

Die PAK-68 im Amiga.

Beschreibung der Prozessor-Austausch-Karte aus der Zeitschrift c't

Richtig anPAKken - PAK-68 im Amiga: Probleme und Lösungen		Seite 2
Tausch mit Turbo - PAK-68/2 - mehr Dampf für 68000-Rechner	Teil 1	Seite 10
Tausch mit Turbo - PAK-68/2 - PAK-68/2 in Mac, Amiga und Atari	Teil 2	Seite 21
Breit-bereit - Schnelles 32-Bit-RAM für die PAK68/2 im Amiga		Seite 35
Doppel-PAK - PAK-68/3 mit Cache u. 68020/030 für 68000-Rechner	Teil 1	Seite 39
Doppel-PAK - PAK-68/3 mit Cache u. 68020/030 für 68000-Rechner	Teil 2	Seite 56

Richtig anPAKken

PAK-68 im Amiga: Probleme und Lösungen

Jürgen Wentz

Das sogenannte Rechnertuning steht zur Zeit hoch im Kurs. Eine effektive Möglichkeit, die 'Freundin' Marke Commodore zu beschleunigen, ist der Einsatz eines Motorola MC68020, zum Beispiel mit der PAK-68. Da die PAK-68 allgemein für 68000-Systeme entwickelt wurde, gibt es bei der Installation in den Amiga ein paar systemspezifische Hürden. Eine bessere Integration des 68020 in das Betriebssystem des Amiga überwindet schon einige dieser Hürden.

Die ersten Schwierigkeiten bei der Inbetriebnahme der PAK-68 [1] können schon beim mechanischen Aufbau eintreten. Während der Einbau in den Amiga 2000 problemlos erfolgt (einfach den alten Prozessor heraushebeln und die PAK-68 einstecken), steht diesem Unterfangen beim Amiga 500 die beschränkte Höhe des Gehäuses entgegen. Benutzer dieses Modells müssen sich leider nach einem neuen Gehäuse umsehen. Beim alten Amiga 1000 geht es sehr eng zu, denn die Floppy ist im Wege; jedoch paßt die PAK-68 gerade noch hinein [5].

Falls man die Floating Point Unit (FPU, MC68881) gleich mitgekauft hat, kann man den getunten Amiga sofort in Betrieb nehmen. Kickstart und Workbench sind ab Version 1.2 für diese Prozessorerweiterung ausgelegt, das heißt, der Rechner sollte hochfahren und alle Workbench-Funktionen und Programme ohne 'Guru' bearbeiten können. Mit dem Performance Monitor (PM) der 'ExtrasD'-Diskette kann man sich auf einfache Weise überzeugen, daß der 68020 vom Amiga erkannt wurde. Statt mit dem gewöhnlichen Text '68000 idle' wird der Prozessor-Leerlauf nun mit '68020 idle' verkündet.

Soll aber zuerst die 'Billiglösung', also der 68020 allein, für mehr Leistung sorgen, fängt der Ärger an. Das Betriebssystem des Amiga enthält in der Reset-Routine der Kickstart ein kleines Unterprogramm, das feststellen soll, welche Prozessorkonfiguration sich im Rechner befindet.

Beim ROM-Kickstart, Version 1.2 33.180 (fest installiert in den Modellen 500 und 2000, welche mit Kickstart 1.2 ausgeliefert wurden), erfolgt der Einsprung in das hier wiedergegebene Unterprogramm bei \$FC02A8. Das Ergebnis der Routine wird in Register D0 übergeben und mit dem Offset \$128 zur ExecBase gespeichert.

Wer ist da?

Hier können User-Programme einfach abfragen, welcher Prozessor eingesetzt ist, um kritische Befehle zu vermeiden - was bisher leider nur wenige tun. Das Amiga-Betriebssystem selbst benutzt diese als 'AttnFlgs' bezeichnete Variable im weiteren Verlauf der Reset-Routine, um die innerhalb der Prozessorfamilie bestehenden kleinen Unterschiede weitgehend abzufangen.

Die Arbeitsweise dieser Testroutine ist ebenso elegant wie einfach. Es werden bewußt Befehle benutzt, welche nicht allen Prozessoren der Familie geläufig sind. Falls nun der aktuelle Prozessor auf einen ihm unbekannten Operationscode stößt, wird eine sogenannte Exception (Ausnahmezustand) ausgelöst. Aber gerade diese Exception-Vektoren werden von der Prozessor-Testroutine auf ihr eigenes Ende verbogen. Im Klartext bedeutet dies: beim ersten, für den aktuellen Prozessor unbekannten Befehl bricht die Prozessor-Testroutine ab, im anderen Fall wird das nächste Mitglied der Familie ausprobiert.

Versucht man, die PAK-68 ohne FPU zu betreiben, entsteht hierbei ein kleines Mißverständnis. Der Amiga testet den MC68881 mit einem Fließkommabefehl. Die PAK-68 meint generell, daß die FPU vorhanden sei, und signalisiert daher keinen Fehler, was den MC68020 zum Auslösen der Exception veranlassen würde. Es entsteht die unglückliche Situation, daß der Amiga bis zum Stromentzug auf die Antwort des nicht vorhandenen Fließkomma-Prozessors wartet.

Man kann dieses Fehlverhalten zum einen mit einer Modifikation an der PAK-68 beseitigen, welche das zum Auslösen der Exception notwendige Fehlersignal liefert. Dazu führt man einfach das Signal /CS (Chip Select), das an den Coprozessor geht, an die Leitung /BERR am 68020 ; /CS ist Pin J3 des Coprozessor-Steckplatzes. Die Diode (4148) ist notwendig, damit es keinen Kurzschluß gibt, wenn ein anderer Baustein einen Bus-Error auslöst.

Es besteht aber auch die Möglichkeit, das Problem durch Patchen des Betriebssystems (Kickstart) zu umfahren. Die hierzu nötigen Schritte werden im weiteren Verlauf des Artikels noch eingehend erklärt.

Der Patch ist sehr einfach: es genügt, die Befehlsworte zwischen \$FC0576 bis einschließlich \$FC0580 durch \$4E71, also den NOP-Befehl des Prozessors, zu ersetzen. Da die nicht vorhandene Floating Point Unit nun nicht mehr angesprochen wird, kann auch das Endloswarten nicht mehr vorkommen, und im Ergebniswort der Routine wird der MC68881 korrekt als nicht vorhanden eingetragen.

Zu viel Speicher

Wenn der Amiga selbst nun auch einwandfrei mit der PAK-68 zusammenarbeitet, kann es bei diversen Systemerweiterungen zu weiteren Schwierigkeiten kommen. Zum einen gab es zeitweise Probleme mit der c't-Festplattenlösung, welche in [3] erkannt und gelöst wurden, zum anderen gab es seit Anbeginn der 'Pa(c)k-erei' immer wieder Ärger mit Speichererweiterungen.

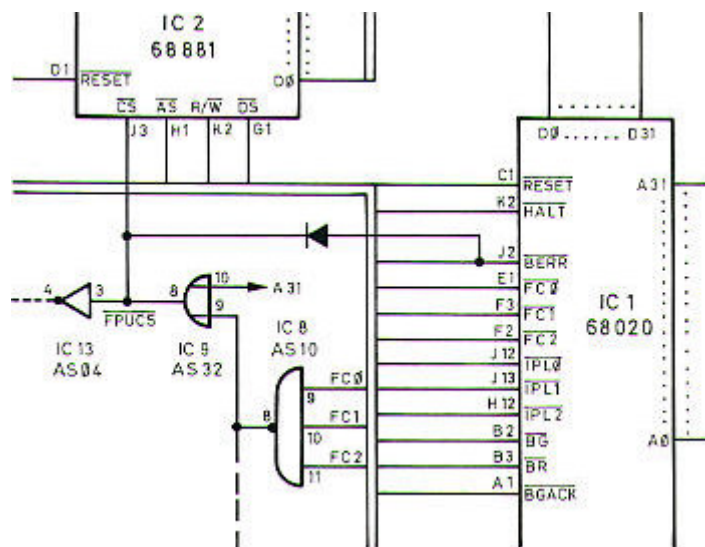
Im Gegensatz zum internen Speicher des Amiga, der viele Manipulation an der PAK-68 problemlos hinnimmt, bestehen viele Speichererweiterungskarten auf einem wesentlich genauer eingehaltenen Protokoll beim zeitlichen Ablauf (timing) für die bei einem RAM-Zugriff nötigen Signale. Bei einigen Karten konnten die Zugriffsprobleme durch eine Verzögerung des /AS-Signals (Gültigkeitserklärung der Adressen) gelöst werden [4], bei anderen wurde die Fehlerrate hierdurch eher größer.

Durch Untersuchung der RAM-Zyklen der PAK-68 und denen eines 68000 mit Hilfe eines Logic-Analyzers konnte ich eine weitere Ursache für das Fehlverhalten einiger RAM-Karten aufdecken. Die PAK-68, das heißt im wesentlichen der 68020, generiert die für den Datenverkehr mit den RAMs nötigen Signale im Gegensatz zum 68000 um 180° phasenverschoben zum Takt. Mit dieser Erkenntnis lassen sich die Probleme mancher Speichererweiterungskarten, welche interne Signale abhängig vom Clock generieren, sehr leicht durch einen Inverterbaustein (z. B. 74F04) lösen.

Auf einem kleinen Stück Lochraster-Platine wird der 74F04 nebst einem Kondensator über der Spannungsversorgung aufgebaut, und einer der 6 Inverter, zum Beispiel Pin 1 (Eingang) und Pin 2 (Ausgang), zusätzlich zu den Kabeln für +5 V und GND mit Anschlußkabeln versehen. Der Inverter wird nun anstelle des als Treiber geschalteten AND-Gatters (Pins 12, 13 und 11 von IC 6) in die Schaltung integriert.

Hierzu biegen Sie die Pins 12, 13 und 11 des alten AND-Gatters ab, um dieses totzulegen. Auf den Lötäugen an der Unterseite der PAK-68 kann man nun die Kabel der kleinen Inverterschaltung anlöten; wobei Pin 1 des Inverters mit dem Lötauge von Pin 12 oder 13 des alten AND und Pin 2 des Inverters mit dem Lötauge für Pin 11 des AND verbunden werden.

Selbstverständlich ist auf Grund der relativ hohen Frequenz (7,16 MHz) auf kurze Kabelwege zu achten. Nachdem man noch einmal überprüft hat, daß bei der Löterei keine Kurzschlüsse entstanden sind, sollte sich die PAK-68 nun auch mit Speichererweiterungen vertragen.



Diese Änderung sorgt für das Fehlersignal bei fehlendem Coprozessor.

Verbotene Befehle

Jetzt sollte man doch ungetr bt `32bittig` loslegen k nnen. Denkste! Nicht jedes liebgewonnene Programm verh lt sich so brav wie die Workbench. Denn die Prozessoren der 680x0-Familie verhalten sich leider (notwendigerweise) nicht in allen Lebenslagen gleich. W hrend der 68000 ein Auslesen des Statusregisters SR (move sr,<ea>) im User-Modus  ber sich ergehen l sst, ohne mit der Wimper zu zucken, mu te bei der Weiterentwicklung des Prozessors dieser Befehl in den Supervisor-Modus verlegt werden.

