil von DM 5,90

Anschluß der Bus- und Speichererweiterung 8501

Mit den nachfolgend beschriebenen Änderungen bauen Sie den Arbeitsspeicher Ihres UMS-85 auf 1,25 KBytes aus; er belegt dann folgende Adreßbereiche:

X800...XBFF	1024	Bytes	(in den 2114-RAMs)
XF00...XFFF	256	Bytes	(in den 2112-RAMs)

Gleichzeitig ändern sich durch die erweiterte Chip-Select-Erzeugung die Port-Adressen für die Timer-Karte 8540, sofern diese auf der Bus- und Speichererweiterungskarte betrieben wird:

Port-Adresse	Zähler 0:	von 01	auf 81
Port-Adresse	Zähler 1:	von 11	auf 91
Port-Adresse	Zähler 2:	von 21	auf A1
Port-Adresse	Control:	von 31	auf B1

Umrüstung:

1. Versorgungsspannung abschalten.
2. Beide RAMs 2112 aus den Sockeln der CPU-Karte 8523 herausnehmen und in die freien Sockel der Erweiterungskarte einstecken (Markierungskerbe links).
3. Erweiterungskarte mit der 31poligen Stiftleiste in die Buchsenleiste der CPU-Karte stecken.
4. Gelbe Chip-Select-Leitung der Erweiterungskarte mit dem korrespondierenden Stift auf der CPU-Karte verbinden (74LS00, Pins 3/4).

Wenn Sie keine der Interface-Karten (Drucker-Interface, Timer oder Analog-Interface) in dem dafür vorgesehenen Steckplatz der Erweiterungskarte betreiben, kann die +5-V-Stromversorgung für die Erweiterungskarte über Steckverbindungen von der CPU-Karte abgenommen werden (+5 V und GND am rechten Rand).

Beim Betrieb einer oder mehrerer Interface-Karten kann die Stromversorgung nicht mehr direkt von der CPU erfolgen; in diesem Fall müssen Sie eine geeignete 12-V-Versorgung an die Erweiterungskarte anschließen (+12 V und GND am linken Rand).

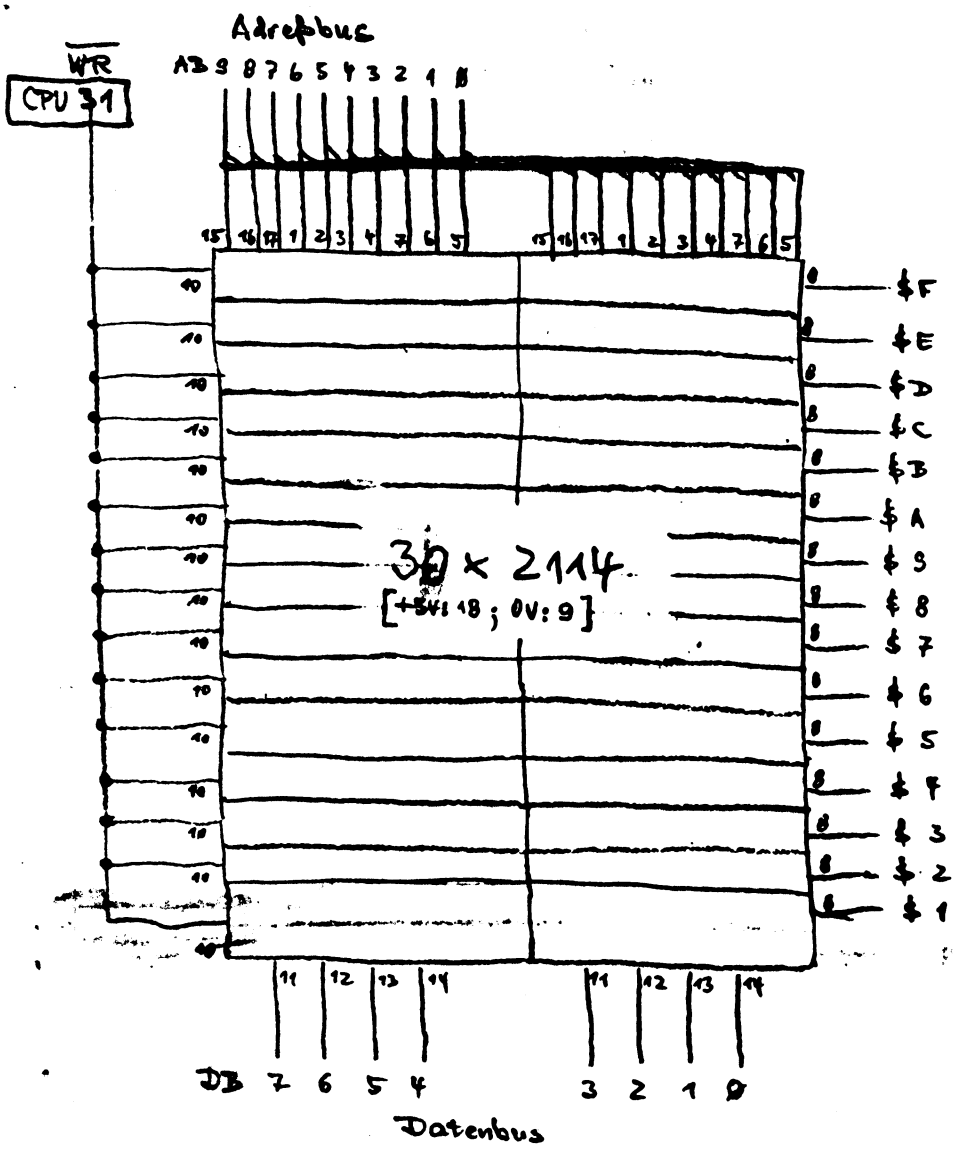
0	2	RAMs 2114
1	1	74LS14 (74LS04)
2	2	74LS00
3	1	74LS155
4	1	7805 (LM340 T5.0)
5	1	Becherelko 470 uF/25 V
6	1	Elko ca. 47 uF/10 V
7	1	MKM-Kondensator 100 nF, RM 7,5/10
8	1	31polige Stiftleiste, gerade Stifte
9	3	31polige Buchsenleiste, abgesetzt
10	2	Schraubklemme, 3polig
11	1	Führungsblech für Platinen
12	2	Fassung, 18polig
13	2	Fassung, 16polig
14	1	Kühlkörper 104527 KL (Assmann)
15	2	Schraube M3 x 25
16	2	Schraube M3 x 15
17	2	Schraube M3 x 10
18	8	Mutter M3
19	8	Federring M3
20	3	Lötstift Ø1,3 mm
21	1	Verbindungskabel, gelb mit Lötbuchse
22	2	Isolierscheiben, Innen-Ø 3,2 mm
23	1	Platine 8501

