CPU68020

Die Zentraleinheit mit dem 32-Bit Mikroprozessor 68020 für den NDR-Computer

Ausgabe 3

Graf Elektronik Systeme GmbH

Inhalt

1. Einführung

1.1 Zum NDR-Computer

Der NDR-Computer wird in der Fernsehserie "Mikroelektronik - Mikrocomputer selbstgebaut und programmiert" und in der neu überarbeiteten Fernsehserie "Computer Modular - Schritt für Schritt" aufgebaut, erklärt und in Betrieb genommen. Diese Serie wird vom Norddeutschen Rundfunk, vom Sender Freies Berlin, vom Bayerischen Fernsehen und von Radio Bremen ausgestrahlt. Es werden bald auch die Regionalsender anderer Bundesländer die Sendung in ihr Programm aufnehmen.

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

- *Buch:* *Rolf-Dieter Klein,*
  "Mikrocomputer selbstgebaut und programmiert"
  2., neu bearbeitete und erweiterte Auflage
  ISBN 3-7723-7162-0, DM 38,-
  erschienen im Franzis-Verlag, München
  Bestellnummer: 10078
  Auf diesem Buch baut die NDR-Serie auf

- *Buch:* *Rolf-Dieter Klein,*
  "Die Prozessoren 68000 und 68008"
  ISBN 3-7723-7651-7, DM 78,-
  erschienen im Franzis-Verlag, München
  Bestellnummer: 10588

- *Sonderhefte der "mc"*
  "Mikrocomputer Schritt für Schritt"
  Bestellnummer: 10399
  "Mikrocomputer Schritt für Schritt Teil 2"
  Bestellnummer: 10398

- *Zeitschriften "mc" und "ELO" des Franzis-Verlages*

- *Zeitschrift "LOOP" der Firma Graf (siehe Kapitel 7.1)*

- *Videocassetten:*
  lizensierte Originalcassetten für den privaten Gebrauch. Auf diesen zwei Cassetten sind die 26 Folgen der Fernsehserie enthalten.
  Systeme: VHS, Beta, Video 2000
  Preise: siehe gültige Preisliste

Inhalt

1.2 Die Baugruppe CPU68020

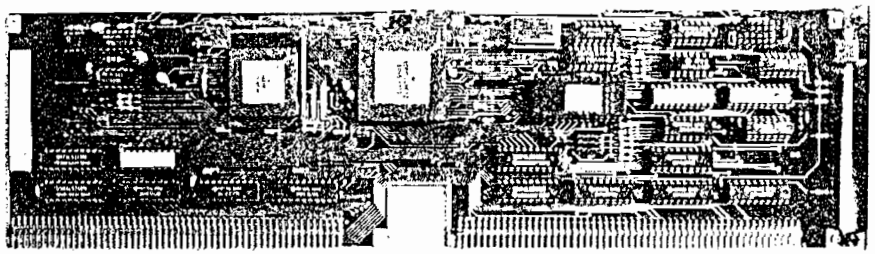


Bild 1: *Die Baugruppe CPU68020*

Der 68020 ist der zur Zeit interessanteste 32-Bit-Mikroprozessor auf dem Markt, denn durch seine Aufwärtskompatibilität zum 68000 laufen die meisten für den 68000 und 68008 geschriebenen Programme auch auf dem 68020. Durch den internen und externen 32-Bit Datenbus, neue leistungsstarke Befehle und einen kleinen superschnellen internen Speicher (Cache) ist er aber bis zu zehn mal so schnell wie der 68000. Auch durch erhöhte Taktfrequenzen (z.Zt. bis 20MHz) ist eine wesentlich höhere Arbeitsgeschwindigkeit erzielbar.

Daß auch eine 32-Bit CPU im NDR-Computer eingesetzt werden kann, ist seinem modularen Konzept zu verdanken: Die 32 Bit Datenleitungen werden einfach aufgeteilt in vier Teile mit jeweils 8 Bit. Alle bisher für den NDR-Computer erhältlichen Speicher- und Peripheriekarten benutzen 8 Datenleitungen. Somit sind diese weiterhin verwendbar!

Für eine Speichererweiterung müssen allerdings immer vier neue Karten auf einmal eingesetzt werden, damit die CPU auch wirklich vier mal 8 Datenbits bearbeiten kann.

Peripheriekarten brauchen wie bisher nur einmal vorhanden zu sein - die CPU kann hier auch 8 Bit Daten bearbeiten.

Man benötigt zu einem lauffähigen minimalen System zwei Busbaugruppen, die nebeneinander angeordnet werden, empfehlenswert sind hier ein BUS3 und ein BUS4 (BUS3a und BUS4a); man sollte hier schon an eine spätere Erweiterung denken (auch mit BUS2a und BUS3a kann schon ein lauffähiges Minimalsystem in Betrieb genommen werden). Beide müssen in der Mitte doppelte Buchsenleisten (wie für den 68000) haben, wobei die Leitungen D0-D7, sowie -IORQ, -MREQ, -RD und -WR getrennt werden müssen. Ferner braucht man vier ROA64 mit dem neuen Grundprogramm (in 8 EPROMs) und vier RAM8 (6264). BANKBOOT-Baugruppen werden nicht benötigt, da die komplette BANKBOOT-Logik incl. BOOT-EPROM auf der CPU-Karte integriert ist. Nur noch eine KEY und eine GDP64K sind nötig, um mit dem Grundprogramm wie beim 68008 in Assembler programmieren zu können.

Ein Sockel für den 68881 ist vorgesehen, er braucht also nur noch eingesteckt zu werden, damit man die neuen Befehle nutzen kann. Da der 68881 ein Mathematik-Coprozessor ist, und der 68020 so mit dem 68881 zusammenarbeitet, als wären sie ein (!) Chip, kann man die neuen Möglichkeiten wie ganz normale Assemblerbefehle nutzen. Es stehen dann neue Befehle zur Verfügung, um Integer- und Floating-Point-Zahlen, z.B. '7.6' oder '1.2E002' mit bis zu 80 Bit Genauigkeit zu bearbeiten, z.B. ADD, SUB, MUL, DIV, SIN, COS, TAN, LOG, Wurzel und vieles mehr. Diese Berechnungen werden mit sehr hoher Genauigkeit extrem schnell durchgeführt - gegenüber Softwarelösungen sind Steigerungen über 2000% möglich! Der 68881 erfüllt den durch IEEE (P754) gesetzten Standard.

Das Grundprogramm für den 68020 nennt sich EG68020. Dieses Programm wird auf acht EPROMs geliefert. Auf jede der vier dazu nötigen ROA64 werden 2 EPROMs eingesetzt (siehe Kapitel 3). Es stehen weiterhin alle Möglichkeiten des EASS0-3 (für 68008 und 68000) zur Verfügung, allerdings wurden einige Erweiterungen zur Unterstützung der FPU 68881 (neue Assemblerbefehle) vorgenommen. Es stehen wesentlich umfangreichere Debug-Möglichkeiten (Einzelschritt usw.) zur Verfügung. Es ist möglich, CP/M68K zu booten, auch JADOS ist lauffähig.

2. Technische Daten

Spannung:	+5 V
Stromaufnahme:	ca. 850 mA
CPU:	68020, 32-Bit-Mikroprozessor
Taktfrequenz CPU:	12 MHz, beliebig wählbar durch Austausch des Q-Osz.
FPU:	68881, Floating-Point Coprozessor, Standardisierte Datenformate, acht 80-bit-Floating-Point-Register
Taktfrequenz FPU:	12 MHz standard wie CPU, beliebig wählbar durch Einbau eines Q-Osz.
Ansprechbarer Speicher:	4 MByte (RAM und EPROM)
I/O-Bereich:	4 MByte, davon können die I/O-Baugruppen des NDR-Computers jedoch nur vier mal 256 Byte ausdekodieren (nur 256 Byte werden genutzt)
Leiterplattengröße:	100 mm * 345 mm
Busformat:	NDR-Bus 216-polig (Spezial-Ausführung für 32 Bit)

3. Inbetriebnahme

Wenn Sie ein Fertiggerät besitzen, dann lesen Sie bitte bei Kapitel 3.2 weiter.

3.1 Zusammenstellung der Komponenten

Zum Test der Baugruppe benötigen Sie ein Minimalsystem. Dies besteht aus folgenden Komponenten:

- *1 * Baugruppe CPU68020*
  alle Jumper default-eingestellt laut Kapitel 5
- *2 * BUS3 (oder 1 * BUS3 und 1 * BUS4)*
  wir empfehlen einen BUS3a und einen BUS4a, es funktionieren auch ein BUS2a mit einem BUS3a oder einem BUS4a, wenn Sie nur zwei Speicherbaugruppen pro Busviertel benötigen
- *4 * ROA64 (auf Adresse $00000, alle Jumper eingesetzt)*
- *4 * R8 (8K * 8 statisches RAM)*
- *1 * EG68020 (8 EPROMs 2764)*
- *1 * KEY (mit ASCII-Tastatur)*
- *1 * GDP64K (mit Monitor)*

Diese Einzelteile werden wie folgt zusammengesetzt:

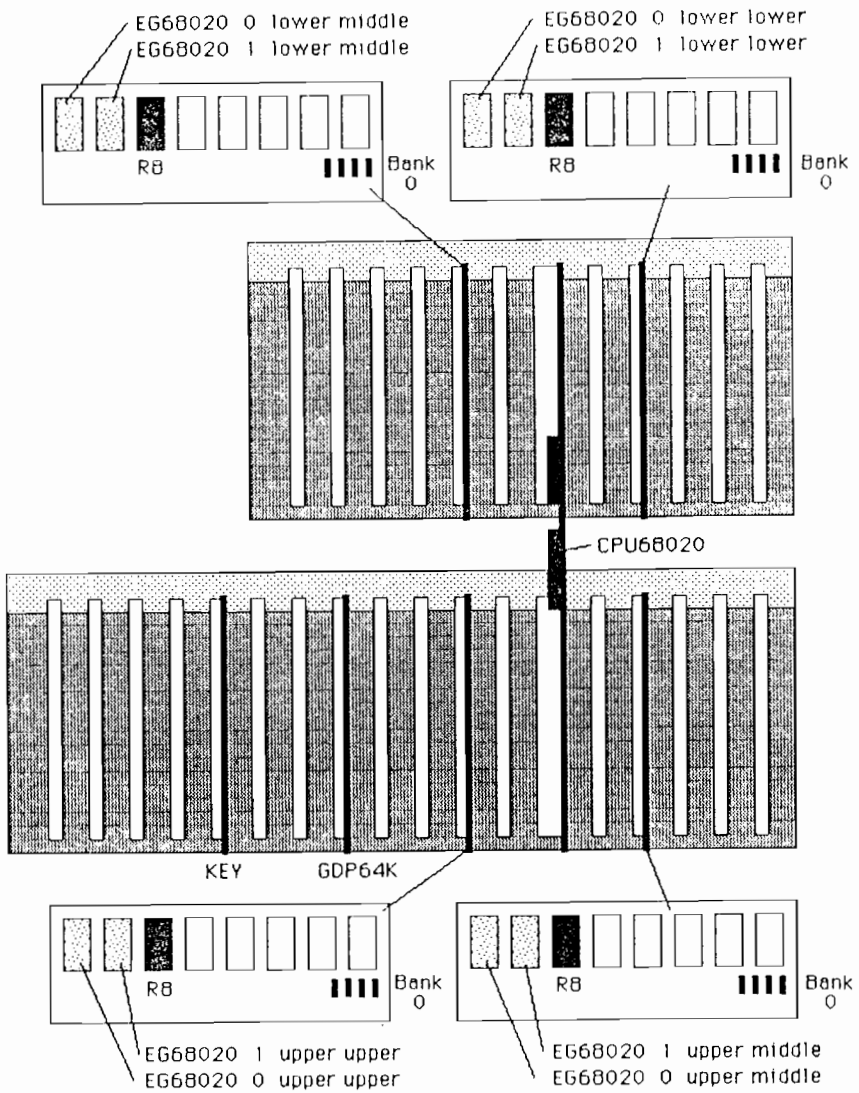


Bild 2: *Bus und Speicherkarten einer Minimalkonfiguration*

3.2 Inbetriebnahme eines Fertigcomputers

3.2.1 Anschließen des Computers an das Stromnetz

Wenn Sie Ihren Computer ausgepackt haben, dann stellen Sie ihn bitte an den vorgesehenen, geeigneten Platz. Unter einem geeigneten Platz stellen wir uns vor:

- *erschütterungsfrei*
- *vor direkter Sonneneinstrahlung geschützt*
- *nicht in der Nähe von Öfen bzw. Heizungen*
- *gute Luftzirkulation*
- *vor Feuchtigkeit geschützt*

Das beigefügte Netzkabel ist hinten am Gehäuse in die Buchse unter dem Netzschalter zu stecken und mit einer 220V-Steckdose zu verbinden.

3.2.2 Anschließen des Monitors

Der Computer wird mit dem Monitor durch ein sog. BAS-Kabel verbunden. Zu diesem Zweck ist das Kabel in die Cinch-Buchse im Monitor und in die Cinch-Buchse im Gehäuse Ihres Computers zu stecken. Der Monitor kann nun eingeschaltet werden.

