CPU 8088

die PC- Baugruppe für den NDR- Computer

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und durfen nicht gewerblich genutzt werden

Alle Schaltungen und technischen Angaben in diesem Handbuch wurden von dem Autor mit großter Sorgfalt erarbeitet bzw zusammengestellt und unter Einschaltung von wirksamen Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen Der Lizenzinhaber und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, daß sie weder Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zuruckzuführen sind, ubernehmen.

1.Ausgabe
Text: Klaus Bischof, Kempten GES
Gestaltung und Layout: Wolfgang Fink, Kempten GES
Druck und Bindung. Druckerei Rieder, Kempten
Bestellnr.: 11267

Erstellt mit "MicrosoftWord" und "PageMaker" auf einem mc-modular-AT/386

Inhaltsverzeichnis

1. Einführung

1.1 Zum NDR-Computer

Der NDR-Computer wird in der Fernsehserie "Rechner modular" aufgebaut, erklärt und in Betrieb genommen. Diese Serie wird vom Norddeutschen Rundfunk, vom Sender Freies Berlin, vom Bayerischen Fernsehen und von Radio Bremen ausgestrahlt. Der NDR-Computer wurde für diese Sendereihe entwickelt. Von Anfang an wurde bei diesem Computer das modulare Konzept in den Vordergrund gestellt. Mittlerweile laufen auf dem NDR-Computer nicht weniger als sechs CPUs und fünf Betriebssysteme. Kein anderes Computersystem kann eine solche Universalität aufweisen. Nähere Angaben zum modularen Konzept und zu den verfügbaren Baugruppen zum NDR-Computer finden Sie in unserem Farbkatalog mit über 200 Seiten.

Zur Serie gibt es einige Begleitmaterialien, daher ist es nicht unbedingt notwendig, die Fernsehserie gesehen zu haben, um den NDR-Computer zu bauen und zu begreifen:

- **Buch:** Rolf-Dieter Klein,
  "Rechner modular"
  ISBN 3-7723-8721-7, DM 68,-
  erschienen im Franzis-Verlag, München
  Bestellnummer: 10 991

- **Sonderhefte der Zeitschrift "mc":**

  "Mikrocomputer Schritt für Schritt"
  Bestellnummer: 10 399

  "Mikrocomputer Schritt für Schritt Teil 2"
  Bestellnummer: 10 398

- **Zeitschriften:** "mc" und "ELO" des Franzis-Verlags

- **Zeitschrift:** "LOOP" der Firma Graf Elektronik Systeme GmbH
  Die Zeitschrift LOOP ist eine Kundenzeitschrift und enthält Neuerungen, Änderungen, Tips und Tricks, Software usw. zum NDR-Computer und auch speziell zu dieser Baugruppe.

- **Viedeokassetten:** lizensierte Original-Kassetten für den privaten Gebrauch. Auf diesen Kassetten sind die 26 Folgen der Fernsehserie enthalten.

  System:
  VHS zwei Kassetten Bestell Nr 10 439
  VIDEO 2000 Bestell Nr.:10 438

1.2 Entwicklung der CPU8088

Bei der Entwicklung der CPU8088 war uns von Anfang an klar, daß diese nur dann sinnvoll ist, wenn der NDR-Computer damit kompatibel zum IBM-PC wird. Doch dies schien noch vor einigen Jahren ein Ding der Unmöglichkeit. Viele Hersteller bemühten sich damals den IBM-PC möglichst mit identischer Hardware nachzubauen. Dies war natürlich für den NDR-Computer nicht möglich.
Hier wäre das modulare Konzept des NDR-Computers in einer Sackgasse gelandet.

Mit diesem Zustand waren wir natürlich nicht zufrieden und stellten Überlegungen an, wie die Baugruppen des NDR-Computers weiter verwendet werden könnten und der NDR-Computer dennoch IBM-kompatibel gebaut werden konnte.

Wir haben es geschafft!

Wir haben uns überlegt, was ein PC unbedingt benötigt, und was durch das BIOS (das Betriebsprogramm eines IBM-Rechners) anpaßbar wäre. Bei den Schnittstellen für Graphik, Festplatte, Laufwerke und Speichern konnten wir durch "jonglieren" auf die hardwarekompatiblen Bausteine verzichten. Nicht so bei den Bausteinen für Interrupt, Timer, Parallelbaustein und DMA. Diese Bausteine mußten möglichst auch beim NDR-Computer verwendet werden, d.h. die entsprechenden Bausteine müssen direkt auf die CPU-Karte oder auf einer neuen Karte untergebracht werden. Diese komplette Schaltung mit allen Bausteinen war nicht unterzubringen. Eine Lösung mit zwei Karten erschien indiskutabel und ist viel zu teuer. Also begannen wir die Karte zu optimieren. Dabei legten wir zuerst den DMA auf Eis. Dieser wird nämlich nur beim Festplattencontroller benutzt, wird aber nicht unbedingt benötigt. Selbst mit dieser Lösung war die Schaltung noch nicht auf einer Europakarte unterzubringen. Eine weitere Möglichkeit in einer solchen Situation ist eine höhere Integrationsdichte. Es gibt natürlich Gate-Array's, in denen die Bausteine, die für einen PC benötigt werden, alle integriert sind. Diese Gate-Array's haben allerdings bis zu 100 Pins und mehr und sind daher nicht zum Selbstbau geeignet. Außerdem sind diese Mammutchips nur

schwer zu durchschauen und widersprechen eigentlich dem einfachen modularen Konzept des NDR-Computers.

Wir haben uns deshalb darauf beschränkt, die meisten TTL-Bausteine in 5 PALs zusammen zu fassen und haben damit wieder einige Bausteine gespart. Doch damit noch nicht genug. Die Europakarte war immer noch so voll, daß wir einen weiteren Kompromiß eingehen mußten. Dieser Kompromiß fiel uns besonders schwer, da wir immer darauf geachtet haben, daß sämtliche unserer Baugruppen in der GES Norm gefertigt werden. Bei dieser Baugruppe mußten wir darauf verzichten, d.h. die Baugruppe wird nur für den NDR-Bus, nicht für den ECB-Bus geliefert werden können. Aber damit war es möglich, auf einer Karte alle nötigen Teile unterzubringen.

Die Anpassung steckt im BIOS

Die Anpassung der gängigen NDR Standardbaugruppen, wie GDP64, FLO3 oder KEY steckt im BIOS der CPU8088. Nun kurz noch einige Anmerkungen zum BIOS allgemein. BIOS steht für "Basic Input Output System" und stellt die Schnittstelle zwischen dem Betriebssystem (hier MS-DOS) und der Hardware dar. Um das etwas näher zu erläutern, sei hier ein kleines Beispiel angeführt: Ein Anwenderprogramm verlangt, daß ein Punkt auf den Bildschirm geschrieben wird. Dabei könnte jetzt an dieser Stelle des Anwenderprogrammes direkt die Hardware angesprochen werden. Dies hätte allerdings zwei gewaltige Nachteile: Zum Ersten müssen Sie immer dieselbe Hardware verwenden, und zum Zweiten müßte der Programmierer genaue Hardwarekenntnisse haben, ganz abgesehen vom Programmieraufwand. Aus diesem Grund gibt es die BIOS (Programm-)Schnittstelle für den Befehl "Punkt auf der Graphikkarte setzten". Im BIOS steht dann an dieser Stelle die Routine, die die Graphikkarte direkt anspricht.

Für die GDP heißt dies nun, daß in unserem BIOS für die entsprechenden Befehle die Hardwaretreiber für die GDP64 stehen. Sind nun die Anwenderprogramme "sauber" geschrieben (über die BIOS Schnittstelle), so laufen die Programme, die auf einem kompatiblen PC laufen, auch auf dem NDR-Rechner mit der CPU8088 und der GDP64. Dasselbe gilt nun für die Baugruppen FLO3 und KEY, bzw. FLO2 und KEY2. Dies ist bei den meisten Programmen, die wir bisher getestet haben, auch gewährleistet. Problematisch sind allerdings die parallele und serielle Schnittstelle. Bei Ein- oder Ausgaben auf diese beiden Schnittstellen greifen doch mehrere Programme direkt auf die Hardware zu. Dies ist sicherlich noch eine Unsitte aus der Zeit des "Protectionismus", womit versucht wurde, Software nur auf original IBM-PCs zum Laufen zu bringen. Mittlerweile hat sich ja gezeigt, daß diese Rechnung nicht aufging, aber diese "Unsitte" ist nicht mehr auszumerzen. Aus diesem Grund haben wir diese Schnittstellen mithilfe einer Buskopplung (NDR-PC-Bus) unterstützt (billige serielle und parallele Schnittstellen für den PC Bus). Andererseits ist unsere CENT2 und unsere SER im BIOS angepaßt.

Außerdem werden wir im BIOS noch folgende Zusatzhardware unterstützen:

- SMART Watch (nach dem Booten wird die aktuelle Zeit im DOS eingetragen).
- Festplatte: Hier unterstützen wir eine Standardfestplatte (Seagate 225) mit dem OMTI-Controller 5520B

Ein Nachteil der GDP beim Betrieb mit der CPU8088 sei nicht verschwiegen: Die GDP unterstützt nicht den vollen IBM-Zeichensatz. Da wir aber kompatibel bleiben wollten, werden nun die IBM-Zeichen einzeln durch das BIOS "gezeichnet". Nur kostet das Zeit.

Fur diejenigen, die sich mit der Geschwindigkeit der GDP nicht anfreunden konnen, werden wir mit der Buskopplung NDR PC-Bus auch eine Hercules, EGA, CGA oder VGA Karte unterstützen. Wer mit der KEY und der ASCII-Tastatur nicht klar kommt, oder unbedingt eine MF II Tastatur haben möchte, kann diese direkt an der CPU8088 anschließen Diese Tastatur wird dann vom BIOS erkannt.

Hier noch einmal zusammengefaßt eine Liste der vom BIOS unterstützten PC-Baugruppen:

- *Graphikkarten:*	MGA (Monochrom Graphik Adapter) Hercules Karte CGA (Colour Graphik Adapter) EGA (Enhanced Graphik Adapter) VGA (Video Graphics Adapter)
- *Festplatten:*	OMTI 5520 (Adapter für gängige Festplattenlaufwerke)
- *Sonstge:*	Parallele Schnittstelle (Centronics) Serielle Schnittstelle

Wie Sie sicher jetzt erkannt haben, gibt es zahlreiche Möglichkeiten mit dem NDR-Computer PC-kompatibel zu werden Die einfachste ist sicher, nur die CPU zu wechseln. Aber es gibt auch Moglichkeiten, sich seinen NDR-PC bis zur Festplatte aufzurusten und dies zu einem relativ gunstigen Preis

Wir hoffen Ihnen hier die Tragweite dieser Entwicklung etwas naher gebracht zu haben, und Ihnen auch einen Uberblick uber die Moglichkeiten des NDR-Rechner auf PC-Basis gegeben zu haben

1.3 Wie setzt man die CPU8088 ein

Bisher war es relativ schwierig, den NDR-Computer wirklich IBM- und damit MS-DOS kompatibel zu machen. Mit der neuen CPU8088 ist es jetzt ganz einfach, mit dem Betriebssystem MS-DOS zu arbeiten:

Die CPU, mit der bisher gearbeitet wurde, einfach herausnehmen, die Neue reinstekken und damit ist eigentlich alles Wesentliche getan. Alle Baugruppen, die vorher verwendet wurden, funktionieren noch immer und sogar mit dem alten NDR-Laufwerk ist es jetzt möglich, IBM-Format zu lesen.

Für die Zukunft wird es weitere Möglichkeiten der Systemkonfiguration geben: Über die Buskopplung NDR-IBM wird mit Hilfe des OMTI-Controller 5520B eine Festplatte unterstützt. Ebenfalls über die Buskopplung sollen parallele und serielle PC-Schnittstellen, sowie PC-Graphikkarten unterstützt werden. Die Erkennung, welche Karten im System stecken, wird das System (BIOS) selbst durchführen. Unten sehen Sie drei Möglichkeiten der Systemkonfiguration.

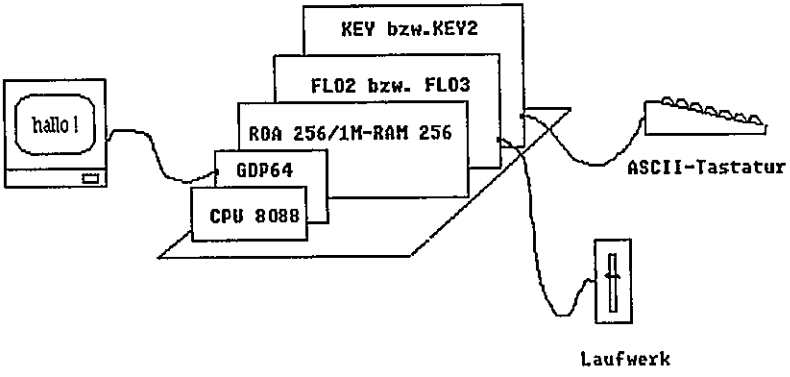


Abb.: Grundkonfiguration NDR mit CPU8088

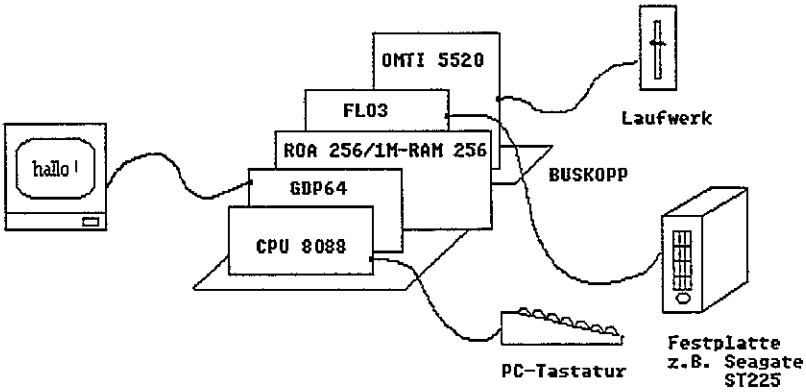


Abb mogliche Konfiguration NDR CPU8088

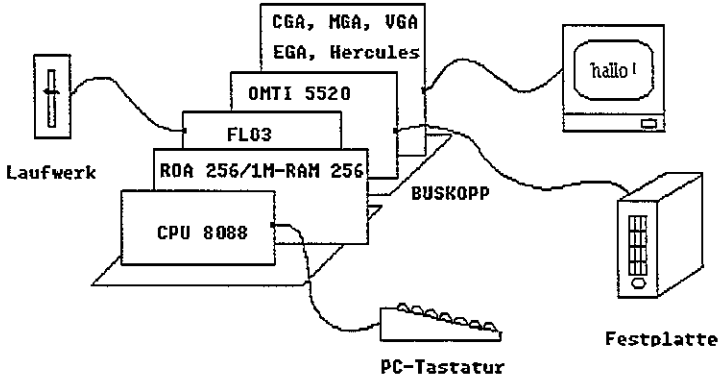


Abb mögliche Konfiguration mit Festplatte und PC-Graphikkarte

2. Technische Daten

- CPU8088-1 = 8088 CPU mit 10 MHz (auch NEC V20)
- Sockel für 8087-1 vorgesehen
- 3 Timer 8253
- Interruptcontroller 8259
- Taktgenerator und RESET-Baustein 8284-1 (10 MHz)
- Buscontroller 8288
- paralleler Schnittstellen-Baustein 8255
- Keyboard-Controller für PC-Tastatur
- Speaker Ausgang (8 Ohm Lautsprecher)
- WAIT-Logik
- 8K ROM (BIOS), max. 32K ROM
- 32K RAM
- BIOS unterstützt IBM monochrom Zeichensatz (Pseudographik)
- BUS: NDR-Bus
- Stromaufnahme +5V: 700mA
- Leiterplattenformat: 100 x 160

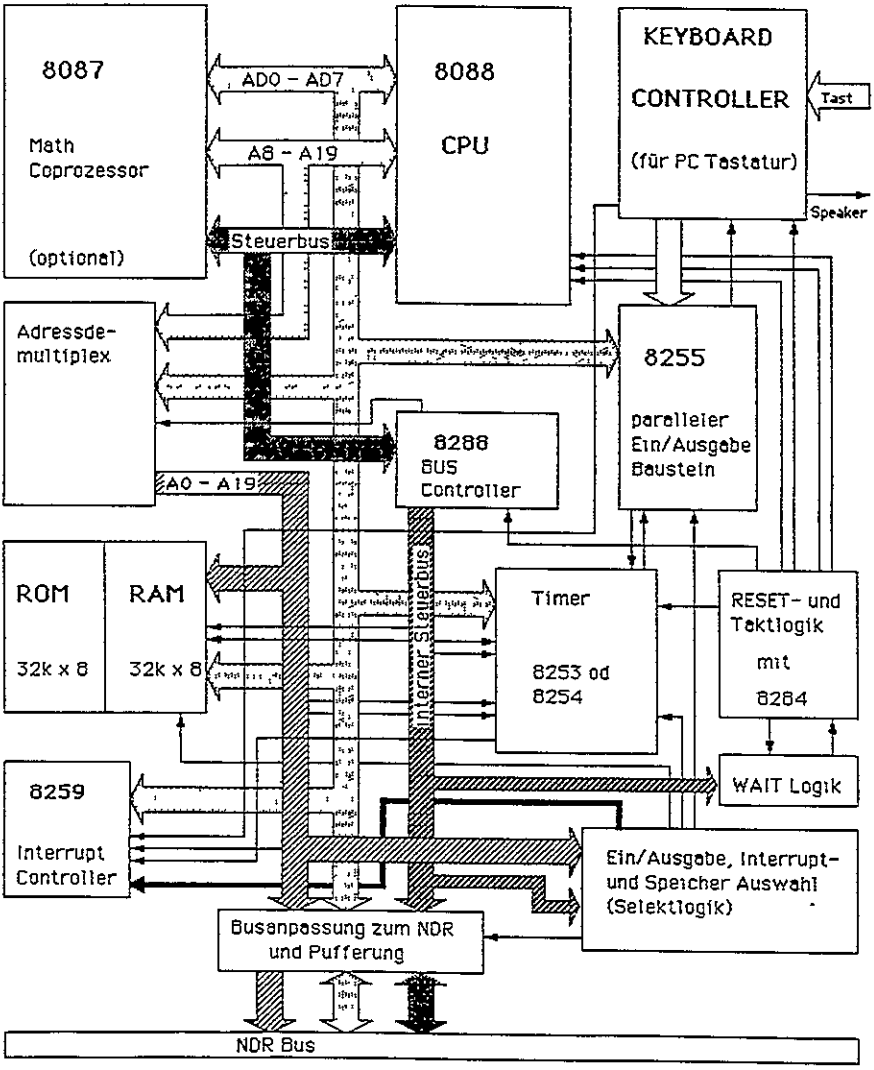
Sämtliche Karten des NDR-Computers können verwendet werden:

GDP64k
KEY mit paralleler Tastatur
ROA64, RAM64/256, ROA256/1M
FLO3 mit TEAC-Laufwerken (FD55F..)
sämtliche I/O Karten

Benchmarks:	Nortonfaktor 3,4
Landmark Test:	CPU Speed entspricht der eines AT's bei 3,6 MHz Landmark Faktor im Vergleich zum IBM PC (4,77 MHz): 1,8

3. Prinzipbeschreibung

3.1 Blockschaltbild



3.2 Beschreibung des Blockschaltbildes

Die Schaltung der CPU8088 ist relativ komplex und dementsprechend undurchsichtig. Aus diesem Grund ist ein übersichtliches Blockschaltbild unerläßlich. Selbst das Blockschaltbild ist noch relativ voll, aber es läßt sich leider nicht mehr vereinfachen, ohne dabei ins Triviale und Nichtssagende zu verfallen.

3.2.1 Die CPU

Die 8088 CPU ist das Herz der Baugruppe und des Rechners. Sie kommt entweder von INTEL und ist mit 8088-1 (10 MHz) bezeichnet, oder sie kommt von NEC und trägt die Aufschrift V20-10MHz. Die CPU steuert sämtliche Vorgänge im Rechner (Verwaltung der Speicher und Ein/Ausgabe Einheiten, Ablauf von Programmen, Rechenarbeit...).
Der Prozessor 8088 ist ein 8-Bit-Prozessor mit interner 16-Bit-Architektur. Der Prozessor ist softwarekompatibel zum 8086 und kann 1 MByte direkt adressieren. Nähere Angaben zur 8088 CPU siehe unter 10.2 Datenblatt 8088 CPU.

3.2.2 Der Coprozessor

Der 8087 ist ein Mathematik Coprozessor, der standardmäßig nicht im Lieferumfang eines PC's enthalten ist. Deshalb haben auch wir darauf verzichtet, den 8087, der doch relativ teuer ist, standardmäßig mitzuliefern. Bei Software-Problemen mit vielen mathematischen Gleichungen bringt der 8087 allerdings einen großen Zeitgewinn, da der Prozessor mathematische Funktionen direkt berechnet (Das heißt, dem Coprozessor wird nur ein Befehl übergeben und anschließend das Ergebnis abgeholt).

3.2.3 Der "Adreßdemultiplex"

Der Block Adreßdemultiplex macht vom Prinzip her nichts anderes, als die gemultiplexten Adreß- und Datenleitung AD0 bis AD7 zu trennen. Am Ausgang sind dann nur noch die Adreßleitungen A0 bis A7 und zusätzlich A8 bis A19, die außerdem noch gepuffert werden. Die Steuerung dieses "Adreßdemultiplexers" wird vom Signal ALE (Adress Latch Enable) gesteuert, das der Bus-Controller 8288 erzeugt.

3.2.4 Der Bus-Controller

Der BUS-Controller 8288 erzeugt aus den Prozessorsignalen S0, S1 und S2, die für den PC-Bus benötigten Signale:

-IOR	(Ein/Ausgabe lesen)
-IOW	(Ein/Ausgabe schreiben)
-MEMR	(Speicher lesen)
-MEMW	(Speicher schreiben)
-DT/R	(Daten senden oder empfangen)
-DEN	(Daten Freigabe)
-ALE	(Adreßfreigabe und Zwischenspeicherung)
-INTA	(Quittieren eines Interrupts)

3.2 Beschreibung des Blockschaltbildes

Die Schaltung der CPU8088 ist relativ komplex und dementsprechend undurchsichtig Aus diesem Grund ist ein übersichtliches Blockschaltbild unerläßlich Selbst das Blockschaltbild ist noch relativ voll, aber es laßt sich leider nicht mehr vereinfachen, ohne dabei ins Triviale und Nichtssagende zu verfallen

3.2.1 Die CPU

Die 8088 CPU ist das Herz der Baugruppe und des Rechners Sie kommt entweder von INTEL und ist mit 8088-1 (10 MHz) bezeichnet, oder sie kommt von NEC und tragt die Aufschrift V20-10MHz. Die CPU steuert samtliche Vorgange im Rechner (Verwaltung der Speicher und Ein/Ausgabe Einheiten, Ablauf von Programmen, Rechenarbeit .).
Der Prozessor 8088 ist ein 8-Bit-Prozessor mit interner 16-Bit-Architektur Der Prozessor ist softwarekompatibel zum 8086 und kann 1 MByte direkt adressieren Nahere Angaben zur 8088 CPU siehe unter 10.2 Datenblatt 8088 CPU

3.2.2 Der Coprozessor

Der 8087 ist ein Mathematik Coprozessor, der standardmaßig nicht im Lieferumfang eines PC's enthalten ist. Deshalb haben auch wir darauf verzichtet, den 8087, der doch relativ teuer ist, standardmaßig mitzuliefern Bei Software-Problemen mit vielen mathematischen Gleichungen bringt der 8087 allerdings einen großen Zeitgewinn, da der Prozessor mathematische Funktionen direkt berechnet (Das heißt, dem Coprozessor wird nur ein Befehl ubergeben und anschließend das Ergebnis abgeholt)

3.2.3 Der "Adreßdemultiplex"

Der Block Adreßdemultiplex macht vom Prinzip her nichts anderes, als die gemultiplexten Adreß- und Datenleitung AD0 bis AD7 zu trennen. Am Ausgang sind dann nur noch die Adreßleitungen A0 bis A7 und zusatzlich A8 bis A19, die außerdem noch gepuffert werden Die Steuerung dieses "Adreßdemultiplexers" wird vom Signal ALE (Adress Latch Enable) gesteuert, das der Bus-Controller 8288 erzeugt

3.2.4 Der Bus-Controller

Der BUS-Controller 8288 erzeugt aus den Prozessorsignalen S0, S1 und S2, die für den PC-Bus benotigten Signale:

-IOR	(Ein/Ausgabe lesen)
-IOW	(Ein/Ausgabe schreiben)
-MEMR	(Speicher lesen)
-MEMW	(Speicher schreiben)
-DT/R	(Daten senden oder empfangen)
-DEN	(Daten Freigabe)
-ALE	(Adreßfreigabe und Zwischenspeicherung)
-INTA	(Quittieren eines Interrupts)

Diese Signale werden zum Ablauf von Speicher- und Ein/Ausgabe-Zugriffen, sowie zur Interruptabwicklung benötigt.

3.2.5 Die Speicher

An Speichern befinden sich auf der Baugruppe maximal 64KB, die sich in 32K ROM und 32 K RAM aufsplitten. Der ROM-Bereich wird in der Regel für das BIOS genutzt. Wer die Karte aber als Single Board Computer für irgendwelche Steuerungen verwenden will, kann hier natürlich auch sein eigenes EPROM stecken. Der RAM Bereich (statisches RAM 32K * 8), dient als "Shadow RAM". Dieser Speicher simuliert eine Graphikkarte, die normalerweise bei einem PC auf diesem Speicherbereich liegt. Dadurch ist es möglich, im Hintergrund (Schattenspeicher) die Graphik aufzubauen und durch Vergleich mit dem GDP Speicher auf der GDP64 zu aktualisieren.

3.2.6 Der Interrupt-Controller

Der Interrupt-Controller 8259 ist für die Abwicklung der Interrupts verantwortlich. Dieser Baustein stellt 8 Interrupt-Eingänge zur Verfügung. In unserer Schaltung sind nur 5 belegt:

Zeitgeber
Tastatur
Parallel-Drucker
2 Asynchron-Adapter

Wird nun ein Interrupt von einem dieser Geräte ausgelöst, so gelangt dieser zum 8259, der dann den Prozessorinterrupt aktiviert und auf das Quittierungssignal (INTA) wartet. Ist dieses aktiv, so wird der Interrupt-Vektor eingelesen und in die Interrupt-Routine verzweigt. Außerdem bietet der Baustein noch einige Extras, wie Prioritätsbehandlung, Kaskadierbarkeit ...

3.2.7 Timer

Der Baustein 8253 oder 8254 (schnelle Version) beinhaltet 3 Timer, wobei bei unserer Schaltung nur zwei verwendet werden. Der erste Timer wird ausschließlich für die Systemuhr benutzt. Dabei wird ein Takt von 1,19 MHz angelegt und der Timer macht dann nichts anderes, als die Taktzyklen zu zählen und in bestimmten Zeitabständen einen Interrupt zu senden. Dabei wird dann bei jedem 18. Interrupt der Sekundenzähler um Eins erhöht. Der zweite Timer wird zur Synchronisation der seriellen Tastaturdaten verwendet.