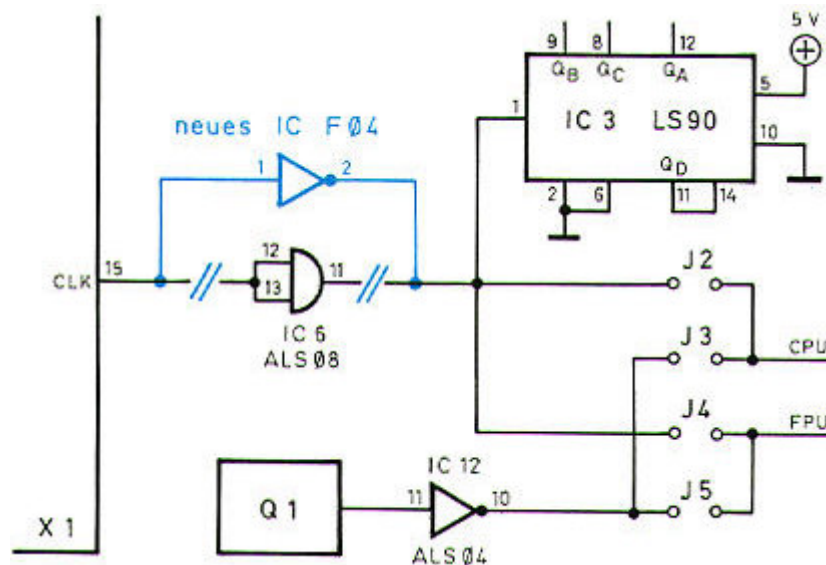
Dies ist eine notwendige Sicherheitsanforderung f r den Einsatz des Prozessors bei multiuserf higen Betriebssystemen. Ein ordentlich geschriebenes Programm f r den 68000 sollte daher diesen Befehl nur im Supervisor-Modus benutzen und sich im User-Modus auf den eigentlich ausreichenden Befehl move ccr,<ea> beschr nken, welcher ja auch die gew nschten Flags holt.

Was tut nun ein MC68010 oder MC68020, wenn ihm ein Programm im User-Modus partout den verbotenen Befehl vorsetzt? Er antwortet mit der daf r vorgesehenen Exception f r eine Privilegverletzung und springt  ber den an der Speicherstelle \$20 stehenden Vektor. An dieser Stelle sollte ein gutes Betriebssystem in einen Handler verzweigen, der den Benutzer auf den Fehler hinweist, die Task schlie t und die angeforderten Ressourcen zur ckgibt.

Beim Amiga sind diese Handler leider etwas spartanisch ausgefallen. Im Normalfall bestehen sie aus einem `JWR` (Jump Without Return) in den wohlbekannten Requester `Software error - Task held` oder enden gleich in der Guru-Meditation des Rechners.

Mittlerweile gibt es eine Reihe von Programmen, die entsprechende Handler anbieten. Als gutes Beispiel sei hier `Gomf` erw hnt, das im Falle der Privilegverletzung dem Anwender diese angezeigt und die Task aus dem System wirft. F r unser Problem ist dies leider nicht das Ei des Columbus, da wir ja nicht das Programm aus dem Rechner werfen wollen, im Gegenteil, das Programm soll so arbeiten, als w rde es am 68000 laufen.

Das Problem l sst sich mit dem in [4] vorgeschlagenen zus tzlichen Handler l sen, welcher per Programm installiert wird. Dies hilft aber leider nicht immer, denn wie installiere ich den Handler bei einem autobootenden Programm? Zu diesem Zweck habe ich einen Patch f r die Kickstart entwickelt, bei dem der Handler immer aktiviert wird, wenn ein 68020-Prozessor im System arbeitet. Dies ist wohl die universellste Methode, der Privilegverletzung Herr zu werden.



**68000 und 68020 generieren Signale phasenverschoben.
Ein Inverter biegt dies so weit hin, da  auch Speicherkarten mit dem 68020 laufen.**

Keine Schleifen

Der eigentlich erwünschte Leistungszuwachs des Amiga ist leider auch Ursache für eine weitere Inkompatibilität. Es gibt nämlich nicht wenige Programme, die Wartezeiten etwa bei parallel laufenden Prozessen in Leerschleifen verbringen. Dies führt beim 68020, der eine Leerschleife mit Hilfe seines prozessor-internen Speichers (Cache) und des Prefetches (Dekodieren eines Befehls, während der letzte noch bearbeitet wird) bis zu einem Faktor 2 schneller abarbeitet, unausweichlich zum Chaos.

Es wäre natürlich vermeidbar, wenn die Programme die entsprechenden Systemroutinen oder Timer-Bausteine zum Warten benutzen würden - aber ... Da dies nun mal nicht immer der Fall ist, muß der 68020, und das bedeutet im Falle der PAK-68 der Cache, heruntergebremsst werden. Das Ausschalten des Cache kann zum Beispiel mit dem in [5] vorgestellten kleinen Programm geschehen. Aufgrund eines kleinen Fehlers im alten Handler arbeitet es leider nicht, wenn dieser installiert ist; der alte Exception-Handler schaltet den Cache ständig wieder an. Mit dem hier gezeigten neuen, in die Kickstart integrierten Handler passiert dies nun nicht mehr.

Bei den meisten zeitkritischen Programmen reicht es völlig aus, innerhalb einer kleinen Batchdatei den Cache vor Aufruf des eigentlichen Programmes aus- und anschließend wieder einzuschalten. Für autobootende Programme ist dies aber keine Lösung. Daher wurde das Cache-Aus mit einer simplen Abfrage der linken Maustaste in die Kickstart integriert. Hält man während des Resets, also bevor die Floppy anläuft, die linke Maustaste gedrückt, so ist der Cache so lange aus, bis er explizit durch den nächsten Reset oder das Cache-an-Programm wieder aktiviert wird.

Das mit dem Handler korrespondierende Listing besteht im wesentlichen nur aus einem Jump in den Handler beziehungsweise dessen Einbindung. Diese Springerei ist nicht besonders elegant, aber der einzig praktikable Weg, da innerhalb der Kickstart zu viele Adressen umgesetzt werden müßten, falls man einfach etwas einschieben wollte.

Der Aussprung erfolgt an dem Punkt, an dem der Amiga gerade die Zusatzfunktionen für den MC68020 einrichtet, der zusätzliche Handler wird auch nur aktiv, wenn die PAK-68 installiert ist. Nach dem Aussprung müssen zuerst die durch den JMP-Befehl überschriebenen Bytes nachgeholt werden. Danach wird der Privilegverletzungs-Vektor auf die neue Handler-Routine umgebogen, in Abhängigkeit vom linken Mausknopf der Cache ausgeschaltet und in der Resetroutine wieder aufgesetzt. Kommt es nun zu einer Privilegverletzung im System, so tritt der neue Handler in Aktion.

Dieser rettet zuerst die von ihm benötigten Register auf den Stack und setzt den Cache zurück. Hier liegt auch der Unterschied zum ursprünglichen Handler; dieser schrieb einfach den Wert 9 in das 'Cache Control Register' (CACR). Bei diesem Parameter wird der Cache aber nicht nur zurückgesetzt, sondern auch wieder angeworfen, egal ob er ein- oder ausgeschaltet war.

Im neuen Handler hingegen wird erst der momentane Zustand des CACR gelesen, Bit 3 (Zurücksetzen des Caches) gesetzt und durch Zurückschreiben in das CACR ausgeführt. Der Zustand des Cache (Bit 0: ein oder aus) wird hierdurch nicht verändert. Nun wird der die Exception auslösende Opcode vom Stack geholt und untersucht, ob es ein 'move sr,<ea>' war. Falls ja, wird er in einen unschädlichen 'move ccr,<ea>' umgewandelt, falls nicht, springt der Handler nach dem Wiederherstellen der Register in den Original-Handler des Amiga ein.

Do it

Die beschriebenen Routinen müssen nun der Kickstart einverleibt werden. Besitzer der Modelle Amiga 500 und 2000 haben es hier schwerer. Im Gegensatz zum Amiga 1000, bei welchem nur die Kickstart-Diskette zu ändern ist, müssen sie neue ROMs generieren; oder aber die Kickstart in 28-Pin-EPROMs bannen, die auf einer (erhältlichen) Adapterplatine betrieben werden.

Dies ist natürlich insbesondere beim Ausprobieren mühsam. Der empfohlene Weg ist daher, das Ganze erst einmal beim Amiga 1000 eines Freundes mit Disketten auszuprobieren und anschließend erst in die (E)PROMs zu brennen.

Für das Patchen der Kickstart-Diskette benötigen Sie einen Diskmonitor oder einen normalen Monitor mit 'Read/Write'-Option für Diskettenblöcke. Die Kickstart-Diskette ist im normalen Amiga-Format beschrieben und kann daher mit den Standardfunktionen bearbeitet werden.

Die Organisation der Daten selbst unterscheidet sich allerdings von der bei normalen Disketten. In Block 0 steht zur Kennung der Diskette nur das Wort 'KICK', der Rest des Blocks ist mit Nullen gefüllt. Die 256 KB Daten der Kickstart folgen ab Block 1, und zwar mit vollen 512 Byte Inhalt pro Block. Die Blöcke sind einfach der Reihe nach angeordnet, sie enthalten keine Zeiger oder Kennungen. Auch die sonst übliche Prüfsumme eines Diskettenblocks entfällt.

Dieser Punkt muß beim Editieren der Kickstart unbedingt beachtet werden. Man muß die veränderten Blöcke ohne Neuberechnung der Checksumme abspeichern. Falls der Diskmonitor eine solche berechnet und speichert, überschreibt er damit ein Daten-Langwort.

Für den Patch selbst müssen zunächst die in den Listings angegebenen absoluten Adressen überprüft werden. Zu diesem Zweck startet man den Computer mit der zu patchenden Kickstart. Der Patch sollte selbstverständlich nur an einer Kopie probiert werden. Mit einem Monitor (z. B. 'C-Mon') überprüft man zuerst einmal die Position des Original-Handlers.

Den Zeiger auf diesen finden Sie bei der absoluten Adresse \$20; falls hier \$00FC08D2 steht, ist das schon ein guter Anfang. Für den Patch muß nun noch ein Platz in der Kickstart gefunden werden. Innerhalb der 256 KB ist aber leider nicht allzuviel freier Raum, also muß irgend etwas herausfliegen.

Was aber wirft man raus, ohne es anschließend doch schmerzlich zu vermissen? Ein Programm innerhalb der Kickstart bietet sich hierfür besonders an. Es ist der Debugger RomWack, mit dem man beim Guru-Zustand des Amiga über die serielle Schnittstelle mittels eines zweiten Rechners oder Terminals im System nach der Ursache des Gurus fahnden kann. Da die meisten Amiga-Besitzer mangels Terminal diese Möglichkeit nicht nutzen, scheint hier eine entbehrliche Routine gefunden zu sein.