3.2.3 Anschließen der Tastatur

Den 15-pol. Stecker der Tastatur stecken Sie bitte in die dafür vorgesehene Buchse am Gehäuse. Damit ist die Tastatur betriebsbereit.

3.3 Inbetriebnahme des Computers

Betätigen Sie den Netzschalter. Nach dem Einschalten des Gerätes sollte sich normalerweise sofort das Grundprogramm melden. Bleibt der Bildschirm auch nach einigen Sekunden und Drücken des RESET-Tasters noch dunkel, so sollten Sie folgendes überprüfen:

- *Ist das Netzkabel angeschlossen?*
- *Ist der Netzschalter eingeschaltet?*
- *Ist der Monitor angeschaltet?*
- *Ist das Kabel zum Monitor in Ordnung und eingesteckt?*
- *Sind Helligkeit und Kontrast am Monitor richtig eingestellt?*
- *Liegt die 5V-Spannungsversorgung am Bus an?*
- *Sitzen alle Baugruppen fest im Bus?*
- *Sind alle Baugruppen an den richtigen Stellen eingesetzt?*
- *Sind die Grundprogramm-EPROMs richtig gesteckt?*
- *Sind die RAMs direkt hinter dem Grundprogramm richtig gesteckt?*
- *Sind die Shuntstecker auf allen Baugruppen richtig gesteckt?*

Sind Sie ganz sicher, daß Sie alle Möglichkeiten überprüft haben, und fest davon überzeugt, daß die Baugruppe CPU68020 defekt ist, so bitten wir Sie, die Baugruppe sorgfältig verpackt mit möglichst viel Information über Ihr System und Ihre durchgeführten Tests an uns einzuschicken.

4. Schaltungsbeschreibung

Die Schaltungsbeschreibung gliedert sich in Beschreibungen für die verschiedenen Teilschaltungen, die einzeln besprochen werden, ein Gesamtschaltplan ist im Anhang A dieses Handbuchs zu finden.

Der Bus auf der CPU68020 Baugruppe ist wie folgt belegt:

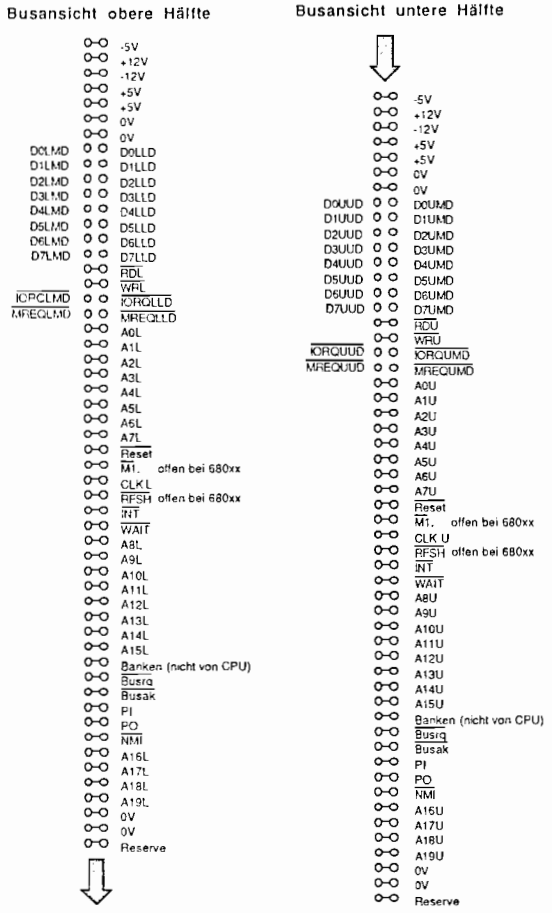


Bild 3: *Busbelegung auf der Baugruppe CPU68020*

4.1 Die Taktlogik

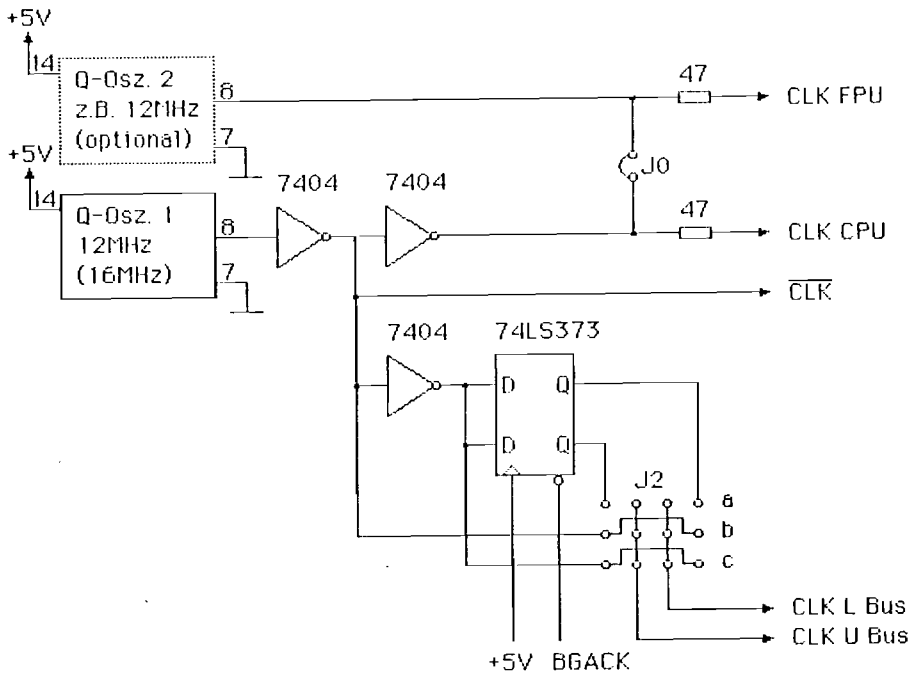


Bild 4: *Taktlogik*

Der Quarzoszillator Q-Osz. 1 liefert der CPU einen Takt von 12 MHz. Alle Schaltungsteile wurden auch bereits mit 16 MHz getestet; hat man also schnelle 68020-Chips, kann man für Q-Osz. 1 auch einen 16 MHz-Quarzoszillator einsetzen. Der Ausgang des Quarzoszillators wird über Nicht-Glieder gepuffert. Die CPU und die FPU werden über je einen 47 Ohm Widerstand mit dem Takt verbunden. Auf der Bestückungsseite der CPU68020 mitten über dem 68881 ist der Jumper J0. Entfernt man hier den Shunt-Stecker und setzt bei Q-Osz. 2 einen Quarzoszillator ein, so kann man den 68881 auch mit einem anderen Takt als den 68020 betreiben.

Der Bus wird über eigene Verstärker mit Takt versorgt. J2 bietet hierbei besondere Einstellmöglichkeiten: zum einen kann ausgewählt werden zwischen dem normalen und dem negierten Takt, zum anderen kann man auch einen nicht-negierten Takt für DMA-Zugriff in den Tri-State-Zustand setzen. Dazu dient das Latch 74LS373. Die Tri-State-Ausgänge werden vom BGACK-Signal gesteuert, das auch andere 74LS373-ICs steuert.

Wenn möglich, sollte man J2 offen lassen, denn alle neuen Baugruppen des NDR-Klein-Computers benötigen diesen Takt nicht. Ein Takt von 16 MHz "belastet" den Bus schon relativ stark (Übersprechen, Verunreinigung der anderen Bussignale usw.).

4.2 Die RESET-Logik

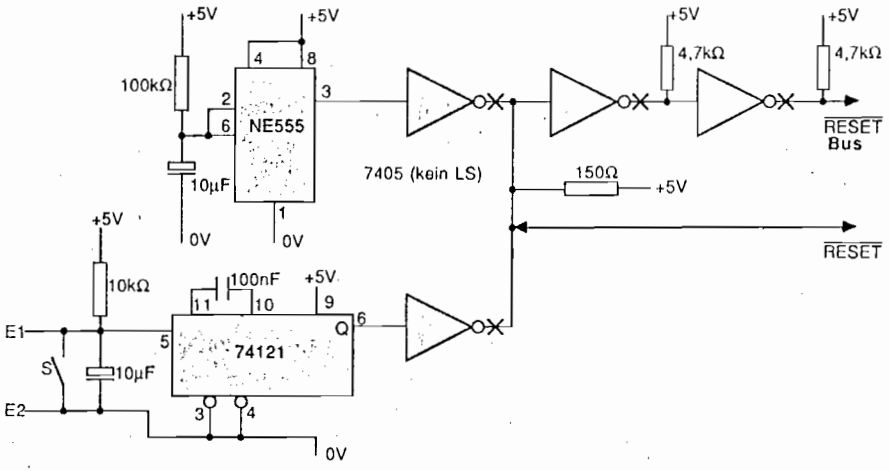


Bild 5: *Reset-Logik*

Es gibt drei RESET-Quellen: Einschalt-RESET, Benutzer-RESET und CPU-RESET. Der Timer NE555 liefert den RESET nach dem Einschalten der Spannung. Er ist so beschaltet, das er einen relativ langen Impuls erzeugt. Das IC 74121 hat die Aufgabe, einen Schalter zu entprellen und einen kurzen RESET-Impuls zu erzeugen, der so kurz ist, daß dynamische RAMs in der Zwischenzeit ihren Inhalt nicht verlieren. Der Taster schließt einen Kondensator kurz, der sich über einen 10KOhm Widerstand langsam wieder auflädt. Wenn der verwendete Taster stark prellt, so kann man statt des 10uF auch einen 47uF Kondensator verwenden.

Die beiden RESET-Signale vom Timer 555 und Monoflop 74121 werden über Open-Collector Inverter zusammengeführt (Wired-Or). Das Signal gelangt von dort aus direkt zum 68020 und 68881. Der 68020 kann per Befehl auch selbst einen RESET auslösen, der auch auf diese Leitung führt (Wired-Or).

Über zwei Inverter wird das RESET-Signal auf den NDR-Computer-Bus geleitet. Auch hier liegt ein Open-Collector-Inverter am Bus, jedoch kann ein von einer anderen Baugruppe ausgelöster RESET nicht die CPU68020-Baugruppe zurücksetzen.

4.3 Die DMA-Logik

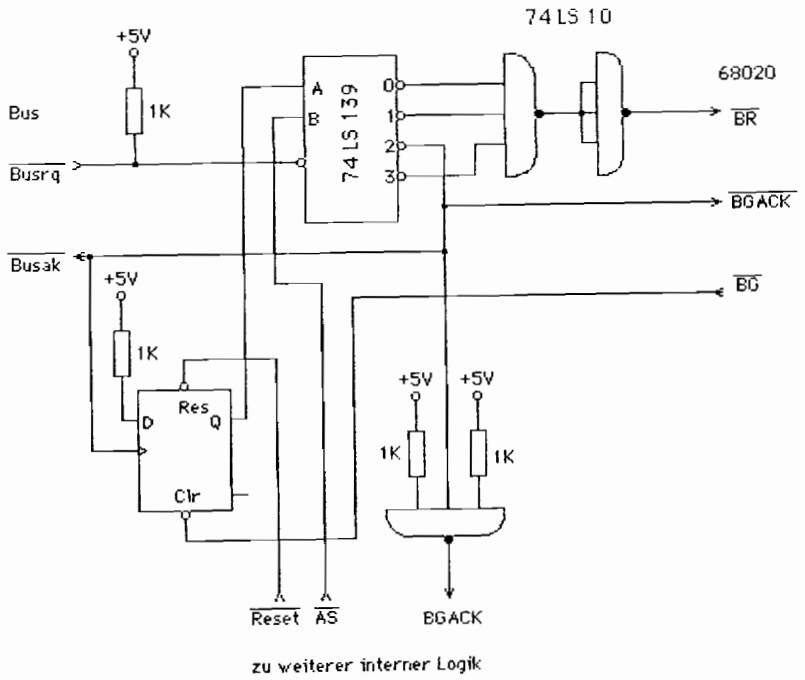


Bild 6: *DMA-Logik*

Diese Schaltung verwaltet den DMA-Zugriff über die Signale BUSRQ und BUSAK. Wird das Signal BUSRQ am Bus angelegt (aktiv LOW), dann wird das Signal BUSAK von der Schaltung erst dann auf LOW gelegt, wenn der Bus wirklich frei ist. Wenn danach das Signal BUSRQ wieder inaktiv (HIGH) wird, so wird das Signal BUSAK zurückgenommen (HIGH); dann darf die CPU68020 wieder auf den Bus zugreifen.

Die Schaltung sorgt dafür, daß das BUSRQ-Signal erst dann LOW wird, wenn alle Signale des 68020 inaktiv sind. Dazu wird das von außen kommende Signal BUSRQ mit den internen Signalen AS (Adreß-Strobe) und BG (Bus Grant) verknüpft. Das Signal BR (Bus Request) für den 68020 wird erzeugt: nach dem Anlegen von BUSRQ wird BR zunächst auf LOW gelegt, damit die CPU die Anforderung mitbekommt. Sie bestätigt den Erhalt der Anforderung, indem sie BG auf LOW legt; dies kann aber auch mitten in einem laufenden Bus-Zyklus geschehen. Daher wird durch den 74LS139 das Signal noch mit AS verknüpft. Ist AS HIGH, so ist der Bus frei und BUSAK und BGACK werden erzeugt. BGACK wird auch invertiert benötigt.