3.2.8 Die RESET- und Taktlogik

Die RESET- und Taktlogik erzeugt zunächst einmal den Systemtakt für den Prozessor 8088 und den Coprozessor 8087. Dabei wird die Quarzfrequenz von 28,62 MHz durch drei geteilt, also 9,54 MHz. Außerdem wird über diesen Baustein auch der Takt für den Timer erzeugt. Dabei wird die Quarzfrequenz vom 8284 erst durch drei, dann durch zwei und anschließend mit zwei Flip-Flops noch zweimal durch zwei geteilt. Dadurch erhalt man den Timertakt von 1,1925 MHz, der für die Systemuhr verwendet wird.

Zum Zweiten erzeugt dieser Baustein ein RESET-Signal aus einem analog erzeugten RESET (RC-Glied für Power on RESET und RESET-Taster). Dieses analog ansteigende Signal wird an den Eingang -RES (mit Schmitt-Trigger) angelegt und auf dem Ausgang RESET mit einer bestimmten Periodendauer an den Prozessor weitergeleitet

Zum Dritten enthalt der Baustein eine Logik zum Synchronisieren des READY-Signals. Dabei können zwei READY-Signale von außen angelegt werden, die dann mit dem Systemclock synchronisiert an den Prozessor weitergeleitet werden. Hier wird nur ein READY-Signal verwendet. Dieses READY-Signal wird von der WAIT-Logik erzeugt.

3.2.9 Die WAIT-Logik

Damit wären wir bereits bei der WAIT-Logik angelangt. Der Ausdruck WAIT hat bei den Computer-Anwendern einen negativen Beigeschmack, weil damit immer eine Verlangsamung des Rechners gemeint ist. Aus systemspezifischen Gründen sind aber WAITs bei bestimmten Zugriffen unerläßlich, um schnelle teure Bausteine zu sparen. Natürlich versucht man, WAITs nur dort einzubauen, wo sie am wenigsten Zeit kosten. Dies ist bei den Ein/Ausgaben der Fall, da auf diese Einheiten im Vergleich zu Speichern nur sehr selten zugegriffen wird. WAIT-States werden in unserer Schaltung bei Zugriffen auf den Timer, auf den Interrupt-Baustein und auf den parallelen Ein/Ausgabe-Baustein eingefügt. Außerdem natürlich auch, wenn vom NDR-Bus ein WAIT-Signal anliegt.

3.2.10 Die Ein/Ausgabe

Der parallele Ein/Ausgabe-Baustein 8255 dient nur zum Einlesen der Tastatur, zur Steuerung des Ablaufs des Keyboard-Controllers, sowie zur Datenausgabe an den Lautsprecher.

3.2.11 Der Keyboard-Controller

Der KEYBOARD-Controller auf der CPU8088 ist so ausgelegt, daß eine PC-Tastatur (auch MF-II Tastaturen, die auf PC- oder XT-Modus umschaltbar sind) angeschlossen werden kann. Dabei muß diese PC-Tastatur aber nicht verwendet werden, denn es funktioniert auch die KEY und daran die Tastaturen TAST1, TAST2 und TAST3. Das BIOS erkennt selbst welche Tastatur vorhanden ist. Die PC-Tastatur hat die höhere

Priorität, d.h. steckt die KEY aus Versehen im Rechner und ist eine PC-Tastatur an der CPU8088 angeschlossen, so wird die PC-Tastatur erkannt und verwendet. Eine PC-Tastatur bietet natürlich mehr Bedienungskomfort. Vor allem die Darstellung der Sonderzeichen und das Betätigen der Funktionstasten verlangt bei den bisherigen NDR-Tastaturen doch einige Klimmzüge, aber es können mit TAST1, TAST2 und TAST3 alle möglichen Zeichen dargestellt werden.

Nun zurück zum KEYBOARD-Controller: Dieser macht nichts Anderes, als die seriell ankommenden Zeichen in ein paralleles Format umzuwandeln. Diese parallelen Zeichen werden dann an den 8255 weitergegeben und eingelesen. Der Ablauf beim Einlesen eines Zeichens wird über Interrupt gesteuert. Das heißt, kommt ein Zeichen von der Tastatur, wird ein Interrupt ausgelöst, das Zeichen eingelesen und in einem Puffer abgelegt (Softwarepuffer). Sollte keine PC-Tastatur angeschlossen sein, sondern eine NDR-Tastatur, so verfügt auch diese über den Softwarepuffer. TAST1, TAST2 und TAST3 haben also bei Betrieb mit MS-DOS einen Zeichenpuffer (Größe des Zeichenpuffers im DOS definiert).

3.2.12 Die Select-Logik

Die Select-Logik wählt, wie der Name schon sagt, die einzelnen Speicher oder Ein/Ausgabe-Bausteine aus. Das heißt, liegt z.B. der ROM-Speicher von Adresse 0 bis 7FFFh, so wird, wenn eine Adresse in diesem Adreßraum angesprochen wird, dieser Baustein aktiviert (Chip-Select-Signal). Ebenso verhält es sich bei den Portadressen von Ein/Ausgabe Bausteinen. Diese Chip-Select Signale werden von der Select-Logik erzeugt.

3.2.13 Die Busanpassung

Die Busanpassung zum NDR-Bus besteht im wesentlichen aus zwei Bausteinen.
Zum Einen werden die Signale

-IOR, -IOW, -MEMR, -MEMW

auf die NDR-Bussignale

-RD, -WR, -IORQ, -MREQ

umgelegt und das RESET-Signal invertiert,
zum Andern werden die Daten über einen bidirektionalen Bustreiber gepuffert und bei Zugriff auf externe Baugruppen (Speicher, Ein/Ausgabe) freigegeben.

Die Beschreibung des Blockschaltbildes wurde absichtlich etwas ausführlicher gehalten, um die einzelnen Blöcke etwas zu durchleuchten. Wollen Sie noch tiefer einsteigen, können Sie in der Schaltungsbeschreibung unter Kapitel 7 und in den Datenblätter unter Kapitel 10 Detailinformationen nachlesen.

4. Aufbauanleitung

4.1 CMOS-Warnung

CMOS-Bausteine sind hochempfindlich gegen elektrostatische Aufladung! Bewahren oder transportieren Sie CMOS-Bausteine nur auf dem leitenden Schaumstoff! Alle Pins müssen kurzgeschlossen sein.

Tip: Fassen Sie an ein geerdetes Teil (z B Heizung, Wasserleitung), bevor Sie einen Baustein berühren.

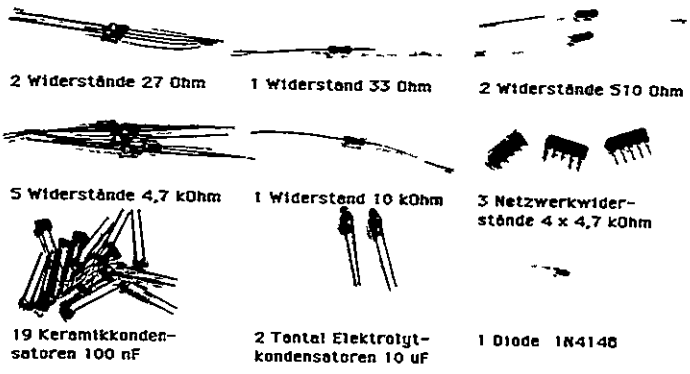
Bitte beachten Sie hierzu auch den Artikel "Schutzmaßnahmen für MOS-Schaltungen" in unserer Zeitschrift LOOP3.

4.2 Stückliste

1	11267	CPU8088H	CPU8088-Handbuch	
1	61256	CPU8088P	CPU8088-Leiterplatte mit Lötstoplack und Bestuckungsdruck	
2	60183	SO14	IC Sockel 14 polig	
1	60186	SO18	IC Sockel 18 polig	
11	60187	SO20	IC Sockel 20 polig	
3	60190	SO28	IC Sockel 28 polig	
1	60188	SO24	IC Sockel 24 polig	
3	60193	SO40	IC Sockel 40 polig	
2	69167	74LS74	J9,10	D-Flip Flop mit Preset
1	60115	74LS245	J14	8-Bit Bus-Transceiver
1	61243	74LS322	J5	8-Bit Schiebereg. mit 'sign extend'
3	60125	74LS373	J15,17, J18	8-Bit D Register mit Tri-State Ausgängen
1	61242	8253	J3	Timerbaustein
1	61241	8255	J4	PIO
1	61240	8259	J8	Interrupt-Controller
1	61239	8284-1	J12	Clock Generator
1	61238	8288	J13	Buscontroller 8MHz
1	61257	2764	J6	BIOS-EPROM
1	11068	82256	J7	RAM 32KB, 120ns, stat.
1	61237	NEC V20	J2	CPU 8088-kompatibel
1	61269	PAL16L8	J19	PAL1

1	61256	PAL16L8	J16	PAL2
1	61270	PAL16L8	J21	PAL3
1	61271	GAL16L8	J11	PAL4
1	61272	PAL16R4	J20	PAL5
1	61247	XTAL	Q1	Quarz 28.62 MHz
1	60290	1N4148	D1	Silizium-Diode
1		BC107	T1	NPN-Transistor
3	60758	RN4*4K7	RN1...3	Netzwerkwid. 4x4.7 kOhm
5	60648	R *R7 - R10, R12*		Widerstand 4.7 kOhm (gelb, violett, rot)
1	60617	R *R1*		Widerstand 10 kOhm (braun, schwarz, orange)
1	60626	R *R5*		Widerstand 1 kOhm (braun, schwarz, rot)
2	61248	R *R2,3*		Widerstand 510 Ohm (grün, braun, braun)
1	60642	R *R4*		Widerstand 33 Ohm (orange, orange, schwarz)
2	60635	R *R6,11*		Widerstand 27 Ohm (rot, violett, schwarz)
2	60248	C 10uF *C2,21*		Tantalelko 10uF/16V
19	60239	C 100nF *C1,3...20*		Keramikkond. 100nF
1	60301	Taster	S1	Resettaster
1	61255	Stecker	BU1	Tastaturanschl. DIN41524
2	60493	Stifte JMP3,	ST1	Stiftleiste 2 x 1 pol. gerade
1	60502	Stifte JMP1,	ST	Stiftleiste 4 x 2 pol. gerade
1	10405	Stifte ST2	ST2	Stiftleiste 18 x 1 pol. gew.
1	10406	Stifte ST2	ST2	Stiftleiste 36 x 1 pol. gew.
2	60486	Jumper		Shuntstecker

Passive Bauelemente CPU8088



Passive Bauelemente CPU8088

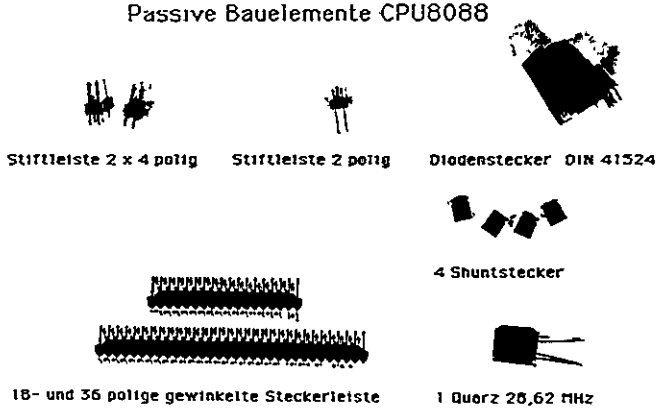


Abb 6: Bauteile CPU8088

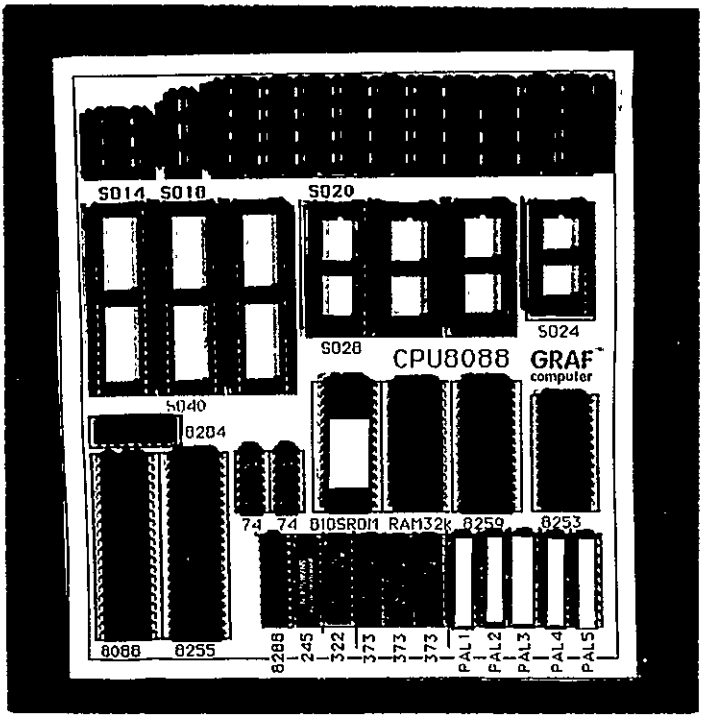


Abb.: ICs und Sockel

4.3 Aufbau Schritt für Schritt

Benötigtes Werkzeug.

Lotkolben mit temperaturgeregelter Spitze
Lötzinn, sauerefrei, mit Kolophonium-Seele
Pinzette
Elektroniker-Seitenschneider

Die Leiterplatte ist sehr eng bestuckt. Bitte bauen Sie nur dann selber auf, wenn Sie über ausreichend Erfahrung im Aufbau von eng bestückten Leiterplatten verfügen.

Die Lotseite erkennt man an der Aufschrift "Lotseite". Ausschließlich auf dieser Seite der Leiterplatte wird gelötet. Die Bauteile werden nur auf der Bestuckungsseite bestuckt.

Beim Aufbauen der Baugruppe arbeiten Sie sich am gunstigsten etappenweise voran, das soll heißen. alle Teile einer Bauteilgruppe werden auf einmal bestuckt. Die Anschlußdrähte der Bauteile werden durch die entsprechenden Lotaugen geschoben, bis der Bauteilkörper flach auf der Platinenoberseite aufliegt. Die Anschlußdrahte werden, wenn moglich, auf der Lotseite leicht abgewinkelt, damit das betreffende Bauteil eine erste Fixierung erhalt Nachdem Sie sicher sind, daß keine Fehlbestückung (falsche Bauteile, Position oder Lage) vorliegt, werden alle bisherigen Anschlusse verlotet und danach, wenn notwendig, zu weit uberstehende Anschlußdrahte gekurzt.

Eine etwas andere Vorgehensweise empfiehlt sich bei den IC-Sockeln
Sie werden zuerst alle, von der Bauteileseite her, eingesetzt und danach mit einem Stuck Karton abgedeckt, damit sie beim Umdrehen der Baugruppe in ihrer Position gehalten werden Bevor nun endgültig alle Anschlußbeine verlötet werden, sollten erst einmal zwei Beinchen pro Sockel gelotet werden (diagonal versetzt) Ungleichmäßigkeiten im Abstand vom Sockel zur Platinenoberflache und - ganz wichtig - eine eventuell falsche Lage einzelner Sockel (Kerbe in der Fassung muß in die selbe Richtung weisen, wie die 'Nase' auf dem Bestuckungsdruck) konnen zu diesem Zeitpunkt noch korrigiert werden Das vorlaufige Anheften empfiehlt sich übrigens bei allen Bauteilen mit mehr als zwei Anschlußdrahten (Transistoren, Stiftleisten etc)!

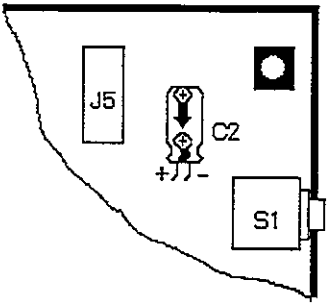
Nach den vielen Ratschlägen, frisch ans Werk:

Beginnen Sie mit dem Einloten der liegenden Einzelwiderstande (***R1 ..R12***) und der Diode ***D1*** Die Widerstandswerte können anhand der aufgedruckten Farbringe abgelesen werden. Die Bedeutung der Farben, bzw die erforderliche Farbenfolge ist fur jeden Widerstandswert in der Stuckliste angegeben. Achtung Der Widerstand ***R6*** ist auf dem Bestuckungsdruck CPU8088 r2 falsch bezeichnet. Hier darf kein 4,7 kOhm

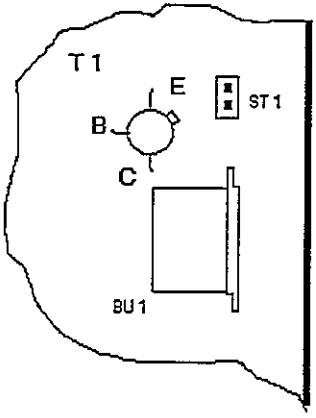
Widerstand eingelötet werden, sondern ein 27 Ohm Widerstand. Beim Einsetzen der Diode ist auf die richige Polung zu achten! Ein Strich rund um die Diode markiert die Kathode. Auf dem Bestückungsdruck ist die Kathode mit dem Strich auf der Pfeilspitze gekennzeichnet.

Nun folgt die Bestückung der IC-Sockel, wie oben beschrieben. Die Orientierung der Fassungen ist auf der Bestückungsseite der Leiterplatte durch den Aufdruck sehr deutlich zu erkennen: Die eingezeichnete 'Nase' muß sich mit der am Sockel eingearbeiteten Kerbe (falls vorhanden) decken. Achten Sie bitte darauf, daß Sie nicht einen 14-poligen Sockel auf den Platz eines 16- oder 18-poligen setzen.
Beim anschließenden Einlöten der 54-poligen (18 + 36) NDR-Bus-Steckerleiste ST2, der Jumper 1...3 sowie des Steckers ST1 müssen Sie, wie oben angedeutet, darauf achten, daß die Isolierkörper parallel zur Leiterplatte liegen. Jumper 1 und 2 sind doppelreihige Stiftleisten, Jumper 3 ist einreihig.
Bei den Tantalkondensatoren ***C2, C21,*** muß auf deren Polung geachtet werden: Der Pluspol ist auf der Platine und auf dem Kondensator entsprechend gekennzeichnet (siehe auch Skizze 1). Die Keramikkondensatoren ***C1,C3..C20*** sind ungepolt und können, ohne auf die Polung zu achten, eingelötet werden.
Die Netzwerkwiderstände ***RN1...RN3*** haben jeweils einen Anschluß für den Pluspol, der auf dem Netzwerkwiderstand und auf der Leiterplatte mit einem (kleinen) Punkt gekennzeichnet ist. Es handelt sich um 4 * 4,7 kOhm Netzwerkwiderstände u.a. mit dem Aufdruck: 472 (= 47 * 10^2 Ohm).
Die Anschlüsse des Transistors T1 sind ebenfalls auf der Platine, bzw im Bestückungsplan, gekennzeichnet (siehe Skizze 2). Betrachtet man den Transistor (Metall-Gehäuse TO18), so ist der Emitter dasjenige Beinchen, das der Lasche am Gehäuserand am nächsten ist. Bei Transistortypen mit Kunststoffgehäuse liegt der Emitter in der Regel bei der rechten Körperkante (wenn der Transitor in Normallage stehend betrachtet wird).
Abschließend werden noch der Resettaster S1, der Tastaturanschlußstecker BU1 und der Quarz Q1 eingelötet.

Bevor Sie jetzt schon ungeduldig die IC's einstecken wollen, sollten Sie mit Kapitel 5.1 weiter weitermachen!



Skizze 1. Polung C2



Skizze 2. Polung Transistor

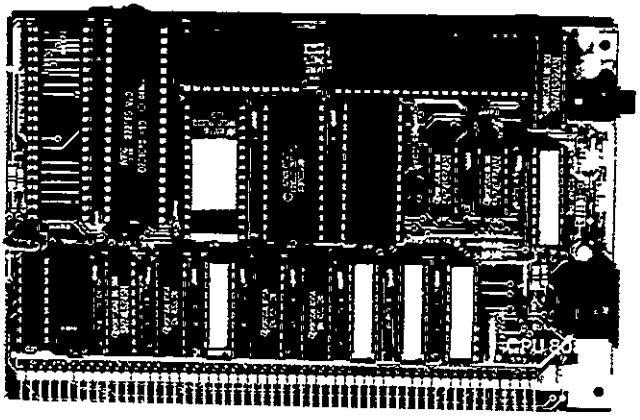


Abb.: Foto Fertiggerät CPU 8088

5. Testanleitung

5.1 Erste Prüfung ohne IC 's

Dieser Test wird auf der, mit den Sockeln und den passiven Bauelementen bestückten, Leiterplatte durchgeführt (Der Transistor Q1 und die Diode können natürlich auch schon eingelötet sein).

Überprüfen Sie zunächst mit einem Ohmmeter den Widerstand zwischen den Versorgungsleitungen: +5V (ST2/Pins 4,5) und Masse (ST2/Pins 6, 7, 52, 53). Hier darf der gemessene Wert keinesfalls niederohmig sein! Werte größer als 10kOhm sind normal. Wenn Sie einen abweichenden Wert feststellen, fahren Sie bitte mit Kapitel 6 fort (Fehler suchen)!

Wenn Sie die Möglichkeit haben, Ihr System auch ohne die neue CPU8088 zu betreiben (Z80- oder 680xx-System), stecken Sie die bis jetzt halbfertige Baugruppe CPU8088 erst einmal dort dazu. Nach dem Einschalten des Systems muß der Rechner auch weiterhin problemlos funktionieren (Test der Daten- und Adressleitungen auf Kurzschluß). Falls nein - weiter mit Kapitel 6.

Haben Sie diese Möglichkeit nicht, können Sie als vorsichtiger Mensch die BUS-Kontakte an ST2 mit dem Ohmmeter überprüfen (Jeder gegen Jeden). Auch hier müssen hochohmige Werte zu messen sein. Anschließend wird die Baugruppe auf den BUS gesteckt (wenn sie nicht bereits in einem anderen System steckt) und die Spannungsversorgung an den BUS gelegt.

Alle IC-Sockel werden auf die Versorgungsspannung von +5V geprüft. Bei Standard-TTL-Bausteinen, den fünf PALs und den Speicherbausteinen J6 und J7 liegt die Versorgungsspannung jeweils auf dem letzten Pin einer Fassung (z.B. bei 14-poligen Fassungen an Pin 14; bei 20-poligen Fassungen an Pin 20) an. Die Masse liegt jeweils auf dem letzten Pin der ersten Reihe (bei 14-poligen Fassungen Pin 7; bei 20-poligen Fassungen an Pin 10). Auch bei den INTEL-Bausteinen 8088 (bzw. V20, J2), 8087 (J1), 8253 (J3), 8288 (J13), 8284 (J12) und 8259 (J8) herrschen die gleichen Verhältnisse vor. Lediglich der INTEL-PIO 8255 (J4) macht hier eine Ausnahme: +5V liegt an Pin 26 und Masse (GND) liegt an Pin 7. Sind die Versorgungsspannungen an den richtigen Pins gemessen worden, können die IC's eingesetzt werden. Schalten Sie dazu den Rechner aus und ziehen Sie die Baugruppe heraus. Beim Einsetzen muß auf die Richtung der IC's geachtet werden. Die Markierung auf dem IC muß mit der Kerbe in der Fassung übereinstimmen (vgl. Bestückungsplan).

Prüfen Sie noch einmal die Lötseite der Baugruppe auf eventuelle Lötbrücken! Zur Stellung der Jumper 1...3, siehe Kapitel 5.4.

5.2 Test im System

5.2.1 Startkonfiguration

Um der CPU8088 erste Lebenszeichen zu entlocken, ist folgende Mindestkonfiguration erforderlich:

BUS2,
GDP64xx + Monitor,
PC-Tastatur oder 'NDR-Tastatur' + KEY,
RAM-Speicher (ROA64),
Netzteil mit 5V/2A (POW5V).

5.2.2 Zum ersten Mal Einschalten

Wenn Sie eine PC-Tastatur einsetzen wollen, wird diese direkt an der CPU8088-Baugruppe (BU1) eingesteckt, ansonsten ist die NDR-übliche Kombination (Tastatur + KEY-Baugruppe) zu verwenden. Der RAM-Speicher muß ab Adresse 00000h zu adressieren sein (Jumper auf der ROA64, ROA256 oder RAM256 entsprechend gesteckt). Nachdem die Spannungsversorgung eingeschaltet wurde, sollte sich am betriebsbereiten Monitor eine BIOS-Copyrightmeldung zeigen. Wenn an ST1 der CPU8088 ein (Miniatur-) Lautsprecher angeschlossen ist, ist außerdem ein kurzer 'Piepser' zu hören. Gibt der Lautsprecher keinen Ton von sich oder "piepst" er zweimal, liegt im ersten Fall ein Hardwarefehler auf der CPU vor oder im zweiten Fall ein Konfigurationsfehler vor.

5.2.3 Vorbereitungen für MS-DOS

Um nun von einer MS-DOS Systemdiskette (V3.3) zu booten, sind weitere Voraussetzungen bezüglich der Systemkonfiguration zu erfüllen:
Das System muß um eine FLO2, bzw. FLO3 und um ein TEAC-Laufwerk 'NDR-Standard' ergänzt werden (Interrupt auf der FLO2 bzw. FLO3 Baugruppe geöffnet).

Achtung!
Setzen Sie die EGA- Karte mit der unter 1.2 angesprochenen Buskopplung ein, so mussen Sie die Standardadressen der FLO3 (auch FLO2) ändern: von bisher C0h auf F0h.

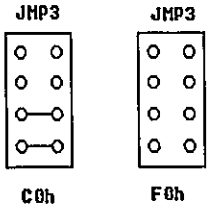


Abb.: Porteinstellung FLO3

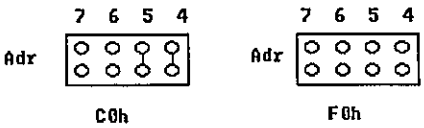


Abb.: Porteinstellung FLO2

Damit einhergehend ist auch ein Netzteil mit zusätzlich +/-12V/2A (z.B.: NE3) erforderlich. Der System-RAM-Speicher muß auf mindestens 256KB aufgestockt werden. Der 'NDR-PC' ist jetzt in der Lage von 5 1/4"-Disketten im IBM-Standardformat (360KB) das Betriebsystem MS-DOS zu booten. Nachdem Sie sich in der MS-DOS-Umgebung befinden, können Sie noch die Arbeitsweise der System-Softwareuhr überprüfen, indem Sie gewisse Zeit nach dem Booten die momentane Uhrzeit mit: TIME<Return> abfragen. Wenn dieser Test die Uhrzeit aktuell anzeigt, ist gewährleistet, daß die wesentlichen Funktionen der CPU8088 erfüllt sind (Timer- und Interruptfunktionen).

5.4. Jumpereinstellungen

Mit nur drei einzustellenden Jumpern ist hier relativ wenig zu tun, jedoch ist keiner voreingestellt, wie es bisher üblich gewesen ist.