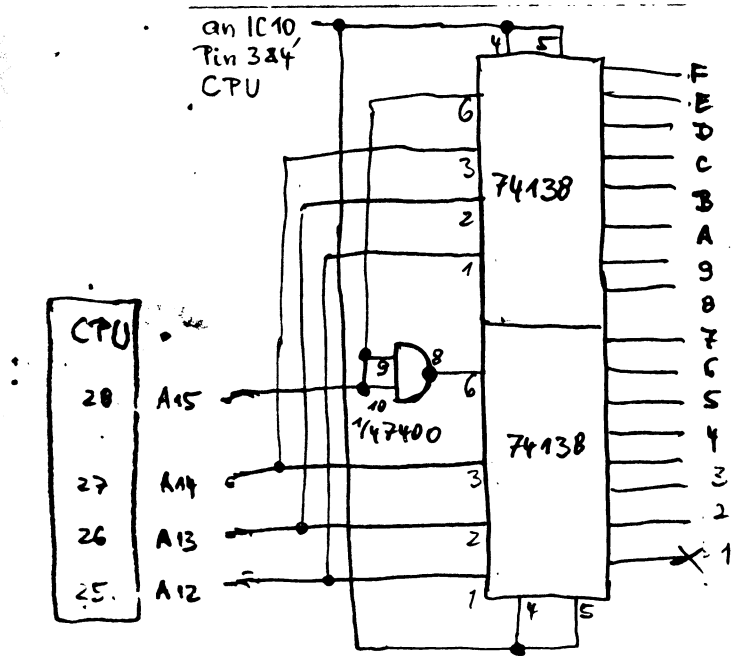
MÖ

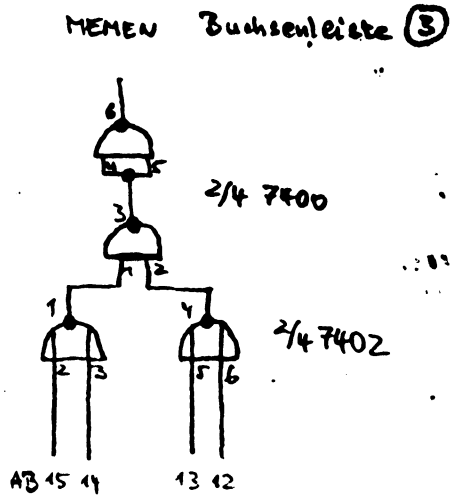
15.12.80

Bus- und Speichererweiterung

Stückliste







15-K-RAM-Ausbau

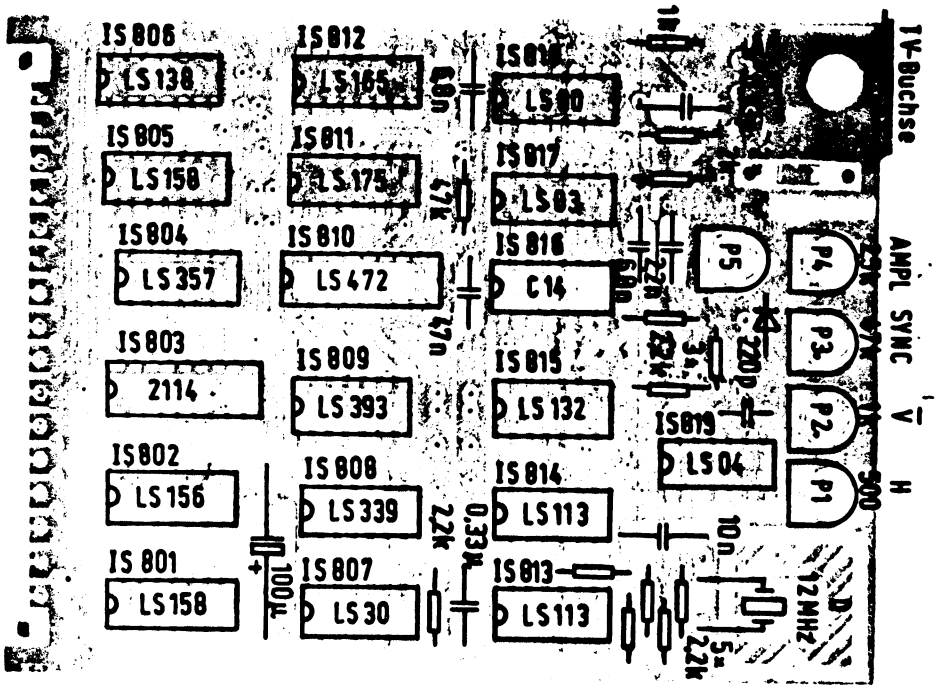
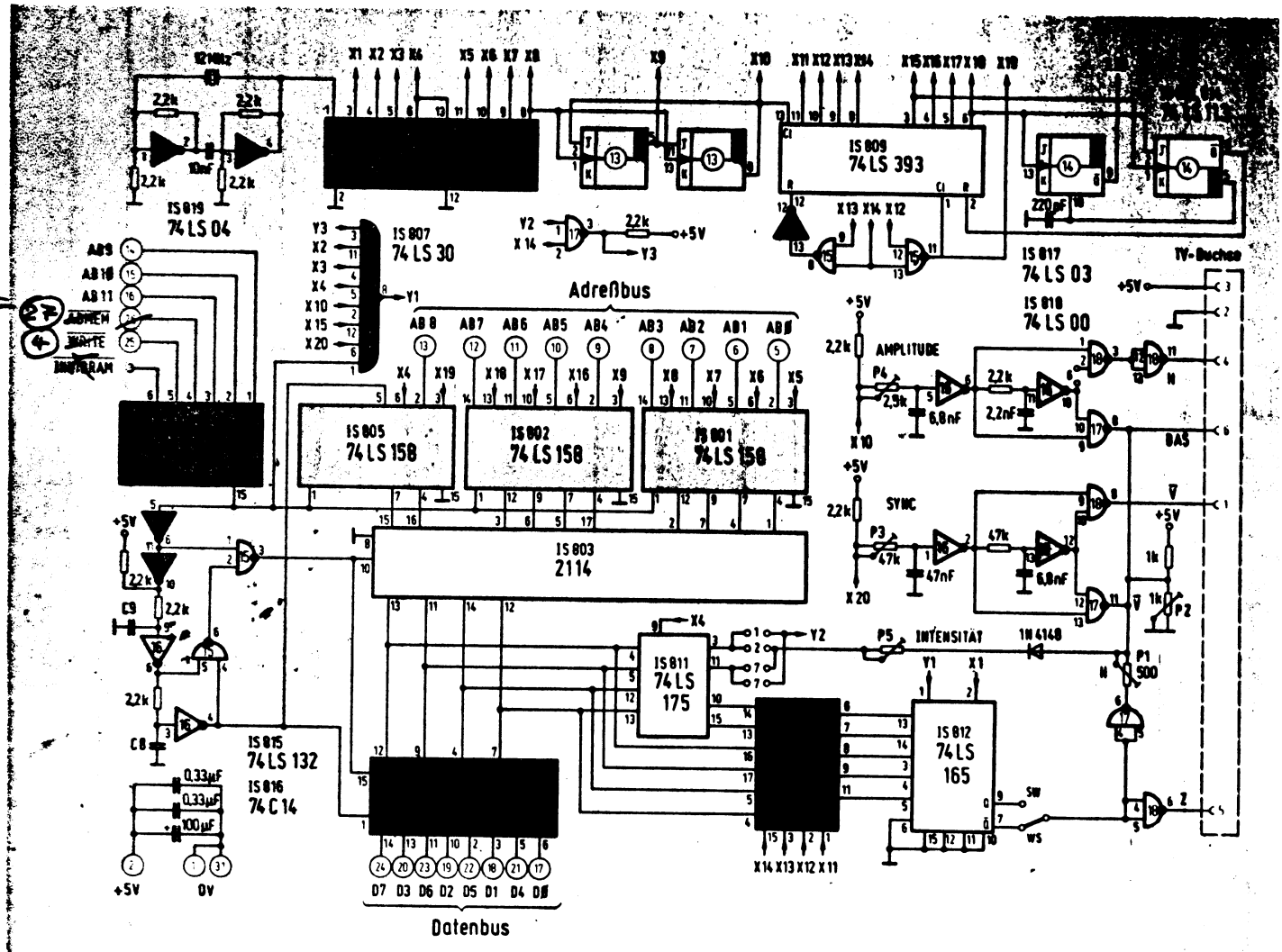
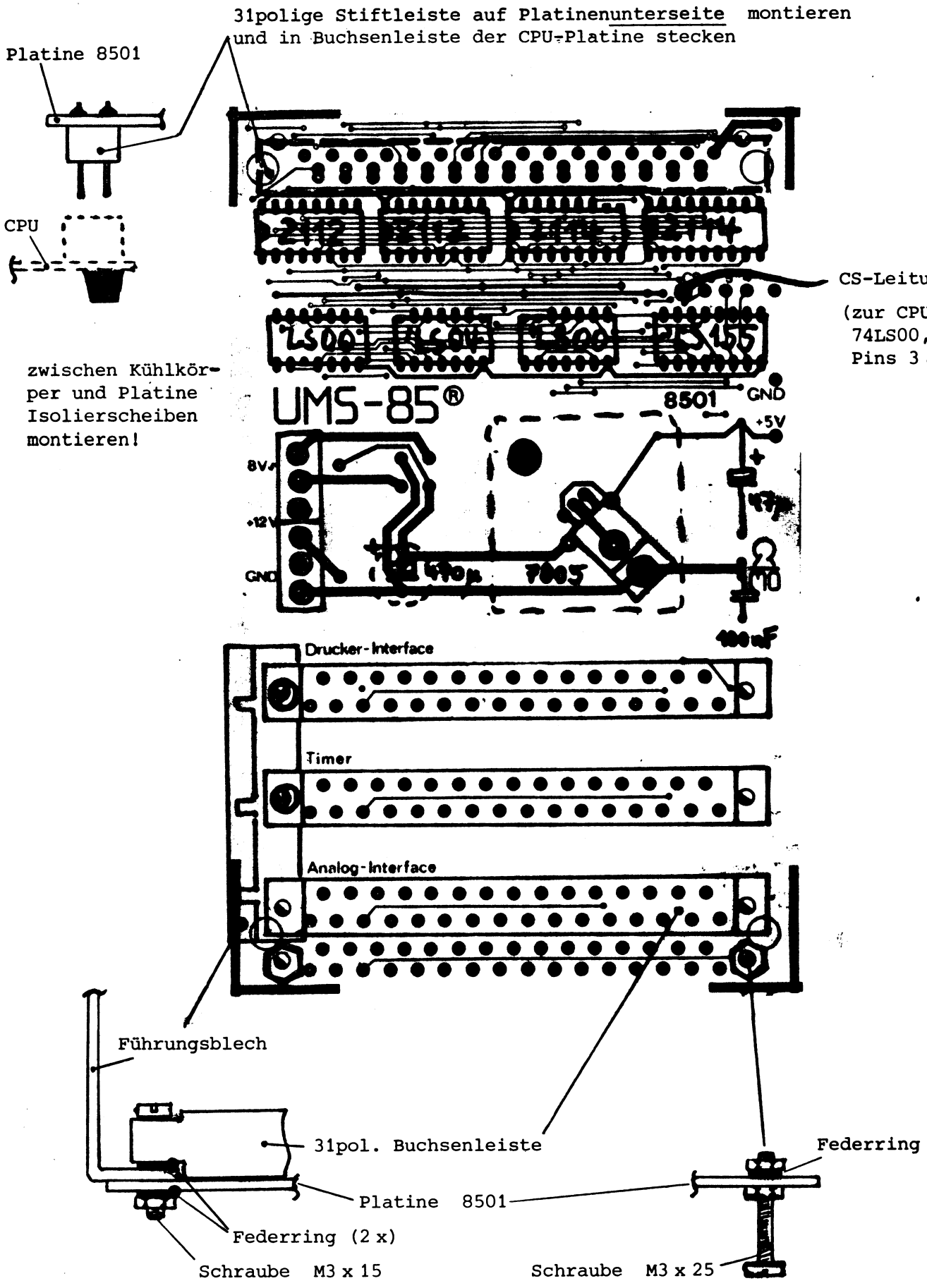


Bild A.19: Video-Interface (Bestückungsplan).

Bild A.20: Schaltbild des Video-Interface'.

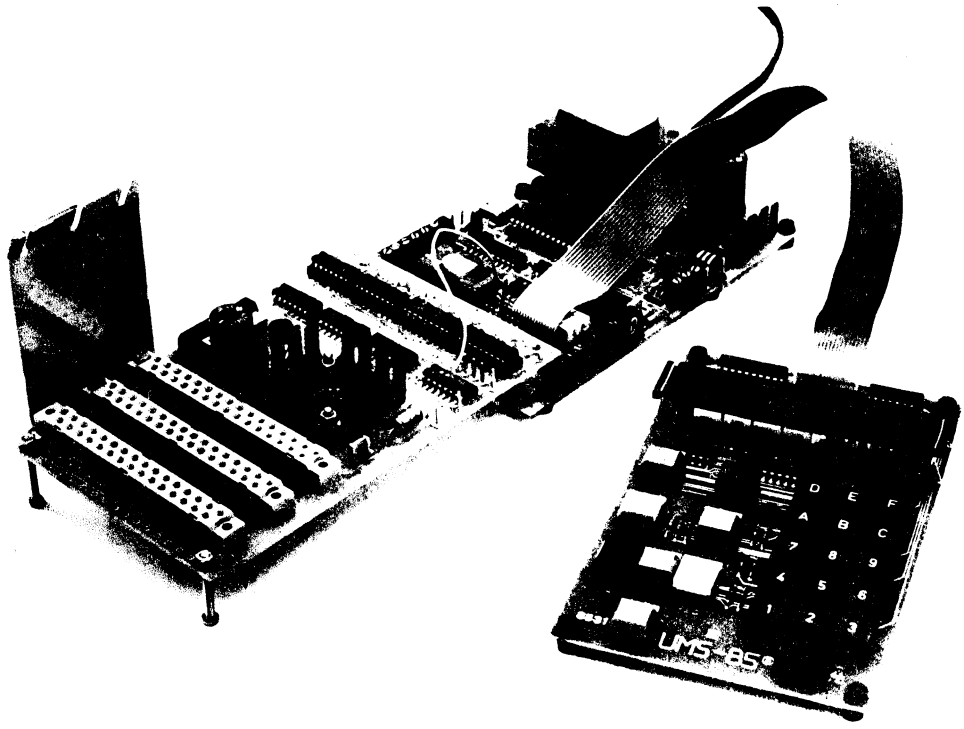


Adressen: C000 H.