Da die CPU nun BGACK erhalten hat, kann das Signal BR zurückgenommen werden. Wenn der Bus nicht mehr von dem peripheren Gerät benötigt wird, legt dieses das Signal BUSRQ wieder auf HIGH. Die Schaltung sorgt dann dafür, daß BGACK und BUSAK inaktiv werden.

Mit dieser Schaltung kann man den DMA-Zugriff völlig asynchron gestalten. Ein einfacher Test der Schaltung gelingt mit dem Signal HSYNC (von der GDP64K-Baugruppe), das man an den Eingang BUSRQ legt. Mit dem Oszilloskop kann man beobachten, daß kurz nachdem BUSRQ auf LOW geht der Bus freigegeben wird. Das System darf bei dieser "harten" Probe nicht abstürzen. Wenn man VSYNC an die Leitung BUSRQ legt, so ist eine synchrone Abfrage des GDP nicht mehr möglich, daher kann der Cursor nicht mehr richtig blinken und verschwindet eventuell sogar ganz. Das System arbeitet dennoch weiter, nur Floppy-Zugriffe sind nicht mehr möglich.

4.4 Die BOOT-Logik

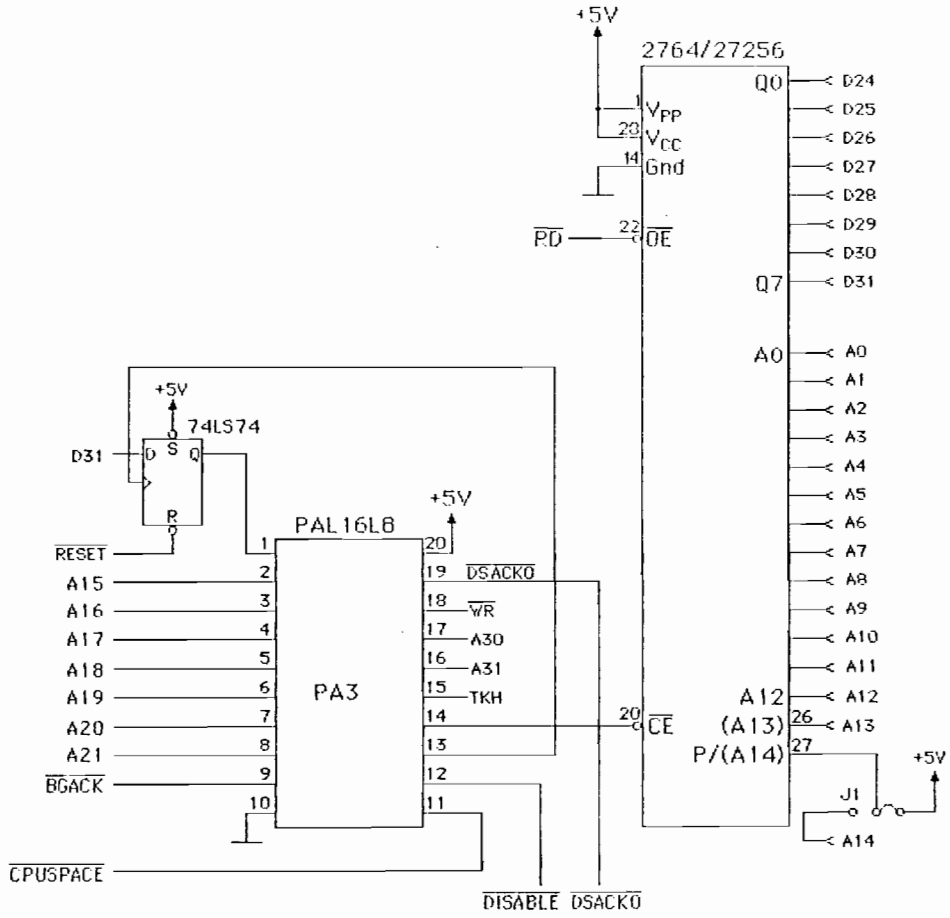


Bild 7: *Boot-Logik*

Der 68020 kann mit unterschiedlichen Datenbusbreiten arbeiten: mit 32-Bit-Datenbus, mit 16-Bit-Datenbus oder mit 8-Bit-Datenbus. Wie breit der aktuelle Datenbus gerade sein soll, das kann man durch die Signale DSACK0 und DSACK1 bestimmen. Dabei kann die Breite für jeden neuen Buszugriff neu bestimmt werden.
Von dieser Möglichkeit macht die Boot-Logik Gebrauch. Ein einziges 2764-EPROM genügt nämlich reichlich für das Boot-Programm, es wäre Verschwendung, vier solcher EPROMs am 32-Bit

Datenbus vorzusehen. Das Boot-EPROM schaltet sich nach dem Reset in den unteren Adreßbereich (0 bis 1FFF, bzw. 0 bis 7FFF, um auch 27256 -ICs zuzulassen). Ein PAL-Baustein sorgt für die Adreßdekodierung. Ein Flipflop 74LS74 wird dazu durch einen System-RESET rückgesetzt. Dadurch gelangt ein LOW-Signal an Pin 1 des PALs. Die Bilder zeigen die PAL-Gleichungen von PA3, die PAL-Belegung, wie sie vom PAL-Assembler erzeugt wird und das Fuse-Diagramm.

```
PAL16L8

Pa 3 Pal fuer CPU68020 NDR-Computer (C) 1986, Rolf-D. Klein

FFQAUS A15 A16 A17 A18 A19 A20 A21 NBGACK GND
NCPUSPAC NDISABLE NIOWRT NEPROMCE TKH A31 A30 NWR NDSACK0 VCC

IF (VCC) /NEPROMCE = /FFQAUS * /A15 * /A16 * /A17 * /A18 * /A19
                     * /A20 * /A21 * NCPUSPAC * /A30 * /A31
IF (VCC) /NIOWRT   = /NWR * /A31 * A31 * NCPUSPAC
IF (VCC) /NDISABLE = /FFQAUS * /A15 * /A16 * /A17 * /A18 * /A19
                     * /A20 * /A21 * NCPUSPAC * /A30 * /A31
                     + /NBGACK

IF (/NEPROMCE * TKH) /NDSACK0 = /NEPROMCE * TKH

DESCRIPTION:    Erzeugung der Selekt und Disable Signale
```

Bild 8: *PA3 Gleichungen*

```
            ****   ****
 FFQAUS    1*   ***   *20   VCC
            *         *
    A15    2*  P A L  *19   NDSACK0
            * 1 6 L 8 *
    A16    3*         *18   NWR
            *         *
    A17    4*         *17   A30
            *         *
    A18    5*         *16   A31
            *         *
    A19    6*         *15   TKH
            *         *
    A20    7*         *14   NEPROMCE
            *         *
    A21    8*         *13   NIOWRT
            *         *
 NBGACK    9*         *12   NDISABLE
            *         *
    GND   10*         *11   NCPUSPAC
           ***********
```

Bild 9: *PA3 Pinbelegung*


```
                    11 1111 1111 2222 2222 2222
   0123 4567 8901 2345 6789 0123 4567 8901

 0 ---- ---- ---- ---- --X- ---X ---- ---- /NEPROMCE*TKH
 1 ---- ---- ---- ---- --X- ---X ---- ---- /NEPROMCE*TKH
 2 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
 3 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
 .
 .
39 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

40 ---- ---- ---- ---- ---- ---- ---- ----
41 -X-X -X-- -X-X -X-X -X-- -X-- -X-- --X- /FFQAUS*/A15*/A16*...
42 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
43 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
44 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
45 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
46 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
47 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

48 ---- ---- ---- ---- ---- ---- ---- ----
49 ---- ---X ---X --X- ---- ---- ---- --X- /NWR*/A31*A31*NCPUSPAC
50 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
51 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
52 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
53 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
54 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
55 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

56 ---- ---- ---- ---- ---- ---- ---- ----
57 -X-X -X-- -X-X -X-X -X-- -X-- -X-- --X- /FFQAUS*/A15*/A16*...
58 ---- ---- ---- ---- ---- ---- ---- -X-- /NBGACK
59 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
60 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
61 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
62 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

LEGEND:  X : FUSE NOT BLOWN (L,N,0)
         - : FUSE BLOWN     (H,P,1)

NUMBER OF FUSES BLOWN: 257
```

Bild 10: *Fuse-Diagramm PA3*

In den PAL-Gleichungen sind die Funktionen der Ausgänge in Abhängigkeit von Eingängen definiert. Der Ausgang NEPROMCE führt an das EPROM. Er liegt immer dann auf LOW, selektiert also das EPROM, wenn FFQAUS aktiv ist (also Pin 1 auf LOW liegt), die richtige Adresse anliegt (A16 bis A21 auf LOW, sowie A30 und A31 auf LOW) und NCPUSPAC auf LOW liegt. Das Signal NCPUSPAC wird erzeugt wenn die Adreßleitungen der CPU eine besondere Bedeutung haben (s.u.).

Das Flipflop kann nun durch einen Speicherzugriff gesetzt werden, womit das EPROM ausgeschaltet wird. Dazu wird der Takteingang des Flipflops mit einem Ausgang des PALs verbunden, der mit NIOWRT bezeichnet ist. Er wird bei Anlegen einer bestimmten Adresse (A31=1 und A30=0, also z.B. auch BFFFFFC8, siehe auch BOOT-Programm) und einem Schreibsignal (WR) auf LOW gelegt. Damit wird D31 in das Flipflop übernommen. Mit dem Befehl

MOVE.B #$80,$BFFFFFC8

kann man also das Flipflop setzen und somit das BOOT-EPROM ausblenden.

Das Signal DSACK0 dient zur Busbreitenumschaltung beim 68020. Wenn dieses Signal auf 0 liegt und das Signal DSACK1 auf 1, erwartet die CPU einen 8-Bit-Datenbus. Alle Daten werden dabei auf den Leitungen D31 bis D24 übertragen. Daher sind auch die Datenausgänge des EPROMs mit diesen Leitungen verbunden. Das Signal NDSACK0 in der PAL-Gleichung wird noch mit dem Eingang TKH verknüpft. Dieser Eingang erhält sein Signal von einem Schieberegister, das die Aufgabe besitzt, den Zugriff an das EPROM anzupassen, also Warte-Zyklen einzufügen. Damit können auch sehr langsame EPROMs als BOOT-EPROM verwendet werden. Der Ausgang DSACK0 ist dabei als offener Kollektor geschaltet. In der PAL-Gleichung steht daher als IF-Bedingung die gleiche Formel wie als Wertzuweisung. Der Ausgang wird in den hochohmigen Zustand geschaltet, wenn diese Gleichung nicht erfüllt ist, und sonst auf LOW.

Das Signal NDISABLE in der PAL-Gleichung hat die Aufgabe, bei Zugriff auf das EPROM die internen Treiber und die Zugriffe nach außen auf den Bus zu sperren. Auch bei einem DMA-Zugriff muß das der Fall sein. Daher wird einmal mit der Formel für NEPROMCE, aber auch mit BGACK verknüpft.

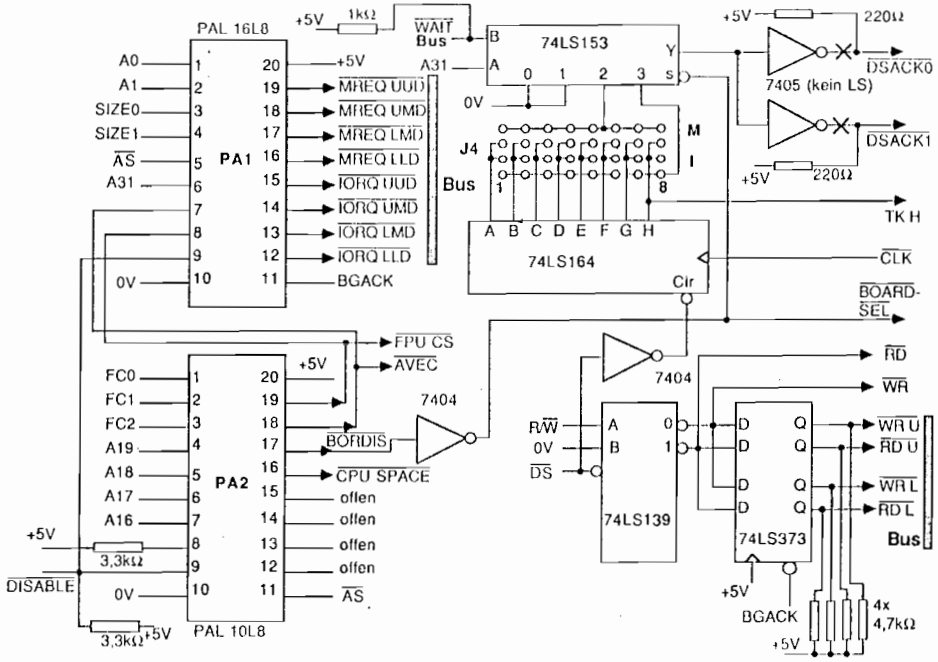


Bild 11: *Timing- und Steuersignal-Logik*

In dieser Schaltung werden alle wichtigen Timing-Signale erzeugt. Dazu werden zwei PAL-Bausteine verwendet. PA1 hat die Aufgabe, die Sinale MREQ und IORQ für den Bus zu erzeugen. Jede Bushälfte besitzt ihre eigenen Steuersignale. MREQ (Memory Request) zeigt an, daß auf den Speicher zugegriffen werden soll und IORQ (Input/Output Request) zeigt einen Zugriff auf die Peripherie an. Die Signale entsprechen in ihrer Bedeutung den Signale des Z80, wo ebenfalls Speicher und Peripherie getrennt adressiert werden. Da die Signale des 68020 von denen des Z80 verschieden sind, müssen die Bus-Signale von dieser Schaltung erzeugt werden. Die 68020-CPU unterscheidet verschiedene Zugriffsarten nach Byte, Wort, Langwort und 3/4 Langwort. Dazu liefert sie die Signale SIZE0 und SIZE1. Zusammen lassen sich daraus die Steuersignale für die Bushälften errechnen.