JMP1:

Hiermit wird die Größe des eingesetzten BIOS-EPROMs eingestellt (vorläufig reichen noch 8K-EPROMs '2764' für das BIOS).

```
JMP1
o--o      für 8K-EPROM 2764 (Standardeinstellung)
o  o
```

```
JMP1
o  o      für 32K-EPROM 27256
o--o
```

JMP2:

Damit kann die Taktrate für den Timerbaustein (und damit auch für die Softwareuhr) umgeschaltet werden. Normalerweise (mit der 10 MHz-Standardbestuckung) muß JMP2 wie folgt gesteckt sein, damit sich eine 'Triggerfrequenz' von ca. 18Hz ergibt:

JMP2
o o für Standardbestückung mit
o--o Quarz 28,62MHz

Alternativ kann aber auch ein Quarz mit der Frequenz von 14,31 MHz eingesetzt werden, um einen CPU-Takt von 4,77 MHz zu erhalten (Kompatibilität zu bestimmten Programmen, preiswertere Bausteine). Damit aber die Systemuhr trotzdem nicht um die Hälfte zu langsam geht, kann mit JMP2 eine Timertakthalbierung umgangen werden, indem er so umgesteckt wird.

JMP2
o--o für alternative Bestückung mit Quarz 14,31MHz
o o

JMP3:

Wenn Sie einen mathmatischen CO-Prozessor (8087-1) einsetzen, muß JMP3 gebrückt werden, ansonsten bleibt er offen.

6. Anweisungen zur Fehlersuche

Zuallererst: Entfernen Sie die Flußmittelrückstände auf der Lötseite der Baugruppe. Am besten mit Spiritus und einer kleinen, weichen Bürste. Beachten Sie aber die Hinweise auf der Spiritusflasche zum Schutz Ihrer Gesundheit und der Ihrer Mitmenschen!

Wir wollen Ihnen hier Vorschläge machen, wie eine systematische Fehlersuche mit und ohne Oszilloskop vor sich gehen kann:

1. Altes System: Sind die bisher verwendeten Baugruppen in Ordnung? (Falls Sie noch eine andere CPU haben: Funktionierte das System ohne die CPU8088?)

2. Jumper: Kontrollieren Sie noch einmal die Stellung der Jumper nach.

3. Sichtprüfung: Können Sie bei einer Sichtprobe unsaubere Lötstellen entdekken? Solche unsauberen Lötstellen, zuviel Zinn oder sogenannte 'Fäden', können Kurzschlüsse verursachen. Löten Sie solche Stellen noch einmal nach und entfernen Sie alles überflüssige Zinn.

4. ICs: Überprüfen Sie noch einmal, ob alle ICs an der richtigen Stelle in der richtigen Richtung stecken. Am besten vergleichen Sie deren Lage nochmal mit dem Bestückungsplan.

5. Gepolte Bauteile: Überprüfen Sie die Lage der gepolten Bauteile (Elkos, Dioden, usw.).

6. Lötstellen: Schauen Sie sich Ihre Platine noch einmal genau an: Können Sie irgendwo 'kalte' Lötstellen entdecken? Kalte Lötstellen erkennt man daran, daß sie nicht glänzen, sondern matt und trüb sind. Oder entdecken Sie etwa vergessene, gar nicht gelötete Stellen? Im Zweifelsfall alle 'verdächtigen' Stellen nochmal nachlöten.

7. Zu heiß gelötet? Wenn der Lötkolben zu heiß eingestellt war, oder wenn Sie beim Löten zu lange auf dem Lötauge geblieben sind, kann es passieren, daß sich das Lötauge oder Leiterbahnen von der Leiterplatte gelöst haben und Unterbrechungen bilden. Eventuell wurden auch Durchkontaktierungen oder sogar einzelne Bauteile zerstört.

Beheben Sie solche Fehler, wenn Sie sie finden, mit einem kleinen Drahtstuck, um die Unterbrechung zu uberbrucken. Bauteilefehler, besonders defekte ICs, sind ohne Meßwerkzeuge wie Oszilloskop oder Logikanalysator nur schwierig festzustellen.

8. Leiterbahnen

Prüfen Sie mit einem Durchgangsprufer anhand des Layouts alle Leiterbahnen auf Durchgang. Sie tun sich leichter, wenn Sie vorher alle ICs aus ihren Fassungen nehmen. Der Übersicht wegen empfiehlt es sich, überprüfte Leiterbahnen mit Bleistift oder mit Farbstift zu markieren.

9 Versorgungsspannung

Messen Sie mit einem Digitalvoltmeter am Bus die Versorgungsspannung nach. Toleranzen von +/- 5% sind erlaubt.

Haben Sie alles uberpruft und nichts gefunden, so ist vermutlich ein Bauteil defekt. Wenn Sie einen Prufstift oder ein Oszilloskop haben, dann können Sie jetzt nachsehen, ob an den Ausgängen jeweils die entsprechend richtigen Signale anliegen Welche Signale wo anliegen mussen, können Sie der Schaltungsbeschreibung, dem Schaltplan und Ihren eigenen Uberlegungen entnehmen.
Ohne Messgerate müssen Sie alle Bauteile systematisch austauschen, bis Sie das fehlerhafte Bauteil gefunden haben. Verwenden Sie dazu eventuell eine zweite Baugruppe, beispielsweise die eines Freundes oder eines Bekannten.
Sollten Sie uberhaupt nicht zurande kommen, hilft Ihnen unser Pauschal-Reparatur-Service, dessen Bedingungen Sie der Preisliste entnehmen können

7. Schaltungsbeschreibung

Die Schaltung der CPU8088 kann in 13 Blöcke zerlegt werden, wie Sie im Blockschaltbild *(Kapitel 3)* bereits erkennen konnten. Diese Aufsplittung in Blöcke ist nötig, um die Funktion der Schaltung allgemein begreifbar zu machen. Ausgehend von diesen Blöcken wollen wir hier versuchen, den Stromlaufplan dieser komplexen Schaltung zu erklären. Wir müssen hier allerdings voraussetzen, daß Sie die Beschreibung des Blockschaltbildes gelesen und verstanden haben.

7.1 Der Prozessor

Der Prozessor 8088 von Intel (auch V20 von NEC) ist ein 8-Bit-Mikroprozessor, der der 8080- und 8085-Familie entspringt, aber zu diesen nicht aufwärtskompatibel ist. Dieser Prozessor steckte in dem ersten IBM-Mikrocomputer, dem legendären PC, den schließlich alle Welt nachzubauen versuchte. Mittlerweile wird dieser 8 Bit Mikroprozessor aber mit 10 MHz betrieben, und ist dann doch um einiges schneller als der damalige IBM-PC. Aber zurück zum Prozessor. Die Datenleitungen und Adreßleitungen 0 bis 7 sind gemultiplext und werden auf jeweils einem Pin herausgeführt. Ob nun Daten oder Adressen an diesen Eingängen anliegen, wird durch das Signal ALE gesteuert, das wiederum vom BUS-Controller 8288 erzeugt wird. Das Timing Diagramm hierzu kann im Datenblatt zum 8288 unter Kapitel 10 nachgeschlagen werden. Die Adreßleitung A8 bis A19 sind nicht gemultiplext. Mit den 20 Adreßleitungen kann der 8088 1 MByte direkt adressieren.

7.2 Die Statussignale

Die Statusleitungen -S0 bis -S2 zeigen an, was der Prozessor im Moment macht. Die Statussignale werden ebenfalls vom 8288 decodiert.

Tabelle 1: Wahrheitstabelle der Statussignale **-S0, -S1** und **-S2:**

-S0	**-S1**	**-S2**	*auszuführende Operation*
0	0	0	Interrupt Quittierung
0	0	1	Ein/Ausgabe-Lese-Zyklus
0	1	0	Ein/Ausgabe-Schreib-Zyklus
0	1	1	Halt
1	0	0	OP-Code Speicher-Lese-Zyklus
1	0	1	Speicher-Lese-Zyklus
1	1	0	Speicher-Schreib-Zyklus
1	1	1	Passiver Zustand

Vom 8288 werden aber nur die Ein/Ausgabe- und Speicher-, Lese- und Schreibzugriffe decodiert, sowie das Interrupt-Quittierungssignal.
Die beiden Statussignale OP-Code Speicher-Lese-Zyklus und Speicher-Lese-Zyklus werden zusammengefaßt zum Speicher-Lese-Signal (J13/7)

Die beiden Statussignale QS1 und QS0 steuern die Abarbeitung von Stapelbefehlen. Diese beiden Signale führen lediglich zum math. Coprozessor 8087-1. Nähere Einzelheiten siehe unter *Kapitel 10*.

Die Eingänge CLK, READY und RESET werden vom 8284 gesteuert. Auf den CLK Eingang wird ein Takt von 28,62 MHz gelegt, der durch drei (=9,54 MHz) geteilt wird, was dem Arbeitstakt der CPU entspricht. Das READY-Signal ist nichts Anderes, als das invertierte WAIT Signal, das bei Zugriffen auf Speicher oder Ein/Ausgaben erzeugt wird, und vom 8284 synchronisiert wird.

Der Befehlsatz des 8088 ist dem des 8085 sehr ähnlich. Es würde den Rahmen dieser Beschreibung sprengen, wenn wir hier näher darauf eingehen würden. Im Datenblatt unter Kapitel 10 ist eine Kurzbeschreibung der möglichen Befehle zu finden.

7.3 Der Coprozessor

Der 8087 ist ein mathematischer Coprozessor, der direkt mit dem 8088 verbunden wird. Ist der Coprozessor eingesteckt, verhalten sich diese beiden wie ein einziger Prozessor, das heißt, es kommen die math. Befehle hinzu, die aber so ausgeführt werden, als ob sie im Prozessor selbst ausgeführt wurden. Dadurch ergeben sich natürlich große Zeitgewinne, wenn häufig mathematische Operationen verwendet werden. Allerdings muß die Software natürlich die Coprozessor-Befehle unterstützen.

7.4 Die Reset- und Taktlogik

Kehren wir noch einmal kurz zum Takt- und RESET-Baustein 8284 zurück. Unter Kapitel 10 ist das Schaltbild dieses Bausteines aufgeführt. Er enthält einen internen Oszillator, der nur noch durch einen externen Quarz stabilisiert werden muß Dieser Quarz hat bei uns eine Frequenz von 28,62 MHz und liegt an J1/16/17. Es kann allerdings auch ein externer Oszillator angeschlossen werden (am Eingang J12/14).
Über den Eingang F/-C wird dem Baustein mitgeteilt, ob der externe oder der interne Oszillator verwendet wird. Da hier der interne Oszillator verwendet wird, liegt dieser Eingang (J12/13) auf Masse.

7.4.1 Der Takt

Der Takt wird an einem synchronen Teiler durch drei geteilt und auf den Ausgang CLK (J12/8) gelegt. Diese Taktfrequenz von 9,54 MHz wird dem Prozessor zur Verfügung gestellt. Ein zweiter synchroner Teiler teilt die Taktfrequenz noch einmal durch zwei und legt diese auf den Ausgang PCLK (J12/2) (=Peripherial Clock). Die Taktfrequenz von 4,77 MHz wird einmal auf den NDR-Bus (ST2/30) gelegt und außerdem über zwei weitere 1:2 Teiler (J10) dem Timer zur Verfügung gestellt.
Die beiden Flip-Flops sind jeweils als 1:2 Teiler beschaltet; der Jumper JMP2 ist so eingestellt, daß der Taktausgang des zweiten Teilers (=1,19 MHz) auf die Takteingänge des Timers gelegt wird. Diesen Takt von 1,19 MHz benötigt der Timer zum Betreiben der Uhr.

7.4.2 Die Reset-Logik

Die RESET-Logik des Bausteines hat nur zwei Funktionen. Zum Einen hat der Eingang J12/11 einen Schmitt-Trigger Eingang, das heißt, daß an diesem Eingang ein analog ansteigendes Signal, wie es beim Einschalten oder beim Drücken der RESET-Taste S1 entsteht, an dem RC-Glied R1 und C2 anliegt. Zum Zweiten wird das RESET-Signal mit dem Takt (Prozessortakt) synchronisiert und invertiert, sodaß es am Ausgang J12/10 in der positiven Logik anliegt. Die Länge des RESET-Signales wird allerdings von dem schon erwähnten RC-Glied bestimmt. Die Zeit ist in etwa das Produkt aus R1 und C2 mal 0,8.

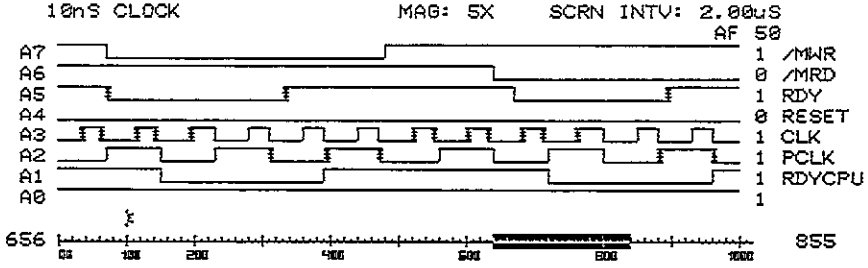
7.4.3 Das READY-Signal

Die dritte Funktion dieses Bausteines ist die Synchronisation des READY-Signales. Die Logik bietet doch einige Möglichkeiten. Der Baustein stellt zwei READY-Eingänge zur Verfügung (J11/4 und J11/6), die über die beiden Eingänge -AEN (J11/3 und J11/7) freigegeben werden können. Die beiden Eingänge werden Oder-verknüpft, das heißt, wenn einer der Eingänge LOW ist (das bedeutet nicht "ready"), wird dieses Signal auf den Ausgang durchgeschaltet.

7.4.4 Das WAIT-Signal

Nun sind aber auf diesem Baustein noch zwei nützliche Eigenschaften integriert: Zum Einen kann dieses READY-Signal über ein FLIP-FLOP mit dem Prozessortakt synchronisiert werden und auf den Ausgang J11/5 gelegt werden. Zum Zweiten kann aber dieses READY-Signal noch um einen Taktzyklus verzögert werden, es wird also ein WAIT-State eingefügt. Diese beiden Modi werden mit dem Eingang ASYNC (J11/15) gesteuert. In unserer Schaltung liegt ASYNC auf Masse, das heißt, daß ein WAIT-State eingefügt wird.

Timing Diagramm CLK, PCLK, RESET, READY



7.5 Der Bus-Controller

Der 8288, auf den weiter oben schon kurz eingegangen wurde, ist eigentlich nichts Anderes als ein Decoder mit einigen Besonderheiten Der Decoderteil dieses Bausteines erzeugt aus den Statussignalen -S0, -S1 und -S2 (J1/26/27/28) die Steuersignale -MRD, -MWR, -IORD, -IOWR und -INTA nach obenstehender Wahrheitstabelle (Tabelle 1). Zusätzlich hat der Baustein noch die Eingänge CLK, -AEN, CEN und IOB. Mit Hilfe des CLK-Signals werden die oben erwähnten Steuersignale und die sonstigen Ausgänge synchronisiert.
Über den Eingang -AEN (Adress Enable) werden die Steuersignale MRD, MWR, usw. freigegeben oder gesperrt (hochohmig). Dies ist bei DMA-Zugriffen nötig. Der Eingang IOB dient dazu, den Baustein zwischen den beiden Bus-Modi (I/O-Bus Mode und System-Bus Mode) umzuschalten. In unserer Schaltung wurde dieser Eingang (J13/1) auf Masse gelegt, also auf System-Bus Mode. Der Eingang CEN (Command Enable) (J13/15) dient ebenfalls zur Freigabe der Steuersignale und der beiden Signale DEN und -PDEN. Dieser Eingang liegt hier auf HIGH, das heißt, die Signale sind freigegeben.

7.6 Der Daten- und Adressenmultiplex

Außer den Steuersignalen gibt es noch vier weitere Ausgänge, die zur Steuerung des Daten- und Adreßmultiplex der Prozessor-Signale AD0 bis AD7 dienen. Der Ausgang DEN (Data Enable J13/16) zeigt an, daß auf den Signalleitungen AD0 bis AD7 gültige Daten anliegen, wenn dieser Ausgang auf HIGH liegt. Der Ausgang DT/-R (Data Transmit/ Receive) zeigt die Richtung des Datenflußes an. Ist dieses Signal LOW, werden Daten gelesen, ist das Signal HIGH, werden Daten geschrieben. Diese beiden Signale steuern den Datenbustreiber zum NDR-Bus (J14). Der Ausgang ALE (Adress Latch Enable) zeigt an, daß gültige Adressen auf den Adreßleitungen der CPU anliegen, auch daß auf den Signalleitungen AD0 bis AD7 im Moment Adressen anliegen. Dieses Signal wird dazu verwendet, die Adreßlatches mit den aktuellen Adressen zu laden (J15/11, J17/11 und J18/11). Der Ausgang MCE (Master Cascade Enable) oder -PDEN (Peripherial Data Enable) ist abhängig vom Eingang IOB. Da dieser Eingang auf LOW liegt, ist der Ausgang MCE aktiv. Dieser Ausgang dient zur Interruptabarbeitung, wird aber hier nicht benutzt.

7.7 Der Decodier-/Select-Logik

Der nächste Block, den wir näher durchleuchten wollen, ist die Dekodier- bzw. Selekt-Logik. Diese Logik ist in den beiden PALs J16 und J19 zusammengefaßt. Über J16 werden die I/O Bausteine dekodiert. Dazu werden an den Eingängen des Bausteines die Adressen A3 bis A9 (J16/1 bis J16/7) und die beiden Signale -IORD und -IOWR (J16/8 und J16/9) verwendet. Hier vielleicht noch kurz einige Erklärungen allgemein zum Thema I/O-Zugriff.

7.7.1 Die Select-Signale

Greift der Prozessor auf I/O zu, so aktiviert er die Adreßleitungen A0 bis A9 und das Signal -IORD oder -IOWR, je nachdem ob er einlesen oder ausgeben will. Auf den Adressen A0 bis A9 liegt die Adresse der I/O-Einheit, die sogenannte Portadresse. Die Dekodier-Logik hat nun die Aufgabe, aus dieser Portadresse und den Signalen -IORD und -IOWR ein Selekt-Signal für den I/O-Baustein zu erzeugen. An den Ausgängen dieser Logik liegen die Selekt-Signale für folgende Bausteine:

-SNMI	(Select Non maskable Interrupt)	J16/16
-SPIO	(Select Parallel Input Output)	J16/15
-STIM	(Select Timer)	J16/14
-SINT	(Select Interrupt-Controller)	J16/13
-SELIO	(Select Input Output)	J16/12

7.7.2 Die Portadressen

Bei der Selektierung der Standard I/O, wie z B. Timer, Interruptcontroller und PIO, haben wir uns naturlich an die Standardadressen, die vom IBM-PC vorgegeben sind, gehalten In der untenstehenden Tabelle sehen Sie die Portadressen der einzelnen Bausteine-

Baustein	*Portadressen*
PIO	60h...7Fh
Timer	40h...5Fh
Interrupt-Controller	20h...3Fh
NMI	A0h ..BFh
SELIO	00h .FFh

Der Prozessor 8088 kann im Gegensatz zum Z80 nicht nur 256 I/O-Ports adressieren, sondern 1024. Deshalb werden hier auch die Adressen A8 und A9 zusätzlich verwendet. Außerdem tritt hier das Problem der Portdoppelbelegung auf, z. B. GDP64k und PIO. Diese Doppelbelegungen werden durch das sogenannte -SELIO-Signal verhindert. Wird auf die internen I/Os zugegriffen, muß auf die Adressen 00h bis FFh zugegriffen werden. Soll auf externe I/Os zugegriffen werden, müssen die Portadressen 100h bis 3FFh verwendet werden. Probleme treten dann auf, wenn später mit der Buskopplung PC-Baugruppen verwendet werden, die sich in der Portbelegung mit den NDR-Baugruppen überschneiden. Bei den Standardbaugruppen.

Hercules mit Bildschirm,
Seriell-Adapter,
OMTI Controller und
Farbgraphikadapter

treten keine Probleme auf. Schwierigkeiten kann es lediglich mit der parallelen Druckerschnittstelle geben. Hier muß dann gewährleistet sein, daß keine GDP im System steckt, wenn eine Herculeskarte eingesetzt wird.

7.7.3 Die Decodierung der Speicherbereiche

Die Dekodierung der Speicherbereiche erfolgt durch den PAL J19. Dazu werden die Signale A14 bis A19, sowie die Steuersignale -MRD und -MWR benötigt. Analog zum I/O Zugriff wird beim Speicherzugriff vom Prozessor ebenfalls eine Adresse und eines der beiden Steuersignale angelegt. Die Select-Signale für die Speicher sind:

-SELROM	(Select ROM)	J19/13,
-SELRAM	(Select RAM)	J19/12,
-SELMEM	(Select Memory)	J19/14.

Das Select Signal -SELROM selektiert das BIOS EPROM. Das -SELRAM Signal wählt den internen RAM (J31) aus. -SELMEM dient zur Auswahl des externen Speichers (nicht auf der CPU8088, z.B. ROA64, ROA256 oder RAM64/256).

Adressen der verschiedenen Speicher

Speicher	*Adressen*		*Speicherplatz*
BIOS ROM	F0000h ... FFFFFh		64K
internes RAM	B0000h ... B3FFFh,	BC000h ...BFFFFh	32K
externes RAM	00000h B0000h,	B4000h ... BBFFFh,	
	C0000h ... EFFFFh		950K

Außerdem übernimmt dieser Baustein noch die Invertierung der vom NDR-Bus kommenden Interruptsignale -INT, -NMI und Reserve (ST2/32/47/54).

7.8 Der Interrupt-Controller

Der Interrupt Controller 8259 besitzt 8 Interrupteingänge, auf denen die Interrupt-Anforderungen der einzelnen Blöcke (Tastatur, Paralleldrucker, zwei Asynchron-Adapter und der Zeitgeber) eintreffen.

7.8.1 Der Ablauf eines Interrupts

Kurz noch einige Worte zum generellen Ablauf des Interrupt mit dem Controller 8259. Löst einer der 5 angeschlossenen Blöcke einen Interrupt aus, wird der INT-Ausgang des Controllers J8/17 aktiviert. Die CPU erkennt den Interrupt und quittiert diesen mit den Statussignalen -S0, -S1 und -S2 (siehe CPU8088). Der Bus Controller 8288 generiert aus dieser Statusmeldung das -INTA-Signal (Interrupt Acknowledge), das dem 8259 (J8/26) mitteilt, daß der Interupt erkannt wurde. Der 8259 erniedrigt das Priority Byte (Einfrieren des momentanen Status) und wartet, bis die CPU das zweite -INTA Signal generiert. Jetzt erkennt der 8259, daß die CPU bereit ist, den Interruptvektor entgegen zu nehmen. Der 8259 legt jetzt den Vektor auf den Datenbus und die CPU 8088 liest diesen ein und verzweigt in die entsprechende Interruptroutine

Belegung des Interruptcontrollers

Eingang	*Priorität*	*Wer löst aus*
IR0 (J8/18)	0	Systemuhr
IR1 (J8/19)	1	Tastatur (Zeichen steht an)
IR2 (J8/20)	2	nicht belegt
IR3 (J8/21)	3	Asynchron Adapter 1
IR4 (J8/22)	4	Asynchron Adapter 2
IR5 (J8/23)	5	nicht belegt
IR6 (J8/24)	6	nicht belegt
IR7 (J8/25)	7	Centronics Schnittstelle

Dabei ist 0 die höchste und 7 die niedrigste Priorität.

Treten nun aber zwei Interrupts gleichzeitig am 8259 auf, so muß der Controller erkennen, welchen er zuerst abarbeiten muß. Dazu kann dem 8259 eine Prioritätsliste übergeben werden, welcher Interrupt die höchste, die zweithöchste, usw. Priorität hat.

7.8.2 Die Beschaltung

Auf die Beschaltung des 8259 soll noch kurz eingegangen werden. Die Eingänge IR0 bis IR7 sind die Interruptrequest-Eingänge von den Peripherieblöcken. Die Datenleitungen D0 bis D7, sowie die Signale -IORD, -IOWR und A0 sind Signale, mit deren Hilfe mit dem Bus kommuniziert werden kann. Mit -IORD und -IOWR wird festgelegt, ob in eines der 8259 Register geschrieben wird oder ob von ihnen gelesen wird. In welches Register geschrieben bzw. gelesen wird, hängt von A0 ab.

Wahrheitstabelle Registerauswahl

-IORD	*-IOWR*	*A0*	*Register*
0	1	0	Statusregister
0	1	1	Statusregister
1	0	0	Commandregister
1	0	1	Commandregister

Das Signal SP/-EN dient zur Steuerung des Datenbustreibers während des Anlegens des Interruptvektors. Hier wird dieses Signal dazu verwendet den Datenbustreiber (J14) zu sperren, um eine Datenkollision zu vermeiden. Die Eingänge CAS0 bis CAS2 dienen zum Kaskadieren von Interrupt Controllern. Insgesamt können bis zu 8 Bausteine angeschlossen werden. Diese drei Pins sind hier offen, da nur ein 8259 verwendet wird.

7.9 Der Timer

Der Timer 8253 beinhaltet drei Timer, die unabhängig voneinander gesteuert werden können. Der Ablauf der Timerfunktion hängt vom eingestellten Modus ab. Es gibt 6 verschiedene Modi, den Timerablauf festzulegen. Die einzelnen Modi sind im Datenblatt nachzulesen, ebenfalls die Programmierung des 8253.

Die Busschnittstelle beinhaltet die Datenleitungen D0 bis D7, die Signale -RD und -WR (J3/22/23) und die Adreßleitungen A0 und A1 (J3/19/20). Die 8 Register werden wie folgt adressiert:

A0	*A1*	*-RD*	*-WR*	*Register*
0	0	1	0	Zähler 0 laden
1	0	1	0	Zähler 1 laden
0	1	1	0	Zähler 2 laden
1	1	1	0	"Mode Wort" schreiben
0	0	0	1	Zähler 0 lesen
1	0	0	1	Zähler 1 lesen
0	1	0	1	Zähler 2 lesen
1	1	0	1	nicht belegt

Die Eingänge GATE0, GATE1 und GATE2 sind Triggereingänge für die drei Timer. Die Eingange CLK0, CLK1 und CLK2 sind die Takteingänge der 3 Zähler. Dieser Clock bestimmt die Grundzykluszeit der Zähler. Die Outputs OUT0, OUT1 und OUT2 sind die Ausgange, die je nach eingestelltem Timermodus ein Strobe-Signal oder ein definiert langes Signal ausgeben.

In unserer Schaltung sind nur die beiden Timer 0 und 2 benutzt. Timer 0 dient dazu, die Softwareuhr des Rechners zu triggern. Dazu wird im Timer der Modus 2 verwendet. In diesem Modus wird der Timer mit einem bestimmten Wert (=65535) geladen und damit auch gestartet. Ist der Timer abgelaufen, wird ein Strobe-Signal ausgegeben, das auf den Interrupt-Controller (J8/18 Zeitgeber) gelegt wird. Mit der gewählten Taktfrequenz von 1,19 MHz erfolgen rund 18 Interruptanforderungen pro Sekunde (Die 'Triggerfrequenz' beträgt 18Hz). Die Interruptroutine macht dann nichts Anderes als nach jedem 18ten Aufruf den Uhrenstand um 1 Sekunde zu erhöhen.

Der zweite Timer dient zur Erzeugung der Lautsprecherfrequenz. Damit können durch Laden verschiedener Zählerwerte verschiedene Frequenzen ausgewählt werden und damit das ganze Tonspektrum abgedeckt werden.