Die Position des RomWack im Amiga steht in dem JMP-Befehl bei Offset -114 zur ExecBase, deren Position über den Zeiger an Speicherstelle 0004 zu finden ist; die Jump-Adresse sollte \$00FC232E sein. Zum dritten und letzten sollte man noch überprüfen, ob sich der abgedruckte Auszug aus der Resetroutine (bis auf den JMP) an der dort angegebenen Position befindet.

Falls diese Adressen alle stimmen, kann man damit beginnen, den Code für den neuen Handler per Diskmonitor in die Kopie der Kickstart einzugeben. Der Patch startet bei Offset \$12E in Block 18. Nachdem dies getan ist, speichert man den Block ohne Neuberechnung der Prüfsumme ab.

Der Patch ist damit zwar noch nicht vollständig, es empfiehlt sich aber, die Kickstart vor dem Weiterpatchen erst einmal auszuprobieren. Also den Rechner ausschalten und ihn mit der neuen Kickstart anfahren. Geht dies, so ist zumindest nichts Lebenswichtiges überschrieben.

Zum weiteren Testen kann der Handler per Hand in das System eingebunden werden, indem man mit einem Monitor die Speicherstelle \$20 mit \$00FC2330, also der Adresse des neuen Handlers, überschreibt (Vorsicht: Das Langwort muß auf einen Rutsch geschrieben werden). Stürzt der Rechner hierdurch nicht ab und überlebt kritische Programme, wie den alten Taschenrechner der Workbench 1.1, so läuft schon mal der neue Handler.

Der zweite Teil des Patches, der JMP-Befehl in der Resetroutine, befindet sich in Block 2 bei Offset \$1F0. Nun ist der Patch eigentlich komplett, bis auf ... Denn nach einem Reset verlangt der Rechner wiederum die Kickstart. Mit der ganzen Patcherei wurde nämlich die Gesamtprüfsumme der Kickstart ungültig, welche in der Resetroutine überprüft wird.

Eine unkorrekte Kickstart bootet erst mal klaglos, beim nächsten Reset nimmt der Amiga aber aufgrund der falschen Kickstart an, daß der Kickstart-Bereich (\$FC0000 bis \$FFFFFF) zerstört wurde, und verlangt wieder die Kickstart-Diskette.

Falls Sie den Patch inklusive der NOPs an exakt den abgedruckten Adressen benutzen konnten, also auf die ROM-Version der Kickstart 1.2 angewendet haben, so können Sie ohne Umstände die Prüfsumme der Kickstart durch Patchen der Adresse \$FFFFE8 mit \$4C2643A2 korrigieren. Auf der Diskette befindet sich dieses Langwort in Block 512 bei Offset \$1E8.

Selber suchen

Falls eine andere Kickstart-Version vorliegt, so sind lediglich folgende Punkte zu beachten: die Position des RomWack kann verschoben sein. In diesem Fall muß der Handler an die entsprechende Stelle verschoben sowie der Einsprungvektor (in der Resetroutine) auf die neue Position mitverändert werden.

Die Position dieser Resetroutine kann sich an leicht veränderter Position befinden, in diesem Fall muß der JMP-Befehl an der gleichen logischen Position eingebaut und der Rücksprung am Ende des neuen Handlers entsprechend angepaßt werden.

Der letzte mögliche Unterschied ist die Position des Original-Handlers, aufgerufen über den Vektor bei \$20. Falls erforderlich, muß der entsprechende JMP-Befehl im neuen Handler hierauf umgebogen werden. Eine Änderung macht natürlich auch eine Neuberechnung der Kickstart-Prüfsumme nötig.

Die Kickstart verfügt, außer den Prüfsummen für diverse Listen, über eine generelle Prüfsumme für den gesamten 256-KB-Bereich. Die Bildung der Prüfsumme besteht in einer einfachen Addition aller Langworte mit einer gesonderten Behandlung des Übertrags. Falls diese Summe gleich 0 ist, geht das System von einer intakten Kickstart aus. Der Korrekturwert, um die Summe zu 0 zu machen, befindet sich bei der Speicherstelle \$FFFFE8. Die genaue Berechnung ist in dem letzten abgedruckten Programm gezeigt. Es bildet die Prüfsumme der als System geladenen Kickstart und errechnet den Korrekturwert, der dann im Register D1 steht.

Dieser Wert wird nun ebenfalls gepatcht, also einfach in den Block 512 ab Byte \$1E8 geschrieben. Die hieraus resultierende Kickstart überlebt dann auch einen Reset. Mit der neuen Kickstart sollte der Workstation-Freude auf Amiga-Basis nichts mehr entgegenstehen. (bb)

Literatur

- [1] J. Assenbaum, Mehr CPU, c't 8/87, S. 68
- [2] J. Assenbaum, PAK-MEM, c't 1/88, S. 214
- [3] D. Göhler, Floppy, nein danke, c't 2/89, S. 222
- [4] J. Assenbaum, GePAKt, c't 4/88, S. 172
- [5] D. Göhler, Unerwartetes vom Amiga, c't 10/87, S. 116

Kasten 1

Die Prozessor-Testroutine benutzt prozessorabhängige Befehle, um festzustellen, welcher Prozessor Dienst tut.

```
FC0546: 48e7 0030    movem.l  A3-A2,-(A7)           ;retten der Register
FC054a: 2078 0010    move.l   $0010,A0             ;Exception-Vektor für illegal
                                   ;instruction nach A0 retten
FC054e: 2478 002c    move.l   $002c,A2             ;Exception für Line 1111-Bef.
FC0552: 43fa 002e    lea      46(PC)($fc0582),A1   ;Endadr. der Subroutine
FC0556: 21c9 0010    move.l   A1,$0010             ;illegal-instruction-Vektor
                                   ;auf Ende der Routine biegen
FC055a: 21c9 002c    move.l   A1,$002c             ;Exception-Vektor für Line
                                   ;1111 auf Ende der Routine
FC055e: 224f          move.l   A7,A1             ;Stackpointer nach A1
FC0560: 7000          moveq    #$00,D0             ;D0 löschen
----- D0 = 0 für MC68000 -----
FC0562: 7200          moveq    #$00,d1             ;D1 löschen
FC0564: 4e7b 1801    movec    D1,VBR               ;D1 ins Vektor Base Register
                                   ;erst ab MC68010 implementiert
FC0568: 08c0 0000    bset     #0,D0                 ;Bit 0 in D0 setzen
----- D0 = 1 für MC68010 -----
FC056c: 7201          moveq    #$01,D1             ;#1 nach D1
FC056e: 4e7b 1002    movec    D1,CACR              ;Cache einschalten; ist erst
                                   ;ab MC68020 implementiert
FC0572: 08c0 0001    bset     #1,D0                 ;Bit 1 in D0 setzen
----- D0 = 3 für MC68020 -----
FC0576: f201 b000    fmove.l  FPCR,d1              ;Inhalt des Floating Point
                                   ;Control Register nach D1
FC057a: 4a81          tst.l     D1                     ;Steht etwas in D1?
FC057c: 6604          bne.s    $fc0582              ;Falls nein, ans Ende d. Rout.
FC057e: 08c0 0004    bset     #4,D0                 ;Bit 4 in D0 setzen
----- D0 = $13 für MC68020 + MC68881 -----
FC0582: 2e49          move.l   A1,A7                     ;alten Stack restaurieren
FC0584: 21c8 0010    move.l   A0,$0010              ;alte Exceptions wieder
FC0588: 21ca 002c    move.l   A2,$002c              ;restaurieren
FC058c: 4cdf 0c00    movem.l  (A7)+,A3-A2           ;Register wieder vom Stack
FC0590: 4e75          rts                      ;Ende
```

Kasten 2

Ein neuer Handler, der nur dann installiert wird, wenn ein 68020 eingebaut ist, und Exceptions geschickter auffängt.

```
;-----
; Neuer Handler für Exception nach MOVE SR,<ea>, angepaßt für 68020
;-----
FC232e: 4e75      rts                      ;Dummy für RomWack-Aufrufe
FC2330: 48e7 8080 movem.l A0/D0,-(A7)        ;A0 und D0 retten
FC2334: 4e7a 0002 movec CACR,D0            ;Cache Control Reg. nach D0
FC2338: 08c0 0003 bset #3,D0              ;Steuerbit für Cache-Reset
FC233c: 4e7b 0002 movec D0,CACR            ;und ausführen
FC2340: 206f 000a move.l 10(A7),A0         ;Opcode-Adr. des Exception-
FC2344: 3010      move.w (A0),D0           ;Befehls holen
FC2346: 0240 ffc0 andi.w #$ffc0,D0        ;ea-Feld ausblenden
FC234a: 0c40 40c0 cmpi.w #$40c0,D0        ;ist es ein move sr,<ea>?
FC234e: 660a      bne.s $fc235a           ;falls nein
FC2350: 08d0 0001 bset #1,(A0)            ;verwandle move sr in move cr
FC2354: 4cdf 0101 movem.l (A7)+,A0/D0      ;Register zurückholen
FC2358: 4e73      rte                    ;Exception-Handler verlassen
FC235a: 4cdf 0101 movem.l (A7)+,A0/D0      ;A0 und D0 wieder vom Stack
FC235e: 4ef9 00fc jmp $fc08d2             ;Die Privilegverletzung war
      08d2      ;nicht durch 68020 entstanden,
FC2364: 4e71      nop                    ;darum Sprung in Original-
FC2366: 4e71      nop                    ;Handler
; Der folgende Teil wird nur einmal in der Reset-Routine aufgerufen. Die
; ersten zwei Befehle entsprechen dem, was durch den Aussprungbefehl aus
; der Original-Routine verloren geht.
FC2368: 327c 0008 move.w #$0008,A1         ;Zeiger auf zu ändernde Vekt.
FC236c: 22c8      move.l A0,(A1)+         ;neue Exception einrichten
FC236e: 23fc 00fc move.l #$fc2330,$20     ;Privileg-Verletzungs-Vektor
      2330 0000 ;auf neuen Handler biegen
      0020
FC2378: 0839 0006 btst #6,$bfe001         ;Linke Maustaste gedrückt?
      00bf e001
FC2380: 6600 0008 bne.l $fc238a           ;Falls nein, normal weiter
FC2384: 7000      moveq #$00,D0           ;Falls ja, D0=0
FC2386: 4e7b 0002 movec D0,CACR            ;Cache außer Kraft setzen
FC238a: 302e 0128 move.w 296(A6),D0       ;D0 restaurieren
FC238e: 4ef9 00fc jmp $fc03f6             ;Wiedereinsprung in Original-
      03f6      ;Routine
FC2394: 4e71      nop
FC2396: 4e71      nop
Kasten 3
```