Die Ausgangssignale des PAL-Bausteines können durch das Signal BGACK in den Tri-State-Zustand geschaltet werden, um z.B. bei einem DMA-Zugriff nicht zu stören. Speicher- und Peripheriezugriff werden durch die Adreßleitung A31 unterschieden. Wenn A31 auf LOW liegt, handelt es sich um einen Speicherzugriff, also werden die Signale MREQ aktiv, sonst um einen IO-Zugriff, und die Signale IORQ werden aktiv. Zugegriffen wird aber auf jeden Fall nur dann, wenn NDIS auf HIGH liegt, also das Disable-Signal inaktiv (HIGH) ist, NFPUCS auf HIGH liegt, also die FPU nicht angesprochen wird, NAVEC auf HIGH ist, also kein Interrupt-Zyklus vorliegt und NAS auf LOW liegt, also der Adreß-Strobe aktiv ist und eine gültige Adresse anliegt.

Die Auswahl der einzelnen Bereiche (UUD..LLD) erfolgt durch zusätzliche Verknüpfung mit den Leitungen A0, A1, SIZE0 und SIZE1.

Die Bilder zeigen PAL-Gleichungen, Belegung und Fuse-Diagramm von PA1.

```
PAL16L8

Pa 1 PAL fuer CPU68020 NDR-Computer (C) 1986, Rolf-D. Klein

A0 A1 SIZE0 SIZE1 NAS A31 NAVEC NFPUCS NDIS GND
BGACK IOLLD IOLMD IOUMD IOUUD MRLLD MRLMD MRUMD MRUUD VCC

IF (/BGACK) /MRUUD =   NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /A0 * /A1
IF (/BGACK) /MRUMD =   NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /SIZE0 * /A1
                     + NDIS * FPUCS * NAVEC * /NAS
                         * /A31 * A0 * /A1
                     + NDIS * FPUCS * NAVEC * /NAS
                         * /A31 * /A1 * SIZE1
IF (/BGACK) /MRLMD =   NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /A0 * A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /A1 * /SIZE0 * /SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /A1 * SIZE0 * SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * A0 * /A1 * /SIZE0
IF (/BGACK) /MRLLD =   NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * A0 * SIZE0 * SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * /SIZE0 * /SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * A0 * A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A31 * A1 * SIZE1
IF (/BGACK) /IOUUD =   NDIS * NFPUCS * NAVEC * /NAS
                         * A31 * /A0 * /A1
```



```
IF (/BGACK) /IOUMD =   NDIS * NFPUCS * NAVEC * /NAS
                         * A31 * /SIZE0 * /A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * A31 * A0 * /A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * A31 * /A1 * /SIZE1
IF (/BGACK) /IOLMD =   NDIS * NFPUCS * NAVEC * /NAS
                         * /A0 * A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A1 * /SIZE0 * /SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /A1 * SIZE0 * SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * A0 * /A1 * /SIZE0
IF (/BGACK) /IOLLD =   NDIS * NFPUCS * NAVEC * /NAS
                         * A0 * SIZE0 * SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * /SIZE0 * /SIZE1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * A0 * A1
                     + NDIS * NFPUCS * NAVEC * /NAS
                         * A1 * SIZE1

DESCRIPTION:  Erzeugt die Signale IORQ und MREQ fuer den Bus,
              die in UUD UMD LMD und LLD geteilt sind.
```

Bild 12: *PA1-Gleichungen*

Bild 13: *PA1-Pinbelegung*

```
                11 1111 1111 2222 2222 2233
   0123 4567 8901 2345 6789 0123 4567 8901

 0 ---- ---- ---- ---- ---- ---- ---- ---X /BGACK
 1 -X-X ---- ---- -X-- -X-- X--- X--- X--- ...*/A31*/A0*/A1
 2 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
 ...
 7 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

 8 ---- ---- ---- ---- ---- ---- ---- ---X /BGACK
 9 -X-- -X-- ---- -X-- -X-- X--- X--- X--- ...*/SIZE0*/A1
10 -XX- ---- ---- -X-- -X-- X--- X--- X--- ...*/31*A0*/A1
11 -X-- ---- X--- -X-- -X-- X--- X--- X--- ...*/A1*SIZE1
12 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
 ...
15 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

16 ---- ---- ---- ---- ---- ---- ---- ---X /BGACK

usw.

63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Bild **14:** *Auszug aus PA1 Fuse-Diagramm*

Eine weitere wichtige Funktion wird durch PA2 erreicht. Es hat die Aufgabe, die Dekodierung des Selekt-Signals für die FPU und für den Autovektor vorzunehmen. Ferner werden die Signale für die interne Bussteuerung erzeugt. Die CPU zeigt Spezial-Zugriffe mit den Signalen FC0 bis FC2 an. Wenn alle auf HIGH liegen, wird ein sogenannter CPU-SPACE-Zugriff durchgeführt, also ein Zugriff mit einem Adreßraum, der speziell der CPU zugeordnet ist. Der PAL-Baustein erzeugt das Signal NCPUSPAC, das diesen Zustand auch nach außen an andere Schaltungsteile weitergibt, denn für diesen Zugriff besitzt der Adreßbus eine besondere Bedeutung und darf nicht wie gewohnt dekodiert werden. Die CPU kann mit dieser Zugriffsart zum Beispiel Coprozessoren adressieren. Welcher Coprozessor dabei gemeint ist, wird durch die Adreßleitungen A16 bis A19 mitgeteilt. Daher wird im PAL eine weitere Verknüpfung durchgeführt: Wenn A19=0, A18=0, A17=1 und A16=0 sind, so erfolgt ein Zugriff auf die FPU (Floating Point Unit). Dieser Zugriff wird von der CPU automatisch gesteuert, so daß die FPU wie eine Befehlserweiterung im Instruktionssatz aussieht. Die CPU-Einheit versteht also mehr Befehle.

Auf der CPU68020-Baugruppe wird nur der Autovektor-Mechanismus zur Interrupt-Bearbeitung eingesetzt, der die Erzeugung spezieller Interrupt-Vektoren auf dem Bus überflüssig macht. Dadurch ist es möglich, den Bus für unterschiedliche Prozessoren kompatibel zu halten. Eine Interrupt-Bestätigung erhält man, wenn die Adreßleitungen A19 bis A16 auf HIGH liegen und FC0 bis FC2 ebenfalls. Dann wird hier das Signal NAVEC erzeugt, das direkt an die CPU geleitet wird. Dadurch wird eine besondere Bedeutung für die Leitungen IPL0 bis IPL1 der CPU programmiert. Die CPU führt dann Interrupts auf feste Adressen.

Ein weiteres Signal, NBOARDIS, wird auch vom PAL PA2 erzeugt. Es hat die Aufgabe, die Datenbustreiber und einige andere Schaltungsteile zu sperren, wenn entweder ein FPU-Zugriff vorliegt oder das Signal NDISABLE (Disable) anliegt. NDISABLE kommt von PA3 und wird erzeugt, wenn ein DMA-Zugriff oder ein Zugriff auf das BOOT-EPROM vorliegt.

Die Bilder zeigen die PAL-Gleichungen, Belegung und Fuse-Diagramm von PA2.

```
PAL10L8

Pa 2 Pal fuer CPU68020 NDR-Computer (C) 1986 Rolf-D. Klein

FC0 FC1 FC2 A19 A18 A17 A16 NC NDISABLE GND
NAS NC NC NC NC NCPUSPAC NBOARDIS NAVE NFPUCS VCC

/NFPUCS     =   FC0 * FC1 * FC2 * /A19 * /A18 * A17 * /A16
/NAVEC      =   FC0 * FC1 * FC2 * A19 * A18 * A17 * A16 * /AS
/NBOARDIS   =   FC0 * FC1 * FC2 * /A19 * /A18 * A17 * /A16
              + /NDISABLE
/NCPUSPAC   =   FC0 * FC1 * FC2
```

Bild 15: *PA2-Gleichungen*

```
          ****   ****
    FC0   1*   ***   *20   VCC
          *         *
    FC1   2*  P A L  *19   NFPUCS
          * 1 0 L 8 *
    FC2   3*         *18   NAVEC
          *         *
    A19   4*         *17   NBOARDIS
          *         *
    A18   5*         *16   NCPUSPAC
          *         *
    A17   6*         *15   NC
          *         *
    A16   7*         *14   NC
          *         *
     NC   8*         *13   NC
          *         *
NDISABLE   9*         *12   NC
          *         *
    GND  10*         *11   NAS
          ***********
```

Bild 16: *PA2 Pinbelegung*


```
                  11 1111 1111 2222 2222 2233
    0123 4567 8901 2345 6789 0123 4567 8901

 0 X-X- X-oo -Xoo -Xoo X-oo -Xoo --oo ---- FC0*FC1*FC2*/A19*...
 1 XXXX XXoo XXoo XXoo XXoo XXoo XXoo XXXX
 2 0000 0000 0000 0000 0000 0000 0000 0000
 ...
 7 0000 0000 0000 0000 0000 0000 0000 0000

 8 X-X- X-oo X-oo X-oo X-oo X-oo --oo ---X FC0*FC1*FC2*A19*...
 9 XXXX XXoo XXoo XXoo XXoo XXoo XXoo XXXX
10 0000 0000 0000 0000 0000 0000 0000 0000
 ...
15 0000 0000 0000 0000 0000 0000 0000 0000

16 X-X- X-oo -Xoo -Xoo X-oo -Xoo --oo ---- FC0*FC1*FC2*/A19*...
17 ---- --oo --oo --oo --oo --oo --oo -X-- /NDISABLE
18 0000 0000 0000 0000 0000 0000 0000 0000
 ...
23 0000 0000 0000 0000 0000 0000 0000 0000

24 X-X- X-oo --oo --oo --oo --oo --oo ---- FC0*FC1*FC2
25 XXXX XXoo XXoo XXoo XXoo XXoo XXoo XXXX
26 0000 0000 0000 0000 0000 0000 0000 0000
 ....
31 0000 0000 0000 0000 0000 0000 0000 0000

32 XXXX XXoo XXoo XXoo XXoo XXoo XXoo XXXX
33 XXXX XXoo XXoo XXoo XXoo XXoo XXoo XXXX
34 0000 0000 0000 0000 0000 0000 0000 0000
 ...
39 0000 0000 0000 0000 0000 0000 0000 0000

40..47 wie 32..39
48..55 wie 32..39
56..63 wie 32..39

LEGEND:  X : FUSE NOT BLOWN (L,N,0)
         - : FUSE BLOWN     (H,P,1)
         0 : PHANTOM FUSE   (L,N,0)
         o : PHANTOM FUSE   (H,P,1)