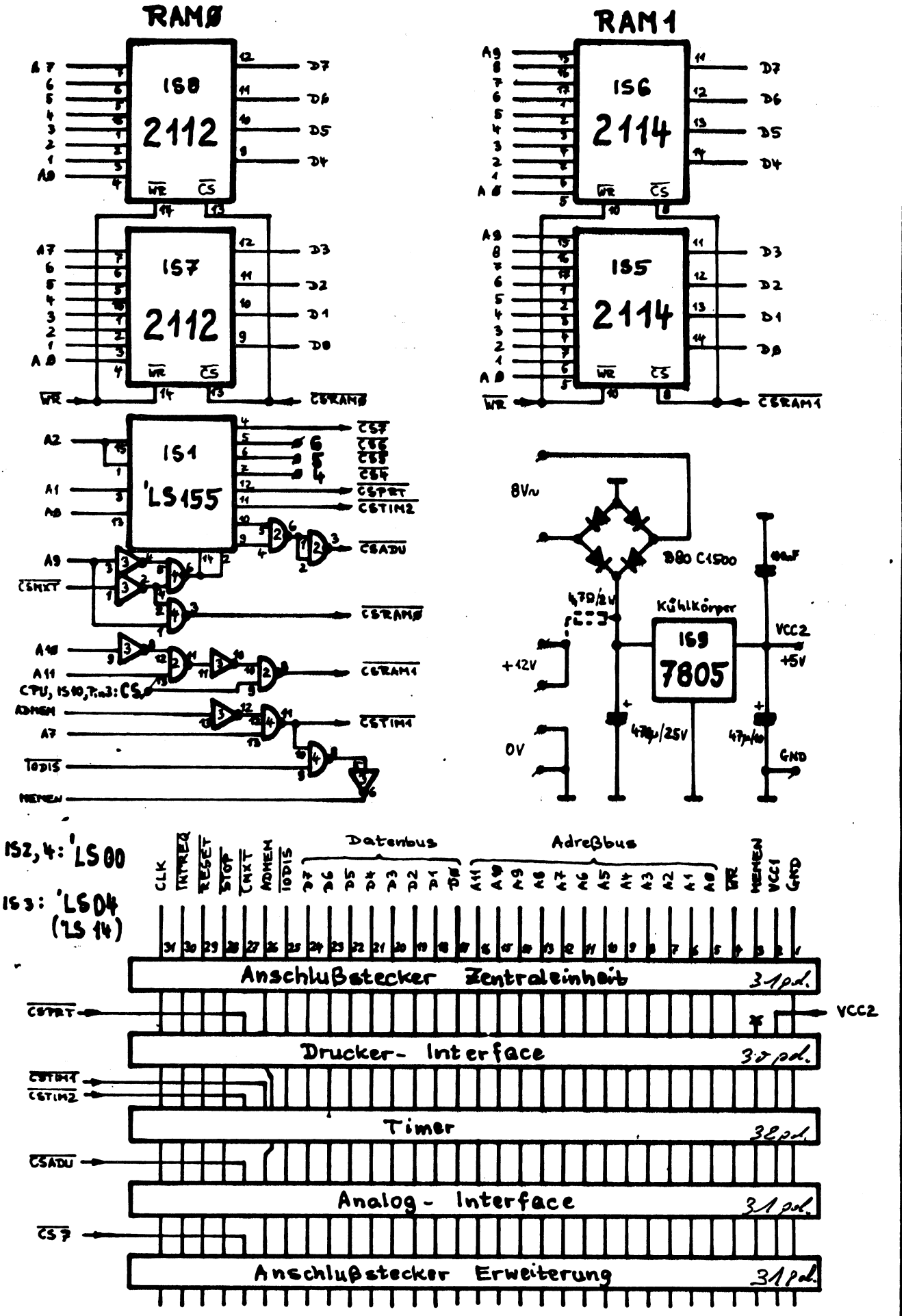


15.12.80	Bus- und Speichererweiterung Bestückungsplan	

UMS-85®



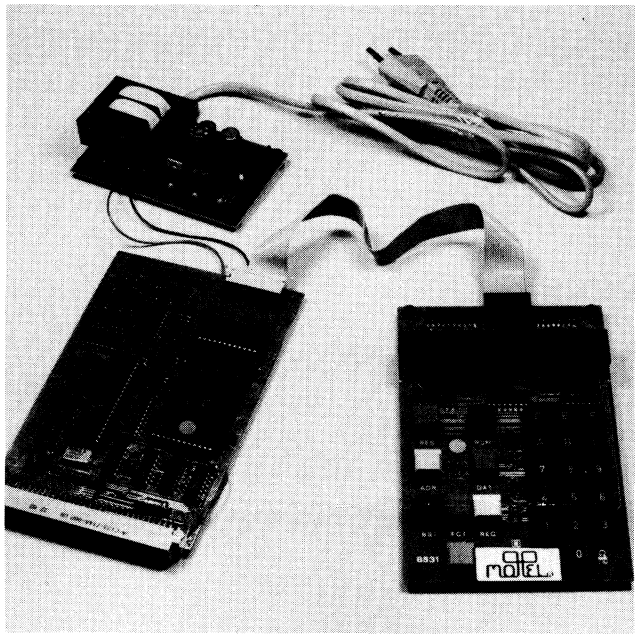
Die Bus- und Speichererweiterung bietet Platz für drei zusätzliche Interface-Karten und baut den System-Arbeitsspeicher (RAM) auf insgesamt 1,25 KBytes aus.

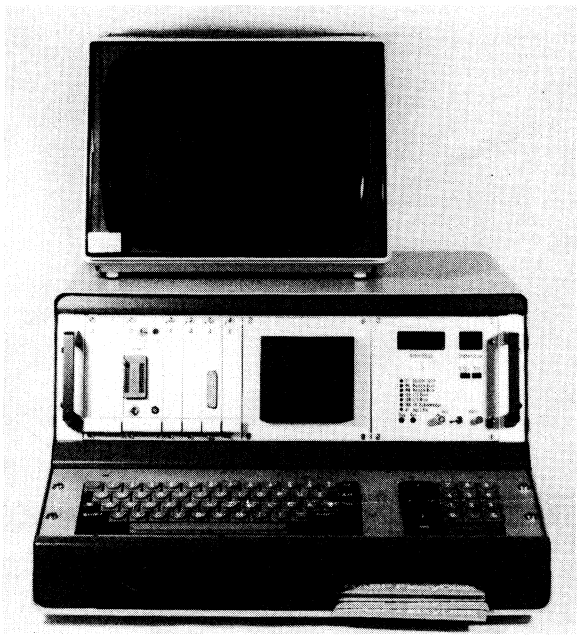


MD	Bus- und Speichererweiterung	8501
18.3.80	Stromlauf	

MOPPEL

Einstieg in die Maschinensprache –
Ausbau bis zum BASIC-System





Preiswerter Einstieg in die Mikrocomputer-Technik – das ist einer der Aufhänger für MOPPEL, das modulare Prozessor-Programm der ELO; Ausbau bis hin zum professionellen Mikrocomputer-System – das ist der andere Aspekt, unter dem dieses System entwickelt wurde. Sie sollen Ihr individuelles System nach und nach, ganz Ihren finanziellen Möglichkeiten angepaßt, zusammenstellen, ohne die zu Anfang erworbenen Baugruppen beim späteren Ausbau wegwerfen zu müssen. Und auf der umfassenden Ausbaufähigkeit dieses Systems lag der Schwerpunkt bei der Entwicklung, die zu keinem Zeitpunkt das Ziel verfolgt hat, ein Billiggerät zu schaffen, das nach kurzer Zeit wegen mangelnder Erweiterungsmöglichkeiten in irgendeiner Ecke landet.

Durch die hier gebotenen vielseitigen Erweiterungskarten ist es ohne weiteres möglich, beim Systemausbau völlig unterschiedliche Schwerpunkte zu setzen. So können sie beispielsweise auf ein handliches Entwicklungssystem hinsteuern, das über effektive Testhilfen verfügt, und das nach einer Programmentwicklung sofort Programmteile oder Daten ins EPROM übernimmt. Oder Sie schaffen sich ein komplettes Datenerfassungssystem, mit dem Sie Umwelt- oder Wetterbedingungen erfassen und per Protokolldrucker dokumentieren, einschließlich der von der internen Echtzeit-Uhr gelieferten Uhrzeit- und Datums-Information. Natürlich können Sie Ihr System auch so gestalten, daß Sie mit dessen Einsatz eine perfekte Terminüberwachung erreichen, angefangen beim Kurzzeit-Wecker bis hin zur Erinnerung an nahende Geburtstage. Und wer sich ausschließlich spielerisch mit seinem MOPPEL beschäftigen will, hat dazu ebenso die Gelegenheit – der modulare Aufbau macht's zu jeder Zeit möglich!

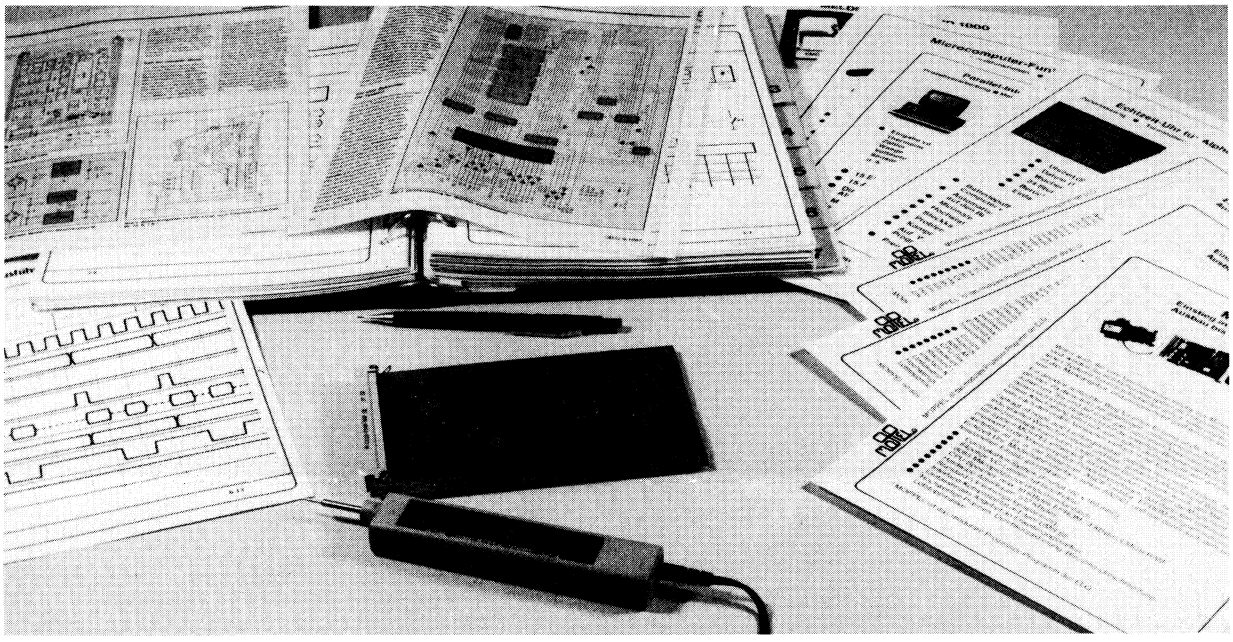
Bereits mit der HEX-Tastatur/Anzeige-Platine (plus CPU) lassen sich Echtzeit-Uhr, EPROM-Programmierzusatz und Thermodrucker ansteuern; zum Einsatz eines TV-Monitors (Bildschirm) ist von vornherein die ASCII-Tastatur sinnvoll, die für die Programmierung in einer höheren Programmiersprache ohnehin erforderlich ist; in diesem Fall benötigen Sie zusätzlich das Video-Interface.

MOPPEL ist das modulare Prozessor-Programm der ELO

hms **Hodenberger Straße 19c · 2800 Bremen 33 · ☎ (04 21) 25 03 47**
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

MOPPEL

Technische Beschreibung der Baugruppen
Das MOPPEL-Handbuch



Für jede Funktionseinheit aus dem MOPPEL-System haben wir ein eigenes Blatt mit einer ausführlichen technischen Beschreibung erstellt. Diese Sammlung informiert Sie über den Entwicklungsstand und die Ausbaumöglichkeiten des Mikrocomputers MOPPEL. Sie erhalten diese umfassenden Informationen gegen Einsendung einer Schutzgebühr von 20,– DM (die bei einer Bestellung angerechnet wird) und eines mit Ihrer Anschrift versehenen Adreßaufklebers für die Rücksendung; die Rückadresse brauchen Sie nicht zu frankieren.