Timer 1 ist unbelegt und kann nach Bedarf frei verwendet werden

Belegung des Timer 8253 (J3)

Timer	*Belegung*
Timer 0	Systemuhr
Timer 1	nicht verwendet
Timer 2	Lautsprechertakt

7.10 Die WAIT-Logik

Die WAIT-Logik des Rechners wurde beim Baustein 8284 schon kurz gestreift. Der zweite Teil dieser WAIT-Logik steckt in dem PAL J20. Die PAL-Logik ist so aufgebaut, daß bei I/O Zugriffen grundsätzlich zwei WAIT-Zyklen und bei externen Speicherzugriffen ein WAIT-Zyklus eingefügt wird. Zum Generieren der I/O-WAITs werden nur die Signale --IORD und -IOWR verwendet, zum Generieren des WAITs für externe Speicher werden die Signale -MRD, -MWR und -SELMEM verwendet. Außerdem wird natürlich ein WAIT, der vom NDR-Bus kommt, ebenfalls weitergeleitet. Das ebenfalls an den PAL gelegte -SELIO-Signal wird nicht verwendet. Der angelegte Takt (J20/1) steuert die internen Register des PALs, das heißt, die Periodendauer des Taktes entspricht einem WAIT-Zyklus.

7.11 Die Busanpassung

Ein weiterer Block ist die Busanpassung zum NDR-Computer (PAL 21). Die auf der Baugruppe erzeugten Steuersignale sind auf den Prozessor und auf den PC bezogen. Diese Steuersignale unterscheiden sich nur unwesentlich vom NDR-Bus, der auf den Z80-Prozessor zugeschnitten ist. Dies sind im wesentlichen die Signale:

-MRD
-MWR
-IORD
-IOWR
-RESET

Die vier Steuersignale -MRD, -MWR, -IORD, und -IOWR werden so verknüpft, daß die NDR-Signale -MREQ, -IORQ, -RD und -WR erzeugt werden. Außerdem werden diese Signale noch mit den Eingängen -SELIO und -SELMEM verknüpt, d.h. -RD und -WR sind nur aktiv, wenn -SELMEM oder -SELIO aktiv sind. -IORQ wird nur aktiv, wenn -SELIO aktiv ist. Das RESET Signal wird lediglich invertiert und auf den NDR Bus gelegt.

Außerdem erzeugt PAL J21 noch das Select-Signal für den Datenbustreiber J14. Der Datenbustreiber wird demnach nur aktiv, wenn das Signal DEN (Data Enable) aktiv ist und auf eine externe I/O oder einen externen Speicher zugreift und kein Interruptacknowledge-Zyklus abläuft. Außerdem wird noch das Banken-Signal über PAL J21 erzeugt. Das Banken-Signal wird bei jedem Speicherzugriff aktiviert.

7.12 Die Adreßpuffer

Der Vollstandigkeit halber gehen wir hier noch kurz einmal auf die Adreßpuffer ein. Dies sind die drei Bausteine J15, J17 und J19. Sie haben im wesentlichen zwei Aufgaben
Zum Einen hat J15 die Aufgabe, die gemultiplexten Daten-Adreß-Leitungen AD0 bis AD7 so aufzusplitten, daß am Ausgang des Puffers nur die Adressen anliegen und zwischengespeichert werden, bis neue Adressen ausgegeben werden.
Die zweite Aufgabe dieser Puffer ist das "Treiben" (verstärken) der Adreßsignale. Gesteuert werden diese Adreß-Latches mit dem ALE-Signal (Adress Latch Enable). Der Eingang -OC (Output Control) gibt die Ausgange der Latches frei. Hier liegt dieser Eingang auf Masse, d.h die Treiber sind ständig frei

7.13 Der Keyboard-Controller

Damit sind wir fast am Ende der eigentlichen CPU-Karte angelangt. Es fehlt nur noch der große Block des KEYBOARD-Controllers. Dieser Block umfaßt den parallelen I/O Baustein 8255 (J4), das Schieberegister 74LS322 (J5), das D-Flip-Flop 74LS74 (J9) und ein PAL (J11). Außerdem ubernimmt der Keyboard Controller noch die Aufgabe, den Lautsprecher zu steuern. Ebenfalls in diesen Block integriert ist noch die NMI-Steuerung (Non Maskable Interrupt) für den 8087.

7.13.1 Die parallele Ein/Ausgabe

Der 8255 ist ein paralleler Schnittstellenbaustein mit drei 8 Bit breiten parallelen Schnittstellen. Diese drei Schnittstellen PA0 bis PA7, PB0 bis PB7 und PC0 bis PC7 sind sowohl als Eingänge als auch als Ausgänge definierbar. Diese Option kann im Steuerwortregister festgelegt werden.

Damit waren wir schon bei der Busschnittstelle und den Registern des 8255. Die Busschnittstelle ist bis auf das zusätzliche RESET-Signal exakt gleich wie die des Timers 8253
Die Register sind wie folgt zugänglich:

A0	A1	-IORD	-IOWR	*Register*
0	0	0	1	Port A lesen
1	0	0	1	Port B lesen
0	1	0	1	Port C lesen
0	0	1	0	Port A ausgeben
1	0	1	0	Port B ausgeben
0	1	1	0	Port C ausgeben
1	1	1	0	Steuerwortregister

Hier wird Port PA0 bis PA7 für die parallelen Tastaturdaten verwendet. Port PB wird für den Lautsprecher und für die Tastatur benötigt. Bit PB0 steuert den GATE Eingang des Timers (Timer 2, J3/16). Über PB1 ist es möglich, parallel zum Timer2 Lautsprecherdaten auszugeben. Auf PB6 wird der Keyboard-Clock ausgegeben, wenn der Rechner etwas an die Tastatur übermitteln will. Dieser Betrieb wird aber bei PCs nicht durchgeführt. Jedoch kann die PC-Tastatur über einen 20ms langen LOW-Impuls zurückgesetzt werden. Über PB7 wird das Schieberegister J5 gelöscht. Auf Port PC wird nur 1 Bit gelesen, dies ist der Lautsprechertakt auf PC5 (J4/12).

Belegung der parallelen Ein/Ausgänge des 8255

Ein/Ausgang	*Belegung*
PA0 bis PA7	Tastaturdaten (System Scan Code), wenn PB7 = LOW oder DIL-Schalter, wenn PB7 = HIGH (hier nicht vorhanden)
PB0 (J4/18)	Lautsprecher Freigabe: LOW = gesperrt HIGH = frei
PB1 (J4/19)	Timer 2: LOW = Takt aus, HIGH = Takt ein Takt (Frequenz) für Lautsprecher
PB2 bis PB5	nicht verwendet
PB6 (J4/24)	Tastaturdatentakt bei Transfer Rechner --> Tastatur HIGH = Taktltg. hochohmig LOW = Taktleitung LOW.
PB7 (J4/25)	Auswahl Tastatur oder DIL-Schalter (siehe PA0 bis PA7)
PC0 bis PC4	nicht verwendet
PC5 (J4/12)	Timer Kanal 2 (Einlesen des Lautsprechertaktes)
PC6 (J4/11)	GND*
PC7 (J4/11)	GND*

*: Die Eingänge PC6 und PC7 zeigen bei PCs im Normalfall einen I/O und Memory Parity Fehler an:

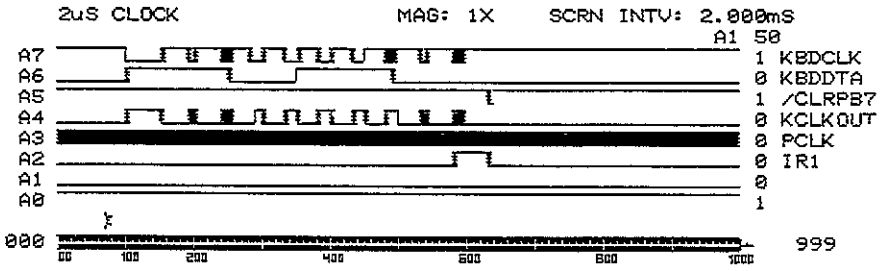
LOW = Kein Parity Fehler,
HIGH = Parity Fehler.

Da hier aber keine Paritätsprufung gemacht wird, werden diese beiden Eingange fest auf Masse gelegt.
Der in der Tabelle angesprochene DIL-Schalter ist bei PCs manchmal vorhanden und wird vom BIOS auch unterstutzt. Physikalisch ist dieser aber auf unserer Baugruppe nicht vorhanden

7.13.2 Das Einlesen eines Zeichens

Das Einlesen eines Zeichens von der Tastatur geht folgendermaßen vor sich:
Beim Einschalten des Rechners wird das Schieberegister J5 gelöscht. Dies geschieht uber PB7 (J4/25), das über das PAL (J11/2/12) invertiert wird und an das Schieberegister (J5/9) gelegt wird. Dieses CLEAR-Signal wird ebenfalls an das Flip-Flop J9/1 gelegt, welches damit rückgesetzt wird. Das bedeutet, daß der Tastaturinterrupt (J9/7) deaktiviert ist. Damit ist die CPU8088 bereit, Tastaturdaten einzulesen. Die Tastaturdaten, die seriell von der Tastatur ankommen, gelangen an das Schieberegister 74LS322 (J5/17). Sie werden dann nacheinander mit dem Keyboardtakt in dem Schieberegister aufgenommen. Der Tastaturtakt, der ebenfalls von der Tastatur kommt, wird an PAL4 (J11) um einen internen Taktzyklus (PCLK2=2.38MHz) verzögert, was aus Timinggründen nötig ist. Dieser Takt (J11/16) wird als Schiebetakt für das Schieberegister verwendet. Der Eingang -OE (Output Enable) wird über das Bit PB7 (J4/25) gesteuert. Ist dieser Ausgang LOW, so wird das Schieberegister seriell geladen oder parallel ausgegeben. Ist das Tastaturbyte vollständig eingelesen, wird das Startbit aus dem Ausgang QH' (J5/12) hinausgeschoben und beim nächsten Taktzyklus übernimmt das Flip-Flop J9 das HIGH-Signal (Start-Bit ist immer HIGH) und setzt den Ausgang J9/5 auf HIGH. Beim Übergang von LOW auf HIGH wird ein Interrupt angefordert (J8/19). Dieser HIGH-Pegel am Interrupteingang (J8/19) deaktiviert auch den Select-Eingang des Schieberegisters (J5/1) und stoppt somit den Schiebevorgang (bei weiteren CLK Zyklen). In der Interruptroutine wird das Tastaturbyte dann über PA0 bis PA7 des parallelen Schnittstellenbausteines eingelesen. Anschließend wird das Schieberegister und das Interruptflipflop wieder gelöscht.

Timingdiagramm mit KBDCLK, KBDDTA...hier einsetzen



Außerdem kann das Schieberegister-Überlaufsignal QH' (J5/12), über das Flip-Flop J9/6 invertiert, wieder auf die Datenleitung KBDDTA der Tastatur gelegt werden. Dazu ist es allerdings notwendig, daß diese Leitung seitens der Tastatur hochohmig geschaltet wird. Das zurückzuführende Signal wird durch PAL J11/4 (-INTD) geschleust.
Der Eingang -INTD J11/4 wirkt in zweierlei Hinsicht auf den PAL-Ausgang KBDDTA (J11/18): Ist -INTD auf HIGH (immer dann, wenn serielle Keyboarddaten eingelesen werden), ist KBDDTA hochohmig geschaltet. Wenn -INTD auf LOW geht (das Startbit wird soeben aus dem Schieberegister J5/12 herausgeschoben), wird der PAL-Ausgang KBDDTA aktiv und auf LOW gezogen. Dieses Verhalten entspricht einem OPEN-KOLLEKTOR-Ausgang. Siehe dazu das Listing der Gleichungen für PAL4 (unter Kap. 10)

PAL4 ist auch in der Lage, die Leitung KBDCLK auf Masse zu ziehen. Gesteuert wird dies über einen LOW-Impuls von PB6 (J4/24). Auch dabei wird wieder durch die Programmierung der PAL-Logik ein OPEN-Kollektor-Verhalten erreicht.

Darüber hinaus führt das PAL noch eine NAND-Verknüpfung zwischen TC2 (J11/7) und SPKDTA (J11/7) und stellt am Ausgang das SPEAKER Signal zur Verfügung, mit dem der Lautsprecher angesteuert wird (J11/17). Dieses Signal wird anschließend über den Transistor T1 verstärkt und kann direkt einen 4 bis 8 Ohm Lautsprecher ansteuern. Die Leistung, die dieser Treiber an den Lautsprecher abgibt, sind ca. 62...125 mW (4 oder 8 Ohm).

Letztlich beinhaltet PAL J11 die erforderliche Logik für die Freigabe des NMI (Non maskable Interrupt), der vom 8087 verwendet wird. Der Interrupt kann über Port 0A0h Bit 7 auf HIGH freigegeben werden (siehe Flipflop J9). Ist der NMI freigegeben, wird noch geprüft, ob JMP3 gesteckt ist oder nicht. Dieser Jumper wird gesteckt, wenn der Coprozessor in der Karte steckt. Kommt jetzt ein Interrupt vom Coprozessor 8087, so wird der NMI Ausgang des PAL aktiviert und an den 8088 weitergegeben (siehe PAL4 J11 unter Kapitel 10).

8. Anwendungen

8.1 Anwendung als Systembaugruppe im NDR-PC

Die häufigste Anwendung dieser Baugruppe durfte wohl mit dem Betriebssystem MS-DOS sein. Dazu finden Sie unter Kapitel 1.3 die entsprechenden Systemkonfigurationen und unter Kapitel 5 den Test und die Inbetriebnahme.

8.2 Anwendung als Single Board Computer (SBC)

Auf der Baugruppe befinden sich alle wesentlichen Teile eines Computers, die ein Rechner zum Betrieb benötigt. Die CPU8088 ist daher auch für Anwendungen interessant, bei welchen nur Minimalkonfigurationen verwendet werden: Bei Steuerungen.

Ein Beispiel für eine Steuerung sei hier kurz genannt: Mit dem SPS-Compiler können SPS-Programme komfortabel auf dem PC erstellt, simuliert und anschließend in ein EPROM gebrannt werden. Das EPROM wird dann auf die Baugruppe CPU8088 gesteckt. Eine eventuell benötigte Ein/Ausgabebaugruppe wird noch in den NDR-Bus gesteckt und schon kann die Steuerung in Betrieb genommen werden (z.B. Heizungssteuerung oder Regelung, Aufzugsteuerung, usw).

9. Kritik, Verbesserungen

9.1 Verbesserungsvorschläge

Keine Baugruppe ist, wenn sie auf den Markt kommt, perfekt und hundertprozentig ausgereift. So wird es auch bei der CPU8088 noch Verbesserungen geben. Sollten Sie irgendwelche Ungereimtheiten in der Hardware oder Software (BIOS) feststellen, möchten wir Sie auf diesem Wege bitten, uns dies schriftlich und detailliert mitzuteilen. Wir werden uns dann bemühen, solche Vorschläge, wenn sie sinnvoll sind, bei einer Revisionsänderung mit aufzunehmen. Wir müssen Sie aber an dieser Stelle auch um Verständnis bitten, daß nicht sämtliche Anwenderwünsche, aufgrund wirtschaftlicher Überlegungen, berücksichtigt werden können.

9.2 Updates zum BIOS

Die Hauptänderungen im Bereich der CPU8088 wird es wohl beim BIOS geben. Hier werden natürlich immer neue Baugruppen angepaßt oder bereits bestehende Treiber für Baugruppen verbessert. Deshalb werden wir bei jeder Änderung des BIOS ein preiswertes Update anbieten. Dadurch haben Sie die Möglichkeit, relativ billig immer auf dem neuesten Stand der Technik zu sein. Den Preis für das Update erfahren Sie aus einer aktuellen Preisliste.

9.3 Kritik

Dem Bausatz bzw. Fertiggerät lag eine Kritikkarte bei. Bitte senden Sie uns die ausgefüllte Kritikkarte an uns zurück. Sie helfen uns damit unsere Produkte noch besser zu machen und unseren Service noch besser zu gestalten. Wir wollen mit dieser Karte nicht auf Adressenfang gehen, sondern wirklich nur prüfen wie unsere Produkte bei unseren Kunden ankommen.

Für Fehlermeldungen und Verbesserungen, die dieses Handbuch betreffen, sind wir immer dankbar.

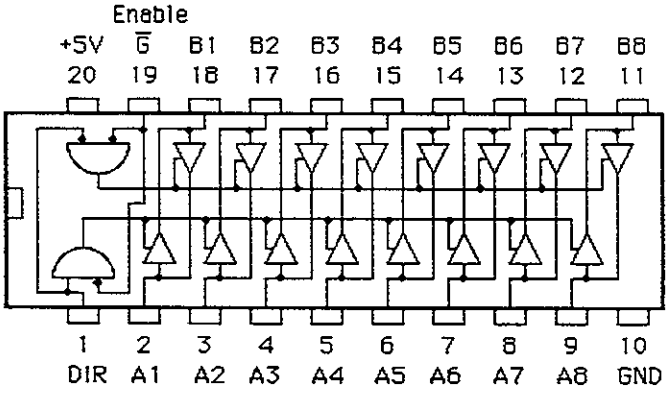
10. Bauelemente

10.1 TTL Baustein

10.1.1 74 LS 245

74LS245

8-fach Bus-Transceiver mit 3-state Ausgängen



Function Table

ENABLE $\overline{G}$	DIRECTION CONTROL DIR	OPERATION
L	L	B data to A bus
L	H	A data to B bus
H	x	Isolation

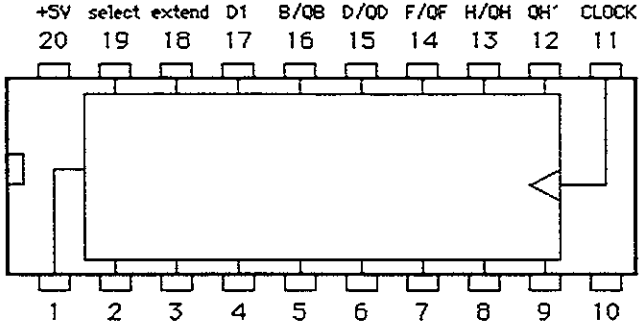
Typ Impuls-Verzogerungszeit 20 ns

Typ Versorgungsstrom 75 mA

10.1.2 74 LS 322

74LS322

8-Bit Schieberegister mit sign extend



Wahrheitstabelle

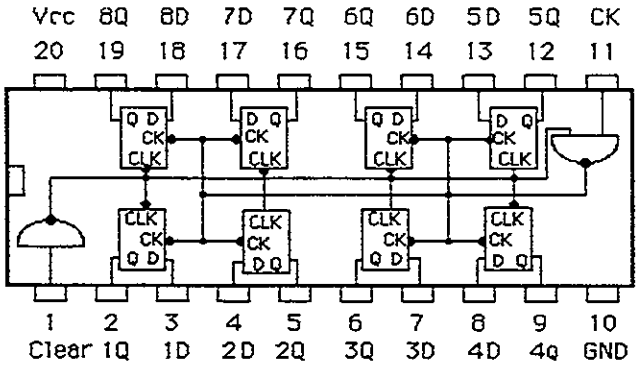
	INPUT							INPUT/OUTPUT				OUTPUT
	9	1	2	18	19	8	11	4	16	5	13	QH'
CLEAR	L	H	X	X	X	L	X	L	L	L	L	L
	L	X	H	X	X	L	X	L	L	L	L	L
HOLD	H	H	X	X	X	L	X	QA0	QB0	QC0	QH0	QH0
SHIFT	H	L	H	H	L	L	↑	D0	QAn	QBn	QGn	QGn
REG	H	L	H	H	H	L	↑	D1	QAn	QBn	QGn	QGn
SIGN EX	H	L	H	L	X	L	↑	QAn	QAn	QBn	QGn	QGn
LOAD	H	L	L	X	X	X	↑	a	b	c	h	h

Max garantierte Schiebefrequenz 35 MHz
Typ Leistungsaufnahme 175 mW
Typ Verzögerungszeit 15 ns

10.1.3 74 LS 373

74 LS 273

8 - Bit D Register mit Clear



Function Table

Inputs			Output
Clear	Clock	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

Typ Impuls-Verzogerungszeit 17,5 ns

Typ Leistungs-aufnahme 85 mW

10.2 PALs

10.2.1 PAL1

```
Name       gspc1;
Partno     U11;
Date       06/30/88;
Revision   01;
Designer   rdk;
Company    rdk;
Assembly   ndrpc;
Location   U11;

/*****************************************************************/
/* Address select generation  memory part                        */
/* NDR PC                                                        */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types: 16L8                          */
/*****************************************************************/

/**  Inputs  **/

Pin 1        = AI19      ;      /* Latched Adresses              */
Pin 2        = AI18      ;      /*                               */
Pin 3        = AI17      ;      /*                               */
Pin 4        = AI16      ;      /*                               */
Pin 5        = AI15      ;      /*                               */
Pin 6        = NMRD      ;      /* read strobe                   */
Pin 7        = NMWR      ;      /* write strobe                  */
Pin 8        = NII1      ;      /* Interrupts from NDR BUS       */
Pin 9        = NII2      ;      /*                               */
Pin 11       = NII3      ;      /*                               */
PIn 15       = AI14      ;      /* additional Address 14         */

/**  Outputs  **/

Pin 12       = NSELRAM   ;      /* select RAM address range      */
Pin 13       = NSELROM   ;      /* select ROM address range      */
Pin 14       = NSELMEM   ;      /* internal memory accessed      */
Pin 16       = O1        ;      /* output ints for 8259          */
Pin 17       = O2        ;      /*                               */
Pin 18       = O3        ;      /*                               */

Pin 10 = GND;
Pin 20 = VCC;

/** Declarations and Intermediate Variable Definitions **/

/**  Logic Equations  **/
/* special range for memory: B0000-B3FFF, BC000-BFFFF (wwordstar etc) */
/* gives a total of 32K */
!NSELRAM = AI19 & !AI18 & AI17 & AI16 & (!AI15 & !AI14 # AI15 & AI14)
         & (!NMRD # !NMWR);
!NSELROM = AI19 & AI18 & AI17 & AI16 & !NMRD; /* F000:0000 ... F000:FFFF wichtig 64k */
!NSELMEM = AI19 & !AI18 & AI17 & AI16 & (!AI15 & !AI14 # AI15 & AI14)   /* RAM */
           # AI19 & AI18 & AI17 & AI16 ; /* internal EPROM , no strobes here */
!O1 = NII1; /* just invert signals */
!O2 = NII2;
!O3 = NII3;
```

10.2.2 PAL2

```
Name     gspc2,
Partno   U13;
Date     06/30/88,
Revision 01;
Designer rdk;
Company  rdk;
Assembly ndrpc;
Location U13,

/*****************************************************************/
/* Address select generation  io part                            */
/* NDR PC                                                        */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types: 16L8                          */
/*****************************************************************/

/**  Inputs  **/

Pin 1      = AI9      ;      /* Latched Adresses                 */
Pin 2      = AI8      ;      /*                                  */
Pin 3      = AI7      ,      /*                                  */
Pin 4      = AI6      ;      /*                                  */
Pin 5      = AI5      ;      /*                                  */
Pin 6      = AI4      ;      /*                                  */
Pin 7      = AI3      ;      /*                                  */
Pin 8      = NIORD    ;      /* read io                          */
Pin 9      = NIOWR    ;      /* write io                         */

/**  Outputs  **/

Pin 12     = NSELIO   ;      /* select IO internal  0..FF        */
Pin 13     = NSINT    ;      /* 8259                             */
Pin 14     = NSTIM    ,      /* 8253                             */
Pin 15     = NSPIO    ;      /* 8255                             */
Pin 16     = NSNMI    ;      /*                                  */

Pin 10 = GND,
Pin 20 = VCC;

/** Declarations and Intermediate Variable Definitions **/

/**  Logic Equations  **/
!NSELIO = !AI9 & !AI8; /* select whole range 0 .FF to be internal no strobe */
!NSINT = !AI9 & !AI8 & !AI7 & !AI6 & AI5 & (!NIORD # !NIOWR); /* range 20h  3fh */
!NSTIM = !AI9 & !AI8 & !AI7 & AI6 & !AI5 & (!NIORD # !NIOWR); /* range 40h  5fh */
!NSPIO = !AI9 & !AI8 & !AI7 & AI6 & AI5 & (!NIORD # !NIOWR);  /* range 60h .7fh */
!NSNMI = !AI8 & !AI8 & AI7 & !AI6 & AI5 & !NIOWR;             /* range a0h  bfh */
```

10.2.3 PAL3

```
Name     gspc3;
Partno   U14;
Date     12/10/88;
Revision 03;
Designer rdk;
Company  rdk;
Assembly ndrpc;
Location U14;

/*****************************************************************/
/* NDR BUS INTERFACE                                             */
/* NDR PC   rev fuer externes BIOS, CGA etc.                     */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types: 16L8                          */
/*****************************************************************/

/**  Inputs  **/

Pin 1      = NMRD      ;      /* memory read                     */
Pin 2      = NMWR      ;      /* memory write                    */
Pin 3      = NIORD     ;      /* io read                         */
Pin 4      = NIOWR     ;      /* io write                        */
Pin 5      = NSELMEM   ;      /* adr internal memory             */
Pin 6      = NSELIO    ;      /* adr internal io                 */
Pin 7      = DEN       ;      /* buffer enable strobe            */
Pin 8      = NSPNEN    ;      /* int controller strobe           */
Pin 9      = RES       ;      /* reset signal                    */

/**  Outputs  **/

Pin 12     = NRD       ;      /* NDR BUS read                    */
Pin 13     = NWR       ;      /* NDR BUS write                   */
Pin 14     = NIORQ     ;      /* NDR BUS io request              */
Pin 15     = NMREQ     ;      /* NDR BUS memory request          */
Pin 16     = NRESET    ;      /* negative reset                  */
Pin 17     = NSEL      ;      /* buffer enable signal            */
Pin 19     = BANKEN    ;      /* Enable on external memory adr   */

Pin 10 = GND;
Pin 20 = VCC;

/** Declarations and Intermediate Variable Definitions **/

/**  Logic Equations  **/

!NRD = !NIORD & NSELIO # !NMRD & NSELMEM;
!NWR = !NIOWR & NSELIO # !NMWR ; /* rev 3.0 wr nicht mehr mit SELMEM verkn. */
!NIORQ = !NIORD & NSELIO # !NIOWR & NSELIO;
!NMREQ = !NMRD # !NMWR; /* used for refresh in NDR BUS no masking */
!NRESET = RES;
!NSEL = DEN & NSPNEN & (NSELIO # (NIORD & NIOWR)) & (NSELMEM # NMRD); /* rev 3.0, no wr */
!BANKEN = NMRD & NMWR; /* always low if not external memory selected (due to dyn256k) */
```

10.2.4 PAL4

```
Name        gspc5;
Partno      U18;
Date        07/07/88;
Revision    01;
Designer    rdk;
Company     rdk;
Assembly    ndrpc;
Location    U18;

/*************************************************************************/
/* Wait state generation                                                 */
/* NDR PC                                                                */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: 16R4                                  */
/*************************************************************************/

/**  Inputs  **/

Pin 1       = PCLK2     ;       /*   clock                              */
Pin 2       = NMRD      ;       /* read memory                          */
Pin 3       = NMWR      ;       /* write memory                         */
Pin 4       = NIORD     ;       /* read io                              */
Pin 5       = NIOWR     ;       /* write io                             */
Pin 6       = NSELMEM   ;       /* select memory                        */
Pin 7       = NSELIO    ;       /* select io                            */
Pin 8       = NWAIT     ;       /* wait from ndr bus                    */
Pin 11      = GND2;

/**  Outputs  **/

Pin 14      = Q1        ;       /* latch intern                         */
Pin 15      = Q2        ;       /* latch intern                         */
Pin 16      = Q3        ;       /* latch intern                         */
Pin 17      = Q4        ;       /* latch intern                         */
Pin 19      = RDY       ;       /* read output to 8284                  */

Pin 10 = GND;
Pin 20 = VCC;

/** Declarations and Intermediate Variable Definitions **/

/** Logic Equations  **/

!Q1.d = !NMRD # !NMWR # !NIORD # !NIOWR;
!Q2 d = !Q1 & (!NMRD # !NMWR # !NIORD # !NIOWR);  /* go to high immed. 1clk */

!RDY = !NWAIT # Q2 & (!NIORD # !NIOWR)            /* IO is always slower */
              # Q1 & NSELMEM & (!NMRD # !NMWR);    /* for external ROM/RAM */

/* ready only after delay and wait high,no access then ready high */
```