In der Resetroutine der Kickstart wird der Handler durch einen JMP eingebunden.

```
;-----
; Teil der Original-Reset-Routine, in der die nötigen Änderungen für den
; MC68020 eingerichtet werden. Hier einfach mit JMP hinauspringen in die
; Einbindung des neuen Handlers (Listing 2)
;-----
FC03e2: 302e 0128 move.w 296(A6),D0        ;AttnFlags holen
FC03e6: 0800 0000 btst #0,D0              ;68010 oder 68020?
FC03ea: 6732      beq.s $fc041e           ;falls 68000, weiterspringen
FC03ec: 41fa 048e lea 1166(PC),A0          ;Zeiger auf neue Exceptions
----- Aussprung aus der Original-Routine -----
FC03f0: 4ef9 00fc jmp $fc2368             ;Einsprung in Einbindung des
      2368      ;neuen Handlers
----- Wiedereinsprung -----
FC03f6: 22c8      move.l A0,(A1)+         ;Exception eintragen
FC03f8: 2d7c 00fc move.l #$fc08ba,-28(A6) ;Erweiterung eintragen
      08ba ffe4
FC0400: 2d7c 42c0 move.l #$42c04e75,-528(A6);dito
      4e75 fdf0
Kasten 4
```

Bei einer Änderung der Kickstart muß deren Gesamtprüfsumme neu berechnet werden.

```
;-----  
; Berechnung der Prüfsumme für das neue Kickstart  
;  
; D0 = Summe aller Longs mit Überlaufbehandlung  
; D1 = Korrektur, D0 + D1 = 0  
; Diesen Wert an die Position $ffffe8 der Kickstart schreiben.  
; Auf der Diskette ist dies das Long bei $1e8 in Block 512  
; D2 = alter Korrekturwert  
;-----  
        lea        $fc0000,A0        ;Anfang der Kickstart  
        lea        $1000000,A1       ;Ende der Kickstart  
        clr.l      D0                ;Sum=0  
loop:    move.l     (A0),D1           ;ein long holen  
        add.l      D0,D1             ;addieren  
        cmp.l      D0,D1             ;Überlauf?  
        bge        skip              ;wenn ja  
        addq.l     #1,D0              ;wenn nicht, D0=D0+1  
skip:    add.l      (A0)+,D0          ;D0 = D0 + longword  
        cmp.l      A1,A0             ;Fertig?  
        bcs        loop              ;wenn nicht  
        move.l     $ffffe8,D2        ;alte Checksumme holen  
        sub.l      D2,D0             ;D0 = D0 - D2  
        clr.l      D1                ;D0 + D1 = 0  
        sub.l      D0,D1             ;D1 ist nun neuer Korrekturwert  
        rts
```

Tausch mit Turbo

PAK-68/2 - mehr Dampf für 68000-Rechner, Teil 1

Manfred Völkel, Carsten Meyer

Auch wenn sich sein Haupteinsatzgebiet ein wenig vom Unix-Mainframe in Richtung Zweitrechner verlagert hat, so bleibt der 68000 doch der meistverwendete Prozessor abseits von Intels x86er-Schiene. Der Leistungshunger zeitgenössischer Applikationen verlangt aber inzwischen nach mehr als 16 Bit und 8 MHz, so daß dem 'gelangweilten' 68K-Anwender nur zwei Alternativen bleiben: Entweder muß ein neuer Computer her - oder ein potenter 'Austauschmotor' für die CPU. Letzteren wollen wir Ihnen hier in Form eines c't-Projekts vorstellen.

Eines der erfolgreichsten c't-Projekte war lange Zeit eine Erweiterungsplatine für 68000-Rechner, die den etwas betagten 16-Bitter (Jahrgang 1980!) durch den modernen 68020 nebst Coprozessor substituierte: PAK-68 (Prozessor-Austausch-Karte) hieß der Nachbrenner, der besonders bei 'rechnenden' Anwendern Gefallen fand [1]. Leider lieferte die alte PAK insbesondere in Atari- und Amiga-Rechnern, bei denen sich Chipsatz-Bestückung, Schaltungsdesign und damit auch das Zeitverhalten häufiger änderten als die vierte Stelle der Seriennummer, bisweilen Anlaß zur Klage - vor allem, wenn man den 68020 tatsächlich mit den 16 MHz füttern wollte, die man beim Prozessorkauf so teuer bezahlt hatte. So waren denn auch zum Teil recht gewagte und umfangreiche Änderungen an der PAK-Schaltung nötig, um in einen dauerhaften Genuß der erhöhten Rechenleistung zu kommen.

Postmaterielles Basteln

Reifliche Überlegungen führten in der Redaktion dann zu dem Entschluß, eine neue, von allen Kinderkrankheiten genesene PAK vorzustellen, deren Peripherie-Logik und Timing auch ohne LötKolben jederzeit durch Umprogrammieren von GALs geändert werden kann. Somit ist die neue PAK gegen die Unwägbarkeiten der Zukunft bestens gewappnet. Damit Sie die alte PAK - so schon vorhanden - sinnvoll 'recyclen' können, verwendet unser Projekt trotz der Verlockungen des leistungsfähigeren (aber auch erheblich teureren) 68030 den guten, alten 68020. Nur beim Macintosh-Betriebssystem 7.0 bringt der 68030 aufgrund des Virtual-Memory-Konzepts und der dafür nötigen PMMU echte Vorteile, ansonsten wirkt er in einem auf nur 16 Bit Busbreite ausgelegten Rechner wie ein Zwölfzylinder in einer Familienkutsche.

Das neue Design unserer PAK-68/2 baut zwar im Prinzip auf der alten Schaltung auf, ist jedoch durch die Verwendung eines modernen Konzeptes (Ablaufsteuerung realisiert durch sog. Statemachines) um einiges toleranter und universeller. So konnte fast die gesamte diskrete Logik der alten Schaltung in kompakte GALs verfrachtet werden, und der gesparte Platz durfte für eine ROM/RAM-Erweiterung erhalten, die früher auf einem etwas wackeligen Hücke-PAK-Türmchen untergebracht war.

Speicher-Platz

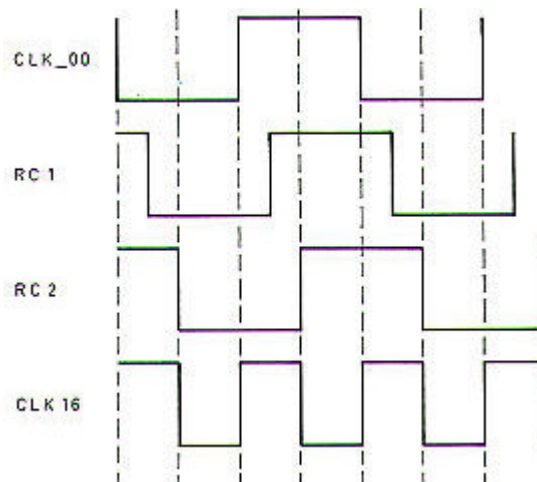
Vier 32polige Fassungen nehmen bis zu 512 KByte ROM/RAM auf, die der Prozessor, ohne durch das 16-Bit-Nadelöhr ausgebremst zu werden, direkt im Zugriff hat. Besonders Rechner mit umfangreichen und gern genutzten Betriebssystemroutinen im ROM (Macintosh, Atari, Amiga) profitieren davon.

Da das normale Atari-ST-TOS auf einem 68020 nur nach sehr umfangreicher Patcherei [3] läuft, können Sie von eMedia dank einer Ausnahmegenehmigung von Atari ein fertig gepatchtes ST-TOS in vier PAK-gerechten EPROMs erhalten. Das beliebte Alternativ-Betriebssystem KAOS 1.42 macht ebenfalls keine Probleme. In der nächsten c't-Ausgabe werden wir Ihnen außerdem zeigen, wie man das auf 32 Bit optimierte TT-TOS für die PAK im ST anpaßt und welche Eigenheiten beim Einbau in Mac und Amiga zu beachten sind.

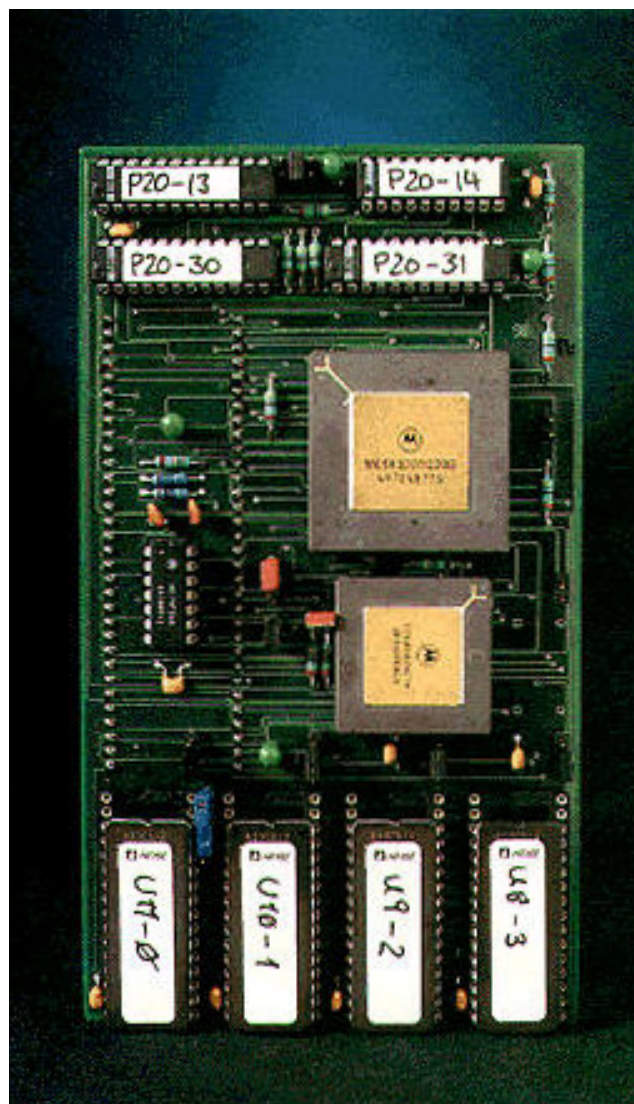
Alternativ können die Fassungen auf der PAK mit statischen RAMs bestückt werden; dafür ist dann Decoder-GAL U12_20 nötig. Der Nutzen eines (im Vergleich zum Hauptspeicher eher kleinen) schnellen RAM-Bereichs dürfte ohne entsprechende Software-Unterstützung allerdings zweifelhaft sein, zumal auch die beim Atari vorhandene DMA nicht an das PAK-RAM herankommt.

Doppelt läuft besser

Ebenfalls auf der Platine untergebracht ist nun eine Schaltung zur Taktverdopplung, die aus den im Rechner angelieferten 7 bis 8 MHz einen phasenstarken 14- beziehungsweise 16-MHz-Takt gewinnt. Das Prinzip der Schaltung ist bestechend einfach: Der Originaltakt wird über zwei RC-Glieder um rund 90° beziehungsweise 32 ns verzögert und schließlich mit sich selbst verEXORt. Solange die Ausgangstaktfrequenz nicht allzusehr von den nominellen 8 MHz abweicht, funktioniert das Ganze ausgezeichnet. Wenn Sie einen noch schnelleren Coprozessor einsetzen möchten, steht dafür ein eigener Quarzoszillator zur Verfügung, der als alternative Taktquelle dienen kann.