NUMBER OF FUSES BLOWN =  134
```

Bild 17: *PA2 Fuse-Diagramm*

Nun zur restlichen Steuer-Logik. Ein Dekoder 74LS139 übernimmt die Erzeugung der Signale RD und WR. RD tritt bei einem Lesevorgang auf und WR bei einem Schreibvorgang. Der Baustein 74LS373 hat die Aufgabe, diese Signale zu puffern. Dabei bekommt jede Bushälfte ein eigenes gepuffertes Signal, um insgesamt mehr Baugruppen ansprechen zu können. Diese Signale WRU/WRL und RDU/RDL sind von den internen Signalen WR und RD abgeleitet. Alle Bausteine 74LS373 werden von BGACK gesteuert, damit bei einem DMA-Zugriff alle Signale auf dem Bus frei zur Verfügung stehen. Pull-Up-Widerstände sorgen dafür, daß nach dem Umschalten kein undefinierter Zustand auf dem Bus auftritt, falls die DMA-Schaltung die Signale erst nach einiger Verzögerungszeit belegt.

4.6 Die WAIT-Logik

Mit einem Schieberegister 74LS164 und einem Multiplexer 74LS153 können Wartezyklen erzeugt werden. In unserer Schaltung kann man Speicher- und IO-Wartezyklen getrennt einstellen. Von Position 1 bis 8 wächst die Zahl der eingestellten Wartezyklen. Die FLO2-Baugruppe benötigt ca. 3 bis 4 Wartezyklen bei 16-MHz-Betrieb. Daher sollte man die IO-Zyklen auf diesen Wert stellen Brücke (I auf 4). Beim Speicher genügt ein Wartezyklus (Brücke M auf 1). Der Multiplexer wählt die jeweilige Wait-Quelle aus. Dazu wird an den A-Eingang das Signal A31 geführt. Es unterscheidet Speicher von Peripherie. Auf den Eingang B ist die Leitung WAIT vom Bus geführt. Damit kann auch eine externe Schaltung ein Wartesignal erzeugen. Die Warte-Steuerung erfolgt über DSACK0 und DSACK1. Da außen immer ein 32-Bit-Bus vorhanden ist, werden beide Signale über die Gatter 7405 gleichzeitig auf LOW gelegt. Nach Beginn eines Bus-Zyklus wartet die CPU, bis DSACK0 oder DSACK1 oder beide auf LOW gehen. Durch das Schieberegister wird ein durchlaufendes 1-Signal an seinen Ausgängen erzeugt, sobald das Signal DS (Data Strobe) den Zugriffsbeginn mitteilt. Über den Multiplexer gelangt dann das Speicher- oder Peripherie-Signal an die Gatter im 7405. Wenn von außen WAIT angelegt wird, d.h. die Leitung WAIT am Bus auf LOW gelegt wird, so schaltet der Multiplexer auf die Eingänge 0 oder 1 um und erhält von dort ein 0-Signal. DSACK0 und DSACK1 bleiben damit auf 1 und die CPU wartet. Das Schieberegister arbeitet normal weiter, und DSACK0 und DSACK1 werden nach Ende des Wait-Signales sofort auf 0 gelegt, falls das Schieberegister ebenfalls schon mit seinen Wartezyklen durch ist. Der Multiplexer wird zusätzlich durch das Signal BOARDSEL gesteuert, denn wenn die CPU auf die FPU zugreift, so erzeugt diese ihre eigenen DSACK-Signale. Bei einem Zugriff auf das BOOT-EPROM wird der Multiplexer über das BOARDSEL-Signal ebenfalls gesperrt, denn das EPROM erzeugt ja ein eigenes DSACK-Signal (wobei nur DSACK0 erzeugt wird, da das EPROM mit einem 8-Bit-Datenbus arbeitet). Die beiden Signale DSACK0 und DSACK1 sind mit einem sehr niederohmigen Widerstand (220 Ohm) nach +5V verbunden, um die Schaltung für 16 MHz tauglich zu machen. Sind die Widerstände nämlich zu groß, so steigen die Signale DSACK0 und DSACK1 zu langsam an, und der nächste Bus-Zyklus erhält noch das 0-Signal als Eingangswert und arbeitet dann ohne Wartezyklen.

4.7 CPU und FPU

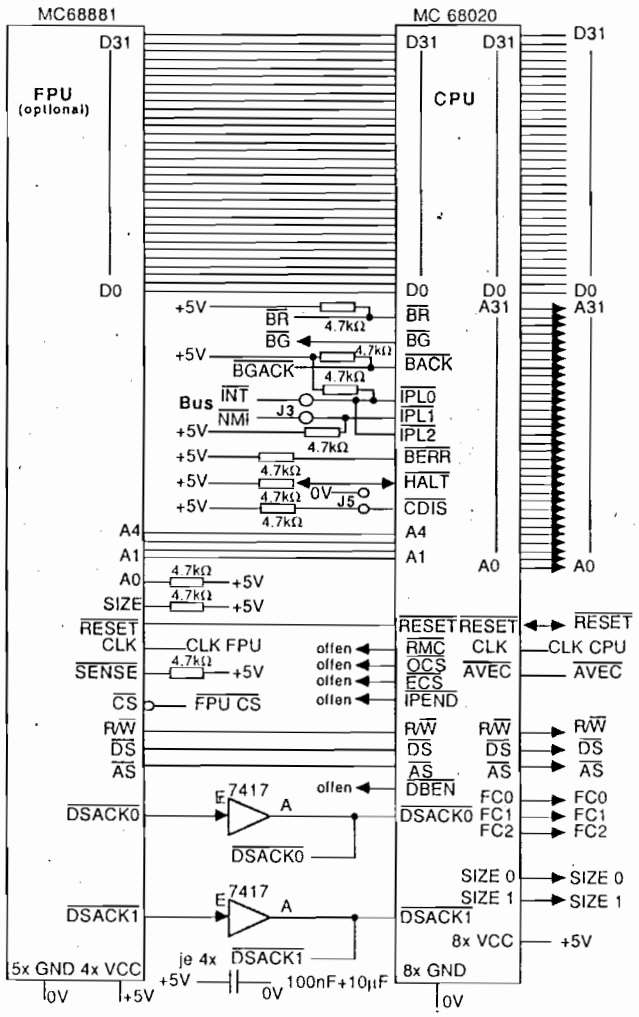


Bild 18: *Verbindung von 68020 und 68881*

4.6 Die WAIT-Logik

Mit einem Schieberegister 74LS164 und einem Multiplexer 74LS153 können Wartezyklen erzeugt werden. In unserer Schaltung kann man Speicher- und IO-Wartezyklen getrennt einstellen. Von Position 1 bis 8 wächst die Zahl der eingestellten Wartezyklen. Die FLO2-Baugruppe benötigt ca. 3 bis 4 Wartezyklen bei 16-MHz-Betrieb. Daher sollte man die IO-Zyklen auf diesen Wert stellen Brücke (I auf 4). Beim Speicher genügt ein Wartezyklus (Brücke M auf 1). Der Multiplexer wählt die jeweilige Wait-Quelle aus. Dazu wird an den A-Eingang das Signal A31 geführt. Es unterscheidet Speicher von Peripherie. Auf den Eingang B ist die Leitung WAIT vom Bus geführt. Damit kann auch eine externe Schaltung ein Wartesignal erzeugen. Die Warte-Steuerung erfolgt über DSACK0 und DSACK1. Da außen immer ein 32-Bit-Bus vorhanden ist, werden beide Signale über die Gatter 7405 gleichzeitig auf LOW gelegt. Nach Beginn eines Bus-Zyklus wartet die CPU, bis DSACK0 oder DSACK1 oder beide auf LOW gehen. Durch das Schieberegister wird ein durchlaufendes 1-Signal an seinen Ausgängen erzeugt, sobald das Signal DS (Data Strobe) den Zugriffsbeginn mitteilt. Über den Multiplexer gelangt dann das Speicher- oder Peripherie-Signal an die Gatter im 7405. Wenn von außen WAIT angelegt wird, d.h. die Leitung WAIT am Bus auf LOW gelegt wird, so schaltet der Multiplexer auf die Eingänge 0 oder 1 um und erhält von dort ein 0-Signal. DSACK0 und DSACK1 bleiben damit auf 1 und die CPU wartet. Das Schieberegister arbeitet normal weiter, und DSACK0 und DSACK1 werden nach Ende des Wait-Signales sofort auf 0 gelegt, falls das Schieberegister ebenfalls schon mit seinen Wartezyklen durch ist. Der Multiplexer wird zusätzlich durch das Signal BOARDSEL gesteuert, denn wenn die CPU auf die FPU zugreift, so erzeugt diese ihre eigenen DSACK-Signale. Bei einem Zugriff auf das BOOT-EPROM wird der Multiplexer über das BOARDSEL-Signal ebenfalls gesperrt, denn das EPROM erzeugt ja ein eigenes DSACK-Signal (wobei nur DSACK0 erzeugt wird, da das EPROM mit einem 8-Bit-Datenbus arbeitet). Die beiden Signale DSACK0 und DSACK1 sind mit einem sehr niederohmigen Widerstand (220 Ohm) nach +5V verbunden, um die Schaltung für 16 MHz tauglich zu machen. Sind die Widerstände nämlich zu groß, so steigen die Signale DSACK0 und DSACK1 zu langsam an, und der nächste Bus-Zyklus erhält noch das 0-Signal als Eingangswert und arbeitet dann ohne Wartezyklen.

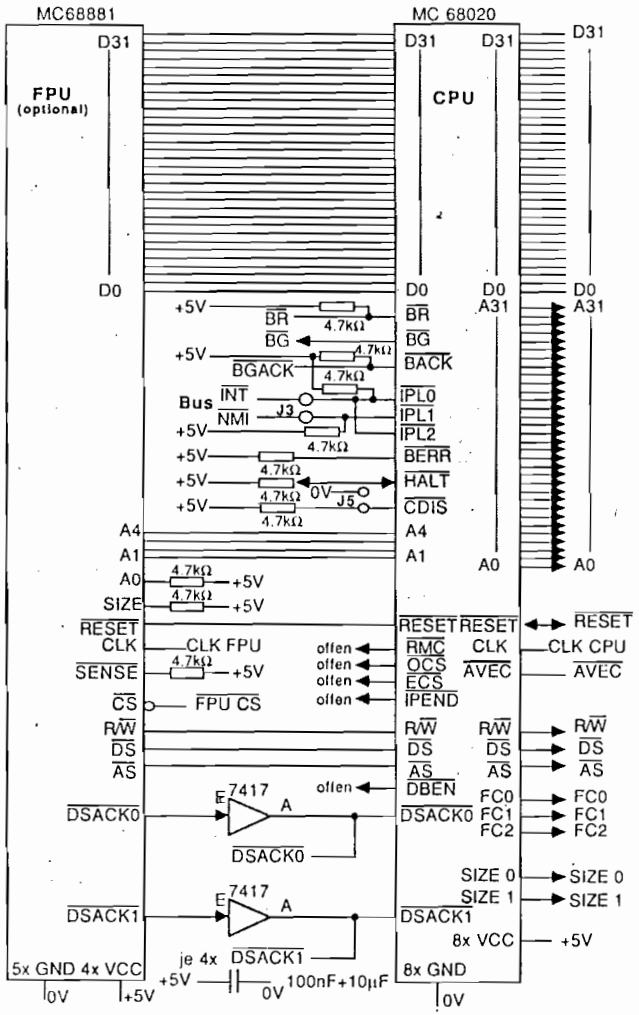


Bild 18: *Verbindung von 68020 und 68881*

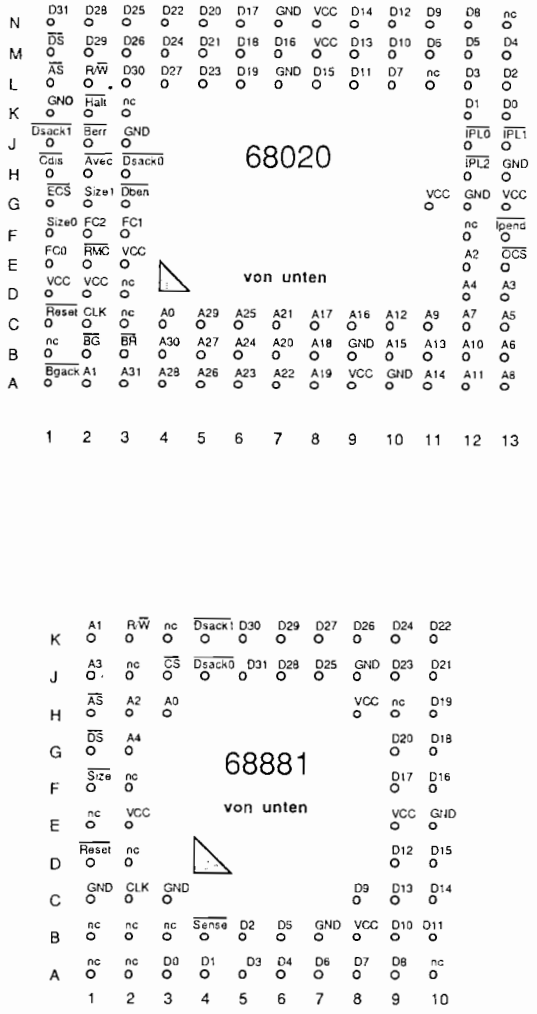


Bild 19: *Pinbelegung 68020 und 68881*

Die FPU ist fast völlig parallel zur CPU geschaltet. Nur die Signale A0, SIZE, CLK, SENSE und CS liegen nicht parallel. A0 und SIZE bestimmen die Datenbusbreite, die hier auf 32 Bit eingestellt ist. SENSE ist von der CPU aus ständig auf Masse gelegt. Damit könnte man über einen Port abfragen, ob die FPU im Sockel steckt, was hier jedoch nicht getan wird. Die Signale CLK, also der Takt, kommen von der Taktlogik und sind dort über einen 47-Ohm-Widerstand mit dem gemeinsamen Takt verbunden. Es ist möglich, CPU und FPU mit unterschiedlicher Taktfrequenz zu betreiben. Das Signal CS ist das Auswahl-Signal an die FPU und wird durch PA1 erzeugt.

Die FPU erzeugt noch zwei Rückmeldesignale, DSACK0 und DSACK1, die über nicht-invertierende Treiber mit den entsprechenden Eingängen der CPU verbunden sind. Darüber werden Datentransfer-Ende und die Datenbusbreite signalisiert.

Bei der CPU sind die Signale IPL0 bis IPL2 bedeutsam. Sie dienen der Interrupt-Anforderung. IPL0 und IPL2 sind zusammengeschaltet, damit man vom Bus her das gleiche Verhalten wie beim 68008 bekommt, zumal nur zwei Interrupt-Leitungen auf dem Bus zur Verfügung stehen. Will man einen nicht-maskierbaren Interrupt erzeugen, so muß man IPL0/2 und IPL1 gleichzeitig auf LOW legen. Dies kann auch durch die Brücke J3 erreichen; dann hat man allerdings nur einen Interrupt-Eingang.

Die Leitung CDIS kann über J5 auf LOW gelegt werden. Damit ist der interne Cache-Speicher gesperrt. Will man den Cache verwenden, muß J5 offen sein und der Befehl zum Einschalten des Cache

MOVE.L #1,D0
MOVEC D0,CACR

gegeben werden.

Die CPU besitzt acht Masse- und acht Versorgungsspannungs-Anschlüsse (+5V), die alle angeschlossen sein müssen. Die FPU besitzt fünf Masse- und vier +5V-Anschlüsse. Einige Entstörkondensatoren sind in der Nähe der CPU und FPU angebracht.

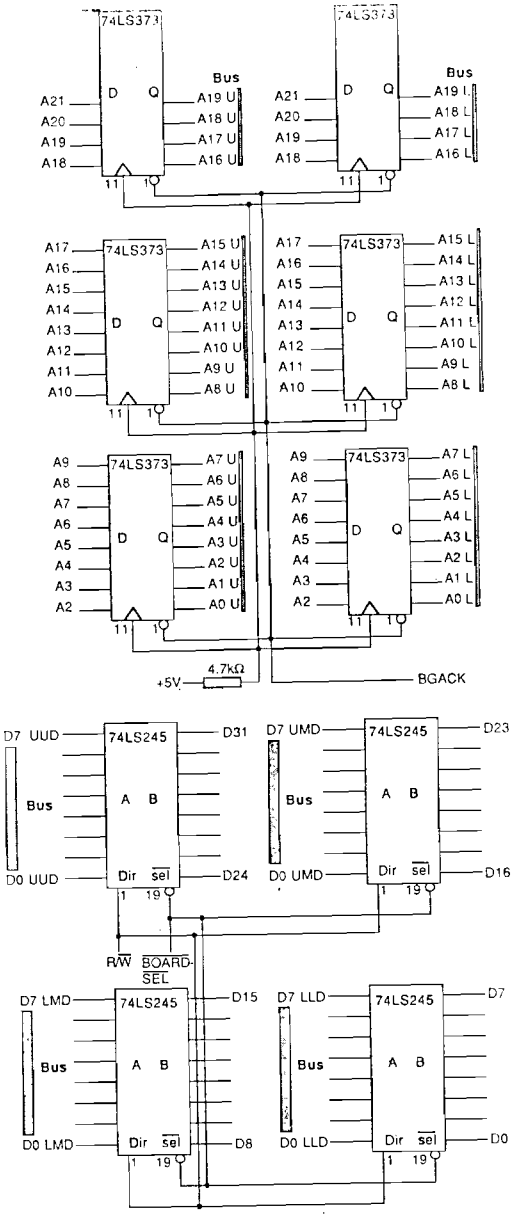


Bild 20, 21: *Adreßbus- und Datenbustreiber*

Die Freigabe der Datenbustreiber erfolgt mit dem Signal BOARDSEL. Durch das Signal R/W wird die Richtung gesteuert.

Für jede Bushälfte sind auch eigene Adreßbustreiber vorgesehen, um insgesammt mehr Baugruppen ansprechen zu können und lange Leitungen auf der Leiterplatte zu puffern. Die Bausteine 74LS373 werden durch das Signal BGACK freigegeben, so daß bei einem DMA-Zugriff der Bus zur Verfügung steht.

Der Adreßbus ist auf besondere Weise verdrahtet. Die Leitung A2 der CPU führt auf die Busleitung A0. Alles ist also um zwei Adreßbits verschoben. Da mit einem 32-Bit-Datenbus gearbeitet wird, aber beim 68020 auch Byte-Adressierung möglich ist, werden die CPU-internen Leitungen A0 und A1 in entsprechende Auswahlsignale für den Bus umgewandelt (PA1).

Beim NDR-Computer kann man mit dem Original-Bus auf diese Weise vier mal 1 MByte, also 4 MByte mit dem 68020 adressieren und im Gegensatz zum 68008 ist der IO-Bereich eigentlich auch genauso groß. Jedoch decodieren alle IO-Baugruppen des NDR-Systems nur die unteren 8 Bit des Adreßbus, so daß "nur" 1024 IO-Adressen ansprechbar sind (auf jedem Busviertel 256 IO-Adressen); da aber die IO-Baugruppen nur auf einem Busviertel angeordnet werden, werden auch hier nur 256 IO-Adressen benutzt.

4.8 Das BOOT-Programm