Gleichzeitig werden Sie damit in einen Verteiler aufgenommen, über den Sie **kostenlos** die in Zukunft erscheinenden technischen Beschreibungen zu den später lieferbaren Baugruppen erhalten. Die Schutzgebühr schließt den damit verbundenen Verwaltungsaufwand und die entstehenden Portokosten voll mit ein.

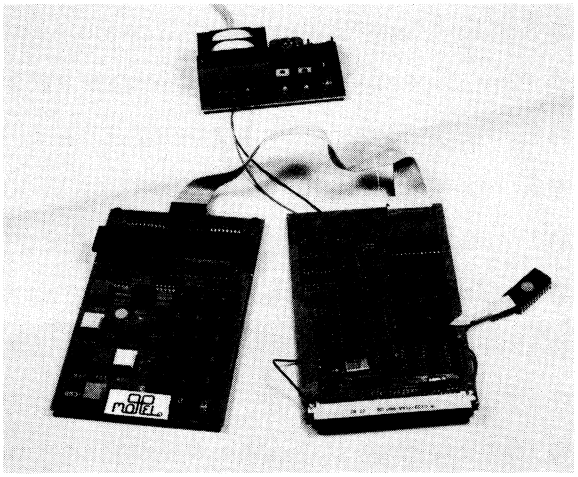
Diese Datenblätter bilden den Grundstein für das MOPPEL-Handbuch, in dem Sie als Blattsammlung alle zu den einzelnen Baugruppen verfügbaren Informationen zusammentragen. Hierzu erscheinen nach und nach Software-Testhilfen und technische Detailbeschreibungen der Zeitabläufe, die für das tiefergehende Verständnis, eigene Erweiterungen oder auch für Servicezwecke unentbehrlich sind. Auf diese Weise entsteht eine umfassende Dokumentation, die Sie mit Nachträgen ständig aktualisieren können, so wie es bei keinem herkömmlichen Handbuch möglich ist und wie es sich aus Platzgründen auch nicht in einer Fachzeitschrift unterbringen läßt. Die über das Maß der technischen Beschreibungen hinausgehenden Beiträge zu diesem Handbuch sind in der Schutzgebühr von DM 20,– **nicht** enthalten; hierüber erhalten Sie gesonderte Informationen.

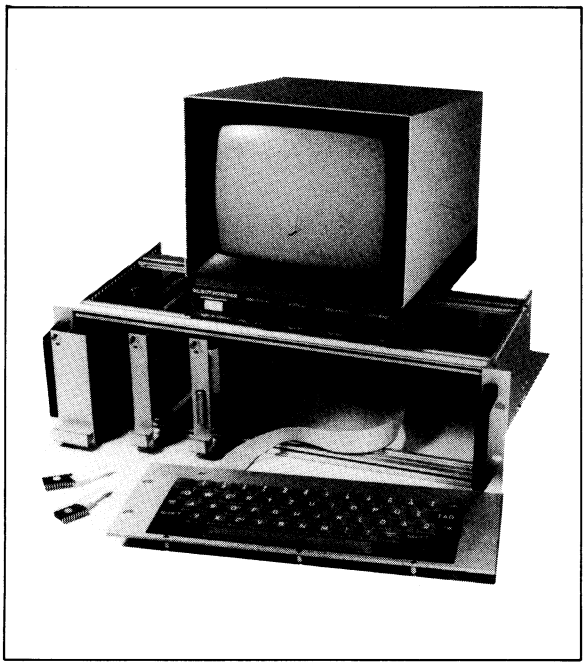
hms **Hodenberger Straße 19c · 2800 Bremen 33 · ☎ (04 21) 25 03 47**
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

MOPPEL-Ausbaumöglichkeiten

Ganz egal, wie Sie den Einstieg in die Mikrocomputer-Technik beginnen wollen und was Ihr Ziel dabei ist, MOPPEL bietet Ihnen immer die geeignete Unterstützung an. Jede Ausbaustufe läßt sich auf die nächsthöhere aufrüsten und mit den anderen Baugruppen nach und nach erweitern. Bei der Zusammenstellung der Bausätze und Fertiggeräte legen wir Wert auf höchsten Qualitätsstandard; so entsprechen die verwendeten Platinen mit Durchkontaktierung, Blei/Zinn-Umschmelzung und Lötstoppmaske industriellen Ansprüchen, und die verwendeten Bauteile sind ausschließlich Industrieprodukte erster Wahl.

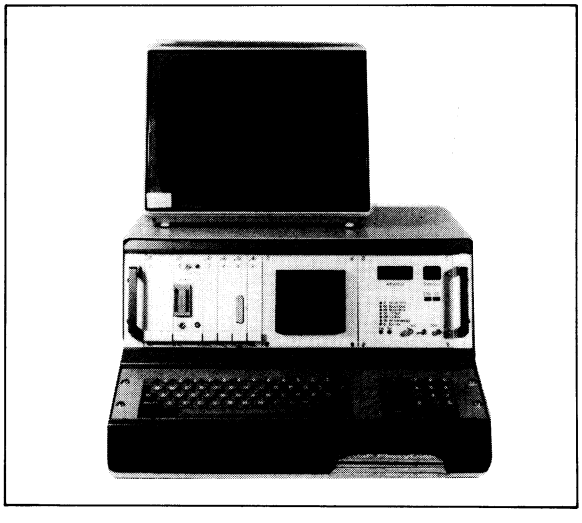
Zum **Einstieg in Maschinensprache** genügen die Zentraleinheit (CPU) mit einem Monitor-EPROM, die HEX-Tastatur mit Anzeige und einem 5-V-Netzteil (z. B. unser Mini-Netzteil). Mit diesem Aufbau können Sie erste Gehversuche unternehmen und die Grundlagen der Programmierung kennenlernen (beachten Sie unser Sonderangebot zum Einsteigen!)





Das **bildschirmorientierte Mikrocomputer-System** besteht aus CPU mit erweitertem Monitor (EPROMs rot plus Video-Monitor gelb), einer ASCII-Tastatur, dem Video-Interface und der Netzteil-Karte. Zweckmäßigerweise bringen Sie die Karten in einem 19-Zoll-Einschubrahmen mit Rückverdrahtung (Busstruktur) unter. Als Sichtgerät können Sie einen herkömmlichen Fernseher oder ein hochauflösendes Datensichtgerät (Monitor) anschließen. Eventuell vorhandene HEX-Tastatur und Mini-Netzteil lassen sich zusammen mit der Funkuhr weiter verwenden.

Mit den übrigen Baugruppen läßt sich MOPPEL zum **professionellen Mikrocomputer-System** ausbauen, das mit Schreibmaschinen-Tastatur, Floppy-Disk-Laufwerk und dem 19-Zoll-Einschubrahmen in einem formschönen und robusten Gehäuse Platz findet.



MOPPEL ist das modulare Prozessor-Programm der ELO

hms **Hodenberger Straße 19c · 2800 Bremen 33 · ☎ (04 21) 25 03 47**
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

Insgesamt sind folgende Baugruppen verfügbar bzw. in Vorbereitung (weitere sind geplant):

- Zentraleinheit (8085A, 2K CMOS-RAM, 10K EPROM, V24-Schnittstelle)
- Netzteil (für Systemversorgung oder das Mini-Sichtgerät)
- Video-Interface (TV- oder Monitor-Anschluß, Grafik-Zeichensatz)
- ASCII-Tastatur (deutsche Normbelegung, drei Ebenen: NOR, SFT, CTL)
- ASCII-Erweiterung (Zehnerblock, Cursor-Steuerung, Funktionstasten)
- 48-K-Speicherkarte (bestückbar mit 16K EPROM und 32K RAM)
- EPROM-Programmierzusatz (Lesen, Vergleichen und Programmieren)
- Echtzeit-Uhr (Uhrzeit, Datum und Wochentag, netzausfallgepuffert)
- Funkuhr (Aufbereitung des Amtlichen Deutschen Zeitzeichens DCF77)
- Thermodrucker (HEX-Listing oder Klartext-Ausdruck, 19-Zoll-Einschub)
- Universal-Interface (parallele Ein/Ausgabe, schnelles serielles Interface)
- Einzelschritt-Modul (Anzeige des gesamten System-Zustandes, Einzelzyklen)
- Mini-Sichtgerät (hochauflösender, grüner Bildschirm, 19-Zoll-Einschub)
- Floppy-Disk (zwei flache 5-Zoll-Laufwerke im Gehäuse einbaubar)

Bis auf wenige Ausnahmen sind die Baugruppen entweder als Bausatz (B vor der Bestell-Nummer) oder fertig bestückte und 100 % geprüfte Einheit (F vor der Bestell-Nummer) erhältlich.

hms **Hodenberger Straße 19c · 2800 Bremen 33 · ☎ (04 21) 25 03 47**
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

Begleitende Literatur zum MOPPEL

Wie bereits angekündigt, erscheint nach und nach ergänzende Literatur zu unserem Mikrocomputer-System MOPPEL, die Sie in einem Sammelordner zu einem umfassenden Handbuch zusammentragen können. Diese Dokumentation entsteht in enger Zusammenarbeit zwischen renommierten Verlagen bzw. Lehrinstituten und uns. Sie können diese Dokumentation auch direkt von uns beziehen. Da wir sie zum Selbstkostenpreis weitergeben, bitten wir Sie, für Bestellungen nur dieses Formular zu benutzen, um uns die Arbeit der Rechnungsschreibung zu ersparen (**Ihren Absender nicht vergessen!**). Wenn Sie gleichzeitig noch MOPPEL-Baugruppen bestellen wollen, legen Sie dieses Blatt bitte Ihrer Bestellung bei.

Anz.	Heft	Einz.Preis	Ges.Preis
	Lehrbriefe zum Selbststudium, die sich ausschließlich mit dem MOPPEL befassen (herausgegeben von der Studiengemeinschaft Darmstadt):		
...	**MIC 1: Hardware und Bedienung des MOPPEL** Blockschaltbilder, Mikrocomputer-Struktur, ausgewählte Befehle	**14,80 DM**	. . . . . **DM**
...	**MIC 2: Programmierung des MOPPEL (1)** Schritte zur Programmentwicklung, Lineare Programme, bedingte Sprünge, Programmschleifen	**14,80 DM**	. . . . . **DM**
...	**MIC 3: Programmierung des MOPPEL (2)** Unterprogrammtechnik, Kellerspeicher, Hard- und Software zu Programmunterbrechungen (Interrupts)	**14,80 DM**	. . . . . **DM**
	ELO-Laborbriefe, die über den Rahmen des MOPPEL hinaus allgemeingültige Bedeutung besitzen (herausgegeben vom FRANZIS-Verlag, München):		
...	**ELB 1: Der 8085-Befehlssatz** Ausführliche Erläuterung des 8085-Befehlssatzes, ergänzt durch je ein Anwendungsbeispiel	**8,00 DM**	. . . . . **DM**
...	**ELB 2: MOPPEL-BASIC** Ausführliche Erläuterung der BASIC-Befehle, ergänzt durch je ein Anwendungsbeispiel	**in Vorbereitung**	
...	**ELB 3: 8085-Ein/Ausgabe-Möglichkeiten** Betrieb der seriellen Schnittstelle; parallele Ein/Ausgaben mit 8255 oder über Ports bzw. Speicheradressen	**in Vorbereitung**	
...	**ELB 4: Mikrocomputer-Lupe** Betrieb des Einzelschritt-Moduls; Prozessor-Timing beim Holen, Decodieren und Ausführen eines Befehls	**in Vorbereitung**	
	Gesamt		. . . . DM
	zzgl. Porto/Verpackungsanteil		3,00 DM
	Summe		**DM**

O Verrechnungsscheck liegt bei
O Zahlung erfolgt auf Ihr Konto 5809 der Bremer Landesbank (290 500 00)
O bitte zusammen mit beiliegender Bestellung berechnen

Absenderangabe nicht vergessen!