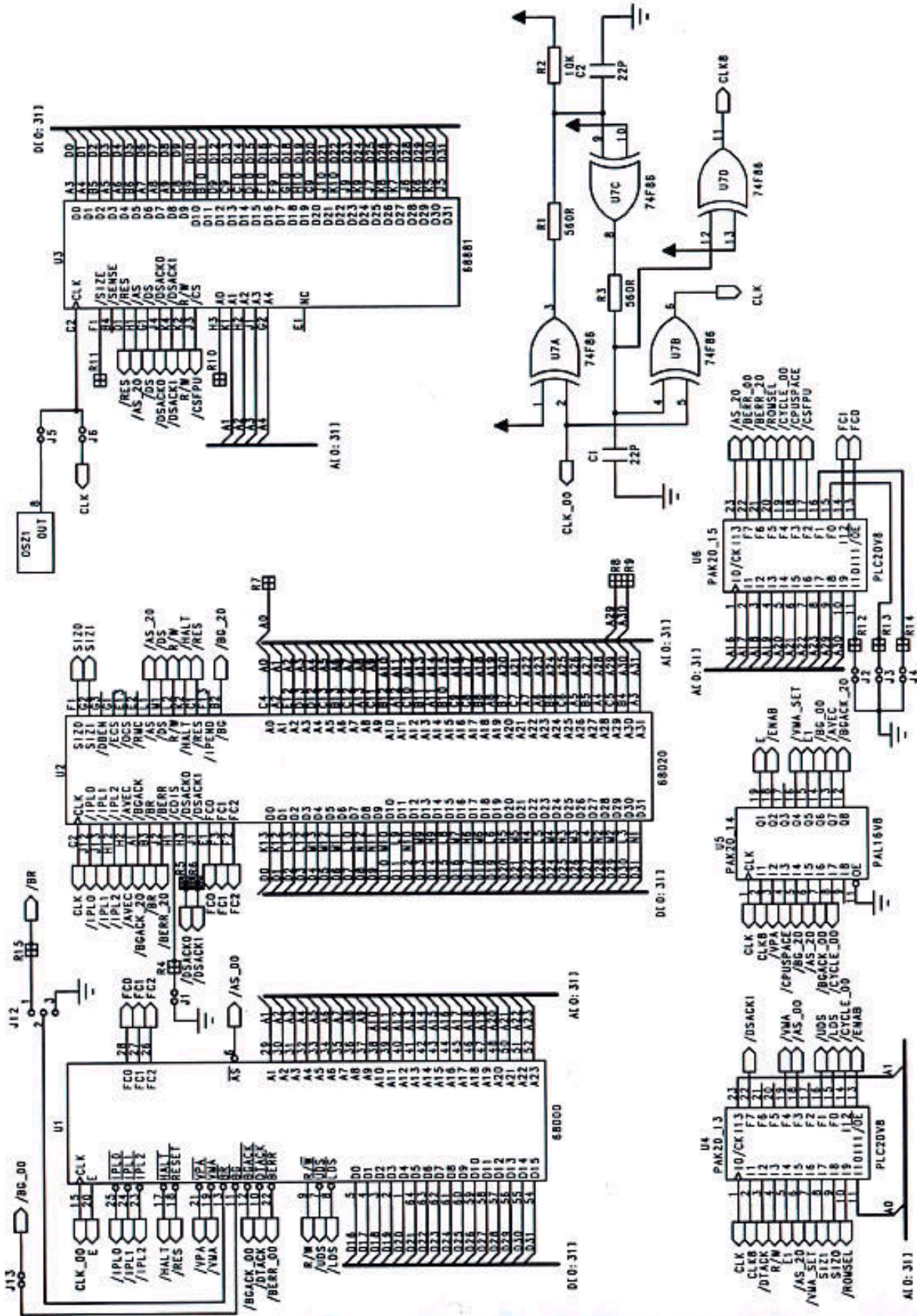
Der Taktverdoppler gewinnt durch Verzögerung des 8-MHz-Clocks um insgesamt 32 ns und anschließende EXOR-Bildung einen phasenrichtigen 16-MHz-Takt.



Beim Layout der Platine wurde versucht, möglichst viele Rechner-Umstände zu berücksichtigen. Probieren Sie vor dem Aufbau mit einer Schablone, ob die PAK in Ihren Computer paßt.

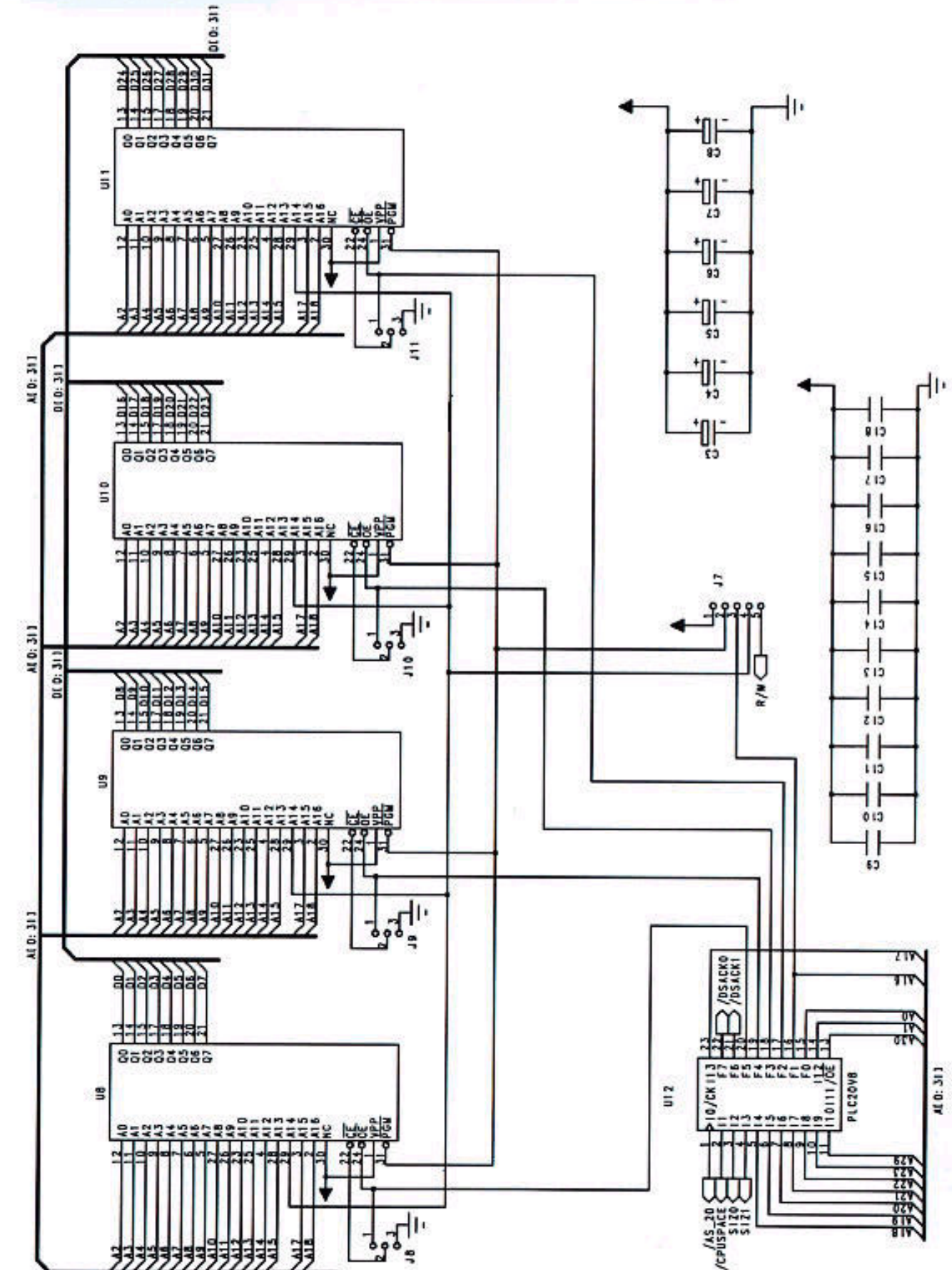
State Of The Machine

Wie bei der alten PAK-68 (c't 8/87) ist auch hier die Emulation des 68000-spezifischen synchronen Buszyklus die größte Herausforderung; die 'Ergatterung' der Data-Strobes und die Synchronisation des /DTACK mit dem 8-MHz-Takt läßt sich dagegen mit wenigen Zeilen GAL-Assembler erschlagen. Für die Erzeugung der 68000-konformen Buszyklen dienen zwei sogenannte State-Machines, realisiert durch die GALs U4 und U5. Dabei sorgt sich U4 um den asynchronen Speicherzyklus, der die Data-Strobes und /VMA mit der gleichen Phasenlage bezogen zum Taktsignal erzeugt wie ein 'echter' 68000. Selbst die /DTACK-Abfrage erfolgt auf der gleichen Clock-Flanke wie beim 68000. Damit sollte die PAK auch in Rechnern laufen, die den 68000-Speicherzyklus allzu genau nehmen. Nachteil: tolerante Computer (z. B. Mac SE) haben nichts davon; eventuell läßt sich hier noch der eine oder andere Taktzyklus herauskitzeln.



Die 'Zusammen-Arbeit' zwischen 68020 und Host-Rechner erledigen im PAK-68/2-Design drei GALs. Der einzige diskrete Schaltungsteil besteht aus dem Taktverdoppler um U7.

GAL U5 (genauer U5_14) liefert ein originalgetreues E-Signal und leitet hieraus gleichzeitig Signale für die /VMA-Erzeugung in GAL U4 ab. Nebenbei erledigt es noch einige HiWi-Jobs: die Verzögerung von /BG bis zum Ende von /AS_20, die Synchronisation von /BGACK mit CLK_8 (wg. Ataris Blitter) und das Freischalten der Treiber (Enable) für /AS_00, /UDS, /LDS, /VMA und /DSACK im 68000-Zyklus.



Das GAL U12 darf nur eingesetzt werden, wenn die PAK-ROMs auch bestückt sind.