```
************************************
* Boot fuer 68020 auf der Baugruppe  *
*                                    *
* CPU 68020/32 V1.0 850710 RDK       *
* V 1.1, 860315 mit RAM Pruefung     *
*               und Erkennung        *
************************************

     dc.l $8ffe                * Vorlaeufiger Stack.
     dc.l start

start:
     lea $8070,a0              * Ziel, (liegt ggf. in ausbuf,
                               * einbuf des Grundprogramms)
                               * dort 264 Bytes Platz (rev 1.1)
     movea.l a0,a2             * dort Start merken
fehlerwdh:                     * Einsprung im Fehlerfall
     lea anf(pc),a1            * Quelle im Boot-EPROM
     move #ende-anf-1,d3
transport:
          move.b (a1)+,d0
          move.b d0,(a0)+
          cmp.b -1(a0),d0
          bne.s errorskip
     dbra d3,transport
     jmp (a2)                  * Gueltige Startadresse

errorskip:
     adda.l #$2000,a2          * in 8K Schritten suchen
     movea.l a2,a0             * neue Zieladresse
     bra.s fehlerwdh           * dort neu versuchen


anf:

boot:
     move.b #$80,$BFFFFFC8     * Boot loeschen
     lea $400,a0               * Start der Suche
loop:
          cmp.l #$a55a8002,(a0)     * Anwender-Kennung
          beq.s chess
          cmp.l #$5aa58001,(a0)     * Grundprog. Kennung
          beq.s gefunden
          adda.l #$400,a0           * 1K Seitengrenzen
     bra.s loop
gefunden:
     cmp #$6000,$20(a0)
     bne.s loop
     cmp #$6000,$24(a0)
     bne.s loop
     jmp $24(a0)               * Start auch wenn im RAM
```

```
goon:
     move.l #$a55a8002,(a0)    * restaurieren
     adda.l #$1000,a0
     bra.s loop

chess:                         * Einsprung fuer Anwender-
     clr.l (a0)                * Software muss in ROM sein
     cmp.l #$a55a8002,(a)
     bne goon
     jmp $4(a0)                * und starten