10.2.5 PAL5

```
Name        gspc4;
Partno      U17;
Date        06/30/88;
Revision    01;
Designer    rdk;
Company     rdk;
Assembly    ndrpc;
Location    U17;

/*****************************************************************/
/* Keyboard select logic                                         */
/* NDR PC                                                        */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types: 16V8                          */
/*****************************************************************/

/**  Inputs  **/

Pin 1       = PCLK2     ;     /*   clock                        */
Pin 2       = PB7       ;     /* from 8255                      */
Pin 3       = PB6       ;     /*                                */
Pin 4       = NINTD     ;     /* from Latch                     */
Pin 5       = INT       ;     /* from FPU                       */
Pin 6       = SPKDTA    ;     /* from 8255                      */
Pin 7       = TC2       ;     /* from 8253                      */
Pin 8       = JUMPER    ;     /* enable FPU                     */
Pin 9       = ENNMI     ;     /* latch for enable NMI           */
Pin 11      = GND2;

/**  Outputs  **/

Pin 12      = NCLRPB7   ;     /* internal                       */
Pin 13      = KBDCLK    ;     /* keyboard clock                 */
Pin 14      = nc        ;     /*                                */
Pin 15      = QINTER    ;     /* latch intern                   */
Pin 16      = KCLKOUT   ;     /* keyboard clock out             */
Pin 17      = SPEAKER   ;     /* load speaker connection        */
Pin 18      = KBDDTA    ;     /* output to keyboard data        */
Pin 19      = NMI       ;     /* output to CPU NMI              */

Pin 10 = GND;
Pin 20 = VCC;

/** Declarations and Intermediate Variable Definitions **/

/**  Logic Equations  **/

!NMI = !(!JUMPER & INT & ENNMI); /* NMI only if all enabled */
!NCLRPB7 = PB7; /* just invert */
!KBDCLK = !PB6; /* open collector */
KBDCLK.oe = !PB6;
!QINTER.d = !KBDCLK; /* read external clock also */
!KCLKOUT.d = QINTER;
!KBDDTA = !NINTD; /* open collector */
KBDDTA.oe = !NINTD;
!SPEAKER = !(SPKDTA & TC2);
```

10.3 Bausteine der Intel-82xx-Reihe

10.3.1 8253

intel

8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- 3 Independent 16-Bit Counters
- DC to 2.6 MHz
- Programmable Counter Modes
- Count Binary or BCD
- Single +5V Supply
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel® 8253 is a programmable counter/timer device designed for use as an Intel microcomputer peripheral. It uses NMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2.6 MHz. All modes of operation are software programmable

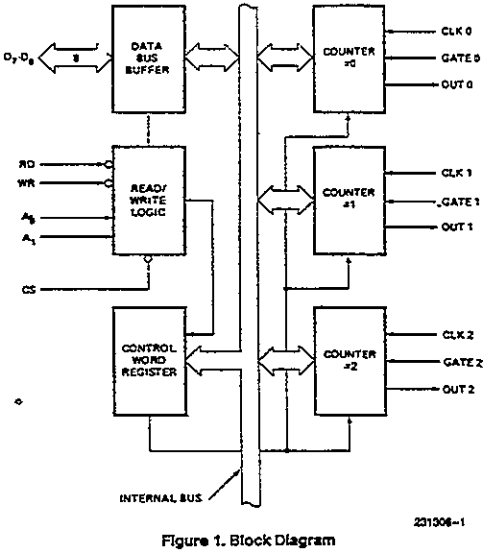


Figure 1. Block Diagram

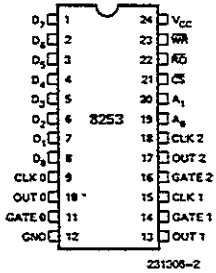


Figure 2. Pin Configuration

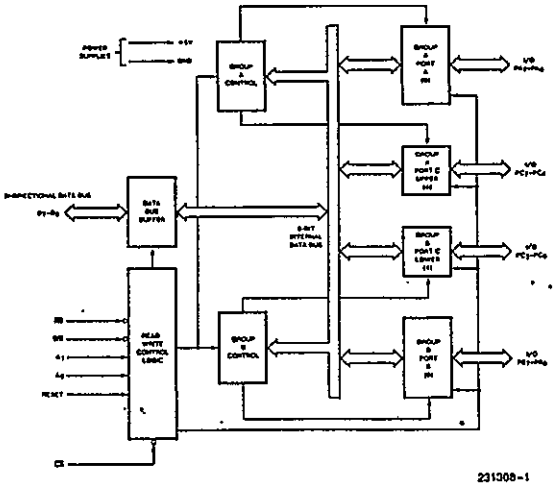
10.3.2 8255

intel

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

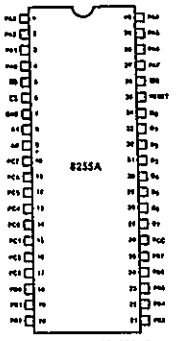
- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- Reduces System Package Count
- Improved DC Driving Capability
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range
- 40 Pin DIP Package or 44 Lead PLCC
  (See Intel Packaging, Order Number 231369)

The Intel 8255A is a general purpose programmable I/O device designed for use with Intel microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.



231308-1

Figure 1. 8255A Block Diagram



231308-2

Figure 2. Pin Configuration

10.3.3 8259

intel

8259A PROGRAMMABLE INTERRUPT CONTROLLER (8259A/8259A-2/8259A-8)

- 8086, 8088 Compatible
- MCS-80®, MCS-85® Compatible
- Eight-Level Priority Controller
- Expandable to 64 Levels
- Programmable Interrupt Modes
- Individual Request Mask Capability
- Single +5V Supply (No Clocks)
- 28-Pin Dual-In-Line Package
- Available in EXPRESS
  - Standard Temperature Range
  - Extended Temperature Range

The Intel 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28 pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

The 8259A is fully upward compatible with the Intel 8259. Software originally written for the 8259 will operate the 8259A in all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered).

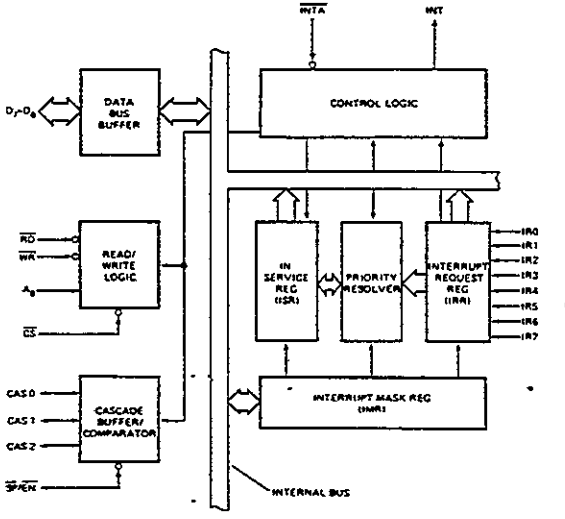


Figure 1 Block Diagram

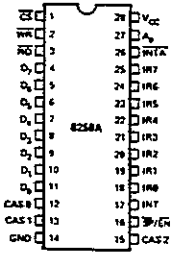


Figure 2. Pin Configuration

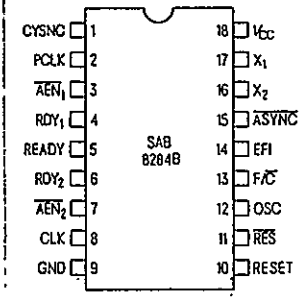
10.3.4 8284

Preliminary

SAB 8284B, SAB 8284B-1 Clock Generator and Driver for SAB 8086 Family Processors

- Fully compatible with SAB 8284A, SAB 8284A-1
- 30% Less Power Supply Current than Standard SAB 8284A, SAB 8284A-1
- Generates the System clock for SAB 8086 and SAB 8088 Processors: upto 8 MHz with SAB 8284B upto 10 MHz with SAB 8284B-1
- Uses a Crystal or a TTL Signal for Frequency Source upto 30 MHz
- Provides Synchronization for Synchronous and Asynchronous READY Signals
- 18-Pin Package
- Single +5V Power Supply
- Generates System Reset Output from Schmitt Trigger Input
- Capable of Clock Synchronization with Other SAB 8284Bs

Figure 1
Pin Configuration



Pin Names	
X_1 X_2	Connections for crystal
F/C	Clock source select
EFI	External clock input
CSYNC	Clock synchronization input
ASYNC	Ready synchronization select
RDY_1 RDY_2	Ready signal
AEN_1 AEN_2	Address enabled qualifiers for $RDY_{1,2}$
RES	Reset input
RESET	Synchronized reset output
OSC	Oscillator output
CLK	MOS Clock for the processor
PCLK	TTL Clock for peripherals
READY	Synchronized ready output
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

SAB 8284B is a bipolar clock generator/driver designed to provide clock signals for SAB 8086 and SAB 8088 processors and peripherals. It also contains READY logic for operation with two bus systems and provides the processors required READY synchronization and timing. Reset logic with hysteresis and synchronization is also provided. This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

847

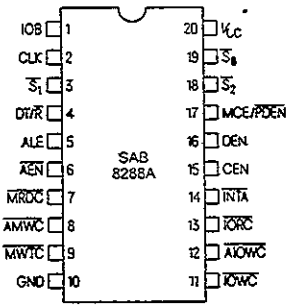
10.3.5 8288

Preliminary

SAB 8288A Bus Controller for SAB 8086 Family Processors

- Fully compatible with SAB 8288
- 40% Less Power Supply Current than Standard SAB 8288
- Bipolar Drive Capability
- Provides Advanced Commands
- Provides Wide Flexibility in System Configurations
- 3-State Command Output Drivers
- Configurable for Use with an I/O Bus
- Facilitates Interface to One or Two Multi Master Busses

Pin Configuration



Pin Names

$\overline{S_0}-\overline{S_2}$	Status
CLK	Clock
ALE	Adress Latch Enable
DEN	Data Enable
DT/$\overline{R}$	Data Transmit/Receive
$\overline{AEN}$	Address Enable
CEN	Command Enable
IOB	Input/Output Bus Mode
$\overline{AIOWC}$	Advanced I/O Write
$\overline{IOWC}$	I/O Write
$\overline{IORC}$	I/O Read
$\overline{AMWC}$	Advanced Memory Write
$\overline{MWTC}$	Memory Write
$\overline{MRDC}$	Memory Read
$\overline{INTA}$	Interrupt Acknowledge
MCE/$\overline{PDEN}$	Master Cascade/Peripheral Data
V_{CC}	Power Supply (+5V)
GND	Ground (0V)

SAB 8288A Bus Controller is a 20 pin bipolar component for use with medium to-large SAB 80186 SAB 80188 SAB 8086 and SAB 8088 processing systems The bus controller provides command and control timing generation as well as bipolar bus drive capability while optimizing system performance

A strapping option on the bus controller configures it for use with a multi master system bus and separate I/O bus.
This device is fabricated in a fast bipolar ASBC (Advanced Standard Buried Collector) process of Siemens.

10.4 Prozessor 8088

NEC

μPD70108 (V20™) 8/16-BIT HIGH-PERFORMANCE CMOS MICROPROCESSOR

Description

The μPD70108 (V20) is a CMOS 16-bit microprocessor with internal 16-bit architecture and an 8-bit external data bus. The μPD70108 instruction set is a superset of the μPD8086/8088; however, mnemonics and execution times are different. The μPD70108 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/division operations. The μPD70108 can also execute the entire 8080 instruction set and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the μPD70116 16-bit microprocessor.

Features

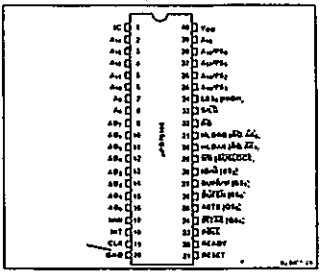
- Minimum instruction execution time: 250 ns (at 8 MHz)
- Maximum addressable memory: 1 Mbyte
- Abundant memory addressing modes
- 14 x 16-bit register set
- 101 instructions
- Instruction set is a superset of μPD8086/8088 instruction set
- Bit, byte, word, and block operations
- Bit field operation instructions
- Packed BCD instructions
- Multiplication/division instruction execution time: 4 μs to 6 μs (at 8 MHz)
- High-speed block transfer instructions: 1 Mbyte/s (at 8 MHz)
- High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- Maskable (INT) and nonmaskable (NMI) interrupt inputs
- IEEE-796 bus compatible interface
- 8080 emulation mode
- CMOS technology
- Low-power consumption
- Low-power standby mode
- Single power supply
- 5 MHz, 8 MHz or 10 MHz clock

Ordering Information

Part Number	Package Type	Max Frequency of Operation
μPD70108C-5	40-pin plastic DIP	5 MHz
μPD70108C-8	40-pin plastic DIP	8 MHz
μPD70108D-5	40-pin ceramic DIP	5 MHz
μPD70108D-8	40-pin ceramic DIP	8 MHz
μPD70108D-10	40-pin ceramic DIP	10 MHz
μPD70108G-5	52-pin plastic flat pack	5 MHz
μPD70108G-8	52-pin plastic flat pack	8 MHz
μPD70108L-5	44-pin PLCC	5 MHz
μPD70108L-8	44-pin PLCC	8 MHz

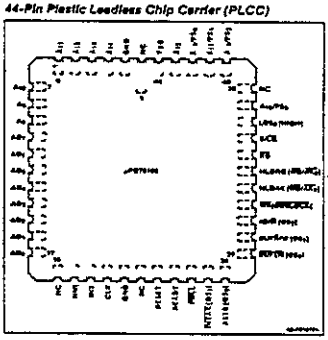
Pin Configurations

40-Pin Plastic DIP/Cerdip

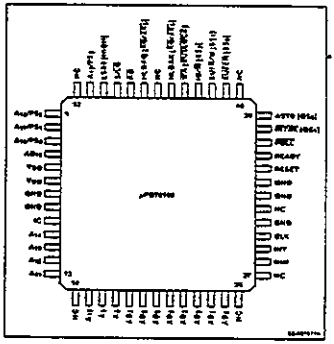


Pin Configurations (cont)

44-Pin Plastic Leadless Chip Carrier (PLCC)



52-Pin Plastic Flat Pack



Pin Identification

Symbol	Direction	Function
IC*		Internally connected
A_{14} - A_8	Out	Address bus, middle bits
AD_7 - AD_0	In/Out	Address/data bus
NMI	In	Nonmaskable interrupt input
INT	In	Maskable interrupt input
CLK	In	Clock input
GND		Ground potential
RESET	In	Reset input
READY	In	Ready input
POLL	In	Poll input
INTAK (QE_1)	Out	Interrupt acknowledge output (queue status bit 1 output)
ASTB (QS_0)	Out	Address strobe output (queue status bit 0 output)
BUFEN (BS_0)	Out	Buffer enable output (bus status bit 0 output)
BUFR/W (BS_1)	Out	Buffer read/write output (bus status bit 1 output)
IO/M (BS_2)	Out	Access is I/O or memory (bus status bit 2 output)
WR (BUSLOCK)	Out	Write strobe output (bus lock output)
HLDAK ($\overline{RQ}$/AK_1)	Out (In/Out)	Hold acknowledge output (bus hold request input/acknowledge output 1)
HLDRQ ($\overline{RQ}$/AK_0)	In (In/Out)	Hold request input (bus hold request input/acknowledge output 0)
$\overline{RD}$	Out	Read strobe output
S/$\overline{LG}$	In	Small-scale/large-scale system input
LBS_0 (HIGH)	Out	Latched bus status output 0 (always high in large-scale systems)
A_{19}/PS_3 - A_{16}/PS_0	Out	Address bus, high bits or processor status output
A_{15}	Out	Address bus, bit 15
V_{DD}		Power supply

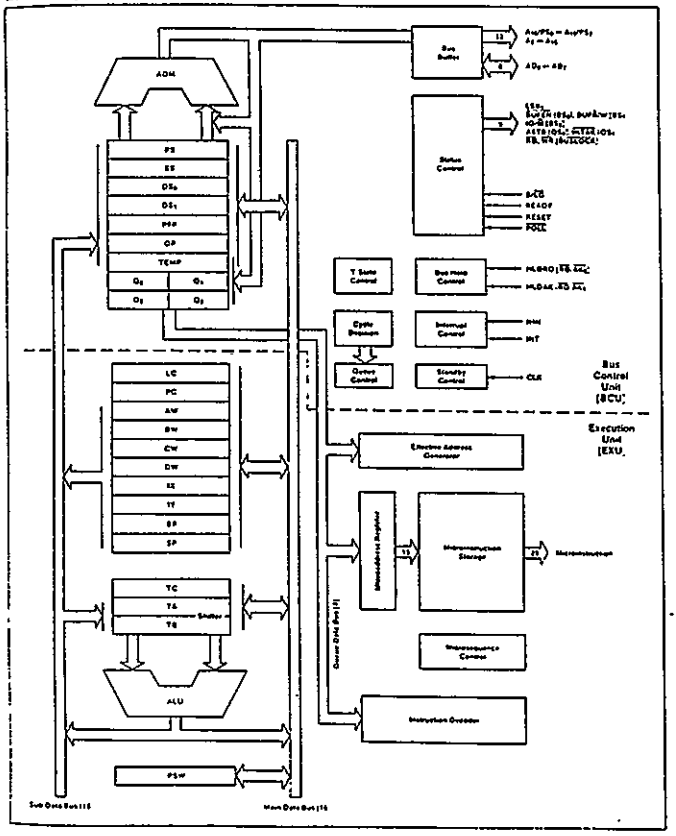
Notes: * IC should be connected to ground

Where pins have different functions in small and large-scale systems the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

NEC μPD70108 (V20)

Block Diagram



Pin Functions

Some pins of the μPD70108 have different function according to whether the microprocessor is used in a small or large scale system. Other pins function the same way in either type of system.

A_{15} - A_8 [Address Bus]
For small and large-scale systems

The CPU uses these pins to output the middle 8 bits of the 20 bit address data. They are three-state outputs and become high impedance during hold acknowledge.

AD_7 - AD_0 [Address/Data Bus]
For small and large-scale systems

The CPU uses these pins as the time-multiplexed address and data bus. When high, an AD bit is a one; when low, an AD bit is a zero. This bus contains the lower 8 bits of the 20-bit address during T1 of the bus cycle and is used as an 8-bit data bus during T2, T3 and T4 of the bus cycle.

Sixteen-bit data I/O is performed in two steps. The low byte is sent first, followed by the high byte. The address/data bus is a three-state bus and can be at a high or low level during standby mode. The bus will be high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]
For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge triggered and must be held high for five clocks to guarantee recognition. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the μPD70108 to exit the standby mode.

INT [Maskable Interrupt]
For small- and large-scale systems

This pin is an interrupt request that can be masked by software.

INT is active high level and is sensed during the last clock of the instruction. The interrupt will be accepted if the interrupt enable flag IE is set. The CPU outputs the INTAK signal to inform external devices that the interrupt request has been granted. INT must be asserted until the interrupt acknowledge is returned.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the μPD70108 to exit the standby mode.

CLK [Clock]
For small and large-scale systems

This pin is used for external clock input.

RESET [Reset]
For small and large scale systems

This pin is used for the CPU reset signal. It is an active high level. Input of this signal has priority over all other operations. After the reset signal input returns to a low level, the CPU begins execution of the program starting at address FFFF0H.

In addition to causing normal CPU start, RESET input will cause the μPD70108 to exit the standby mode.

READY [Ready]
For small and large-scale systems

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (Tw) by setting this signal to inactive (low level) and requesting a read/write cycle delay.

If the READY signal is active (high level) during either the T3 or Tw state, the CPU will not generate a wait state.

$\overline{POLL}$ [Poll]
For small and large-scale systems

The CPU checks this input upon execution of the $\overline{POLL}$ instruction. If the input is low, then execution continues. If the input is high, the CPU will check the $\overline{POLL}$ input every five clock cycles until the input becomes low again.

The $\overline{POLL}$ and READY functions are used to synchronize CPU program execution with the operation of external devices.

$\overline{RD}$ [Read Strobe]
For small and large-scale systems

The CPU outputs this strobe signal during data read from an I/O device or memory. The IO/$\overline{M}$ signal is used to select between I/O and memory.

The three-state output is held high during standby mode and enters the high impedance state during hold acknowledge.

S/$\overline{LG}$ [Small/Large]
For small and large-scale systems

This signal determines the operation mode of the CPU. This signal is fixed at either a high or low level. When

this signal is a high level, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU

$\overline{INTAK}$ [Interrupt Acknowledge]
For small-scale systems

The CPU generates the $\overline{INTAK}$ signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus (AD_7 - AD_0).

ASTB [Address Strobe]
For small-scale systems

The CPU outputs this strobe signal to latch address information at an external latch.

ASTB is held at a low level during standby mode and hold acknowledge.

$\overline{BUFEN}$ [Buffer Enable]
For small-scale systems.

This is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations, with external memory or I/O devices or during input of an interrupt vector.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

BUF$\overline{R}$/W [Buffer Read/Write]
For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

BUF$\overline{R}$/W is a three-state output and becomes high impedance during hold acknowledge.

IO/$\overline{M}$ [IO/Memory]
For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A high-level output specifies I/O and a low-level signal specifies memory.

IO/$\overline{M}$'s output is three state and becomes high impedance during hold acknowledge.

$\overline{WR}$ [Write Strobe]
For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or, memory is performed by the IO/M signal.

This three-state output is held high during standby mode and enters the high-impedance state during hold acknowledge.

HLDAK [Hold Acknowledge]
For small-scale systems.

The HLDAK signal is used to indicate that the CPU accepts the hold request signal (HLDRQ). When this signal is a high level, the address bus, address/data bus, and the control lines become high impedance.

HLDRQ [Hold Request]
For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

LBS_0 [Latched Bus Status 0]
For small-scale systems.

The CPU uses this signal along with the IO/$\overline{M}$ and BUF$\overline{R}$/W signals to inform an external device what the current bus cycle is

IO/$\overline{M}$	BUFR/$\overline{W}$	LBS_0	Bus Cycle
0	0	0	Program fetch
0	0	1	Memory read
0	1	0	Memory write
0	1	1	Passive state
1	0	0	Interrupt acknowledge
1	0	1	I/O read
1	1	0	I/O write
1	1	1	Halt

A_{19}/PS_3 - A_{16}/PS_0 [Address Bus/Processor Status]
For small- and large-scale systems.

These pins are time multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

The processor status signals are provided for both memory and I/O use. PS_3 is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is put on pin PS_2. Pins PS_1 and PS_0 indicate which memory segment is being accessed.

A_{17}/PS_1	A_{16}/PS_0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

The output of these pins is three state and becomes high impedance during hold acknowledge.

QS_1, QS_0 [Queue Status]
For large-scale systems.

The CPU uses these signals to allow external devices such as the floating-point arithmetic processor chip (μPD72091) to monitor the status of the internal CPU instruction queue.

QS_1	QS_0	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions.

BS_2 - BS_0 [Bus Status]
For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is.

The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS_2	BS_1	BS_0	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

The output of these signals is three state and becomes high impedance during hold acknowledge.

$\overline{BUSLOCK}$ [Bus Lock]
For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the BUSLOCK prefix instruction or during an interrupt acknowledge cycle. It is a status signal to the other bus masters in a multiprocessor system, inhibiting them from using the system bus during this time.

The output of this signal is three state and becomes high impedance during hold acknowledge. $\overline{BUSLOCK}$ is high during standby mode except if the HALT instruction has a $\overline{BUSLOCK}$ prefix.

$\overline{RQ}/\overline{AK}_1$, $\overline{RQ}/\overline{AK}_0$ [Hold Request/Acknowledge]
For large-scale systems.

These pins function as bus hold request inputs ($\overline{RQ}$) and as bus hold acknowledge outputs ($\overline{AK}$). $\overline{RQ}$ $\overline{AK}$ has a higher priority than $\overline{RQ}/\overline{AK}_1$.

These pins have three-state outputs with on-chip pull up resistors which keep the pin at a high level when the output is high impedance.

V_{DD} [Power Supply]
For small- and large-scale systems.

This pin is used for the +5 V power supply.

GND [Ground]
For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]
This pin is used for tests performed at the factory by NEC. The μPD70108 is used with this pin at ground potential.