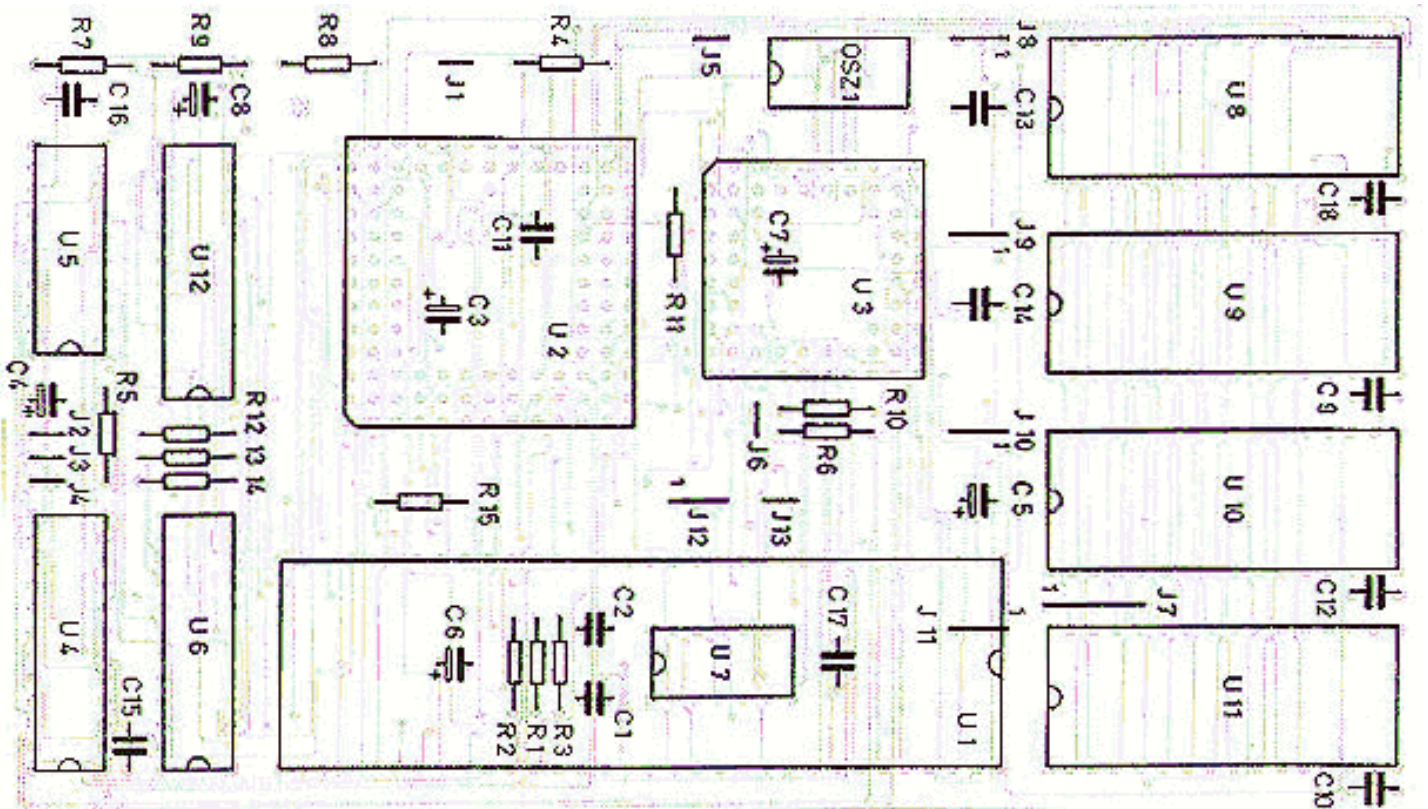
Ein Her(t)z und eine Seele

Eine besondere U5-Variante (U5_16) erzeugt kein E-Signal, sondern erwartet dieses von einem 'lahmgelegten', aber weiterhin in der Schaltung befindlichen 68000-Prozessor. Leider läßt sich beim 68000 der 800kHz-E-Takt nicht von außen abschalten, so daß diese GAL-Version eingesetzt werden muß, wenn man den 68000 unter der PAK nicht auslöten will. Das funktioniert unseres Wissens nur beim Mac, weil der völlig ohne DMA-Zugriffe auskommt.

In den meisten Rechnern ist es möglich, auf die System-ROMs schneller zuzugreifen als auf den von Video-Zugriffen verseuchten RAM-Bereich. Dies wird durch das Signal ROMSel unterstützt, welches die Synchronisation zum 8-MHz-Takt abschaltet. Ein ROM-Lesezyklus dauert somit nur fünf statt acht 16-MHz-Takte. Wann ROMSel aktiv wird, hängt vom Systemdecoder U6 ab, der rechnerabhängig programmiert werden muß, da die Adreßlage vom Rechnermodell abhängig ist. Drei 'Basismodelle' stehen zur Zeit zur Verfügung: U6_22 für Atari ST, U6_30 für Kompakt-Macs und U6_40 für Amiga-Rechner.

Verbleibt noch GAL U12, das sich um die Ausdekodierung des PAK-ROMs beziehungsweise RAMs sorgt. Da es auch die PAK-internen /DSACKs erzeugt, darf es nur eingesteckt werden, wenn auch die ROM/RAM-Fassungen 'belebt' sind - anderenfalls ist dieser Speicherbereich für die Rechnerplatine 'tot', da hier kein /DTACK weitergereicht wird. Auch U12 ist modellabhängig; die hier vorgestellte Version U12_31 gilt für das Mac-ROM (\$40 0000 bis \$47 FFFF, max. 512 KByte). Hinweise für Atari und Amiga sowie deren GAL-Listings finden Sie in der nächsten c't; als 'Anschaungsobjekt' soll uns hier der Mac genügen.

Die GAL-Logik sorgt auf Wunsch (siehe Jumper-Tabelle) dafür, daß Zugriffe auf das System-ROM auf das PAK-interne ROM umgeleitet werden und gar nicht erst auf den 68000-Bus gelangen. Da manche Rechner (z. B. Mac) die System-ROMs beim Reset auf Adresse 0 (Reset-Vektor und Supervisor-Stackpointer) einblenden, dürfen diese nicht aus ihren Fassungen entfernt werden, auch wenn das eigentliche Betriebssystem auf der PAK abläuft.



Achten Sie beim Bestücken darauf, daß die Fassung U1 zuletzt auf der Lötseite eingesetzt wird. Eventuell müssen hier auch C3, C7 und C11 Platz finden.

Springer-Problem

Auf einer universell verwendbaren Platine sind eine erkleckliche Anzahl Jumper natürlich unvermeidlich. J1 deaktiviert den Cache des 68020, damit wird die PAK zahm wie ein echter 68000 (wichtig für zeitkritische Aufgaben). J2 bis J4 konfigurieren den System-Decoder, und zwar wie folgt:

- J2 offen: Die System-ROMs sind auf der Hauptplatine, auf die im verkürzten Buszyklus (fünf Takte) zugegriffen wird.
- J2 gesteckt: Es wird im GAL U6 bestimmten Speicherbereich kein *cycle_00* und *ROMSel* erzeugt. Der 68020 erwartet einen 32 Bit breiten ROM/RAM-Bereich auf der PAK, auf den er ohne Waitstate zugreift.
- J3 offen: Adreßbits A29 und A30 werden ausgewertet (kompatibel zur alten PAK).
- J3 gesteckt: A29 und A30 werden ausmaskiert, damit wird ein Hängen des Rechners bei unsauberer Software (z. B. Mac-ROMs), die die oberen acht Adreßbits als Flags mißbraucht, verhindert.
- J4 offen: FPU vorhanden
- J4 gesteckt: Beim Ansprechen der (nicht vorhandenen) FPU wird /BERR erzeugt (wichtig für Amiga).

Die Art der FPU-Versorgung bestimmen J5 und J6: Über J5 erhält der Rechenknecht seinen Takt vom Quarzoszillator, über J6 dagegen vom 68020 (16 MHz). Durch diese Option ist es möglich, schnellere oder langsamere Coprozessoren einzusetzen. Mit weniger als 12 MHz FPU-Takt dürfen Sie sich aber nicht zufriedengeben, weil der Coprozessor sonst vom Prozessor 'überfahren' wird.

J7 bis J11 bestimmen, ob RAMs oder EPROMs auf der PAK zum Einsatz kommen. Bei EPROMs wird J7 auf Stift 1-2 und 3-4 gejumpert, bei RAMs auf Stift 2-3 und 4-5. J8 bis J11 ziehen auf Position 2-3 den /CS-Eingang der EPROMs dauerhaft auf Masse, somit lassen sich preiswerte 150-ns-Typen verwenden, wenn auf minimalen Stromverbrauch nicht unbedingt Wert gelegt wird. Die Zugriffszeit der RAMs sollte dagegen grundsätzlich kleiner/gleich 100 ns sein, in diesem Fall sind J8 bis J11 auf Position 1-2 zu stecken.

Mit der richtigen Konfiguration von J12 und J13 und dem passenden GAL U5 kann in DMA-freien Systemen (etwa Mac) auf das Entfernen des 68000 verzichtet werden:

- 68000 entfernt: J12 auf Position 1-2, J13 gesteckt, U5_14 verwendet,
- 68000 vorhanden: J12 auf Position 2-3, J13 offen, U5_16 verwendet.

Lötsinnig

Sollten Sie den Begriff GAL vornehmlich mit der bundesdeutschen Parteienlandschaft assoziieren und die letzte Begegnung mit Ihrem Lötkolben in schmerzhafter Erinnerung haben, können wir vom Auf- und Einbau der Schaltung in Eigenregie nur abraten. Immerhin muß (zumindest beim Atari) der 64polige Altprozessor aus der Rechner-Hauptplatine entfernt werden, und das ist ohne langjährige Löterfahrung und gutes Werkzeug ein schwieriges und für den Rechner oft tödliches Unterfangen. Im Mac läßt sich der Einbau etwas entschärfen, indem man einfach einen 64poligen Sockel auf den alten 68000 lötet; ansonsten kneift man in Ermangelung eines professionellen Entlötgeräts sämtliche Beinchen des Prozessors vorsichtig nahe am DIL-Gehäuse ab und zieht die Pins einzeln aus ihren Lötäugen.

Bei der Bestückung der Platine gibt es nur wenig zu beachten; so sollte der 64polige Sockel zur Rechner-Verbindung erst dann auf der Lötseite eingesetzt werden, wenn alle Bauteile auf der Bestückungsseite verlötet sind. Abhängig von den verwendeten Pin-Grid-Fassungen kann es notwendig sein, die darunter befindlichen Abblockkondensatoren ebenfalls auf der Lötseite unterzubringen. Selbstverständlich sollten Sie für alle Halbleiter gute Fassungen verwenden. Die Verbindung zur Rechnerplatine erfolgt schließlich über einen 64poligen 'Adaptersockel', der auf beiden Seiten mit Pins bestückt ist, oder alternativ über ebensolche SIL-Streifen.

Abhängig vom Layout der Rechnerplatine bei Atari und Amiga kann es notwendig sein, für die 'Beinfreiheit' der darunterliegenden Bauteile eine oder zwei 64polige Fassungen unterzulegen. Beim Amiga 2000 muß man die Platine über einen selbstgefertigten 90°-Winkeladapter senkrecht stellen, da unter dem Baugruppenträger für die Floppy-Laufwerke nur sehr wenig Platz ist. Überhaupt ist es sinnvoll, vor der Platinenbestellung mit einer Schablone zu prüfen, ob selbige überhaupt in den aufzurüstenden Rechner paßt; wir haben zwar versucht, beim Layout alle Gegebenheiten zu berücksichtigen, aber man weiß ja nie ...