Preisliste für MOPPEL-Zubehör (gültig bis 30.9.1983)

Sonderangebot zum Kennenlernen:

MOPPEL-Einsteiger-Bausatz, später voll erweiterbar,
bestehend aus Mikrocomputer-Nullversion mit
Monitor grün, HEX-Tastatur und Anzeige (ohne Netzteil)

Bausatzpreis: DM 299,- (nur als Bausatz lieferbar)

Platinen:	87022	CPU (nur zusammen mit Monitor-EPROM)	DM 68,-
	85031	HEX-Tastatur und Anzeige	DM 68,-
	87030	ASCII-Tastatur	DM 68,-
	87032	ASCII-Tastatur-Erweiterung	DM 48,-
	87010	Mini-Netzteil	DM 15,-
	87012	12-V-Netzteil für Mini-Sichtgerät	DM 39,-
	87015	Netzteil-Karte (5 V/3,5 A; +/-12 V)	DM 39,-
	87040	Speicherkarte (für 48-K-Bestückung)	DM 68,-
	87151	EPROM-Programmierzusatz	DM 68,-
	87120	Thermodrucker (nur mit Zeichensatz-EPROM)	DM 68,-
	87091	Video-Interface (nur mit Zeichengenerator)	DM 68,-
	87071	Echtzeit-Uhr	DM 39,-
	87001	Rückwandverdrahtung (Bus)	DM 48,-
	87171	DCF77-Funkuhr	in Vorbereitung
	87050	Universal-Interface	in Vorbereitung
	87160	Einzelschritt-Modul	in Vorbereitung
	87100	Floppy-Disk-Controller	in Vorbereitung

Aufrüstsatz:	von Mikrocomputer-Null- auf Basisversion	DM 23,-
	von Mikrocomputer-Basis- auf Profiversion	DM 51,-
	von Netzteil-Basis- auf Profiversion	DM 67,-
	von Speicher-Basis- auf Profiversion	DM 132,-
	von Thermodrucker-Basis- auf Profiversion	DM 99,-
	Bauteile für UHF-Modulator (Video-Interface)	DM 16,-
	von Universal-Interface Basis- auf Profiversion	in Vorber.
	von Rückwandverdrahtung-(Bus)Basis- auf Profiversion	DM 68,-

Einzelteile:	RAM 2K x 8	DM 19,-
	Puffer-Akku (für CPU-CMOS-Ram und Echtzeit-Uhr)	DM 13,-
	64polige VG-Stiftleiste (Leiterplatten-Seite)	DM 4,-
	64polige VG-Buchsenleiste (Bus-Seite)	DM 7,-

Software:	Monitor-EPROMs grün, blau, rot und gelb siehe umseitig	
	Zeichensatz und Software für Thermodrucker	DM 44,-
	Video-Zeichengenerator (128 ASCII- u.Sonderzeichen)	DM 54,-
	Video-Zeichengenerator (mit Grafik-Zeichensatz)	DM 68,-
	8-K-BASIC in zwei 2732-EPROMs ($4000-5FFF)	DM 148,-
	Text-Editor im EPROM	in Vorbereitung
	Assembler im EPROM	in Vorbereitung

Lieferbedingungen (Mindest-Waren-Bestellwert: DM 50,-):
lle Preisangaben verstehen sich inklusive Mehrwertsteuer und zuzüglich M 6,90 an Porto- und Versandkostenanteil. Die Lieferung erfolgt per Nachahme, Einsendung eines Verrechnungsschecks oder Vorauskasse (Überweisung uf unser Bankkonto; Ihre Anschrift und die Bestellung nicht vergessen!). irmen werden bei schriftlicher Bestellung auf Rechnung beliefert. Ihre Anschrift haben wir (nur zur eigenen Verwendung) auf Datenträger gespeichert.

---

hms, Hodenberger Straße 19 C, 2800 Bremen 33; (04 21) 25 03 47
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

	Bausatz	DM	Fertiggerät	DM
Zentraleinheit (CPU-Nullversion)	B-022/1-0:	138,-	./.	
(CPU-Basisversion)	B-022/2-1:	154,-	F-022/2-1:	198,-
(CPU-Profiversion)	B-022/3-5:	199,-	F-022/3-5:	248,-
(zusätzlich Monitor-EPROM grün, blau oder rot erforderlich; Betrieb mit ASCII-Tastatur und Video-Interface nur mit Monitor rot plus gelb)				
HEX-Tastatur und -Anzeige	B-031/1-1:	149,-	F-031/1-1:	186,-
ASCII-Tastatur	B-030/1-1:	189,-	F-030/1-1:	238,-
ASCII-Tastatur-Erweiterung	B-032/1-5:	83,-	F-032/1-5:	121,-
Mini-Netzteil	B-010/2-1:	45,-	F-010/2-1:	59,-
Netzteil-Karte mit Trafo (Basis)	B-015/3-1:	112,-	(F-015/3-1:	141,-)
(Profi)	B-015/4-5:	173,-	(F-015/4-5:	219,-)
12-V-Netzteil f. Mini-Sichtgerät	B-012/2-5:	147,-	F-012/2-5:	178,-
Speicherkarte (Basis; 4 K RAM)	B-040/1-1:	109,-	F-040/1-1:	138,-
(max.48 K) (Profi; 16 K RAM)	B-040/2-5:	238,-	F-040/2-5:	284,-
EPROM-Programmierzusatz	B-151/2-5:	218,-	F-151/2-5:	269,-
Thermodrucker (Basisversion)	B-120/2-1:	299,-	F-120/2-1:	348,-
(Profiversion)	B-120/3-5:	388,-	F-120/3-5:	446,-
Video-Interface mit BAS-Ausgang	B-091/2-5:	220,-	F-091/2-5:	259,-
m.UHF-Modulator	B-091/3-1:	234,-	F-091/3-1:	279,-
Echtzeit-Uhr	B-071/1-1:	89,-	F-071/1-1:	122,-
DCF77-Empfangsmodul	DCF	49,-	./.	
DCF77-Funkuhr	in Vorbereitung		in Vorbereitung	
Universal-Interface (Basis)	in Vorbereitung		in Vorbereitung	
(Profi)	in Vorbereitung		in Vorbereitung	
Hardware-Einzelschritt-Modul	in Vorbereitung		in Vorbereitung	
Rückwandverdrahtung (Bus, Basis)	B-001/1-1:	58,-	F-001/1-1:	86,-
(Bus, Profi)	B-001/2-5:	129,-	F-001/2-5:	182,-
Floppy-Disk-Controller	in Vorbereitung		in Vorbereitung	
Mini-Floppy-Laufwerk (5 Zoll, flache Bauform)			in Vorbereitung	
Mini-Sichtgerät (grün) als 19-Zoll-Einschub (12 V)			Z-091/1-7:	489,-
19-Zoll-Einschubrahmen	B-100/1-1:	129,-	./.	
Stahlblechgehäuse	in Vorbereitung		./.	
Monitor-EPROM grün (Nullversion)			S-022/0-0:	32,-
blau (Basisversion)			S-022/0-1:	44,-
rot (Profiversion)			S-022/0-5:	59,-
gelb (Video-Monitor; erweitert M.rot)			S-022/1-7:	68,-

Lieferbedingungen und Zubehör siehe Rückseite; die in Vorbereitung befindlichen Baugruppen sind gleichzeitig mit der Publikation in der ELO lieferbar.

hms, Hodenberger Str.19 C, 2800 Bremen 33; (04 21) 25 03 47
Konto 5809 bei der Bremer Landesbank (BLZ 290 500 00)

Einleitung

1. Der Mi SAB 8085

Diese Technische Mitteilung beschreibt die Eigenschaften des Mikroprozessors SAB 8085 und der kombinierten Speicher- und Eingabe-/Ausgabe-Bausteine SAB 8755/8355 und SAB 8155/8156.

Dabei werden die Verbesserungen und die zusätzlichen Funktionen des SAB 8085 gegenüber dem Mikroprozessor SAB 8080 herausgestellt. Das Interrupt-System wird eingehend erläutert und durch ein Logikmodell veranschaulicht. Zu der seriellen Ein-/Ausgabemöglichkeit des SAB 8085 wird ein grundsätzliches Beispiel angeführt.

Anschließend folgt die Darstellung eines leistungsfähigen 3-Chip-Mikrocomputer-Minimalsystem sowie dessen Erweiterung durch zwei Standardbausteine.

Ergänzend wird allgemein der Anschluß von Standard-Speicher- und Peripherie-Bausteinen an den Prozessor SAB 8085 für den Aufbau komplexer Mikrocomputer-Systeme beschrieben.

Der Prozessor SAB 8085 beinhaltet alle Leistungsmerkmale des SAB 8080 sowie zusätzliche Funktionen und einige Verbesserungen.

Die Verbesserungen sind:

- einfache +5 V Spannungsversorgung
- höhere Verarbeitungsgeschwindigkeit (ca. 50 %)
- Taktgenerator und Systemsteuerung mit auf dem Chip integriert
- höherer „high" Ausgangssignalstrom ($I_{OH} = 400\ \mu A$) und ein Eingangsspannungspegel von $V_{IL} = 0{,}8$ V und $V_{IH} = 2{,}0$ V (beim SAB 8080 $I_{OH} = 150\ \mu A$ und $V_{IH} = 3{,}3$ V)

Zusätzliche Funktionen sind:

- 4 Interrupt-Eingänge, bei denen der SAB-8085-Prozessor intern eine feste Adresse erzeugt
- eine serielle Ein-/Ausgabe-Schnittstelle.

Infolge der zusätzlichen Funktionen werden gegenüber dem SAB 8080 mehr Baustein-Anschlüsse benötigt. Um dennoch mit einem 40-Pin-Gehäuse auskommen zu können, werden der untere Teil des Adreßbusses und der Datenbus im Zeitmultiplexverfahren betrieben. Dadurch werden 7 Anschlüsse eingespart (der achte wird für das Adreß-Latch-Signal gebraucht), die nun für andere Zwecke benutzt werden.

Zum Bedienen der seriellen Schnittstelle und der RST-Interrupts sind 2 zusätzliche Befehle, RIM (Read Interrupt Mask) und SIM (Set Interrupt Mask), vorhanden.