ende:
     end
```

Bild 22: *Listing des Boot-Programmes*

Zum Betrieb der Baugruppe benötigt man das BOOT-EPROM. Es hat die Aufgabe, nach dem Grundprogramm zu suchen und dieses zu starten. Durch das BOOT-Prinzip ist es möglich, ab der Adresse $000000 RAM zu betreiben, wie man es für den Betrieb von CP/M68k benötigt.

Das Programm kopiert zunächst das eigentliche Suchprogramm in einen RAM-Bereich. Dabei wird bei Adresse $8070 begonnen nach RAM zu suchen. Die Adresse wurde so gewählt, weil sie in das lokale RAM des Grundprogramms fallen kann (Version 4.3 auf $8000, Version 5.x auf $10000). Dann wird da nur ein unbedeutender Puffer überschrieben. Damit ist ein zerstörungsfreier Warmstart nach einem erneuten RESET möglich. Das Bootprogramm schaltet, sobald es sich im RAM befindet und dort aufgerufen wurde, das BOOT-EPROM ab und beginnt die Suche bei Adresse $400 in 1K-Schritten. Das Grundprogramm besitzt ein Erkennungsmuster ($5aa58001) mit dem das Suchprogramm den Anfang finden kann. Falls das Muster gefunden ist, wird eine Plausibilitätsprüfung durchgeführt und dann wird das Grundprogramm gestartet. Ein weiterer String, der $a55a8002 lautet, wird ebenfalls gesucht. Mit ihm kann man eigene Anwendersoftware kennzeichnen, die dann auch automatisch gestartet werden kann.

5. Die Bedeutung der Jumper

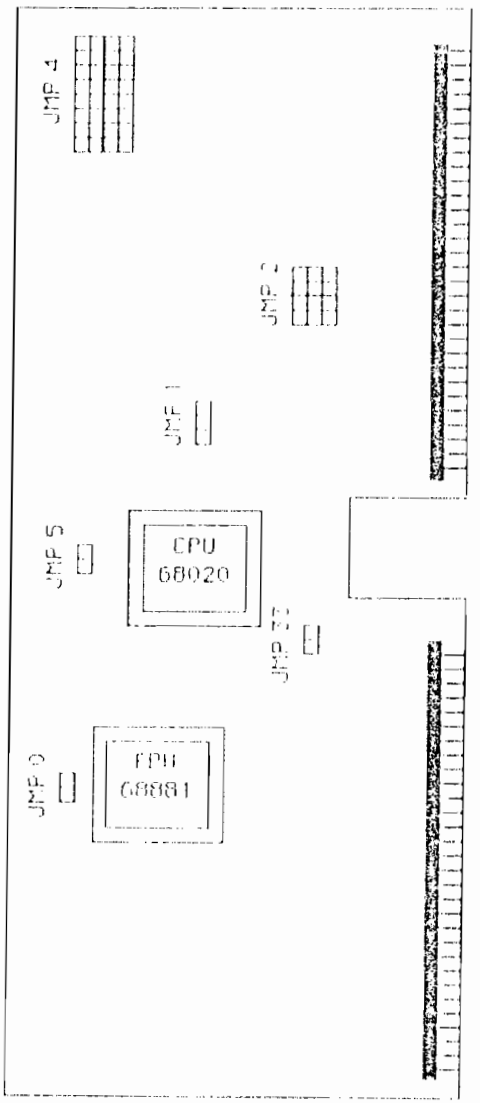


Bild 23: *Lage der Jumper auf der CPU68020*

5.1 JMP0

Mit dieser Steckbrücke kann man die Taktversorgung der FPU umkonfigurieren. Der entsprechende Ausschnitt aus dem Schaltplan zeigt die Bedeutung:

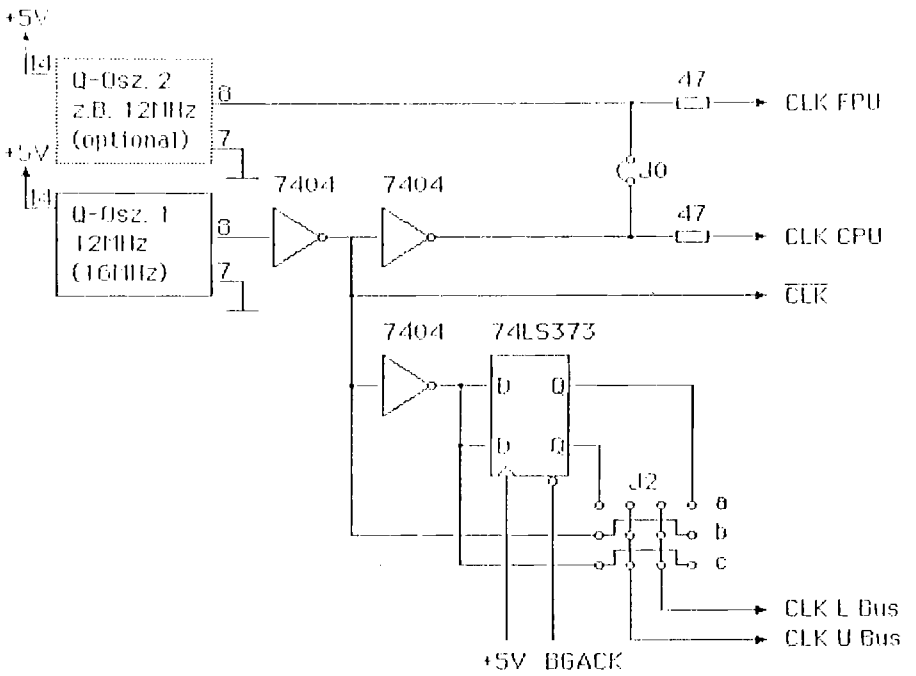


Bild 25: *JMP0*

JMP0 o o offen: Q-Osz. 2 bestückt; FPU bekommt Takt von Q-Osz. 2

JMP0 o-o geschlossen: Q-Osz. 2 darf nicht bestückt sein, FPU bekommt Takt wie CPU von Q-Osz. 1
Diese Einstellung ist die Default-Einstellung

5.2 JMP1

Mit diesem Jumper wird die Größe des Boot-EPROMs auf der Baugruppe CPU68020 eingestellt.

JMP1 o-o o Im EPROM-Sockel sitzt ein 27256

JMP1 o o-o Im EPROM-Sockel sitzt ein 27128 oder 2764
Diese Einstellung ist die Default-Einstellung

5.3 JMP2

Mit diesem Jumper wird die Funktion der CLK-Leitung auf dem Bus eingestellt.

JMP2	o-o o-o o o o o o o o o	nicht negierter Takt (von Q-Osz. 1), der bei DMA-Zugriffen in den hochohmigen Tri-State-Zustand gesetzt wird
JMP2	o o o o o-o o-o o o o o	-CLK, negierter Takt (von Q-Osz. 1) auf den Bus
JMP2	o o o o o o o o o-o o-o	CLK, Takt (von Q-Osz. 1) auf den Bus. Dies ist die Default-Einstellung.
JMP2	o o o o o o o o o o o o	kein Taktsignal auf dem Bus

5.4 JMP3

Mit diesem Jumper wird die Interrupt-Konfiguration eingestellt.

JMP3	o o	offen:	Standard-Interrupt-Konfiguration. Die Bus-Leitung -INT führt am Prozessor an den Pin -IPL0, -NMI führt an das verbundene -IPL1 und IPL2.
JMP3	o-o	geschlossen:	Die Leitungen -INT und -NMI sind miteinander verbunden, am Prozessor sind -IPL0, -IPL1 und -IPL2 verbunden, damit wird immer ein nichtmaskierbarer Interrupt ausgelöst.

5.5 JMP4

Wartezyklen-Generator. Ausschnitt aus dem Schaltplan:

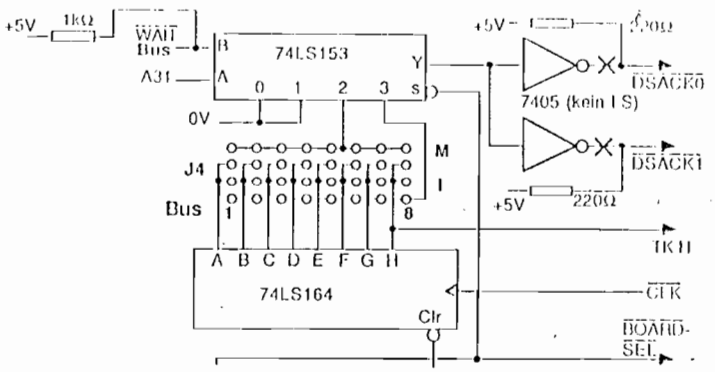


Bild 26: *JMP4*

Beispieleinstellungen:

JMP4 o o o o o o o o es werden keine Wartezyklen bei Speicher- und I/O-Zugriffen eingefügt
o o o o o o o o

o o o o o o o o

o o o o o o o o

JMP4 o o o o o o o o bei Speicherzugriffen werden zwei Wartezyklen eingefügt, bei I/O-Zugriffen werden vier Wartezyklen erzeugt
I
o o o o o o o o

o o o o o o o o

o o o o o o o o

JMP4 o o o o o o o o jeweils 3 Wartezyklen bei Speicher- und I/O-Zugriffen
I
o o o o o o o o default-Einstellung

o o o o o o o o
I
o o o o o o o o

5.6 JMP5

Wenn dieser Jumper geschlossen ist, so kann der Cache-Speicher des 68020 nicht verwendet werden. Dies ist nur sinnvoll bei Test und besonderen Prüfungen der CPU-Baugruppe.

JMP5 o o offen: Cache enable, default

JMP5 o-o geschlossen: Cache disable

6. Anwendungsbeispiele

Nach dem Einschalten des Rechners meldet sich das RDK-Grundprogramm. Hier stehen dem Anwender schon viele Möglichkeiten zur Programmierung des Computers offen. Mit einem Disketten-Betriebssystem erschließt man sich die Welt komerziell erhältlicher Software.

6.1 Grundprogramm

Im Grundprogramm stehen einem folgende Möglichkeiten zur Verfügung:

- ändern (Speicherstellen, Byte-, Wort- und Langwortweise)
- starten (von Anwenderprogrammen)
- ansehen (von Speicherbereichen)
- Symbole (bisher festgelegte Symbole ansehen)
- Editor (Texte bildschirmorientiert editieren)
- Assembler (kompletter 68K-Assembler mit FPU-Unterstützung)
- Bibliothek (Verzeichnis und Start von Anwenderprogrammen, z.B in EPROMs)
- Optionen:
  - Assembler-Optionen
    - Nur Fehlerausgabe
    - Ausgabe auf CRT (Assembler-Ausgabe auf den Bildschirm)
    - Ausgabe auf LST (Assembler-Ausgabe auf den Drucker)
  - Textstart (zum Verwalten mehrerer Texte im Speicher):
    - alter Text
    - neuer Text
  - Löschen Symbole
  - Zeichen/Zeile & Debug
    - 40 Zeichen/Zeile (für Fernseher etc.)
    - 80 Zeichen/Zeile
    - Debug-Info an (für Einzelschritt, Fehlersuche)
    - Debug-Info aus
  - Speichern CAS (Cassettenrecorder Routinen, jeweils Text und Daten)
  - Laden CAS
  - Prüfen CAS
  - Floppy Start (zum Laden eines Diskettenbetriebssystemes oder eines Anwenderprogrammes von Floppy)
  - EPROM programmieren
  - EPROM lesen
  - Speicherbereiche (RAM und EPROM Bereiche herausfinden)
  - Text drucken (auf Drucker mit Centronics-Schnittstelle)
  - IO lesen (Eingabe von externem Gerät oder spezielle Baugruppenfunktion einlesen)
  - IO setzen (externes Gerät oder spezielle Baugruppenfunktion steuern)
  - Einzelschritt (Debug-Möglichkeiten mit Registeranzeige, Möglichkeit des Setzen von Break-Points usw.)

6.3 JADOS

Das Diskettenbetriebssystem JADOS ist eine preisgünstige Alternative zu CP/M68K. Es erscheint dem Anwender in der Bedienung sogar häufig dem CP/M68K sehr ähnlich zu sein, obwohl die Dateien in völlig unterschiedlicher Anordnung auf der Diskette verwaltet werden.

JADOS V2.x bietet dem Anwender folgende Möglichkeiten:

* Laden, Speichern und Löschen von Dateien, Texten und Programmen
* Die Bereiche gelöschter Dateien stehen wieder uneingeschränkt zur Verfügung
* Koppeln von Texten; manuell oder über Antwortdatei
* Ändern von Dateinamen
* Selektives Inhaltsverzeichnis
* Laufwerksumschaltung über Tastatur
* Automatische Abarbeitung einer Startdatei nach dem Booten
* Volle Unterstützung des Assemblers von R.-D. Klein