Im [nächsten Heft](#) finden Sie nähere Hinweise zum Einbau in Mac, Atari und Amiga, eine Patch-Anleitung für das auf 32 Bit optimierte TT-TOS und ein Programm, welches das Betriebssystem des jeweiligen Rechners ausliest und mit entsprechendem Offset auf vier Files verteilt. Mit diesen Files müssen Sie wiederum ein EPROM-Programmiergerät füttern und entsprechende EPROMs programmieren, um das volle Leistungspotential der PAK-68/2 ausnutzen zu können. (cm)

Literatur

[1] Johannes Assenbaum, Mehr CPU, PAK-68 - Die Prozessor-Austausch-Karte für 68000-Rechner, c't 7/87, S. 68

[2] Carsten Meyer, PAK im Mac, 68020 und 68881 im kleinen Macintosh, c't 12/88, S. 126

[3] Detlef Wulf, Die 32-Bit-Kur für TOS und GEM, c't 3/88, S. 108

[4] Leo Drisis, Der Real-Renner, 68881-Einbindung für die PAK im Mac, [c't 5/90, S. 190](#)

Kasten 1

Stückliste

U1	64poliger DIL-Sockel zum System
U2	MC68020RC16
U3	MC68881RC16 oder MC68882RC16 (auch schneller)
U4	GAL20V8 (U4_13)
U5	GAL16V8 (U5_14 oder U5_16)
U6	GAL20V8-15 (U6_30, 22 od. 40)
U7	74F86 (muß F-Typ sein!)
U8-11	27C512-120
U12	GAL20V8-15 (U12_31, 21 od. 41)
OSZ1	optionaler Quarzoszillator, Frequenz je nach FPU-Takt

Widerstände

R1,R3	560R
R2	10k
R5,R6	1k
R4,R7-R15	4k7

Kondensatoren

C1,C2	22p
C3-C8	10µ 16V Ta.
C9-C18	100n

Sonstiges

ca. 10 Jumper-Steckbrücken, 64pol. DIL-Fassung, 64pol. Adapter-Fassung, div, IC-Fassungen, Platine PAK-68/2

Kasten 2

Die GALs wurden mit dem PALASM4 von AMD entwickelt. Dieses Produkt ist zwar eigentlich für die PALCEs aus gleichem Hause vorgesehen, liefert aber auch für die funktionsgleichen GALs hervorragende Ergebnisse.

```

1  TITLE      68020-16 PAK
2  PATTERN    P20-13
3  REVISION   1
4  AUTHOR     Manfred Völkel
5  COMPANY    GK Computer
6  DATE       06/01/91
7
8
9
10 CHIP      _P20_13  PAL20V8
11
12 PIN 1 CLK16 COMB
13 PIN 2 CLK8 COMB
14 PIN 3 /DTACK COMB
15 PIN 4 RW COMB
16 PIN 5 E1 COMB
17 PIN 6 /AS_20 COMB
18 PIN 7 E2 COMB
19 PIN 8 SIZ1 COMB
20 PIN 9 SIZ0 COMB
21 PIN 10 /ROMSEL COMB
22 PIN 11 A0 COMB
23 PIN 12 GND
24 PIN 13 /OE COMB
25 PIN 14 /CYCLE_00 COMB
26 PIN 15 /LDS REG
27 PIN 16 /UDS REG
28 PIN 17 DS_SET COMB
29 PIN 18 /AS_00 REG
30 PIN 19 /VMA REG
31 PIN 20 Q1 REG
32 PIN 21 Q0 REG
33 PIN 22 /DSACK1 REG
34 PIN 23 A1 COMB
35 PIN 24 VCC
36
37
38
39 EQUATIONS
40
41
42
43     /LDS    :=  DSACK1 * /DS_SET
44               +  AS_00 * /DS_SET
45               +  /A0 * SIZ0 * /SIZ1
46               +  /AS_20 * /DS_SET
47               +  /Q0 * /DS_SET
48               +  /RW * /DS_SET
49               +  /CLK8 * /ROMSEL * /DS_SET
50               +  /CYCLE_00
51     UDS     :=  /A0 * RW * AS_00 * /DSACK1 * /Q1
52               +  /A0 * RW * AS_00 * /Q1 * Q0
53               +  /A0 * RW * /DSACK1 * /Q1 * Q0 * AS_20 * CYCLE_00
54               * ROMSEL
55               +  /A0 * RW * /DSACK1 * /Q1 * Q0 * CLK8 * AS_20 * CYCLE_00
56
57               +  /A0 * /RW * AS_00 * /DSACK1 * Q0
58     DS_SET  =   /RW * AS_00 * /DSACK1 * Q0
59               +   RW * AS_00 * /Q1 * Q0
60               +   RW * AS_00 * /DSACK1 * /Q1
61     AS_00   :=  AS_00 * /DSACK1 * Q1 * Q0
62               +  AS_00 * DSACK1 * /Q1 * Q0
63               +  AS_20 * CLK8 * CYCLE_00 * /DSACK1 * /Q1 * Q0
64               +  AS_20 * ROMSEL * /DSACK1 * /Q1 * Q0 * RW
65               +  AS_00 * /DSACK1 * /Q1 * /Q0
66               +  AS_20 * AS_00 * /DSACK1 * /Q1
67     VMA     :=  /CLK8 * /E1 * /E2
68               +  VMA * /E2
69               +  VMA * /E1
70               +  CLK8 * VMA
71     Q1      :=  AS_00 * DSACK1 * Q1 * /Q0
72               +  AS_20 * /CLK8 * /RW * DTACK * AS_00 * /Q1 * Q0
73               +  AS_20 * /CLK8 * /RW * AS_00 * /Q1 * Q0 * E2 *
74               /E1 * VMA
75               +  AS_00 * DSACK1 * /Q1 * Q0
76     Q0      :=  /AS_00 * /DSACK1 * /Q0

```

```

77      + AS_00 * /DSACK1 * /Q1 * AS_20
78      + /AS_00 * /DSACK1 * /Q1 * /AS_20
79      + /AS_00 * /Q1 * /Q0
80      + /DSACK1 * /Q1 * /Q0
81      + /AS_00 * /DSACK1 * /Q1 * ROMSEL * RW
82      + /AS_00 * /DSACK1 * /Q1 * /CLK8
83      + /AS_00 * /DSACK1 * /Q1 * /CYCLE_00
84 DSACK1 := AS_00 * /DSACK1 * Q1 * Q0
85      + AS_00 * DSACK1 * /Q1
86      + AS_20 * /CLK8 * RW * AS_00 * /DSACK1 * Q0 * E2
87      * /E1 * VMA
88      + AS_20 * RW * DTACK * AS_00 * /DSACK1 * Q0 * ROMSEL
89      + AS_20 * /CLK8 * RW * DTACK * AS_00 * /DSACK1 * Q0

```

```

1  TITLE      PAK20 16 MHz
2  PATTERN    P20_14
3  REVISION   1
4  AUTHOR     M.Völkel
5  COMPANY    GK-Computer
6  DATE       08/10/91
7
8
9

```

```

10 CHIP _P20_14 PAL16V8

```

```

11
12 PIN 2 CLK8
13 PIN 3 /VPA
14 PIN 4 /CPUSPACE
15 PIN 5 /BG_20
16 PIN 6 /AS_20
17 PIN 7 /BGACK_00
18 PIN 8 /CYCLE00
19 PIN 10 GND
20 PIN 12 /BGACK_20 REG
21 PIN 13 /AVEC COMB
22 PIN 14 /BG_00 REG
23 PIN 15 E1 REG
24 PIN 16 E2 REG
25 PIN 17 E0 REG
26 PIN 18 /ENAB COMB
27 PIN 19 E REG
28 PIN 20 VCC
29
30
31

```

```

32 EQUATIONS

```

```

33
34
35
36 BGACK_20 := /CLK8 * BGACK_20
37           + BGACK_00
38 AVEC = CPUSPACE * VPA
39 BG_00 := CLK8 * BG_20 * /AS_20
40        + /CLK8 * BG_00
41 /E1 := CLK8 * /E1
42      + /E * E2 * /E1 * /E0
43      + /CLK8 * /E * E2 * E1 * E0
44      + /CLK8 * E * /E2 * E0
45      + E * /E1 * E0
46 /E2 := /E2 * E1 * /E0
47      + /CLK8 * /E * E2 * /E1 * VPA * /CPUSPACE
48      + /CLK8 * /E * /E1 * E0
49      + CLK8 * /E2
50      + /E * /E2 * E1
51 E0 := CLK8 * E0
52     + /CLK8 * /E0
53 ENAB = CYCLE00 * /BG_00 * /BGACK_20
54 E := E * /E1
55     + CLK8 * E
56     + E * /E0
57     + /CLK8 * /E2 * E1 * E0
58     + /CLK8 * /E2 * /E1 * /E0

```

```

0  CHIP _P20_16 PAL16V8

```

```

1
2  PIN 2 CLK8
3  PIN 3 /VPA
4  PIN 4 /CPUSPACE
5  PIN 5 /BG_20

```

```

6 PIN 6 /AS_20
7 PIN 7 /BGACK_00
8 PIN 8 /CYCLE00
9 PIN 10 GND
10 PIN 12 /BGACK_20 REG
11 PIN 13 /AVEC COMB
12 PIN 14 /BG_00 REG
13 PIN 15 E1 REG
14 PIN 16 E2 REG
15 PIN 17 E0 REG
16 PIN 18 /ENAB COMB
17 PIN 19 E
18 PIN 20 VCC
19
20
21
22 EQUATIONS
23
24
25
26 BGACK_20 := /CLK8 * BGACK_20
27           + BGACK_00
28 AVEC = CPUSPACE * VPA
29 BG_00 := CLK8 * BG_20 * /AS_20
30         + /CLK8 * BG_00
31 /E1 := /CLK8 * /E2 * E1 * /E0 * /CPUSPACE * VPA
32       + /CLK8 * E * /E2 * E1 * /E0
33       + CLK8 * /E1
34 E2 := CLK8 * E2
35     + E2 * E0
36     + /CLK8 * E * /E2 * /E0
37     + /CLK8 * /E1 * /E0
38 E0 := /E1 * /CLK8
39     + E * E2 * E0
40     + /E * /E2 * /CLK8
41     + E0 * CLK8
42 ENAB = CYCLE00 * /BG_00 * /BGACK_20

```

```

0 TITLE      PAK 68020-16
1 PATTERN    p20_30
2 REVISION   1
3 AUTHOR     M.Voelkel
4 COMPANY    GK Computer
5 DATE       08/10/91
6
7 CHIP       _p20_30 PALCE20V8
8
9 ;----- PIN Declarations -----
10 PIN 17      /csfpu                      COMBINATORIAL ; OUTPUT
11 PIN 18      /cpuspace                   COMBINATORIAL ; OUTPUT
12 PIN 19      /cycle_00                   COMBINATORIAL ; OUTPUT
13 PIN 20      /romsel                     COMBINATORIAL ; OUTPUT
14 PIN 21      /berr_20                   COMBINATORIAL ; OUTPUT
15 PIN 22      /berr_00                    ; INPUT
16 PIN 23      /as_20                      ; INPUT
17 PIN 11      /jp2                        ; INPUT
18 PIN 15      /jp3                        ; INPUT
19 PIN 16      /jp4                        ; INPUT
20 PIN 1..8    a[16..23]                   ; INPUT
21 PIN 9       a29                         ; INPUT
22 PIN 10      a30                         ; INPUT
23 PIN 13      fc0                         ; INPUT
24 PIN 14      fc1                         ; INPUT
25
26 ; ROM Adressen :
27 ; sysrom = Systemrom      MAC 400000 - 4FFFFFF
28 ; addmem =                "   none
29
30 STRING sysrom '( /a[23] * a[22] * /a[21] * /a[20] )'
31 STRING addmem '( GND )'
32
33
34 ;----- Boolean Equation Segment -----
35 EQUATIONS
36 MINIMIZE_ON
37
38 cycle_00 =/( a30 * /a29 * /JP3
39              + sysrom * /a[19] * jp2 ; nur 40000-47FFFF !!
40              + addmem