Alle anderen Befehle sind unverändert vom SAB-8080-Befehlssatz übernommen worden. Lediglich der zeitliche Ablauf mancher Befehle hat sich gegenüber dem SAB-8080-Prozessor geändert.

Beim SAB 8080 besteht ein Maschinenzyklus innerhalb eines Befehlsablaufs aus drei bis fünf aktiven Operationsschritten (States – siehe auch [2] S. 42–47), während er beim SAB 8085 aus drei bis sechs States besteht (s. Bild 1a). Der erste Maschinenzyklus bei jedem Befehlsablauf ist ein „Opcode Fetch"-Zyklus*. Beim SAB 8080 dauert dieser 4 bzw. 5 States, während beim SAB 8085 der Fetch-Zyklus, bedingt durch den Multiplex-Betrieb auf dem Bus, 4 bzw. 6 States dauert. Alle weiteren Maschinenzyklen beim SAB 8085 dauern 3 States.

Manche Befehle erfordern deshalb beim SAB 8085 einen State mehr. Durch die kürzere Dauer eines States ist der Befehlsablauf trotzdem wesentlich schneller als bei SAB 8080.

Ebenso hat sich der Ablauf der bedingten JUMP-, CALL- und RETURN-Befehle geändert. Bei nicht erfüllter Bedingung wird die Befehlsbearbeitung nach dem zweiten Maschinenzyklus (holen des niederwertigen Adreßbytes der Sprungadresse) abgebrochen.

* „Opcode Fetch"-Zyklus = Befehls-Hol-Zyklus

Charakteristische Daten des SAB 8085

- Software kompatibel mit SAB 8080 A
- Taktgenerator auf dem Chip 6 MHz Taktfrequenz (Quarz od. RC-Glied)
- Erweiterte Interrupteigenschaften
- 3 Chip Minimal System möglich (mit SAB 8156 und SAB 8355/8755)
- Multiplexbetrieb auf Datenbus (Adressen/Daten)
- Einfache Schnittstelle zu Standard-speichern (über SAB 8212)
- 40 Pin, +5 V, N-MOS Depletion Load
- einfache serielle E/A möglich

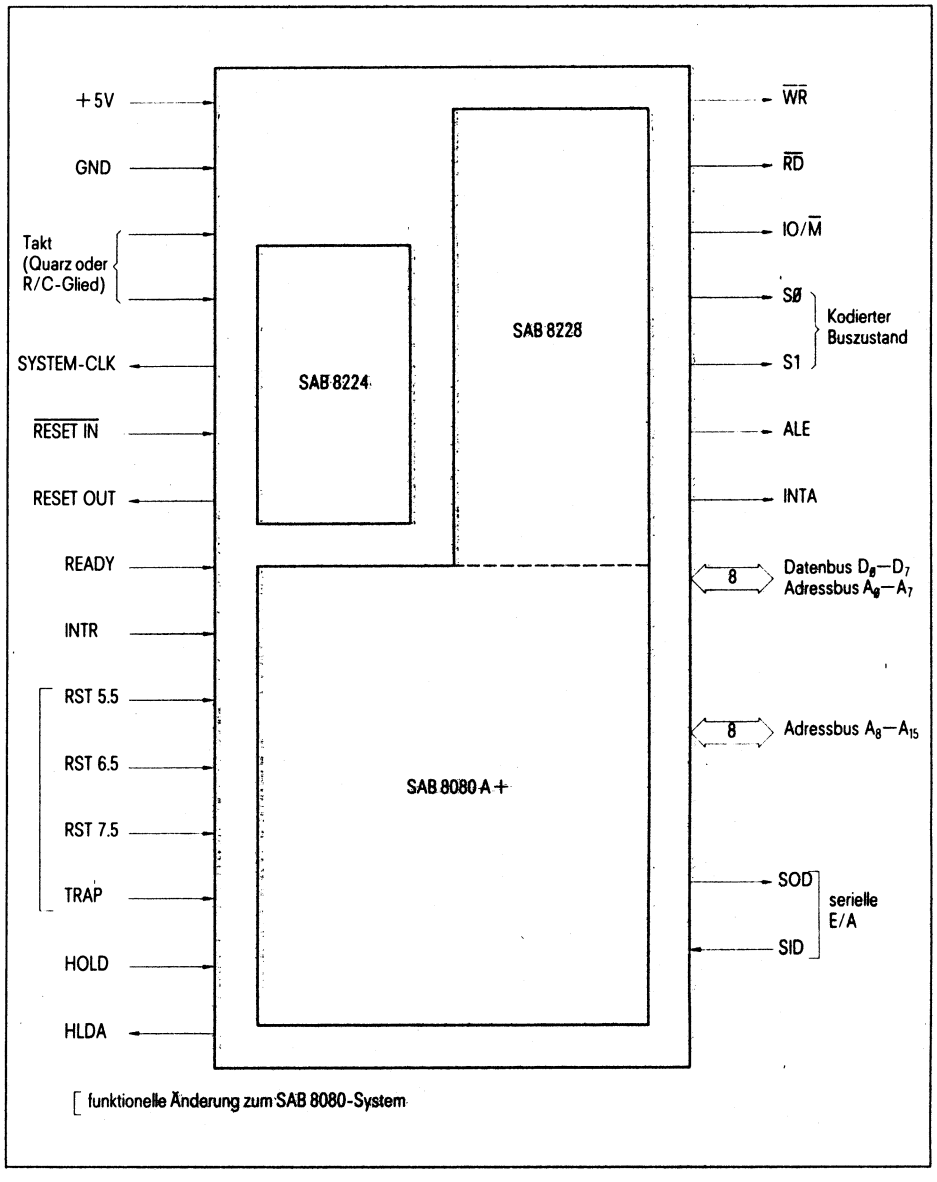


Bild 1 Pin-Anschlußbelegung (funktionell)

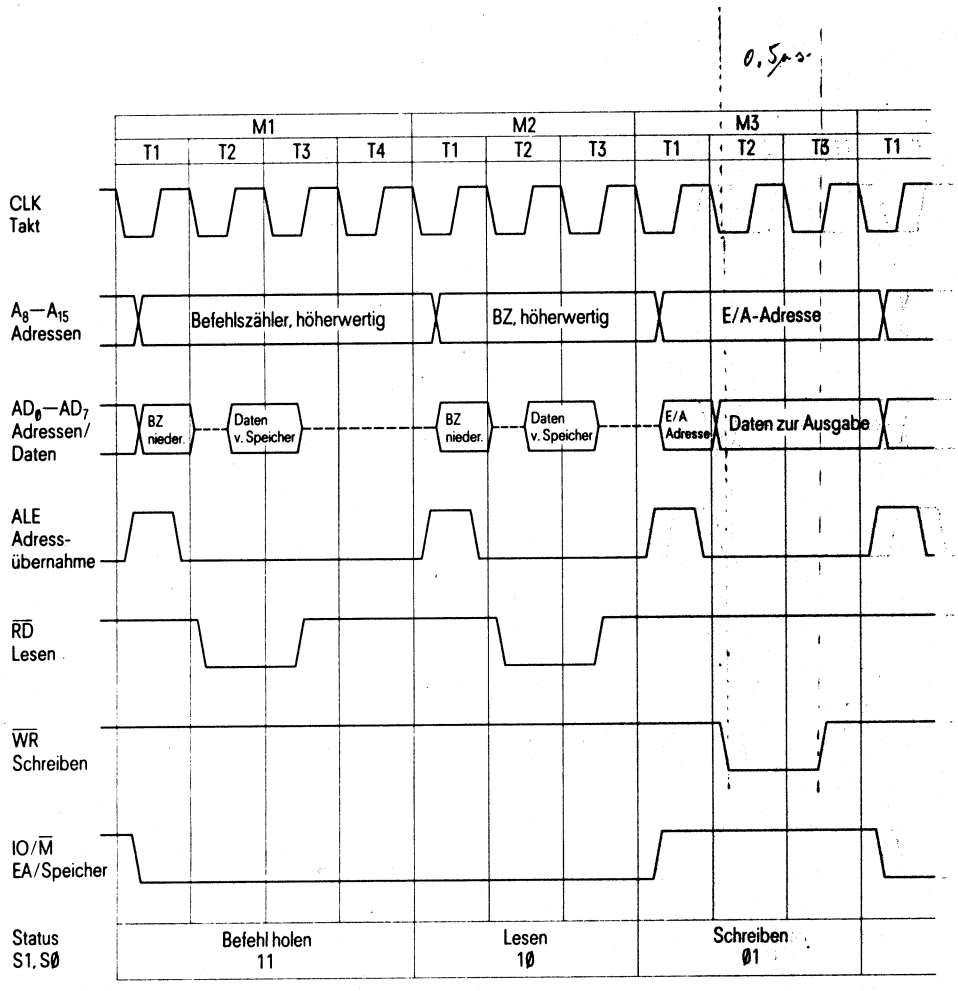


Bild 1a Befehlsablauf (schematisch)
Beispiel: OUT adr.

1.1 Das SAB 8085 Interrupt-System (Bild 2)

Um die Interrupt-Struktur des SAB 8085 durchschaubarer zu machen, wurde ein Funktionsmodell (Bild 2a) aufgestellt. Dieses Modell stimmt jedoch mit der physikalischen Realisierung des Interrupt-Systems auf dem CPU-Chip nur hinsichtlich der Funktion überein.

1.1.1 Maskierbare Interrupts

Der SAB-8085-Prozessor hat außer dem vom SAB 8080 bekannten Interrupt-Request-Eingang (INTR) vier Interrupt-Eingänge, bei deren Aktivierung der Prozessor intern eine RESTART-Adresse erzeugt (s. Bild 2).

Drei dieser Interrupt-Anforderungen (RST 7.5, RST 6.5 und RST 5.5) sind mit dem Befehl SIM (s. Bild 2b) maskierbar. D. h., die Annahme einer Interrupt-Anforderung kann durch Setzen einer 1 in der Interrupt-Maske verhindert werden.

Der TRAP-Interrupt, der ebenfalls eine Restart-Adresse generiert, ist weder durch den SIM-Befehl noch durch einen DI-Befehl (Disable Interrupt) maskierbar.

Mit einem DI-Befehl wird generell die Annahme aller anderen Interrupt-Anforderungen verhindert.

Der Trap-Interrupt wird aktiviert mit einer ansteigenden Flanke **und** mit einem „hig"-Pegel am Eingang. Mit der ansteigenden Flanke des Anforderungsimpulses wird ein Bereit-FF gesetzt. Der Anforderungsimpuls muß anschließend solange anliegen, bis der Interrupt angenommen worden ist. Im ungünstigsten Fall kann das 6 µs dauern (längste Befehlsdauer des SAB 8085). Nach der Annahme des Interrupts durch die CPU wird das Bereit-FF rückgesetzt und verhindert damit ein erneutes Annehmen des TRAP-Interrupts, falls das Anforderungssignal weiterhin anliegt.