* Volle Unterstützung des Editors von R.-D. Klein
* An jeder beliebigen Stelle, an der eine Tastatureingabe möglich ist, kann mit der Tastenkombination Ctrl-P eine Hardcopy des Text-Bildschirminhaltes auf den Drucker erfolgen
* Es wird die Baugruppe SOUND unterstützt, sowohl im Kommandomodus als auch für den Programmierer
* Superschnelle Textausgabe mit Vor- und Rückwärtsblättern
* Formatieren von Disketten im NDR80-Format
* Sehr komfortable Druck-Funktion
* Laden von Programmen mit Autostart
* Unterstützung der CPUs 68008, 68000 und 68020 !!
* Komplette Kommandoabläufe können mit Hilfe von Kommandodateien automatisiert werden; ähnlich Submit bei CP/M
* Deutsche Bedienerführung und Fehlermeldungen
* Umfangreiches, klar verständliches deutsches Handbuch
* Sehr gutes Preis/Leistungsverhältnis

6.2 Anwenderprogramm

Mit dem Boot-Programm des des EB68020 ist es möglich, Anwenderprogramme im EPROM statt des Grundprogrammes zu starten. Diese müssen als erste Kennung $a55a8002 besitzen und werden direkt dahinter gestartet. Diese Anwenderprogramme müssen sich dann um den Stack und die Speicherverwaltung selber kümmern.

Beispiele sind selbstgeschriebene Grundprogramme, aber auch spezielle Steuerprogramme (z.B. Ablaufsteuerung in einer Fertigungsstraße), die sofort nach dem Einschalten bzw. einem Spannungsausfall anlaufen müssen.

6.4 CP/M68K

Das Diskettenbetriebssystem CP/M68K wird von der Firma Digital Research vertrieben. Es ist sehr weit verbreitet, da es auf nahezu allen 68000-Computern lauffähig ist.

Im Lieferumfang enthalten:
- Zeileneditor
- C-Compiler
- 68K-Assembler
- Linker
- Debugger
- Universal-Kopierprogramm PIP
- Bios-Quellistings (allgemein)
- Beispiele zum CP/M68K, C und Assembler

Bei den von uns ausgelieferten Disketten sind zusätzliche Programme vorhanden:
- Bildschirmorientierter Texteditor (wie Grundprogramm)
- Aufruf des Grundprogramm-Assemblers
- BIOS-Quellisting für den NDR-Computer
- Universal-Formatierprogramm
- Terminal-Programm für Minimal-Modembetrieb als Quellisting

6.4.1 Anpassung des CP/M68K an den 68020

Leider funktioniert das CP/M68K in der ausgelieferten Version nicht mit der CPU68020. Das System muß erst neu konfiguriert werden.

Dazu muessen Sie in dem File **MAKECPM.SUB** folgende Zeile durch die nächste ersetzen:

lo68.rel -r -ucpm -o cpm.rel cpmlib ndrbios.o **(alt)**

lo68.rel -r -um68010 -ucpm -o cpm.rel cpmlib ndrbios.o **(neu)**

Dann müssen Sie das File MAKECPM.SUB durch folgende Eingabe starten:

makecpm

Jetzt wird automatisch ein neues System zusammengestellt, mit dem auch der 68020 zurechtkommt.

6.5 Modula-2

Unter dem Betriebssystem CP/M68K ist ein MODULA-2 Compiler lieferbar.

MODULA-2 ist eine konsequente Weiterentwicklung von Pascal und findet weltweit in jüngster Zeit immer größere Anerkennung. Die Sprache bietet unter anderem folgende wichtige Neuerungen:

- Ein Modulkonzept, das es ermöglicht, auf bequeme Weise ein Programm in kleinere Teile zu zerlegen, wobei zwischen der Schnittstellenbeschreibung (DEFINITION MODULE) und dem eigentlichen Programmteil (IMPLEMENTATION MODULE) unterschieden wird. Die Einhaltung dieser Schnittstellen wird überprüft.
- Eine systematische Syntax, z.B. ein END zur Klammerung aller zusammengesetzten Anweisungen.
- Sehr gute Unterstützung der systemnahen Programmierung, z.B.: Adreßrechnung mit ADDRESS, die Arten WORD und BYTE, erzwungene Artanpassungen, Prozedurvariable, Koroutinen für parallele Prozesse.
- Felder variabler Länge als Prozedurparameter (z.B. für Stringverarbeitung).

Die MODULA-2-Implementierung auf dem NDR-Computer enthält neben dem kompletten Sprachumfang, wie in dem Buch 'Programmieren in Modula-2' von N. Wirth (3rd Edition 1985, Springer-Verlag) beschrieben, auch die von Wirth in 'Revisions and Amendments to Modula-2' (in Modula-2 News #0, 1984, Modula-2 User Association) angeführten Erweiterungen. Insbesondere wurden die Datentypen LONGCARD, LONGINT für 32-Bit-Arithmetik und große Sets mit bis zu 256 Elementen realisiert.

Der Compiler erzeugt optimierten 68000-Code (als Assembler-Quellen) und enthält die Möglichkeit, Assembler- und C-Programme anzubinden.

Der Floating-Point Coprozessor 68881 wird, wenn vorhanden, unterstützt!

7. Literatur

7.1 Hinweis auf LOOP

In unserer Zeitschrift LOOP wird regelmäßig über neue Produkte und Änderungen bzw. Verbesserungen berichtet. Es ist für Sie von großem Vorteil, LOOP zu abonnieren, denn dadurch ist sichergestellt, daß Sie auch immer über die neuesten Informationen verfügen.

Ein LOOP-ABO können Sie bei jeder Bestellung einfach mitbestellen.

7.2 Die Zeitschrift mc

Die Zeitschrift mc des FRANZIS-Verlages erscheint regelmäßig jeden Monat. In der Ausgabe 7/1986 hat eine Artikelserie über die Baugruppe CPU68020, den Mikroprozessor 68020 und den Floating-Point Coprozessor 68881 begonnen.

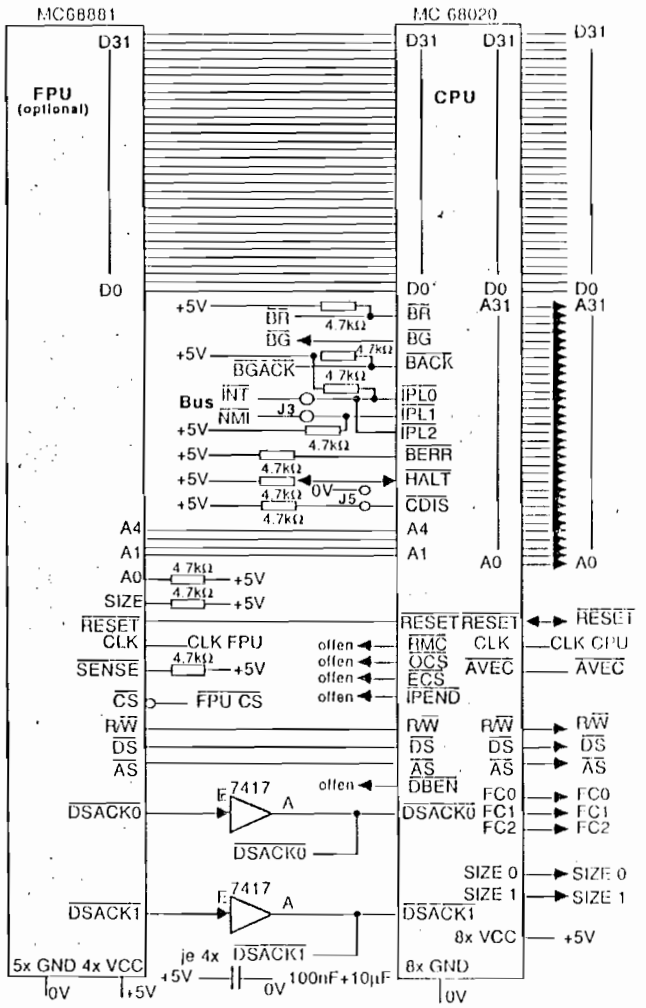
7.3 Empfohlene Fachbücher

Direkt von der Firma Motorola gibt es zwei sehr zu empfehlende Datenbücher zum 68020 und 68881. Diese enthalten sämtliche Hard- und Software Beschreibungen zu diesen beiden ICs, sind allerdings englischsprachig.

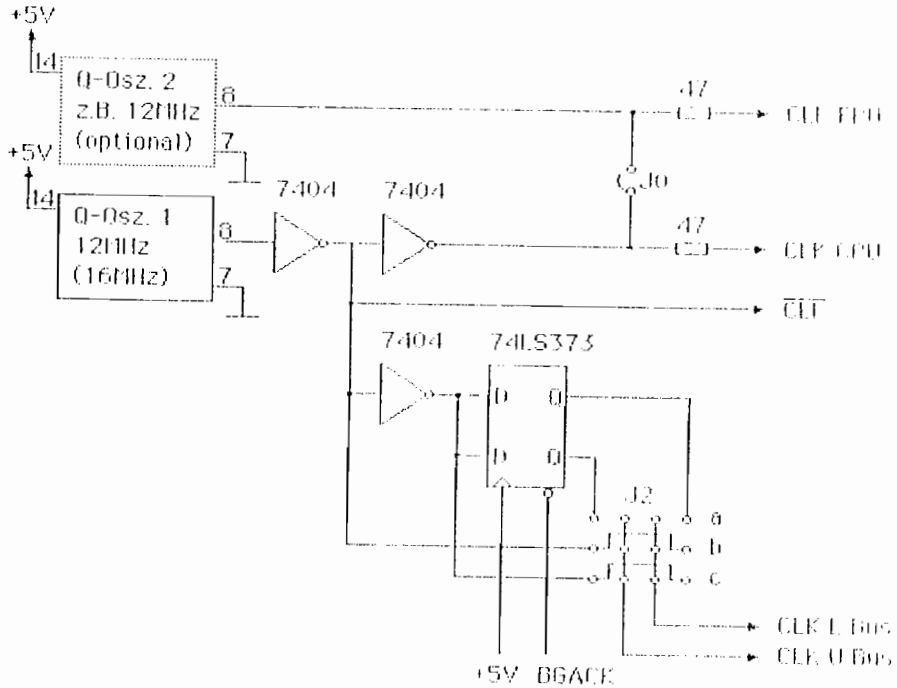
Beide Bücher sind bei uns erhältlich:

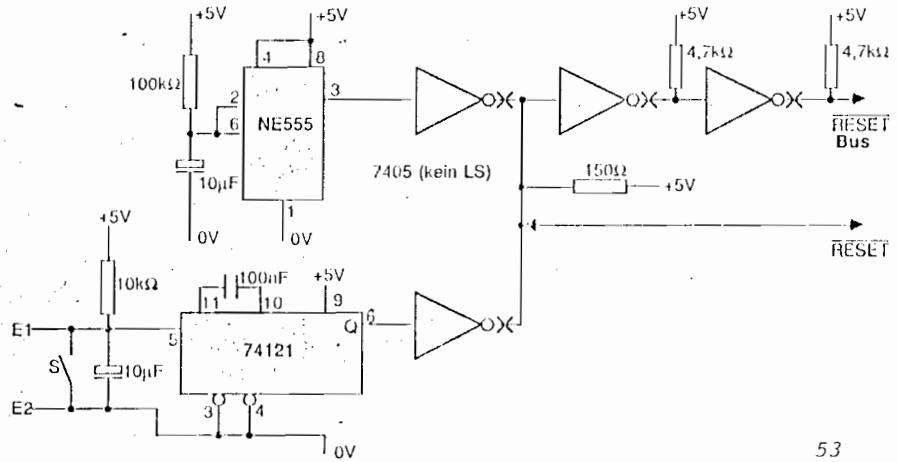
- *MC68020 Datenbuch, Bestell-Nr. 10619*

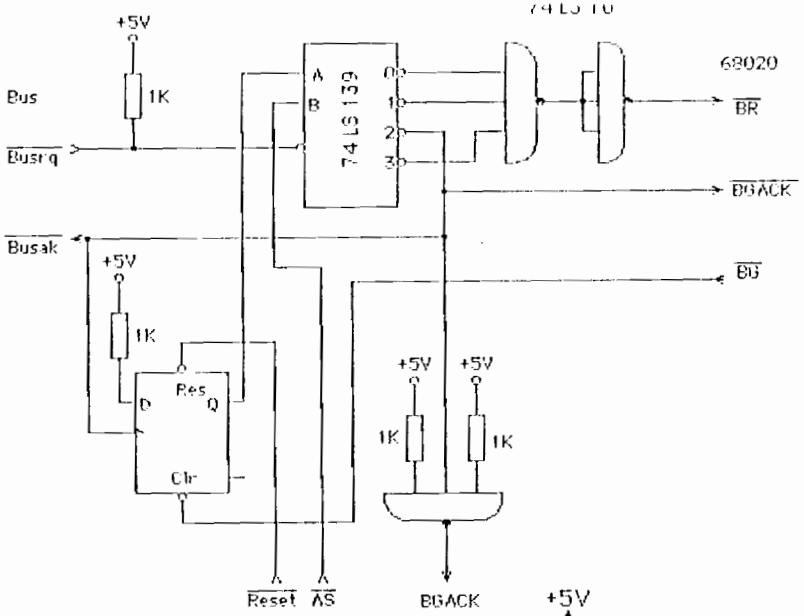
- *MC68881 Datenbuch, Bestell-Nr. 10620*

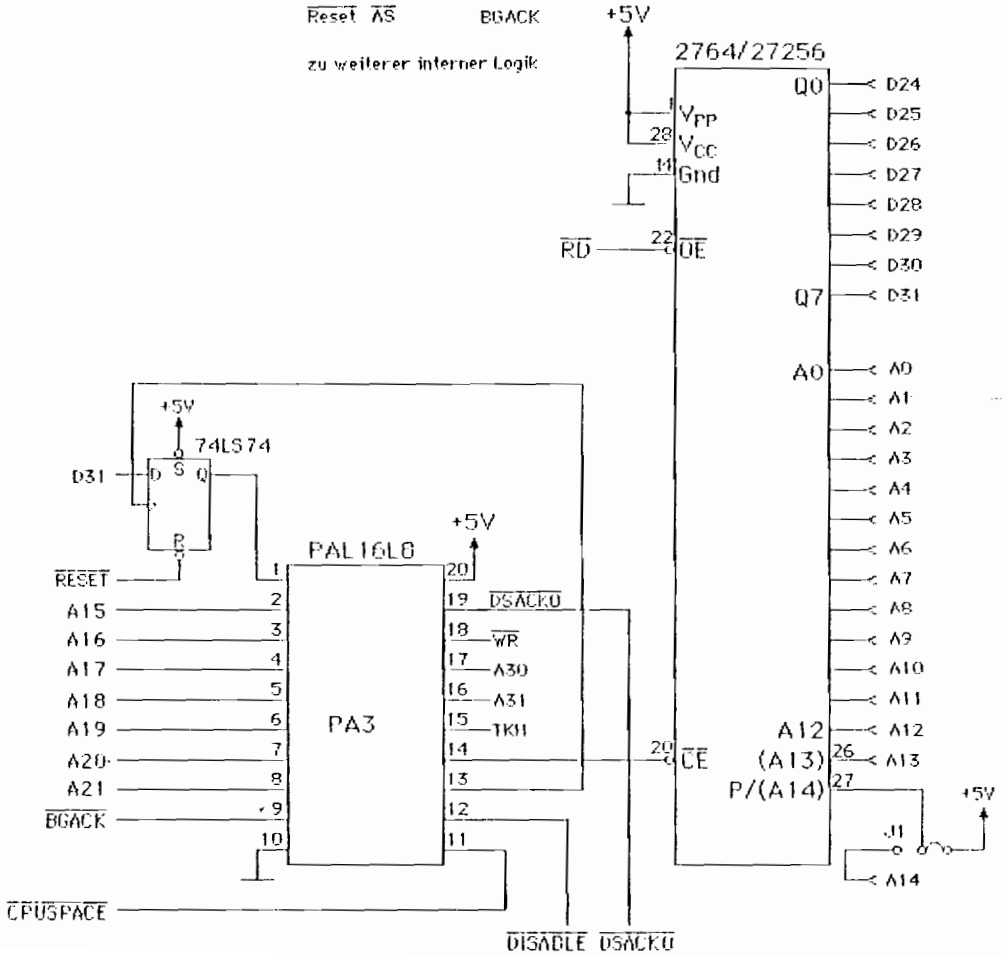


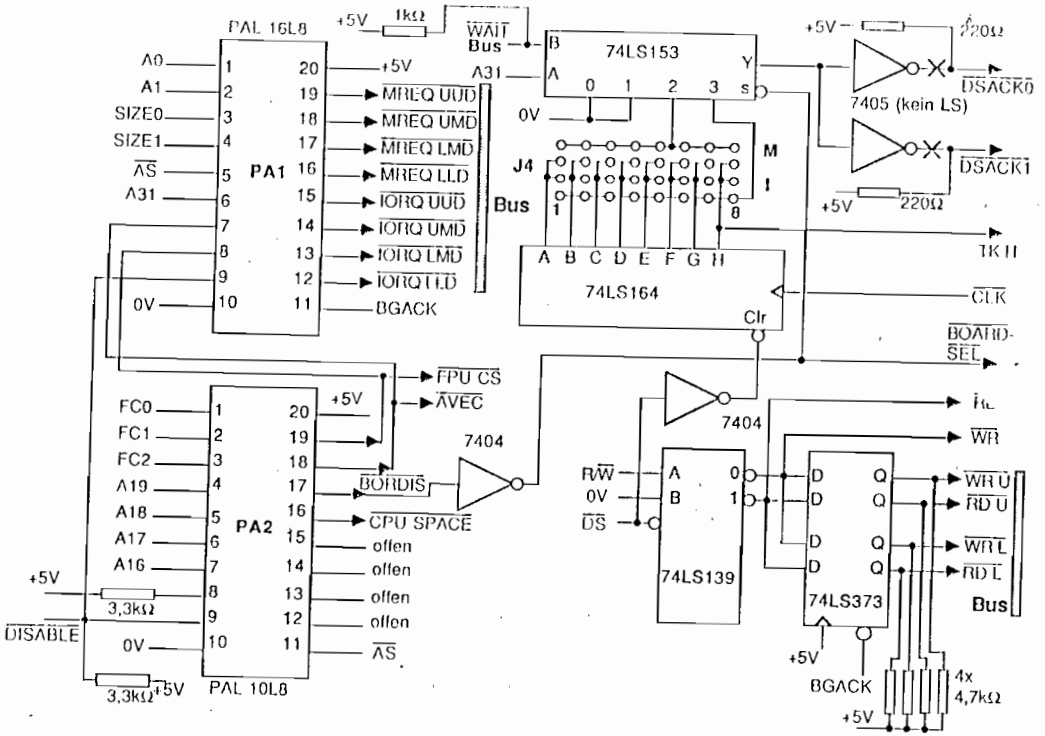
Anhang A: Gesamtschaltplan CPU68020






74LS10
+5V
Bus
1K
Busrq
74LS139
A
B
0
1
2
3
68020
BR
BGACK
Busak
BG
+5V
1K
Res
D
Q
Clr
+5V
+5V
1K
1K
Reset
AS
BGACK
zu weiterer interner Logik


+5V
2764/27256
VPP
28
VCC
14
Gnd
RD
22
OE
Q0
Q7
A0
D24
D25
D26
D27
D28
D29
D30
D31
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A12
(A13)
P/(A14)
26
27
CE
20
+5V
J1
74LS74
D
S
Q
R
D31
+5V
PAL16L8
RESET
A15
A16
A17
A18
A19
A20
A21
BGACK
PA3
1
2
3
4
5
6
7
8
9
10
20
19
18
17
16
15
14
13
12
11
DSACK0
WR
A30
A31
TKH
CPUSPACE
DISABLE
DSACK0


PAL 16L8
+5V
1kΩ
WAIT
Bus
A0
A1
SIZE0
SIZE1
AS
A31
0V
PA1
MREQ UUD
MREQ UMD
MREQ LMD
MREQ LLD
IORQ UUD
IORQ UMD
IORQ LMD
IORQ LLD
BGACK
Bus
74LS153
A31
0V
J4
M
I
74LS164
A B C D E F G H
Clr
CLK
TKH
BOARD-SEL
FPU CS
AVEC
FC0
FC1
FC2
A19
A18
A17
A16
PA2
BORDIS
CPU SPACE
offen
offen
offen
offen
AS
7404
3,3kΩ
DISABLE
PAL 10L8
R/W
0V
DS
74LS139
74LS373
WR U
RD U
WR L
RD L
Bus
BGACK
4x
4,7kΩ
7405 (kein LS)
220Ω
DSACK0
DSACK1