Absolute Maximum Ratings

T_A = +25 °C

Power supply voltage, V_{DD}	−0.5 V to +7.0 V
Power dissipation, PD_{MAX}	0.5 W
Input voltage, V_I	−0.5 V to V_{DD} + 0.3 V
CLK input voltage, V_K	−0.5 V to V_{DD} + 1.0 V
Output voltage, V_O	−0.5 V to V_{DD} + 0.3 V
Operating temperature, T_{OPT}	−40 °C to −85 °C
Storage temperature, T_{STG}	−65 °C to +150 °C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Capacitance

T_A = +25 °C, V_{DD} = 0 V

Parameter	Symbol	Limits Min	Max	Unit	Test Conditions
Input capacitance	C_I		15	pF	fc = 1 MHz Unmeasured pins returned to 0 V
I/O capacitance	C_{IO}		15	pF	

DC Characteristics

μPD70108-5, T_A = −40 °C to +85 °C, V_{DD} = +5 V ± 10%
μPD70108-8, μPD70108-10, T_A = −10 °C to +70 °C, V_{DD} = +5 V ± 5%

Parameter	Symbol		Limits Min	Typ	Max	Unit	Test Conditions
Input voltage high	V_{IH}		2.2		V_{DD} − 0.3	V	
Input voltage low	V_{IL}		−0.5		0.8	V	
CLK input voltage high	V_{KH}		3.9		V_{DD} + 1.0	V	
CLK input voltage low	V_{KL}		−0.5		0.6	V	
Output voltage high	V_{OH}		0.7 x V_{DD}			V	I_{OH} = −400 μA
Output voltage low	V_{OL}				0.4	V	I_{OL} = 2.5 mA
Input leakage current high	I_{LIH}				10	μA	V_I = V_{DD}
Input leakage current low	I_{LIL}				−10	μA	V_I = 0 V
Output leakage current high	I_{LOH}				10	μA	V_O = V_{DD}
Output leakage current low	I_{LOL}				−10	μA	V_O = 0 V
Supply current	I_{DD}	70108-5 5 MHz	30	60		mA	Normal operation
			5	10		mA	Standby mode
		70108-8 8 MHz	45	80		mA	Normal operation
			6	12		mA	Standby mode
		70108-10 10 MHz	60	100		mA	Normal operation
			7	14		mA	Standby mode

Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute

The PC increments each time the microprogram fetches an instruction from the instruction queue A new location value is loaded into the PC each time a branch call return or break instruction is executed At this time the contents of the PC are the same as the Prefetch Pointer (PFP)

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is a 16 bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next byte for the instruction queue The contents of PFP are an offset from the PS (Program Segment) register

The PFP is incremented each time the BCU prefetches an instruction from the program memory A new location will be loaded into the PFP whenever a branch call return or break instruction is executed At that time the contents of the PFP will be the same as those of the PC (Program Counter)

Segment Registers [PS SS DS_0, and DS_1]

The memory addresses accessed by the μPD70108 are divided into 64K byte logical segments The starting (base) address of each segment is specified by a 16 bit segment register and the offset from this starting address is specified by the contents of another register or by the effective address

These are the four types of segment registers used

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP effective address
DS_0 (Data Segment 0)	IX effective address
DS_1 (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW and DW]

There are four 16-bit general-purpose registers Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH AL, BH BL CH CL DH DL)

Each register is also used as a default register for processing specific instructions The default assignments are

- AW Word multiplication/division word I/O data conversion
- AL Byte multiplication/division byte I/O BCD rotation data conversion translation
- AH Byte multiplication/division
- BW Translation
- CW Loop control branch repeat prefix
- CL Shift instructions rotation instructions BCD operations
- DW Word multiplication/division indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing indexed addressing or based indexed addressing

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers They cannot be used as 8 bit registers

Also each of these registers acts as a default register for specific operations The default assignments are

- SP Stack operations
- IX Block transfer (source) BCD string operations
- IY Block transfer (destination) BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags

Status Flags	Control Flags
• V (Overflow)	• MD (Mode)
• S (Sign)	• DIR (Direction)
• Z (Zero)	• IE (Interrupt Enable)
• AC (Auxiliary Carry)	• BRK (Break)
• P (Parity)	
• CY (Carry)	

When the PSW is pushed on the stack the word images of the various flags are as shown here

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M D	1	1	1	V	D IR	I E	B RK	S	Z	0	AC	0	P	1	CY

The status flags are set and reset depending upon the result of each type of instruction executed

Instructions are provided to set reset and complement the CY flag directly

Other instructions set and reset the control flags and control the operation of the CPU

NEC

High-Speed Execution of Instructions

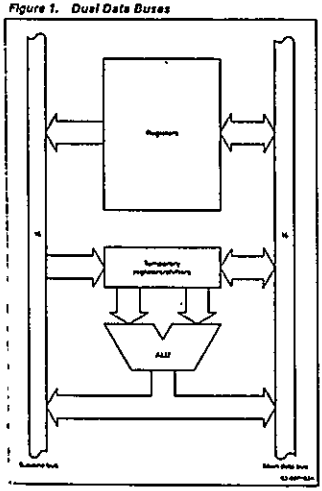
This section highlights the major architectural features that enhance the performance of the μPD70108.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the μPD70108 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been speeded up some 30% over single-bus systems.

Figure 1. Dual Data Buses



Example

ADD AW, BW : AW ← AW + BW

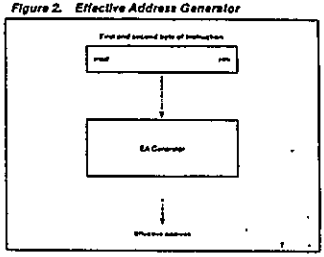
Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective Address Generator

This circuit (figure 2) performs high-speed processing to calculate effective addresses for accessing memory.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

Figure 2. Effective Address Generator



16/32-Bit Temporary Registers/Shifters (TA, TB)

These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction

The processing performed for a multiple bit rotation of a register is shown below The average speed is approximately doubled over the microprogram method

Example

RORC AW CL CL = 5

Microprogram method LC method

8 + (4 x 5) = 28 clocks 7 + 5 = 12 clocks

Program Counter and Prefetch Pointer [PC and PFP]

The μPD70108 microprocessor has a program counter (PC) which addresses the program memory location of the instruction to be executed next and a prefetch pointer(PFP) which addresses the program memory location to be accessed next Both functions are provided in hardware A time saving of several clocks is realized for branch call return and break instruction execution compared with microprocessors that have only one instruction pointer

Enhanced Instructions

In addition to the μPD8085/86 instructions the μPD70108 has the following enhanced instructions

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and closes previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm

This instruction allows immediate data to be pushed onto the stack

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction

Enhanced Multiplication Instructions

MUL reg16 imm16/MUL mem16, imm16

These instructions allow the contents of a register or memory location to be 16-bit multiplied by immediate data

Enhanced Shift and Rotate Instructions

SHL reg imm8/SHR reg imm8/SHRA reg imm8

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data

ROL reg imm8/ROR reg imm8/ROLC reg imm8/ RORC reg imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data

Check Array Boundary Instruction

CHKIND reg16 mem32

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range The lower limit of the array should be in memory location mem32 the upper limit in mem32 + 2 If the index value in reg16 is not between these limits when CHKIND is executed a BRK 5 will occur This causes a jump to the location in interrupt vector 5

Block I/O Instructions

OUTM DW src block/INM dst-block, DW

These instructions are used to output or input a string to or from memory when preceded by a repeat prefix

Stack Frame Instructions

PREPARE imm16, imm8

This instruction is used to generate the stack frames required by block-structured languages such as PASCAL and Ada The stack frame consists of two areas One area has a pointer that points to another frame which has variables that the current frame can access The other is a local variable area for the current procedure

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the μPD8088/86 instructions and the enhanced instructions, the μPD70108 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

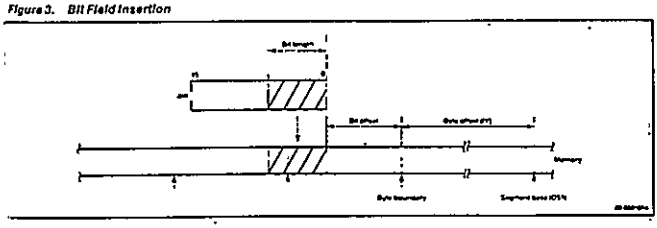
This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS_1 register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Figure 3. Bit Field Insertion



EXT reg8 reg8/EXT reg8 imm4

This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base) the IX index register (byte offset) and the lower 4 bits of the first operand (bit offset)

After the transfer is complete the IX register and the lower 4 bits of the first operand are automatically updated to point to the next bit field

Either immediate data or a register may be specified for the second operand Because the maximum transferrable bit length is 16 bits however only the lower 4 bits of the specified register (0H to 0FH) will be valid

Bit field data may overlap the byte boundary of memory

Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S SUB4S CMP4S) or byte-format operands (ROR4 ROL4) Packed BCD strings may be from 1 to 254 digits in length

When the number of digits is even the zero and carry flags will be set according to the result of the operation When the number of digits is odd the zero and carry flags may not be set correctly in this case (CL = odd) the zero flag will not be set unless the upper 4 bits of the highest byte are all zero The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte When CL is odd the contents of the upper 4 bits of the highest byte of the result are undefined

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register and stores the result in the string addressed by the IY register The length of the string (number of BCD digits) is specified by the CL register and the result of the operation will affect the overflow flag (V) the carry flag (CY) and zero flag (Z)

BCD string (IY CL) ← BCD string (IY CL) + BCD string (IX CL)

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY register and stores the result in the string addressed by the IY register The length of the string (number of BCD digits) is specified by the CL register and the result of the operation will affect the overflow flag (V) the carry flag (CY) and zero flag (Z)

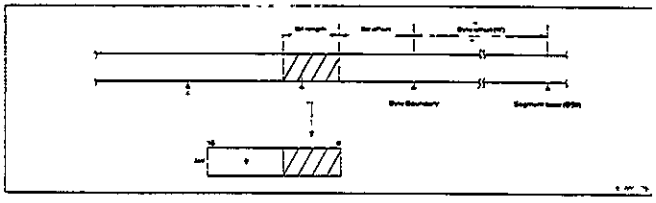
BCD string (IY CL) ← BCD string (IY CL) - BCD String (IX CL)

CMP4S

This instruction performs the same operation as SUB4S except that the result is not stored and only the overflow (V) carry flags (CY) and zero flag (Z) are affected

* BCD string (IY CL) – BCD string (IX CL)

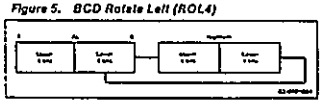
Figure 4 Bit Field Extraction



ROL4

This instruction (figure 5) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL_L) to rotate that data one BCD digit to the left

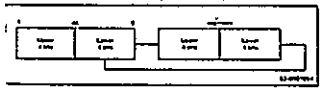
Figure 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL_L) to rotate that data one BCD digit to the right.

Figure 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1

NOT1

This instruction inverts a specific bit in a register or memory location

CLR1

This instruction clears a specific bit in a register or memory location

SET1

This instruction sets a specific bit in a register or memory location

Repeat Prefix Instructions

REPC

This instruction causes the μPD70108 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero

REPNC

This instruction causes the μPD70108 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register is decremented to zero.

Floating Point Instruction

FPO2

This instruction is in addition to the μPD8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instructions

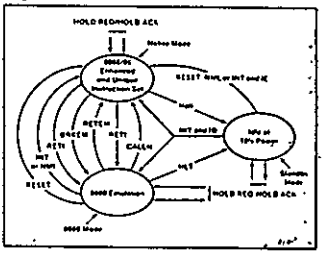
The μPD70108 has two operating modes (figure 7). One is the native mode which executes μPD8088/86 enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the μPD8080AF is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back: BRKEM (Break for Emulation), and RETEM (Return from Emulation)

Two instructions are used to switch from the emulation mode to the native mode and back: CALLN (Call Native Routine), and RETI (Return from Interrupt)

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present or when an external interrupt (NMI or INT) is present

Figure 7. V20 Modes



BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction except that BRKEM resets the mode flag (MD) to 0. PSW, PS and PC are saved to the stack, MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as μPD8080AF instructions.

In 8080 emulation mode, registers and flags of the μPD8080AF are performed by the following registers and flags of the μPD70108.

	μPD8080AF	μPD70108
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	SP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by SP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DS_0, and DS_1) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DS_0 register (set by the programmer immediately before the 8080 emulation mode is entered).

It is prohibited to nest BRKEM instructions.

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as a μPD8080AF instruction), the CPU restores PS, PC, and PSW (as it would when returning from an interrupt processing routine) and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction is restored to MD. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as μPD8080AF instruction) is similar to that performed when a BRK instruction is executed in the native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1, and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored; the CPU is set in emulation mode, and all sobsequent instructions are interpreted and executed as μPD8080AF instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op mem/FPO2 fp-op mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor

Interrupt Operation

The interrupts used in the μPD70108 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External Interrupts

(a) NMI input (nonmaskable)
(b) INT input (maskable)

Software Processing

As the result of instruction execution

- When a divide error occurs during execution of the DIV or DIVU instruction
- When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

- When V = 1 during execution of the BRKV instruction

Unconditional break instructions

- 1-byte break instruction: BRK3
- 2-byte break instruction: BRK imm8

Flag processing

- When stack operations are used to set the BRK flag

8080 Emulation mode instructions

- BRKEM imm8
- CALLN imm8

Interrupt Vectors

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. These vectors consequently cannot be used for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

A single interrupt vector is made up of 4 bytes (figure 9): The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

Figure 8. Interrupt Vector Table

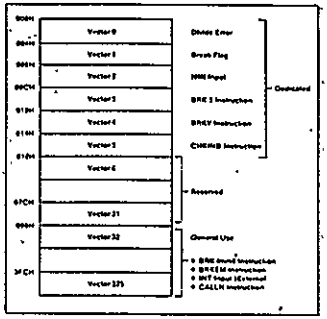
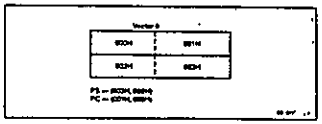


Figure 9. Interrupt Vector 0



Based on this format the contents of each vector should be initialized at the beginning of the program

The basic steps to jump to an interrupt processing routine are now shown

(SP - 1 SP - 2) — PSW
(SP - 3 SP - 4) — PS
(SP - 5 SP - 6) — PC
SP — SP - 6
IE — 0 BRK — 0 MD — 1
PS — vector high bytes
PC — vector low bytes

Standby Function

The μPD70108 has a standby mode to reduce power consumption during program wait states This mode is set by the HALT instruction in both the native and the emulation mode

In the standby mode the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions As a result power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode

The standby mode is released by inputting a RESET signal or an external interrupt (NMI INT)

The bus hold function is effective during standby mode The CPU returns to standby mode when the bus hold request is removed

During standby mode all control outputs are disabled and the addres/data bus will be at either high or low levels

Instruction Set

Symbols

Preceding the instruction set, several tables explain symbols abbreviations and codes

Clocks

In the Clocks column of the instruction set, the numbers cover these operations. instruction decoding effective address calculation operand fetch and instruction execution

Clock timings assume the instruction has been prefetched and is present in the four-byte instruction queue Otherwise add four clocks for each byte not present

For instructions that reference memory operands the number on the left side of the slash (/) is for byte operands and the number on the right side is for word operands

For conditional control transfer or branch instructions the number on the left side of the slash is applicable if the transfer or branch takes place The number on the right side is applicable if it does not take place

If a range of numbers is given the execution time depends on the operands involved

Symbols

Symbol	Meaning
acc	Accumulator (AW or AL)
disp	Displacement (8 or 16 bits)
dmem	Direct memory address
dst	Destination operand or address
ext-disp8	16-bit displacement (sign-extension byte + 8 bit displacement)
far_label	Label within a different program segment
far_proc	Procedure within a different program segment
fp_op	Floating point instruction operat on
imm	8- or 16-bit immediate operand
imm3/4	3/4 bit immediate bit offset
imm8	8-bit immediate operand
imm16	16-bit immediate operand
mem	Memory field (000 to 111) 8- or 16-bit memory location
mem8	8 bit memory location
mem16	16 bit memory location
mem32	32-bit memory location
memptr16	Word containing the destination address within the current segment
memptr32	Double word containing a destination address in another segment
mod	Mode field (00 to 10)
near_label	Label within the current segment
near_proc	Procedure within the current segment
offset	Immediate offset data (16 bits)
pop value	Number of bytes to discard from the stack
reg	Register field (000 to 111) 8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
regptr	16-bit register containing a destination address within the current segment
regptr16	Register containing a destination address within the current segment
seg	Immediate segment data (16 bits)
short_label	Label between −128 and +127 bytes from the end of the current instruction

NEC

Symbols (cont)

Symbol	Meaning
sr	Segment register
src	Source operand or address
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
AC	Auxiliary carry flag
AH	Accumulator (high byte)
AL	Accumulator (low byte)
AND	Logical product
AW	Accumulator (16 bits)
BH	BW register (high byte)
BL	BW register (low byte)
BP	BP register
BRK	Break flag
BW	BW register (16 bits)
CH	CW register (high byte)
CL	CW register (low byte)
CW	CW register (16 bits)
CY	Carry flag
DH	DW register (high byte)
DIR	Direction flag
DL	DW register (low byte)
DS0	Data segment 0 register (16 bits)
DS1	Data segment 1 register (16 bits)
DW	DW register (16 bits)
IE	Interrupt enable flag
IX	Index register (source) (16 bits)

Symbols

Symbol	Meaning
IY	Index register (destination) (16 bits)
MD	Mode flag
OR	Logical sum
P	Parity flag
PC	Program counter (16 bits)
PS	Program segment register (16 bits)
PSW	Program status word (16 bits)
R	Register set
S	Sign extend operand field S = 0 No sign extension S = 1 Sign extend immediate byte operand
S	Sign flag
SP	Stack pointer (16 bits)
SS	Stack segment register (16 bits)
V	Overflow flag
W	Word/byte field (0 to 1)
X, XXX YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value
Z	Zero flag
( )	Values in parentheses are memory contents
←	Transfer direction
+	Addition
-	Subtraction
×	Multiplication
÷	Division
%	Modulo

Flag Operations

Symbol	Meaning
(blank)	No change
0	Cleared to 0
1	Set to 1
*	Set or cleared according to result
U	Undefined
R	Restored to previous state

Memory Addressing Modes

mem	mod = 00	mod = 01	mod = 10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Register Selection (mod = 11)

reg	W = 0	W = 1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Segment Register Selection

sr	Segment Register
00	DS1
01	PS
10	SS
11	DS0

NEC

μPD70108 (V20)

Instruction Set

Mnemonic	Operand	Opcode 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Data Transfer Instructions						
MOV	reg, reg	1 0 0 0 1 0 1 W	1 1 reg reg	2	2	
	mem, reg	1 0 0 0 1 0 0 W	mod reg mem	9/13	2-4	
	reg, mem	1 0 0 0 1 0 1 W	mod reg mem	11/15	2-4	
	mem, imm	1 1 0 0 0 1 1 W	mod reg mem	11/15	3-6	
	reg, imm	1 0 1 1 W reg		4	2-3	
	acc, dmem	1 0 1 0 0 0 0 W		10/14	3	
	dmem, acc	1 0 1 0 0 0 1 W		9/13	3	
	sr, reg16	1 0 0 0 1 1 1 0	1 1 0 sr reg	2	2	
	sr, mem16	1 0 0 0 1 1 1 0	mod 0 sr mem	11/15	2-4	
	reg16, sr	1 0 0 0 1 1 0 0	1 1 0 sr reg	2	2	
	mem16, sr	1 0 0 0 1 1 0 0	mod 0 sr mem	10/14	2-4	
	DS0, reg16, mem32	1 1 0 0 0 1 0 1	mod reg mem	18/26	2-4	
	DS1, reg16, mem32	1 1 0 0 0 1 0 0	mod reg mem	18/26	2-4	
	AH, PSW	1 0 0 1 1 1 1 1		2	1	
	PSW, AH	1 0 0 1 1 1 1 0		3	1	X X X X X
LDEA	reg16, mem16	1 0 0 0 1 1 0 1	mod reg mem	4	2-4	
TRANS	src_table	1 1 0 1 0 1 1 1		9	1	
XCH	reg, reg	1 0 0 0 0 1 1 W	1 1 reg reg	3	2	
	mem, reg	1 0 0 0 0 1 1 W	mod reg mem	16/26	2-4	
	AW, reg16	1 0 0 1 0 reg		3	1	
Repeat Prefixes						
REPC		0 1 1 0 0 1 0 1		2	1	
REPNC		0 1 1 0 0 1 0 0		2	1	
REP REPE REPZ		1 1 1 1 0 0 1 1		2	1	
REPNE REPNZ		1 1 1 1 0 0 1 0		2	1	
Block Transfer Instructions						
MOVBK	dst, src	1 0 1 0 0 1 0 W		11 + 8n	1	
CMPBK	dst, src	1 0 1 0 0 1 1 W		7 + 14n	1	X X X X X X
CMPM	dst	1 0 1 0 1 1 1 W		7 + 10n	1	X X X X X X
LDM	src	1 0 1 0 1 1 0 W		7 + 9n	1	
STM	dst	1 0 1 0 1 0 1 W		7 + 4n	1	
		n = number of transfers				
I/O Instructions						
IN	acc, imm8	1 1 1 0 0 1 0 W		9/13	2	
	acc, DW	1 1 1 0 1 1 0 W		8/12	1	
OUT	imm8, acc	1 1 1 0 0 1 1 W		8/12	2	
	DW, acc	1 1 1 0 1 1 1 W		8/12	1	
INM	dst, DW	0 1 1 0 1 1 0 W		9 + 8n	1	
OUTM	DW, src	0 1 1 0 1 1 1 W		9 + 8n	1	
		n = number of transfers				

Instruction Set (cont)

Mnemonic	Operand	Opcode 7 6 5 4 3 2 1 0	Opcode 7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
BCD Instructions						
ADJBA		0 0 1 1 0 1 1 1		3	1	x x U U U U
ADJ4A		0 0 1 0 0 1 1 1		3	1	x x U U x x
ADJBS		0 0 1 1 1 1 1 1		7	1	x x U U U U
ADJ4S		0 0 1 0 1 1 1 1		7	1	x x U U x x
ADD4S	dst src	0 0 0 0 1 1 1 1	0 0 1 0 0 0 0 0	7 + 19n	2	U x U U U x
SUB4S	dst src	0 0 0 0 1 1 1 1	0 0 1 0 0 0 1 0	7 + 19n	2	U x U U U x
CMP4S	dst src	0 0 0 0 1 1 1 1	0 0 1 0 0 1 1 0	7 + 19n	2	U x U U U x
ROL4	reg8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 0 1 0 0 0	25	3	
	mem8	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 1 0 1 0 0 0	28	3-5	
RORd	reg8	0 0 0 0 1 1 1 1 1 1 0 0 1 reg	0 0 1 0 1 0 1 0	29	3	
	mem8	0 0 0 0 1 1 1 1 mod 0 0 1 mem n = number of BCD digits divided by 2	0 0 1 0 1 0 1 0	33	3-5	
Data Type Conversion Instructions						
CVTBD		1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0	15	2	U U U x x x
CVTDB		1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0	7	2	U U U x x x
CVTBW		1 0 0 1 1 0 0 0		2	1	
CVTWL		1 0 0 1 1 0 0 1		4-5	1	
Arithmetic Instructions						
ADD	reg reg	0 0 0 0 0 0 1 W	1 1 reg reg	2	2	x x x x x x
	mem reg	0 0 0 0 0 0 0 W	mod reg mem	16/24	2-4	x x x x x x
	reg mem	0 0 0 0 0 0 1 W	mod reg mem	11/15	2-4	x x x x x x
	reg imm	1 0 0 0 0 0 S W	1 1 0 0 0 reg	4	3-4	x x x x x x
	mem imm	1 0 0 0 0 0 S W	mod 0 0 0 mem	16/26	3-6	x x x x x x
	acc imm	0 0 0 0 0 1 0 W		4	2-3	x x x x x x
ADDC	reg reg	0 0 0 1 0 0 1 W	1 1 reg reg	2	2	x x x x x x
	mem reg	0 0 0 1 0 0 0 W	mod reg mem	16/24	2-4	x x x x x x
	reg, mem	0 0 0 1 0 0 1 W	mod reg mem	11/15	2-4	x x x x x x
	reg, imm	1 0 0 0 0 0 S W	1 1 0 1 0 reg	4	3-4	x x x x x x
	mem, imm	1 0 0 0 0 0 S W	mod 0 1 0 mem	16/26	3-6	x x x x x x
	acc, imm	0 0 0 1 0 1 0 W		4	2-3	x x x x x x
SUB	reg reg	0 0 1 0 1 0 1 W	1 1 reg reg	2	2	x x x x x x
	mem reg	0 0 1 0 1 0 0 W	mod reg mem	16/24	2-4	x x x x x x
	reg mem	0 0 1 0 1 0 1 W	mod reg mem	11/15	2-4	x x x x x x
	reg imm	1 0 0 0 0 0 S W	1 1 1 0 1 reg	4	3-4	x x x x x x

NEC

Instruction Set (cont)

Mnemonic	Operand	Opcode 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Arithmetic instructions (cont)						
	mem, imm	1 0 0 0 0 0 S W	mod 1 0 1 mem	16/26	3-6	x x x x x x
	acc, imm	0 0 1 0 1 1 0 W		4	2-3	x x x x x x
SUBC	reg, reg	0 0 0 1 1 0 1 W	1 1 reg reg	2	2	x x x x x x
	mem, reg	0 0 0 1 1 0 0 W	mod reg mem	16/24	2-4	x x x x x x
	reg, mem	0 0 0 1 1 0 1 W	mod reg mem	11/15	2-4	x x x x x x
	reg, imm	1 0 0 0 0 0 S W	1 1 0 1 1 reg	4	3-4	x x x x x x
	mem, imm	1 0 0 0 0 0 S W	mod 0 1 1 mem	16/26	3-6	x x x x x x
	acc, imm	0 0 0 1 1 1 0 W		4	2-3	x x x x x x
INC	reg8	1 1 1 1 1 1 1 0	1 1 0 0 0 reg	2	2	x - x x x
	mem	1 1 1 1 1 1 1 W	mod 0 0 0 mem	16/24	2-4	x - x x x
	reg16	0 1 0 0 0 reg		2	1	x - x x x
DEC	reg8	1 1 1 1 1 1 1 0	1 1 0 0 1 reg	2	2	x - x x x
	mem	1 1 1 1 1 1 1 W	mod 0 0 1 mem	16/24	2-4	x - x x x
	reg16	0 1 0 0 1 reg		2	1	x - x x x
MULU	reg	1 1 1 1 0 1 1 W	1 1 1 0 0 reg	21-30	2	u x x u u u
	mem	1 1 1 1 0 1 1 W	mod 1 0 0 mem	27-36	2-4	u x x u u u
MUL	reg	1 1 1 1 0 1 1 W	1 1 1 0 1 reg	33-47	2	u x x u u u
	mem	1 1 1 1 0 1 1 W	mod 1 0 1 mem	33-57	2-4	u x x u u u
	reg16,reg16,imm8	0 1 1 0 1 0 1 1	1 1 reg reg	28-34	3	u x x u u x
	reg16,mem16,imm8	0 1 1 0 1 0 1 1	mod reg mem	34-44	3-5	u x x u u x
	reg16,reg16,imm16	0 1 1 0 1 0 0 1	1 1 reg reg	36-42	4	u x x u u u
	reg16,mem16,imm16	0 1 1 0 1 0 0 1	mod reg mem	44-52	4-6	u x x u u u
DIVU	reg	1 1 1 1 0 1 1 W	1 1 1 1 0 reg	19-25	2	u u u u u u
	mem	1 1 1 1 0 1 1 W	mod 1 1 0 mem	25-35	2-4	u u u u u u
DIV	reg	1 1 1 1 0 1 1 W	1 1 1 1 1 reg	29-43	2	u u u u u u
	mem	1 1 1 1 0 1 1 W	mod 1 1 1 mem	35-53	2-4	u u u u u u
Comparison Instructions						
CMP	reg, reg	0 0 1 1 1 0 1 W	1 1 reg reg	2	2	x x x x x x
	mem, reg	0 0 1 1 1 0 0 W	mod reg mem	11/15	2-4	x x x x x x
	reg, mem	0 0 1 1 1 0 1 W	mod reg mem	11/15	2-4	x x x x x x
	reg, imm	1 0 0 0 0 0 S W	1 1 1 1 1 reg	4	3-4	x x x x x x
	mem, imm	1 0 0 0 0 0 S W	mod 1 1 1 mem	13/17	3-6	x x x x x x
	acc, imm	0 0 1 1 1 1 0 W		4	2-3	x x x x x x
Logical Instructions						
NOT	reg	1 1 1 1 0 1 1 W	1 1 0 1 0 reg	2	2	
	mem	1 1 1 1 0 1 1 W	mod 0 1 0 mem	16/24	2-4	
NEG	reg	1 1 1 1 0 1 1 W	1 1 0 1 1 reg	2	2	x x x x x x
	mem	1 1 1 1 0 1 1 W	mod 0 1 1 mem	16/24	2-4	x x x x x x
TEST	reg, reg	1 0 0 0 0 1 0 W	1 1 reg reg	2	2	u 0 0 x x x
	mem, reg	1 0 0 0 0 1 0 W	mod reg mem	10/14	2-4	u 0 0 x x x
	reg, imm	1 1 1 1 0 1 1 W	1 1 0 0 0 reg	4	3-4	u 0 0 x x x

Instruction Set (cont)