Ein erneuter TRAP-Interrupt kann nur durch eine neue ansteigende Flanke mit anschließendem „high"-Pegel am TRAP-Eingang ausgelöst werden. Beim Trap-Interrupt ist es wichtig zu beachten, daß das INTE-FF des Prozessors nach Annahme einer Trap-Interrupt-Anforderung zurückgesetzt wird. Damit wird der Zustand des INTE-FF's gegebenenfalls geändert und bei Rückkehr aus dem Trap-Unterprogramm der vorherige Zustand nicht wieder hergestellt. Eine softwaremäßige Lösung dieses Problems ist möglich, indem vor jeder Änderung des INTE-FF's dieses per Programm im RAM-Speicher festgehalten wird. Die Änderung des INTE-FF's durch Annahme eines anderen Interrupts muß ebenfalls in der RAM-Speicherstelle abgespeichert werden.

Beim Nachfolgetyp SAB 8085 A kann der vorherige Zustand des INTE-FF's einfacher wieder hergestellt werden [4].

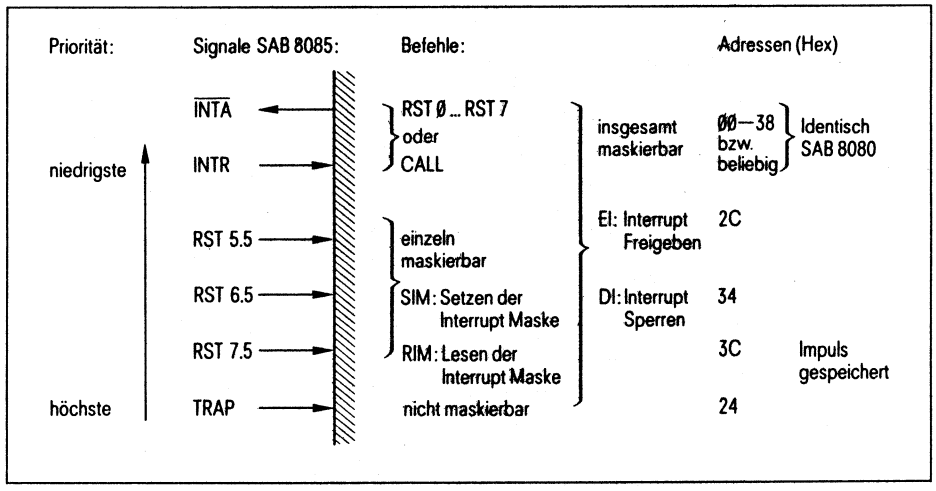


Bild 2 SAB 8085 – Unterbrechungssystem

1.1.4 RST 7.5

Der RST 7.5 Interrupt nimmt unter den 5 Interrupt-Eingängen eine Sonderstellung ein. An diesem Eingang muß eine Interrupt-Anforderung nicht eine bestimmte Zeit anstehen, sondern hier genügt allein eine ansteigende Flanke als Anforderung. Durch diese Flanke wird ein Eingangs-Flip-Flop gesetzt, das die Interrupt-Anforderung bis zur Annahme der Anforderung oder bis zum Rücksetzen des Flip-Flops speichert. Dieses Eingangs-Flip-Flop wird durch den Befehl SIM (Bit 4 des Akkus =1), durch ein System Reset oder durch die Annahme der RST 7.5 Interrupt-Anforderung rückgesetzt.
Die Annahme der RST 7.5 Interrupt-Anforderung kann ebenfalls wie beim RST 5.5 und 6.5 durch Setzen des entsprechenden Bits in der Interrupt-Maske und durch das Rücksetzen des Interrupt-Enable-Flip-Flops (INTE) verhindert werden.

Das RST 7.5 Eingangs-Flip-Flop wird durch die Interrupt-Maske und durch INTE nicht beeinflußt, so daß eine RST 7.5 Interrupt-Anforderung auch zu einem späteren Zeitpunkt bei Freigabe der Interrupt-Anforderung angenommen werden kann. Der Zustand dieses Eingangs-Flip-Flops kann trotzdem mit Hilfe des Befehls RIM (s. Bild 2 und 3) abgefragt werden, so daß auch bei gesperrtem Interrupt-System nach entsprechender Abfrage zu einer Interrupt-Service-Routine verzweigt werden kann. Die Interrupt-Anforderung kann auch durch Rücksetzen des Eingangs-Flip-Flops gelöscht werden.

1.1.5 RST 6.5 und 5.5

Diese Interrupt-Eingänge reagieren allein auf einen positiven Spannungspegel. Der „high"-Pegel muß bis zur Annahme der Interrupt-Anforderung anliegen (mindestens 1 Befehlslänge). Der aktuelle logische Zustand der Eingänge kann mit Hilfe des RIM-Befehls (s. Bild 2a und 2b abgefragt werden.

Die Annahme der RST 6.5 und 5.5 Interrupt-Anforderung kann durch Setzen der Interrupt-Maske und durch Rücksetzen des INTE-Flip-Flops verhindert werden. RST 6.5 und 5.5 Interrupt-Anforderungen werden nicht gespeichert.

1.1.6 Pending Interrupts*

Mit dem Befehl RIM (s. Bild 2a und 2b) läßt sich der Zustand der 3 Interrupt-Anforderungsleistungen RST 5.5, 6.5 und 7.5 abfragen. Beim RST 7.5 handelt es sich dabei um den Ausgang des Eingangs-Flip-Flops, während beim RST 6.5 und 5.5 unmittelbar der logische Zustand der Eingangsleitung, der gerade beim Ausführen des RIM-Befehls vorhanden ist, angezeigt wird.

1.1.7 INTR (Interrupt Request)

Diese Interrupt-Möglichkeit hat dieselben Merkmale wie das SAB 8080 Interrupt-System.

Mit INTR kann die Anzahl der zusätzlich möglichen Interrupt-Anforderungen z. B. mit Hilfe des Interrupt-Bausteins SAB 8259 bis auf 64 erhöht werden, wobei in diesem Fall CALL-Befehle erzeugt werden [5].

1.2 Serielle Daten-Schnittstelle (SID/SOD) (Bild 3)

Die beiden Anschlüsse SID (Serial Input Data) und SOD (Serial Output Data) ermöglichen eine einfache serielle Datenübertragung mit dem SAB 8085.

Dabei wird mit dem Befehl SIM das Bit 7 des Akkumulators in das Ausgangs-FF übertragen und steht dann am SOD-Ausgang des SAB 8085 zur Verfügung. Mit dem Befehl RIM wird der Zustand des SID-Eingangs in das Akkumulator-Bit 7 gelesen.

Ein Beispiel für eine serielle Datenübertragung mit dem SAB 8085 zeigt Bild 3a.

Die hierbei erzielte Übertragungsdauer für 1 Byte beträgt mindestens 94 µs bei einer Taktperiodendauer des SAB 8085 von 330 ns.

1.1.2 Priorität

Die fünf Interrupt-Request-Eingänge der CPU haben untereinander eine fest vorgegebene Priorität, die sich nicht verändern läßt. Beginnend mit dem Trap, der die höchste Priorität hat, folgen RST 7.5, RST 6.5, RST 5.5 und mit der niedrigsten Priorität INTR.

Liegen mehrere Interrupt-Anforderungen gleichzeitig an, wird die Interrupt-Anforderung mit der höchsten Priorität angenommen.

Läuft gerade eine Interrupt-Service-Routine, ausgelöst z. B. durch einen RST 7.5, so kann diese ISR durch eine Interrupt-Anforderung niedrigerer Priorität jederzeit unterbrochen werden. Voraussetzung dazu ist jedoch, daß das Interrupt-System vorher durch den Befehl EI freigegeben und der Interrupt-Eingang nicht durch Setzen der Interrupt-Maske gesperrt wurde.

Die Priorität bezieht sich also nur auf die Annahme von Interrupt-Anforderungen bei gleichzeitigem Anstehen.

1.1.3 TRAP

Der Trap-Interrupt hat vor allen anderen Interrupts Vorrang, da er die höchste Priorität besitzt und weder maskierbar, noch durch den DI-Befehl zu unterbinden ist. Er wird in jeder Phase eines Programms – am Ende des gerade bearbeitenden Befehls – angenommen. Der Trap eignet sich deshalb besonders zur Aktivierung nicht aufschiebbarer Interrupt-Service-Routinen, z. B. zum Retten von aktuellen Daten in einen batteriegepufferten RAM-Speicher bei einem Netzausfall oder in einer Prozeßsteuerung zum Reagieren auf schwerwiegende Fehler im Prozeßablauf.

* pending = schwebend, anhängig

Bild 2a Interrupt-Logik-Modell SAB 8085

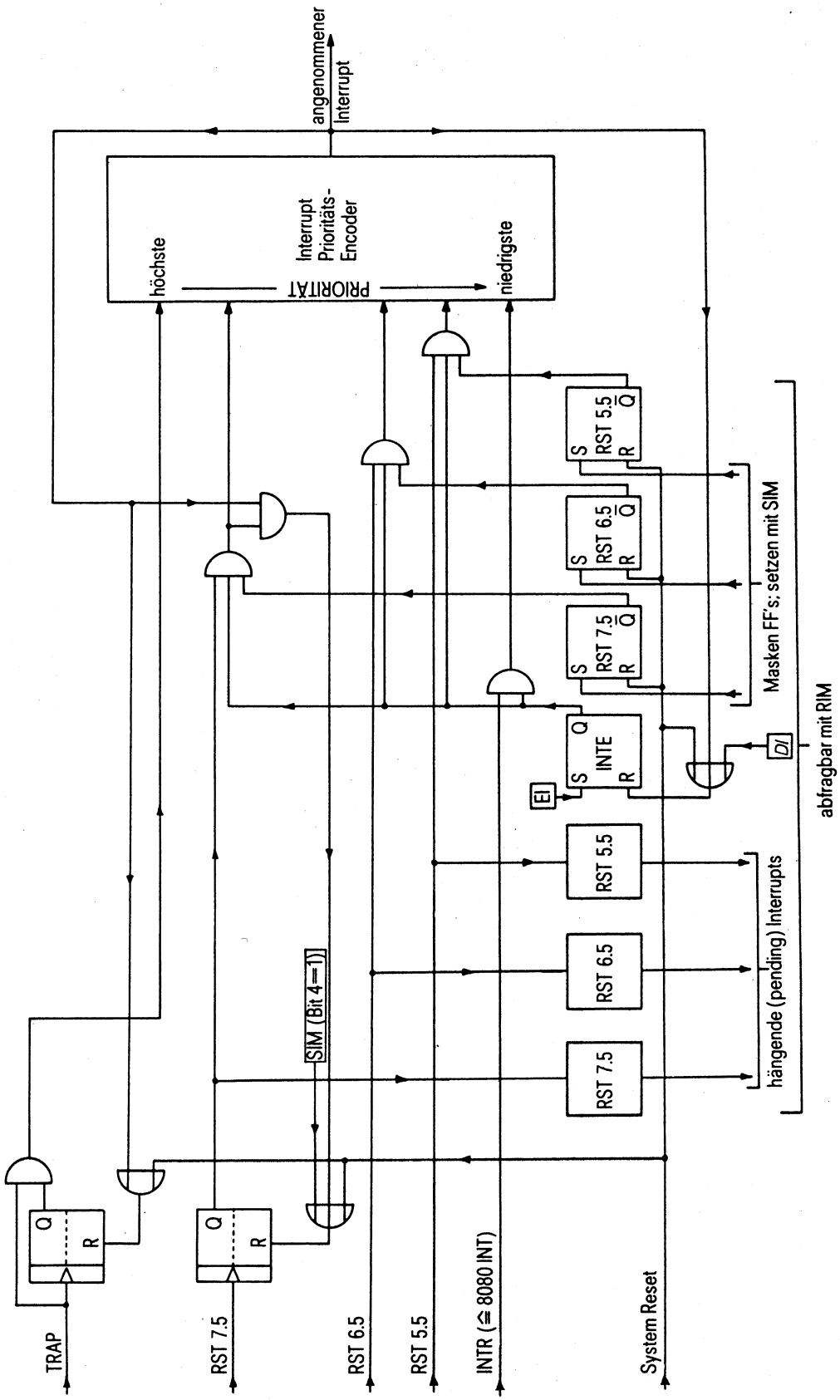
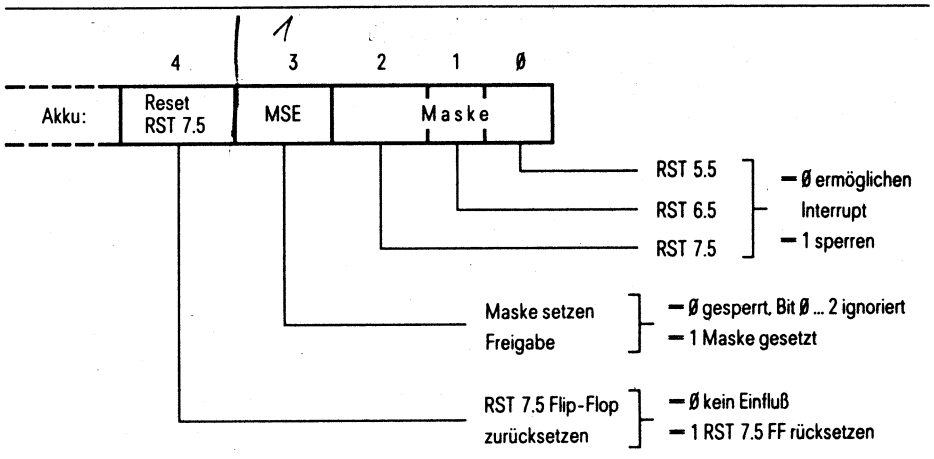


Bild 2b Unterbrechungssystem (SIM/RIM)

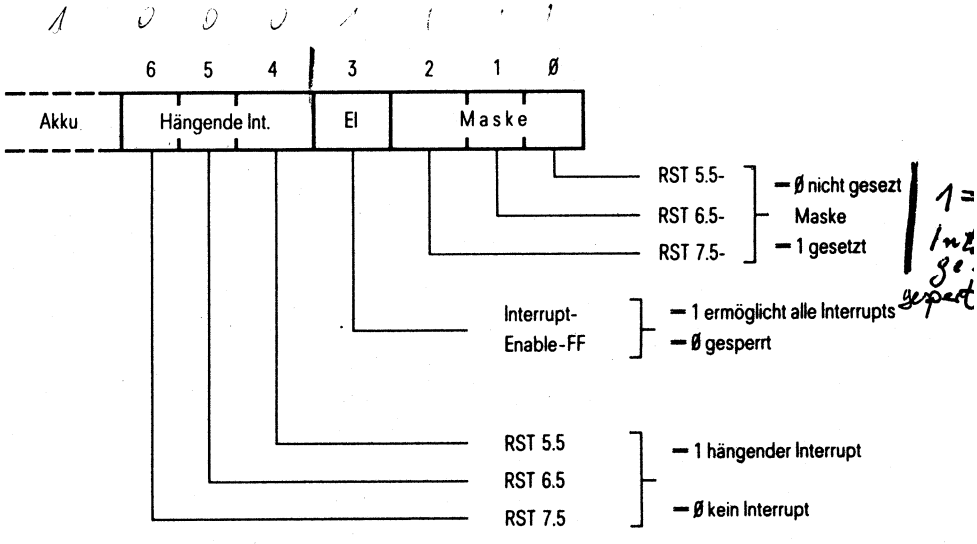
Befehl SIM: Setzen Interrupt-Maskenregister

Der Inhalt des Akkumulators wird wie folgt interpretiert:



Befehl RIM: Lesen Interrupt-Maskenregister

Der Akku wird mit folgender Information geladen:



Mit RIM-Information über:

- Zustand Maskenregister
- Zustand Interrupt-Enable-FF
- Zustand Interrupt-Leitungen RST 5.5 ... RST 7.5

Bild 3 Serielle Daten-Ein-/Ausgabe-Schnittstelle

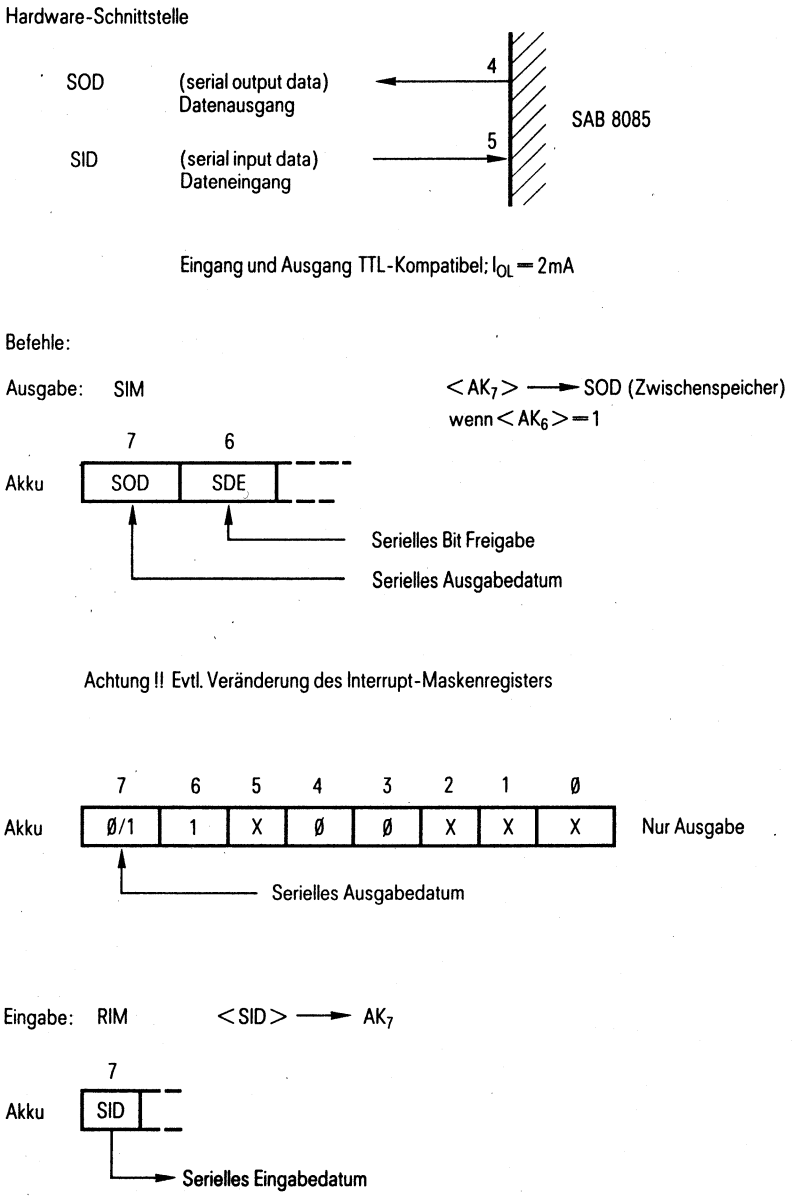
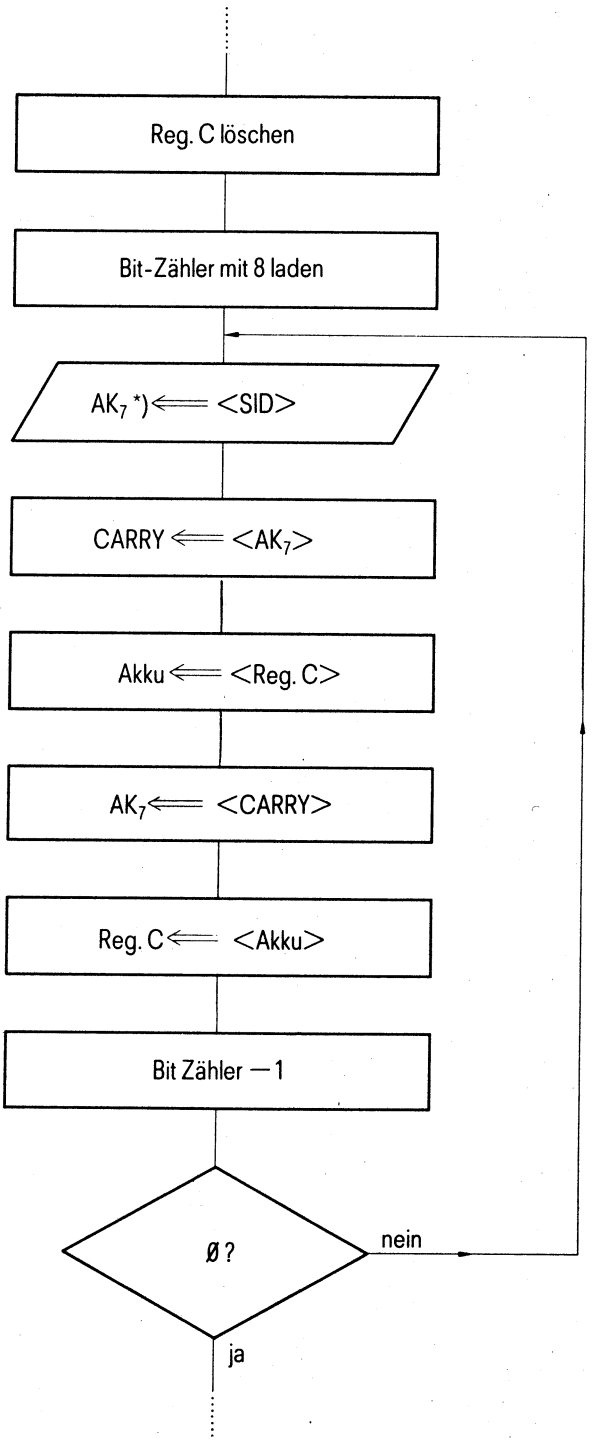


Bild 3a Serielle Datenübertragung
mit dem SAB 8085

Übertragung von 8 Bits



7. Bit	6.	5.	4.	3.	2.	1.	Ø

das Ø. Bit wird als erstes eingelesen

```
ADR. Befehle     Kommentar                    States
                 .
                 .
                 .
                 .
                 .
     MVI C, Ø    ; Reg. C löschen             7
     MVI B, Ø8H  ; Bit Zähler laden           4
A1:  RIM         ; 1 Bit in AK7               4
     RAL         ; Bit ins Carry schieben     4
     MOV A, C    ; Inhalt von Reg. C in Akku  4
     RAR         ; Carry in den Akku          4
     MOV C, A    ; Akku ins Reg. C            4
     DCR B       ; Bit Zähler-1               4
     JNZ A1      ; wenn Zähler ≠ 0, springe   7/10
                 .
                 .
                 .
                 .
```

* AK_7 ... Bit Nr. 7 des Akkus
<.....> ... Inhalt eines Registers