Mnemonic	Operand	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Clocks	Bytes	Flags: AC CY V P S Z
Logical Instructions (cont)						
	mem, imm	1 1 1 1 0 0 1 W	mod 0 0 0 mem	11/15	3-6	u 0 ? ? ?
	acc, imm	1 0 1 0 1 0 0 W		4	2-3	u ? ? ? ? ?
AND	reg, reg	0 0 1 0 0 0 1 W	1 1 reg reg	2	2	u 0 ? ? ? ?
	mem, reg	0 0 1 0 0 0 0 W	mod reg mem	16/24	2-4	u 0 ? ? ? ?
	reg, mem	0 0 1 0 0 0 1 W	mod reg mem	11/15	2-4	u 0 ? ? ? ?
	reg, imm	1 0 0 0 0 0 0 W	1 1 1 0 0 reg	4	3-4	u 0 ? ? ? ?
	mem, imm	1 0 0 0 0 0 0 W	mod 1 0 0 mem	18/26	3-6	u 0 ? ? ? ?
	acc, imm	0 0 1 0 0 1 0 W		4	2-3	u 0 ? ? ? ?
OR	reg, reg	0 0 0 0 1 0 1 W	1 1 reg reg	2	2	u 0 ? ? ? ?
	mem, reg	0 0 0 0 1 0 0 W	mod reg mem	16/24	2-4	u 0 ? ? ? ?
	reg, mem	0 0 0 0 1 0 1 W	mod reg mem	11/15	2-4	u 0 ? ? ? ?
	reg, imm	1 0 0 0 0 0 0 W	1 1 0 0 1 reg	4	3-4	u 0 ? ? ? ?
	mem, imm	1 0 0 0 0 0 0 W	mod 0 0 1 mem	18/26	3-6	u 0 ? ? ? ?
	acc, imm	0 0 0 0 1 0 0 W		4	2-3	u 0 ? ? ? ?
XOR	reg, reg	0 0 1 1 0 0 1 W	1 1 reg reg	2	2	u 0 ? ? ? ?
	mem, reg	0 0 1 1 0 0 0 W	mod reg mem	16/24	2-4	u 0 ? ? ?
	reg, mem	0 0 1 1 0 0 1 W	mod reg mem	11/15	2-4	u 0 ? ? ? ?
	reg, imm	1 0 0 0 0 0 0 W	1 1 1 1 0 reg	4	3-4	u 0 ? ? ? ?
	mem, imm	1 0 0 0 0 0 0 W	mod 1 1 0 mem	18/26	3-6	u 0 ? ? ? ?
	acc, imm	0 0 1 0 1 0 0 W		4	2-3	u 0 ? ? ? ?
Bit Manipulation Instructions						
INS	reg8, reg8	0 0 0 0 1 1 1 1 1 1 reg reg	0 0 1 1 0 0 0 1	35-133	3	
	reg8, imm8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 0 1	35-133	4	
EXT	reg8, reg8	0 0 0 0 1 1 1 1 1 1 reg reg	0 0 1 1 0 0 1 1	34-58	3	
	reg8, imm8	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 1 1 1 0 1 1	34-58	4	
TEST1	reg, CL	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 0 0 0 W	3	3	u 0 ? ? u ?
	mem, CL	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 0 0 0 W	12/16	3-5	u 0 ? ? u ?
	reg, imm3/4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 1 0 0 W	4	4	u 0 ? ? u ?
	mem, imm3/4	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 1 0 0 W	13/21	4-6	u 0 ? ? u ?
RET1	reg, CL	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 0 1 0 W	4	3	
	mem, CL	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 0 1 0 W	13/21	3-5	

NEC

Instruction Set (cont)

Mnemonic	Operands	Opcode								Clocks	Bytes	Flags					
		7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Bit Manipulation Instructions (cont)																	
	reg, imm3/4	0	0	0	0	1	1	1	1								
		1	1	0	0	0	reg			5	4						
	mem, imm3/4	0	0	0	0	1	1	1	1								
		mod			0	0	0	mem		14/22	4-6						
	CY	1	1	1	1	1	0	0	1	2	1		1				
	DIR	1	1	1	1	1	1	0	1	2	1						
CLR1	reg, CL	0	0	0	0	1	1	1	1								
		1	1	0	0	0	reg			5	3						
	mem, CL	0	0	0	0	1	1	1	1								
		mod			0	0	0	mem		14/22	3-5						
	reg, imm3/4	0	0	0	0	1	1	1	1								
		1	1	0	0	0	reg			6	4						
	mem, imm3/4	0	0	0	0	1	1	1	1								
		mod			0	0	0	mem		15/27	4-6						
	CY	1	1	1	1	1	0	0	0	2	1		0				
	DIR	1	1	1	1	1	1	0	0	2	1						
NOT1	reg, CL	0	0	0	0	1	1	1	1								
		1	1	0	0	0	reg			4	3						
	mem, CL	0	0	0	0	1	1	1	1								
		mod			0	0	0	mem		18/26	3-5						
	reg, imm3/4	0	0	0	0	1	1	1	1								
		1	1	0	0	0	reg			5	4						
	mem, imm3/4	0	0	0	0	1	1	1	1								
		mod			0	0	0	mem		19/27	4-6						
	CY	1	1	1	1	0	1	0	1	2	1		x				
Shift/Rotate Instructions																	
SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	reg	2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	0	mem	16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg	7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	0	mem	19/27 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg	7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	0	mem	19/27 + n	3-5	u	x	u	x	x	x
SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg	2	2	u	x	x	x	x	x
	mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem	16/24	2-4	u	x	x	x	x	x
	reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg	7 + n	2	u	x	u	x	x	x
	mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem	19/27 + n	2-4	u	x	u	x	x	x
	reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg	7 + n	3	u	x	u	x	x	x
	mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem	19/27 + n	3-5	u	x	u	x	x	x

n = number of shifts

NEC

μPD70108 (V20)

Instruction Set (cont)

Mnemonic	Operands	Opcode 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Bit Manipulation Instructions (cont)						
	reg, imm3/4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 1 1 0 W	5	4	
	mem, imm3/4	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 1 1 0 W	14/22	4-6	
	CY	1 1 1 1 1 0 0 1		2	1	↕
	DIR	1 1 1 1 1 1 0 1		2	1	
CLR1	reg, CL	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 0 0 1 W	5	3	
	mem, CL	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 0 0 1 W	14/22	3-5	
	reg, imm3/4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 1 0 1 W	6	4	
	mem, imm3/4	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 1 0 1 W	15/27	4-6	
	CY	1 1 1 1 1 0 0 0		2	1	0
	DIR	1 1 1 1 1 1 0 0		2	1	
NOT1	reg, CL	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 0 1 1 W	4	3	
	mem, CL	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 0 1 1 W	14/26	3-5	
	reg, imm3/4	0 0 0 0 1 1 1 1 1 1 0 0 0 reg	0 0 0 1 1 1 1 W	5	4	
	mem, imm3/4	0 0 0 0 1 1 1 1 mod 0 0 0 mem	0 0 0 1 1 1 1 W	15/27	4-6	
	CY	1 1 1 1 0 1 0 1		2	1	x
Shift/Rotate Instructions						
SHL	reg, 1	1 1 0 1 0 0 0 W	1 1 1 0 0 reg	2	2	u x x x x x
	mem, 1	1 1 0 1 0 0 0 W	mod 1 0 0 mem	16/24	2-4	u x x x x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 1 0 0 reg	7 + n	2	u x u x x x
	mem, CL	1 1 0 1 0 0 1 W	mod 1 0 0 mem	19/27 + n	2-4	u x u x x x
	reg, imm8	1 1 0 0 0 0 0 W	1 1 1 0 0 reg	7 + n	3	u x u x x x
	mem, imm8	1 1 0 0 0 0 0 W	mod 1 0 0 mem	19/27 + n	3-5	u x u x x x
SHR	reg, 1	1 1 0 1 0 0 0 W	1 1 1 0 1 reg	2	2	u x x x x x
	mem, 1	1 1 0 1 0 0 0 W	mod 1 0 1 mem	16/24	2-4	u x x x x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 1 0 1 reg	7 + n	2	u x u x x x
	mem, CL	1 1 0 1 0 0 1 W	mod 1 0 1 mem	19/27 + n	2-4	u x u x x x
	reg, imm8	1 1 0 0 0 0 0 W	1 1 1 0 1 reg	7 + n	3	u x u x x x
	mem, imm8	1 1 0 0 0 0 0 W	mod 1 0 1 mem	19/27 + n	3-5	u x u x x x

n = number of shifts

Instruction Set (cont)

Mnemonic	Operands	Opcode 7 6 5 4 3 2 1 0	Opcode 7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Shift/Rotate Instructions (cont)						
SHRA	reg, 1	1 1 0 1 0 0 0 W	1 1 1 1 1 reg	2	2	u x 0 x x x
	mem, 1	1 1 0 1 0 0 0 W	mod 1 1 1 mem	16/24	2-4	u x 0 x x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 1 1 1 reg	7 + n	2	u x u x x x
	mem, CL	1 1 0 1 0 0 1 W	mod 1 1 1 mem	19/27 + n	2-4	u x u x x x
	reg, imm8	1 1 0 0 0 0 0 W	1 1 1 1 1 reg	7 + n	3	u x u x x x
	mem, imm8	1 1 0 0 0 0 0 W	mod 1 1 1 mem	19/27 + n	3-5	u x u x x x
ROL	reg, 1	1 1 0 1 0 0 0 W	1 1 0 0 0 reg	2	2	x x
	mem, 1	1 1 0 1 0 0 0 W	mod 0 0 0 mem	16/24	2-4	x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 0 0 0 reg	7 + n	2	x u
	mem, CL	1 1 0 1 0 0 1 W	mod 0 0 0 mem	19/27 + n	2-4	x u
	reg, imm	1 1 0 0 0 0 0 W	1 1 0 0 0 reg	7 + n	3	x u
	mem, imm	1 1 0 0 0 0 0 W	mod 0 0 0 mem	19/27 + n	3-5	x u
ROR	reg, 1	1 1 0 1 0 0 0 W	1 1 0 0 1 reg	2	2	x u
	mem, 1	1 1 0 1 0 0 0 W	mod 0 0 1 mem	16/24	2-4	x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 0 0 1 reg	7 + n	2	x u
	mem, CL	1 1 0 1 0 0 1 W	mod 0 0 1 mem	19/27 + n	2-4	x u
	reg, imm8	1 1 0 0 0 0 0 W	1 1 0 0 1 reg	7 + n	3	x u
	mem, imm8	1 1 0 0 0 0 0 W	mod 0 0 1 mem	19/27 + n	3-5	x u
ROLC	reg, 1	1 1 0 1 0 0 0 W	1 1 0 1 0 reg	2	2	x x
	mem, 1	1 1 0 1 0 0 0 W	mod 0 1 0 mem	16/24	2-4	x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 0 1 0 reg	7 + n	2	x u
	mem, CL	1 1 0 1 0 0 1 W	mod 0 1 0 mem	19/27 + n	2-4	x u
	reg, imm8	1 1 0 0 0 0 0 W	1 1 0 1 0 reg	7 + n	3	x u
	mem, imm8	1 1 0 0 0 0 0 W	mod 0 1 0 mem	19/27 + n	3-5	x u
RORC	reg, 1	1 1 0 1 0 0 0 W	1 1 0 1 1 reg	2	2	x x
	mem, 1	1 1 0 1 0 0 0 W	mod 0 1 1 mem	16/24	2-4	x x
	reg, CL	1 1 0 1 0 0 1 W	1 1 0 1 1 reg	7 + n	2	x u
	mem, CL	1 1 0 1 0 0 1 W	mod 0 1 1 mem	19/27 + n	2-4	x u
	reg, imm8	1 1 0 0 0 0 0 W	1 1 0 1 1 reg	7 + n	3	x u
	mem, imm8	1 1 0 0 0 0 0 W	mod 0 1 1 mem	19/27 + n	3-5	x u

n = number of shifts

Stack Manipulation Instructions					
PUSH	mem16	1 1 1 1 1 1 1 1	mod 1 1 0 mem	26	2-4
	reg16	0 1 0 1 0 reg		12	1
	sr	0 0 0 sr 1 1 0		12	1
	PSW	1 0 0 1 1 1 0 0		12	1
	R	0 1 1 0 0 0 0 0		67	1
	imm	0 1 1 0 1 0 S 0		11/12	2-3

NEC

μPD70108 (V20)

Instruction Set (cont)

Mnemonic	Operands	Opcode	Clocks	Bytes	Flags
		7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC CY V P S Z
Stack Manipulation Instructions (cont)					
POP	mem16	1 0 0 0 1 1 1 1 / mod 0 0 0 mem	25	2-4	
	reg16	0 1 0 1 1 reg	12	1	
	sr	0 0 0 sr 1 1 1	12	1	
	PSW	1 0 0 1 1 1 0 1	12	1	R R R R R R
	R	0 1 1 0 0 0 0 1	75	1	
PREPARE	imm16 imm8	1 1 0 0 1 0 0 0 / =imm8 = 0 12 imm8 > 1 19 + 8 (imm8 − 1)	-	4	
DISPOSE		1 1 0 0 1 0 0 1	10	1	
Control Transfer Instructions					
CALL	near_proc	1 1 1 0 1 0 0 0	29	3	
	regptr	1 1 1 1 1 1 1 1 / 1 1 0 1 0 reg	16	1	
	memptr16	1 1 1 1 1 1 1 1 / mod 0 1 0 mem	31	2-4	
	far_proc	1 0 0 1 1 0 1 0	29	5	
	memptr32	1 1 1 1 1 1 1 1 / mod 0 1 1 mem	47	2-4	
RET		1 1 0 0 0 0 1 1	18	1	
	pop_value	1 1 0 0 0 0 1 0	24	3	
		1 1 0 0 1 0 1 1	28	1	
	pop_value	1 1 0 0 1 0 1 0	32	3	
BR	near_label	1 1 1 0 1 0 0 1	13	3	
	short_label	1 1 1 0 1 0 1 1	12	2	
	reg	1 1 1 1 1 1 1 1 / 1 1 1 0 0 reg	11	2	
	memptr16	1 1 1 1 1 1 1 1 / mod 1 0 0 mem	24	2-4	
	far_label	1 1 1 0 1 0 1 0	15	5	
	memptr32	1 1 1 1 1 1 1 1 / mod 1 0 1 mem	35	2-4	
BV	near_label	0 1 1 1 0 0 0 0	14/4	2	
BNV	near_label	0 1 1 1 0 0 0 1	14/4	2	
BC, BL	near_label	0 1 1 1 0 0 1 0	14/4	2	
BNC, BNL	near_label	0 1 1 1 0 0 1 1	14/4	2	
BE, BZ	near_label	0 1 1 1 0 1 0 0	14/4	2	
BNE, BNZ	near_label	0 1 1 1 0 1 0 1	14/4	2	
BNH	near_label	0 1 1 1 0 1 1 0	14/4	2	
BH	near_label	0 1 1 1 0 1 1 1	14/4	2	
BN	near_label	0 1 1 1 1 0 0 0	14/4	2	
BP	near_label	0 1 1 1 1 0 0 1	14/4	2	
BPE	near_label	0 1 1 1 1 0 1 0	14/4	2	
BPO	near_label	0 1 1 1 1 0 1 1	14/4	2	
BLT	near_label	0 1 1 1 1 1 0 0	14/4	2	
BGE	near_label	0 1 1 1 1 1 0 1	14/4	2	

Instruction Set (cont)

Mnemonic	Operand	Opcode 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	Clocks	Bytes	Flags AC CY V P S Z
Control Transfer Instructions (cont)						
BLT	near-label	0 1 1 1 1 1 1 0		14/4	2	
BGT	near-label	0 1 1 1 1 1 1 1		14/4	2	
DBNZNE	near-label	1 1 1 0 0 0 0 0		14/5	2	
DBNZE	near-label	1 1 1 0 0 0 0 1		14/5	2	
DBNZ	near-label	1 1 1 0 0 0 1 0		13/5	2	
BCWZ	near-label	1 1 1 0 0 0 1 1		13/5	2	
Interrupt Instructions						
BRK	3	1 1 0 0 1 1 0 0		50	1	
	imm8	1 1 0 0 1 1 0 1		50	2	
BRKV	imm8	1 1 0 0 1 1 1 0		52/3	1	
RETI		1 1 0 0 1 1 1 1		39	1	R R R R R R
CHKIND	reg16, mem32	0 1 1 0 0 0 1 0	mod reg mem	13-16/26	2-4	
BRKEM	imm8	0 0 0 0 1 1 1 1	1 1 1 1 1 1 1 1	50	3	
CPU Control Instructions						
HALT		1 1 1 1 0 1 0 0		2	1	
BUSLOCK		1 1 1 1 0 0 0 0		2	1	
FPO1	fp_op	1 1 0 1 1 X X X	1 Y Y Y Y Z Z Z	2	2	
	fp_op, mem	1 1 0 1 1 X X X	mod Y Y Y mem	15	2-4	
FPO2	fp_op	0 1 1 0 0 1 1 X	1 1 Y Y Y Z Z Z	2	2	
	fp_op, mem	0 1 1 0 0 1 1 X	mod Y Y Y mem	15	2-4	
POLL		1 0 0 1 1 0 1 1		7 + 5n	1	
		n = number of times POLL pin is sampled				
NOP		1 0 0 1 0 0 0 0		3	1	
DI		1 1 1 1 1 0 1 0		2	1	
EI		1 1 1 1 1 0 1 1		2	1	
8080 Instruction Set Enhancements						
RETEM		1 1 1 0 1 1 0 1	1 1 1 1 1 1 0 1	39	2	R R R R R R
CALLN	imm8	1 1 1 0 1 1 0 1	1 1 1 0 1 1 0 1	58	3	

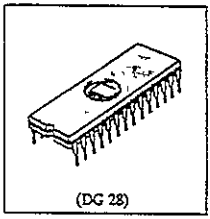
10.5 EPROM, RAM

10.5.1 27128

HN27128AG Series

16384-word X 8 bit UV Erasable and Programmable ROM

The HN27128AG is a 16384-word by 8 bit erasable and electrically programmable ROM. This device is packaged in a 28 pin dual in line package with transparent window. The transparent window allows the user to expose the chip to ultraviolet light to erase the bit pattern whereby a new pattern can then be written into the device.



(DG 28)

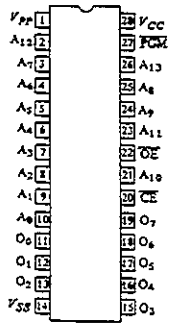
■ FEATURES

- Single Power Supply — +5V ±5%
- High Performance Programming — Program Voltage +12.5V D.C. High-Performance Programming Operations
- Static — No Clocks Required
- Inputs and Outputs TTL Compatible During Both Read and Program Modes
- Access Time — 170/200/250/300ns(max.)
- Absolute Max. Rating of Vpp pin — 14.0V Max
- Low Stand by Current — 35mA Max. (stand-by)
- Device Identifier Mode — Manufacturer Code and Device Code

■ ORDERING INFORMATION

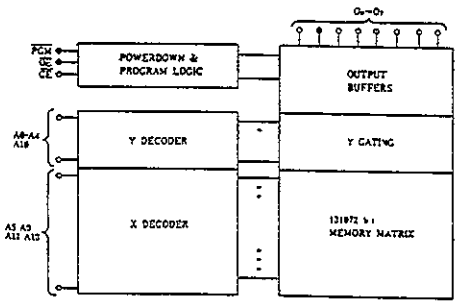
Type No.	Access Time	Package
HN27128AG-17	170ns	600 mil 28 pin Cerdip
HN27128AG-20	200ns	
HN27128AG-25	250ns	
HN27128AG-30	300ns	

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



10.5.2 84256

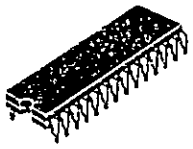
HM62256 Series

32768-word x 8-bit High Speed CMOS Static RAM

FEATURES

- High Speed: Fast Access Time 85/100/120/150ns (max.)
- Low Power Standby and Low Power Operation;
  Standby: 200μW (typ)/10μW (typ) (L-version),
  Operation: 40mW (typ.) (f = 1MHz)
- Single 5V Supply
- Completely Static RAM: No clock or Timing Strobe Required
- Equal Access and Cycle Time
- Common Data Input and Output, Three-state Output
- Directly TTL Compatible: All Input and Output
- Capability of Battery Back Up Operation (L-/L-SL version)

HM62256P Series



(DP-28)

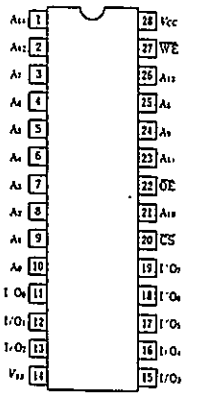
HM62256FP Series

(FP-28DA)

ORDERING INFORMATION

Type No.	Access Time	Package
HM62256P-8	85ns	600 mil 28 pin Plastic DIP
HM62256P-10	100ns	
HM62256P-12	120ns	
HM62256P-15	150ns	
HM62256LP-8	85ns	
HM62256LP-10	100ns	
HM62256LP-12	120ns	
HM62256LP-15	150ns	
HM62256LP-10SL	100ns	
HM62256LP-12SL	120ns	
HM62256LP-15SL	150ns	
HM62256FP-8T	85ns	28 pin Plastic SOP
HM62256FP-10T	100ns	
HM62256FP-12T	120ns	
HM62256FP-15T	150ns	
HM62256LFP-8T	85ns	
HM62256LFP-10T	100ns	
HM62256LFP-12T	120ns	
HM62256LFP-15T	150ns	
HM62256LFP-10SLT	100ns	
HM62256LFP-12SLT	120ns	
HM62256LFP-15SLT	150ns	

PIN ARRANGEMENT



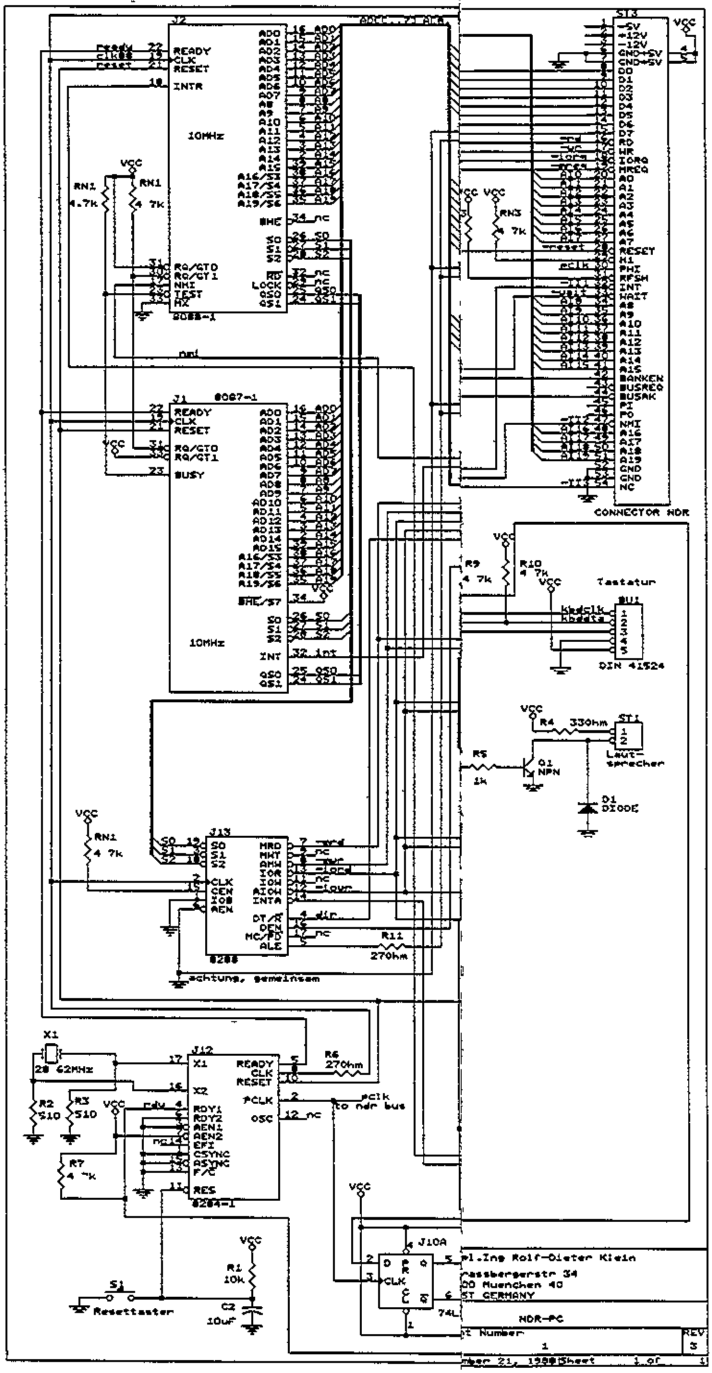
(Top View)

ABSOLUTE MAXIMUM RATINGS

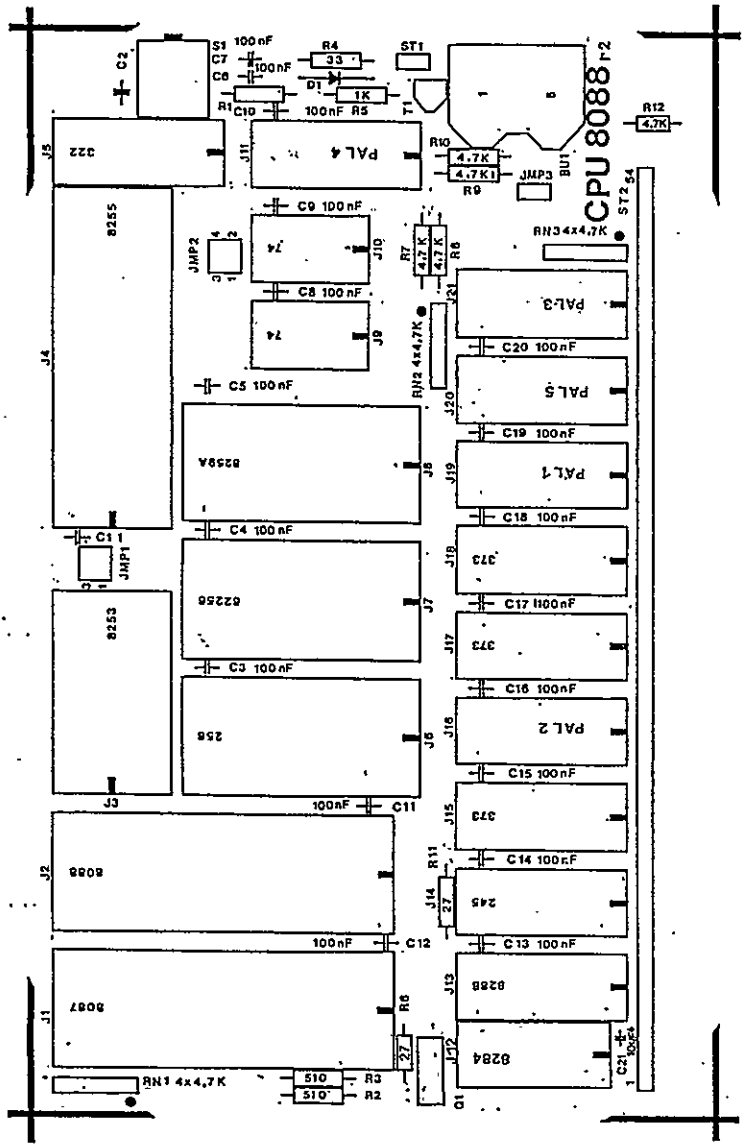
Item	Symbol	Rating	Unit
Voltage on any pin with relative to V_{SS}	V_T	−0.5 *1 to +7.0	V
Power Dissipation	P_T	1.0	W
Operating Temperature	T_{opr}	0 to +70	°C
Storage Temperature	T_{stg}	−55 to +125	°C
Temperature Under Bias	T_{bias}	−10 to +85	°C

Note) *1. −3.0V for pulse width ≦ 50ns

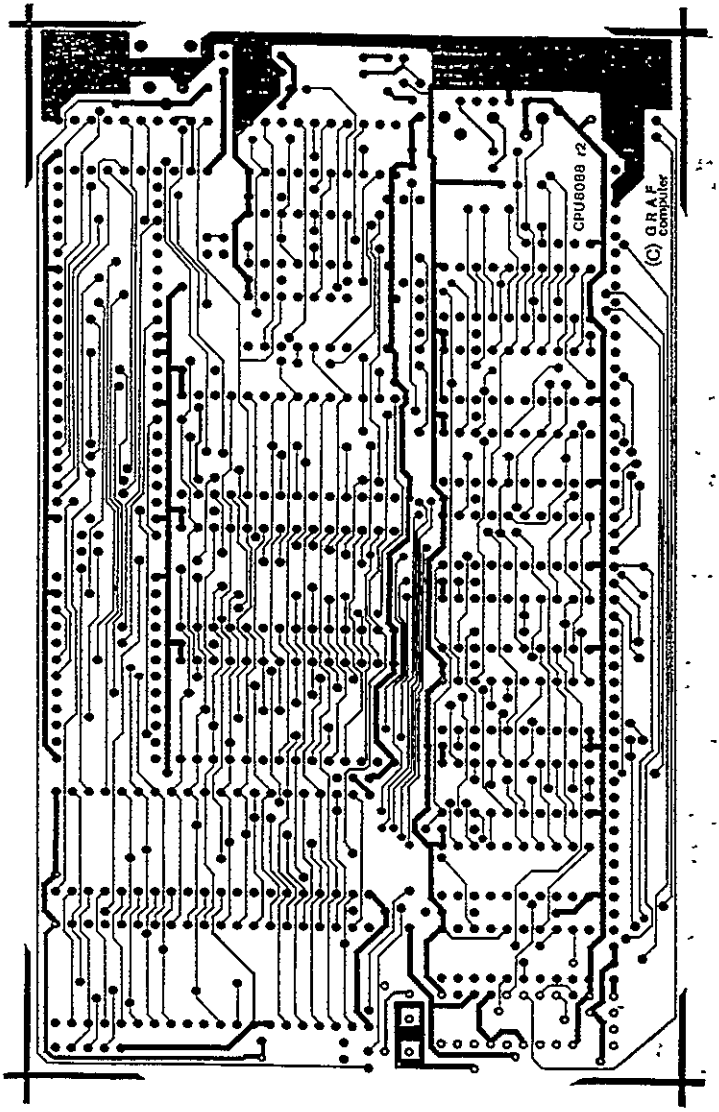
Anhang A: Schaltpla



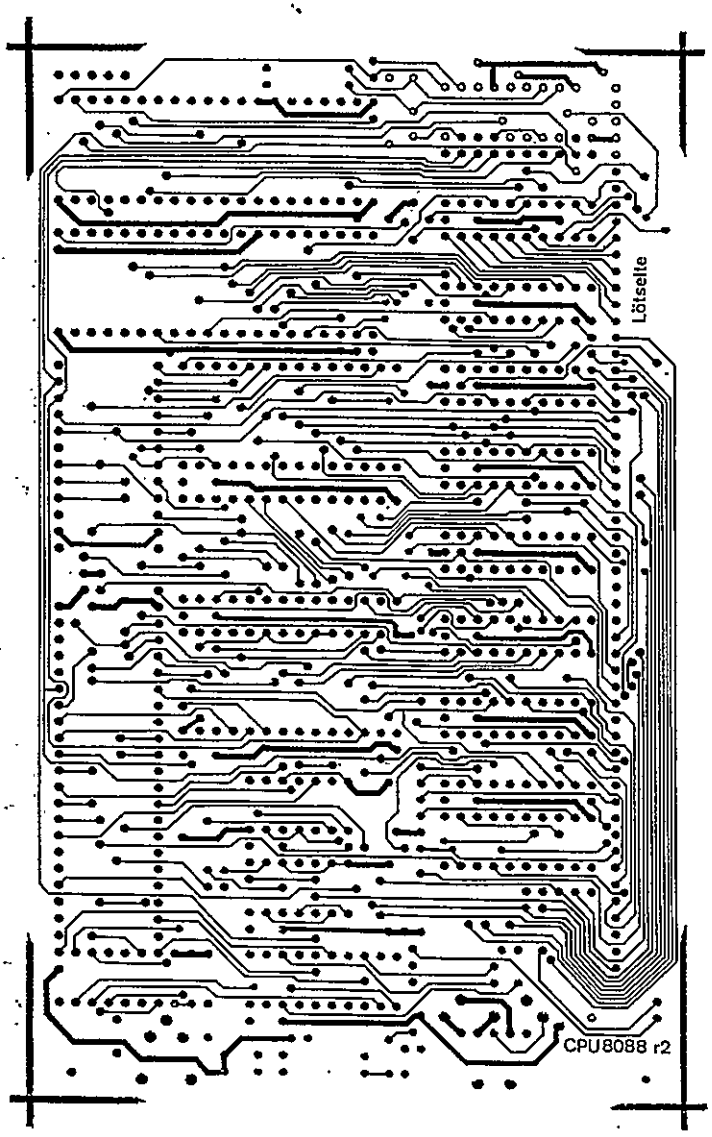
Anhang B: Layout Bestückungsseite mit Bestückungsaufdruck



Anhang C: Layout Bestückungsseite



Anhang D: Layout Lötseite



Belegung der NDR-ASCII Tastatur beim CPU8088 BIOS

Die NDR ASCII Tastatur ist bekanntlich total anders aufgebaut wie die IBM-PC Tastatur. Sie unterscheiden sich in der Art der Übertragung und in den Übertragungscodes. Der gravierendste Unterschied ist wohl, daß bei der IBM-PC Tastatur 8 Bit übertragen werden und bei ASCII Tastaturen nur 7 Bit. Aus diesem Grund muß bei der ASCII Tastatur eine Möglichkeit vorgesehen werden, auch das 8. Bit mit einzubeziehen. Dies wird durch einen Steuercode erledigt (hier CTRL "Klammeraffe"), der zwei Ebenen umschaltet. Diese beiden Ebenen sind in der folgenden Tabelle dargestellt. Nach dem Einschalten befindet man sich in der Standardebene, in der alle alphanumerischen Zeichen und die wichtigsten CTRL-Funktionen zugänglich sind. Mit CTRL "Klammeraffe" (SCAN Code 0) kann auf Zeichen in der höheren Ebene zugegriffen werden. Auf dieser Ebene liegen die PC-üblichen Steuertasten wie, BACKTAB, PRTSCR, NUMLOCK, PG UP, PG DOWN, usw, sowie die Funktionstasten und Kombinationen mit den Funktionstasten, z B SHIFT F1, ALT F1, CTRL F1, usw. Außerdem ist bei den PCs vorgesehen, daß mit Hilfe der ALT Funktion Tastencodes eingegeben werden können und es wird dann das entsprechende Zeichen dargestellt. Auch bei der NDR-ASCII Tastatur ist dies möglich, allerdings doch recht aufwendig. Der Syntax dafür sieht folgendermaßen aus:

CTRL "Klammeraffe", CTRL "Klammeraffe", Zeichencode, RETURN

z.B. "Back Slash"

CTRL "Klammeraffe", CTRL "Klammeraffe", 92, RETURN

Geben Sie diese Tastenkombinationen hintereinander ein, so erscheint an der Stelle an der der Cursor blinkt der "Back Slash".

In der folgenden Tabelle sind die beiden Ebenen dargestellt:

1. Standardebene

```
; ************************************************
;   general ASCII TO Scancode system using NDR KEY *
; ************************************************

tbascl  label byte      ; scan codes for ASCII as index
        db 0            ;  0 CTRL-@, used as  SPEZIAL FUNCTION CODE
        db 30           ;  1 CTRL-A
        db 48           ;  2 CTRL-B
        db 46           ;  3 CTRL-C
        db 32           ;  4 CTRL-D
        db 18           ;  5 CTRL-E
        db 33           ;  6 CTRL-F
        db 34           ;  7 CTRL-G
        db 14           ;  8 CTRL-H BTAB scan code
        db 15           ;  9 CTRL-I TAB scan code
        db 28           ;  a CTRL-J LF scan code
        db 37           ;  b CTRL-K
        db 38           ;  c CTRL-L
        db 28           ;  d CTRL-M CR scan code
        db 49           ;  e CTRL-N
        db 24           ;  f CTRL-O
        db 25           ; 10 CTRL-P
        db 16           ; 11 CTRL-Q
        db 19           ; 12 CTRL-R
        db 31           ; 13 CTRL-S
        db 20           ; 14 CTRL-T
        db 22           ; 15 CTRL-U
        db 47           ; 16 CTRL-V
        db 17           ; 17 CTRL-W
        db 45           ; 18 CTRL-X
        db 21           ; 19 CTRL-Y
        db 44           ; 1a CTRL-Z
        db 26           ; 1b CTRL-[
        db 43           ; 1c CTRL-\
        db 27           ; 1d CTRL-]
        db 7            ; 1e CTRL-^
        db 12           ; 1f CTRL-_
        db 57           ; 20 space
        db 2            ; 21 !
        db 40           ; 22 "
        db 4            ; 23 #
        db 5            ; 24 $
        db 6            ; 25 %
        db 8            ; 26 &
        db 40           ; 27 '
        db 10           ; 28 (
        db 11           ; 29 )
        db 9            ; 2A *
        db 13           ; 2B +
        db 51           ; 2C ,
        db 12           ; 2D -
        db 52           ; 2E .
        db 53           ; 2F /
        db 11           ; 30 0
        db 2            ; 31 1
        db 3            ; 32 2
        db 4            ; 33 3
        db 5            ; 34 4
        db 6            ; 35 5
        db 7            ; 36 6
        db 8            ; 37 7
        db 9            ; 38 8
        db 10           ; 39 9
        db 39           ; 3A :
        db 39           ; 3B ;
        db 51           ; 3C <
        db 13           ; 3D =
        db 52           ; 3E >
        db 53           ; 3F ?
        db 3            ; 40 @
        db 30           ; 41 A
        db 48           ; 42 B
        db 46           ; 43 C
        db 32           ; 44 D
        db 18           ; 45 E
        db 33           ; 46 F
        db 34           ; 47 G
        db 35           ; 48 H
        db 23           ; 49 I
        db 36           ; 4a J
        db 37           ; 4b K
        db 38           ; 4c L
        db 50           ; 4d M
        db 49           ; 4e N
        db 24           ; 4f O
        db 25           ; 50 P
        db 16           ; 51 Q
        db 19           ; 52 R
        db 31           ; 53 S
        db 20           ; 54 T
        db 22           ; 55 U
        db 47           ; 56 V
        db 17           ; 57 W
        db 45           ; 58 X
        db 21           ; 59 Y
        db 44           ; 5a Z
        db 26           ; 5b [
        db 43           ; 5c \
        db 27           ; 5d ]
        db 7            ; 5e ^
        db 12           ; 5f _
        db 41           ; 60 `
        db 30           ; 61 a
        db 48           ; 62 b
        db 46           ; 63 c
        db 32           ; 64 d
        db 18           ; 65 e
        db 33           ; 66 f
        db 34           ; 67 g
        db 35           ; 68 h
        db 23           ; 69 i
        db 36           ; 6a j
        db 37           ; 6b k
        db 38           ; 6c l
        db 50           ; 6d m
        db 49           ; 6e n
        db 24           ; 6f o
        db 25           ; 70 p
        db 16           ; 71 q
        db 19           ; 72 r
        db 31           ; 73 s
        db 20           ; 74 t
        db 22           ; 75 u
        db 47           ; 76 v
        db 17           ; 77 w
        db 45           ; 78 x
        db 21           ; 79 y
        db 44           ; 7a z
        db 26           ; 7b {
        db 43           ; 7c |
        db 27           ; 7d }
        db 41           ; 7e ~
        db 14           ; 7f del
```

2. Zweite Ebene (vor jedem Zeichen CTRL "Klammeraffe" drücken)

```
tbasc2  label byte      ; scan codes for ASCII with ALT KEY as index
        db 0            ;  0 CTRL-@, used as  SPEZIAL FUNCTION CODE
        db 15           ;  1 CTRL-A BACKTAB
        db 55           ;  2 CTRL-B PRTSCR          db 127          ; 38 8
        db 69           ;  3 CTRL-C NUMLOCK         db 128          ; 39 9
        db 70           ;  4 CTRL-D SCROLL          db 0            ; 3A :
        db 73           ;  5 CTRL-E PG UP           db 0            ; 3B ,
        db 81           ;  6 CTRL-F PG DOWN         db 111          ; 3C < ALT-F8
        db 83           ;  7 CTRL-G DEL             db 131          ; 3D =
        db 75           ;  8 CTRL-H LEFT            db 112          ; 3E > ALT-F9
        db 77           ;  9 CTRL-I RIGHT           db 113          ; 3F ? ALT-F10
        db 80           ;  a CTRL-J DOWN            db 132          ; 40 @ shifted codes used for shift-FUNKTION
        db 72           ;  b CTRL-K UP              db 94           ; 41 A CTRL-F1
        db 79           ;  c CTRL-L END             db 108          ; 42 B ALT-F5
        db 74           ;  d CTRL-M ALT RIGHT       db 106          ; 43 C ALT-F3
        db 76           ;  e CTRL-N ALT PGDOWN      db 96           ; 44 D CTRL-F3
        db 71           ;  f CTRL-O HOME            db 86           ; 45 E SHIFT-F3
        db 78           ; 10 CTRL-P GREY +          db 97           ; 46 F CTRL-F4
        db 84           ; 11 CTRL-Q ALT PGUP        db 98           ; 47 G CTRL-F5
        db 0            ; 12 CTRL-R                 db 99           ; 48 H CTRL-F6
        db 0            ; 13 CTRL-S                 db 91           ; 49 I SHIFT-F8
        db 0            ; 14 CTRL-T                 db 100          ; 4a J CTRL-F7
        db 0            ; 15 CTRL-U                 db 101          ; 4b K CTRL-F8
        db 82           ; 16 CTRL-V INS             db 102          ; 4c L CTRL-F9
        db 0            ; 17 CTRL-W                 db 110          ; 4d M ALT-F7
        db 0            ; 18 CTRL-X                 db 109          ; 4e N ALT-F6
        db 0            ; 19 CTRL-Y                 db 92           ; 4f O SHIFT-F9
        db 0            ; 1a CTRL-Z                 db 93           ; 50 P SHIFT-F10
        db 68           ; 1b CTRL-[   - F10         db 84           ; 51 Q SHIFT-F1
        db 0            ; 1c CTRL-\                 db 87           ; 52 R SHIFT-F4
        db 0            ; 1d CTRL-]                 db 95           ; 53 S CTRL-F2
        db 0            ; 1e CTRL-^                 db 88           ; 54 T SHIFT-F5
        db 0            ; 1f CTRL-_                 db 90           ; 55 U SHIFT-F7
        db 57           ; 20 space  ++++            db 107          ; 56 V ALT-F4
        db 59           ; 21 ! F1                   db 85           ; 57 W SHIFT-F2
        db 60           ; 22 " F2                   db 105          ; 58 X ALT-F2
        db 61           ; 23 # F3                   db 104          ; 59 Y ALT-F1 (deutsch)
        db 62           ; 24 $ F4                   db 89           ; 5a Z SHIFT-F6 (deutsch)
        db 63           ; 25 % F5                   db 0            ; 5b [
        db 64           ; 26 & F6                   db 103          ; 5c \ CTRL-F10  db 50          ; 6d m
        db 65           ; 27 ' F7                   db 27           ; 5d ]           db 49          ; 6e n
        db 66           ; 28 ( F8                   db 131          ; 5e ^           db 24          ; 6f o
        db 67           ; 29 ) F9                   db 12           ; 5f             db 25          ; 70 p
        db 68           ; 2A * (F10 also)           db 41           ; 60 `           db 16          ; 71 q
        db 131          ; 2B +                      db 30           ; 61 a           db 19          ; 72 r
        db 0            ; 2C ,                      db 48           ; 62 b           db 31          ; 73 s
        db 130          ; 2D -                      db 46           ; 63 c           db 20          ; 74 t
        db 0            ; 2E .                      db 32           ; 64 d           db 22          ; 75 u
        db 0            ; 2F /                      db 18           ; 65 e           db 47          ; 76 v
        db 129          ; 30 0                      db 33           ; 66 f           db 17          ; 77 w
        db 120          ; 31 1                      db 34           ; 67 g           db 45          ; 78 x
        db 121          ; 32 2                      db 35           ; 68 h           db 21          ; 79 y
        db 122          ; 33 3                      db 23           ; 69 i           db 44          ; 7a z
        db 123          ; 34 4                      db 36           ; 6a j           db 0           ; 7b {
        db 124          ; 35 5                      db 37           ; 6b k           db 0           ; 7c |
        db 125          ; 36 6                      db 38           ; 6c l           db 0           ; 7d }
        db 126          ; 37 7                                                       db 0           ; 7e ~
                                                                                     db 83          ; 7f del ok
```

Erläuterungen zum BIOS V1.3

Das BIOS V1.3 unterstützt bereits die meisten der im Handbuch beschriebenen Spezifikationen. In der folgenden Tabelle sehen Sie die bereits unterstützten Hardware Komponenten:

- CPU8088
- NDR Speicherbaugruppen (ROA64, RAM64/256, ROA256/1M)
- NDR Graphik (GDP64k)
- NDR ASCII Tastatur mit KEY
- NDR FLO3 mit Standard TEAC Laufwerken FD55F..
- NDR SER und CENT

Parallel zu den oben genannten Graphik und Parallel- und Serielladapter sind über die BUSKOPP-Baugruppe auch die PC Graphikkarten und Seriell- und Paralleladapter einsetzbar.

- PC-Graphik über BUSKOPP einsetzbar (MGA, Hercules, CGA, EGA und VGA)
- PC-parallele und serielle Schnittstelle

Von denen im Handbuch angegebenen Spezifikationen sind nur noch wenige Funktionen noch nicht erfüllt.

1. Die angesprochene Festplatte mit dem OMTI Controller ist noch nicht implementiert

2. Die Smart Watch zum automatischen Laden der Uhrzeit nach dem "Power ON" ist ebenfalls noch nicht realisiert.

3. Für den NDR-Computer mit der FLO3 existiert noch keine Formatierroutine für Disketten. Diese Routine wir später Diskette erhältlich sein.

Achtung!

Im Handbuch auf den Seiten 27 und 28 wird die Jumperstellung der FLO3 bei Verwendung der EGA-Karte beschrieben. Bei der jetzt ausgelieferten BIOS Version 1.3 muß die FLO3 bzw. FLO2 generell auf die Adresse 0F0H umgestellt werden (siehe Handbuch). Die SER des NDR-Computers, die normalerweise auf dieser Adresse liegt, muß, wenn Sie verwendet wird, auf die Adresse 0F8h gelegt werden.

Wir wünschen Ihnen viel Spaß mit dem NDR-PC

Ihr GRAF Computer Team

Ergänzung zum Handbuch CPU8088

BIOS 1.5

Sie erhalten hiermit die BIOS-Version 1.5 für die CPU 8088. Diese neue Version ermöglicht nun neben einigen anderen Verbesserungen u. a. auch den Betrieb einer Festplatte.

Das Formatierprogramm UFORM

Auf der mitgelieferten Diskette befindet sich das Formatierprogramm UFORM.EXE. Es ermöglicht Ihnen, auf dem NDR-Computer Disketten zu formatieren, die auch von anderen PC's gelesen werden können.
Das Programm bietet Ihnen drei Möglichkeiten zum Formatieren: Sie können auf einem 40-Spur-Laufwerk 40-Spur-Disketten formatieren. Weiterhin können auf einem 80-Spur-Laufwerk sowohl 40-Spur- als auch 80-Spur-Disketten formatiert werden. Eine Überprüfung des Formatiervorgangs ist ebenfalls in das Programm integriert.
Die Formatier-Routine arbeitet mit der FLO-2 und der FLO-3 zusammen.
Auf der FLO-3 müssen an JMP2 die Anzahl der WAIT-States auf '3' eingestellt sein. Dies ist der Standardwert, so daß normalerweise keine Neueinstellung notwendig ist. Bei Unklarheiten können Sie die genaue Lage und Einstellung von JMP2 dem FLO-3 Handbuch entnehmen.

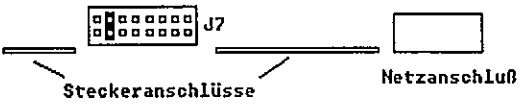
Anschluß einer Harddisk

Mit Hilfe der neu entwickelten Buskopplung kann u. a. auch ein Festplatten-Controller am System betrieben werden. Dadurch haben nun auch Sie als NDR-Anwender Zugang zu einem schnellen und ausreichend dimensionierten Speichermedium.
Momentan bieten wir zwei Festplatten an, die am NDR-Computer betrieben werden können.

ST 251 mit 820 Zylindern, 6 Kopfen, 40MB
ST 225 mit 615 Zylindern, 4 Kopfen, 20MB

Beachten Sie bei der Inbetriebnahme der Festplatte, daß die Verbindungskabel richtig angeschlossen werden. Sie benötigen ein 20-adriges unverdrilltes und ein 34-adriges verdrilltes Kabel.

Zur Orientierung sind auf den Leiterplatten der Harddisk und des Controllers die Pinnummern neben den Steckern angegeben.
Das Drive-Select muß an dem Jumper J7 auf 2 eingestellt werden. Dieser Jumper befindet sich an der Rückseite der Harddisk zwischen den beiden Steckeranschlüssen.



Skizze: Einstellung von J7
(Rückansicht der Harddisk)

Mit dem Programm HDFORMAT, das sich auf der mitgelieferten Diskette befindet, werden Festplatten von 20MB und 40MB Kapazität an einem Omti-Controller formatiert.

Formatieren einer neuen Platte:

1. Jumpereinstellung für Harddisk auf Omti-5520B-Controller:

W1	W2	W3	W4	
ON	ON	ON	ON	für ST 251, 40MB
OFF	ON	OFF	ON	für ST 225, 20MB

2. HDFORMAT aufrufen.

Die verwendete Platte wird anhand der Jumper automatisch eingestellt. Das Programm fragt zusätzlich, ob defekte Sektoren auf der Platte vorhanden sind. Wenn dies der Fall ist, sind die defekten Sektoren meist an einem Aufkleber auf der Festplatte angegeben. Diese können Sie dem Programm mitteilen, wodurch der Formatiervorgang schneller abläuft.
HDFORMAT führt die Low-Level-Formatierung durch. Das Hilfsprogramm READ.EXE dient zum Test und zum Einkreisen von Fehlern und wird normalerweise nicht benötigt.

3. Festplatte partitionieren

- Legen Sie Ihre DOS-Diskette ein und rufen Sie FDISK C: auf. Daraufhin erscheint das Hauptmenü von FDISK auf dem Bildschirm.
- Wählen Sie die Option (1) 'Erstellen einer DOS-Partition'! Sie befinden sich nun in einem Untermenü.
- Zur Erstellung einer primären DOS-Partition betätigen Sie die RETURN-Taste, um die Standard-Auswahl (1) zu akzeptieren.
- Anschließend geben Sie 'J' ein, um die ganze Festplatte (maximal 32MB) für die DOS-Partition zu verwenden.

- Das System wird jetzt neu gestartet. Dazu muß sich ihre System-Diskette in Laufwerk A: befinden.

4. Formatieren

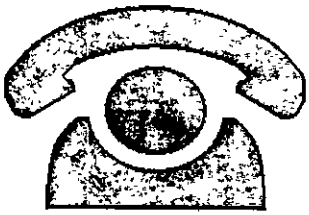
Rufen Sie FORMAT C: /S auf (Das Programm befindet sich auf Ihrer Systemdiskette). Die Festplatte wird jetzt unter DOS formatiert, wobei anschließend noch das System mitkopiert wird.

5. Einrichten eines erweiterten DOS-Speicherbereichs

- Wenn Ihre Festplatte uber mehr als 32MB verfugt, oder wenn Sie mehrere logische Laufwerke fur die Festplatte bestimmen möchten, können Sie mit FDISK einen erweiterten Speicherbereich einrichten
Dazu wahlen Sie im Untermenu von FDISK die Option (2) 'Erstellen einer erweiterten DOS-Partition'
- Das Menu zeigt die Gesamtzylinderzahl an, die für einen erweiterten Speicherbereich verfügbar ist. Der Standardwert für die Speicherbereichsgroße ist der auf der Festplatte maximal verfügbare Speicherplatz Betatigen Sie die RETURN-Taste zur Bestatigung des Standardwertes oder geben Sie die gewunschte Große in Zylindern an.
- Nach dem Einrichten des erweiterten Speicherbereichs mussen Sie ein logisches Laufwerk für den Bereich festlegen Nach Auswahl von Option (3) aus dem Menü 'Erstellen einer DOS-Partition' können Sie die gewunschte Partitionsgröße (Zylinderanzahl) für das logische Laufwerk D: angeben Sie können hier die gesamte verbleibende Kapazität angeben; es ist aber auch möglich, den erweiterten Speicherbereich auf mehrere logische Laufwerke zu verteilen. Dazu fordert Sie FDISK solange auf, die gewünschte Partitionsgröße für das aktuelle Laufwerk anzugeben, bis der gesamte Speicherbereich einem logischen Laufwerk zugeteilt wurde

6. Booten von der Platte

Wenn keine Diskette in Laufwerk A: eingelegt ist, wird ab sofort automatisch von der Platte gebootet. Dazu muß sich, wie oben beschrieben, auf der aktiven DOS-Partition ein System befinden (FORMAT C:/S) Achtung: Beim Hochlaufen des Systems kann dies mit angeschlossener Festplatte lange dauern (1/2 min od ähnl)



Telefonservice
08 31-6211
jeden Mittwochabend
bis 20.00 Uhr

Graf Elektronik Systeme GmbH
Magnusstraße 13 Postfach 1610
8960 Kempten (Allgäu)
Telefon: (0831) 6211
Teletex: 831804 = GRAF
Telex : 17 831804 = GRAF
Datentelefon: (0831) 6 93 30

Geschäftszeiten: GES GmbH+Verkauf
Mo.-Do. 8.00-12.00 Uhr, 13.00 -17.00 Uhr
Freitag 8.00-12.00 Uhr
Telefonservice Mo.- Do. 14.00 -17.00 Uhr
zusätzlich Mittwoch bis 20.00 Uhr

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (040) 38 81 51
Öffnungszeiten:
Mo. - Frei. 16.00-18.00 Uhr

Filiale München
Georgenstraße 61
8000 München 40
Telefon: (089) 2 71 58 58
Öffnungszeiten
Montag - Freitag
10.00 - 12.00 Uhr, 13.00 - 18.00 Uhr
Samstag 10.00 - 14.00 Uhr