```

```

41         + fc0 * fc1 * /a[16] * a[17] * /a[18] * /a[19]      ; fpu
42     )
43 romsel = /JP2 * sysrom
44 csfpu =  fc0 * fc1 * /a[19] * /a[18] * a[17] * /a[16] * /JP4 ;
45 cpuspace = fc0 * fc1 ;
46 ;-----
47 ;      Busererror wird erzeugt bei                               |
48 ;      Busererror vom 68000 System                               |
49 ;      FPU Select wenn JP4 gesteckt d.h. keine FPU im System   |
50 ;      (erlaubt Software Emulation der Line F Befehle)         |
51 ;      Breakpoint ACK cycle ( braucht 68851 PMMU )              |
52 ;      Access Level fetch      "      "      "                  |
53 ;-----
54 berr_20 = as_20 * ( berr_00
55         + fc0 * fc1 * /a[19] * /a[18] * a[17] * /a[16] * JP4
56         + fc0 * fc1 * /a[19] * /a[18] * /a[17] * /a[16]
57         + fc0 * fc1 * /a[19] * /a[18] * /a[17] * a[16]
58         );

```

```

1  TITLE      68020-16 PAK
2  PATTERN    P20-31
3  REVISION   1
4  AUTHOR     Manfred Völkel
5  COMPANY    GK Computer
6  DATE       06/01/91
7
8
9

```

```

10 CHIP _P20_31 PAL20V8

```

```

11
12 PIN 1 /AS_20 COMB
13 PIN 2 /CPUSPACE COMB
14 PIN 3 SIZ0 COMB
15 PIN 4 SIZ1 COMB
16 PIN 5 A[18] COMB
17 PIN 6 A[19] COMB
18 PIN 7 A[20] COMB
19 PIN 8 A[21] COMB
20 PIN 9 A[22] COMB
21 PIN 10 A[23] COMB
22 PIN 11 A[29] COMB
23 PIN 12 GND
24 PIN 13 A[30] COMB
25 PIN 14 A1 COMB
26 PIN 15 A0 COMB
27 PIN 16 A[16] COMB
28 PIN 17 /CS11 COMB
29 PIN 18 /CS10 COMB
30 PIN 19 /CS9 COMB
31 PIN 20 /CS8 COMB
32 PIN 21 /DSACK1 COMB
33 PIN 22 /DSACK0 COMB
34 PIN 23 A[17] COMB
35 PIN 24 VCC
36
37
38

```

```

39 EQUATIONS

```

```

40
41
42
43 CS11 = AS_20 * /CPUSPACE * /A[19] * /A[20] *
44       /A[21] * A[22] * /A[23]
45 CS10 = AS_20 * /CPUSPACE * /A[19] * /A[20] *
46       /A[21] * A[22] * /A[23]
47 CS9  = AS_20 * /CPUSPACE * /A[19] * /A[20] * /A[21] * A[22]
48       * /A[23]
49 CS8  = AS_20 * /CPUSPACE * /A[19] * /A[20] * /A[21] * A[22]
50       * /A[23]
51 DSACK1 = AS_20
52 DSACK1.TRST = CS11
53 DSACK0 = AS_20
54 DSACK0.TRST = CS11

```

Tausch mit Turbo

PAK-68/2 in Mac, Amiga und Atari, Teil 2

Carsten Meyer, Manfred Völkel

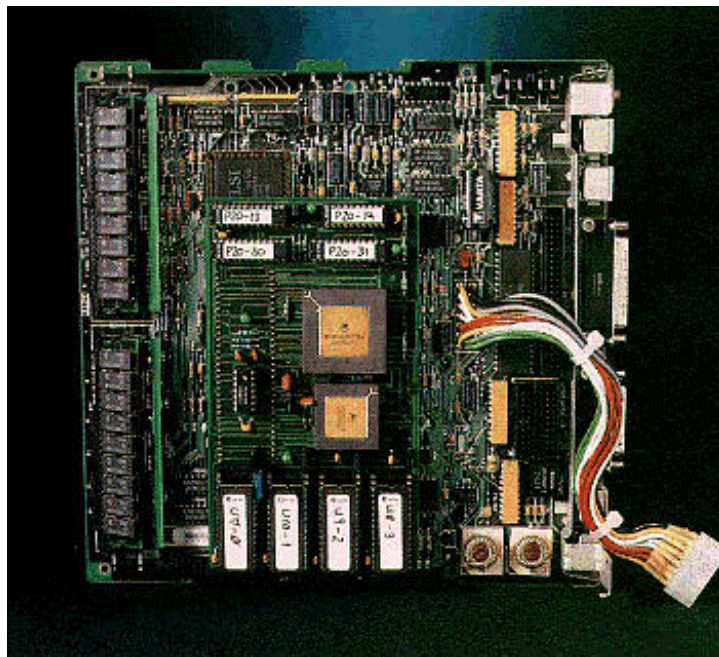
Nachdem wir Ihnen in der letzten c't die Hardware unserer neuen PAK-68/2 vorgestellt haben, ist nach Erkalten des Lötcolbens die Anpassung an die verschiedenen 68K-Rechnerfamilien fällig. Im Zeitalter des postmateriellen Bastelns geschieht dies einfach durch Programmieren von GALs und EPROMs.

Wie schon im ersten Teil angesprochen, besteht bei der neuen PAK die Möglichkeit, das Betriebssystem in vier EPROMs zu packen, auf die der 68020 in seiner ganzen Datenbusbreite zugreifen kann. Allein diese Maßnahme bringt bei sehr vielen Anwendungen gut ein Viertel an Geschwindigkeitszuwachs, beim stark ROM-orientierten Mac sogar noch mehr (siehe Benchmark-Tabelle). Sollten Sie Zugang zu einem EPROM-Brenner haben, lohnt sich diese Option auf jeden Fall.

Als erster Patient dient uns der kompakte Mac, auch bekannt unter dem Pseudonym 'shoebox', der angesichts der kommenden System-7-Euphorie für Leistungssteigerungen besonders dankbar ist. Allerdings liegen uns über den neueren Mac Classic noch keine Erfahrungen vor; außerdem benötigt er einen (selbstzustrickenden) PLCC-DIL-Adapter für die CPU.

SE-Mac, öffne dich!

Zum Öffnen des Macintosh sollten Sie einen sogenannten Torx-Schraubendreher der Größe 15 besorgen; oft sitzen die Gehäuseschrauben sehr fest, und ein passend angeschliffener Flach-Schraubendreher bricht in den Torx-Schraubenköpfen ab. Leider wird im Mac SE der PDS (Prozessor Direct Slot) teilweise von der PAK überdeckt, so daß hier keine Erweiterungskarten mehr eingesteckt werden können. Das ist aber kein Beinbruch, denn beispielsweise ein Bildschirmadapter hat schon mechanisch wegen der ausladenden PAK im Mac keine Chance. Achten Sie aber darauf, daß von der PAK keine Drahtenden in den Slot ragen, die Kurzschlüsse verursachen könnten.



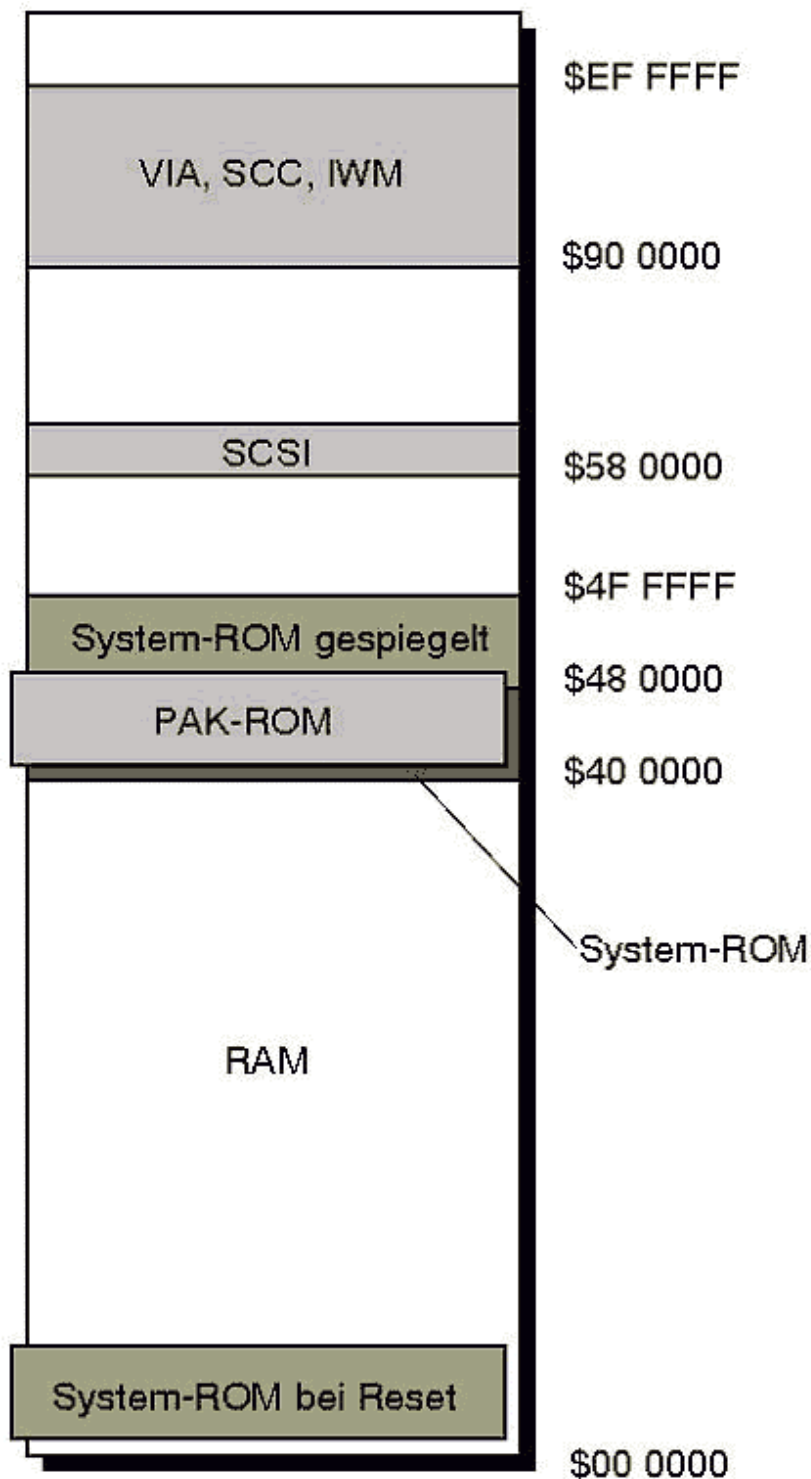
Im Mac SE überdeckt die PAK den PDS etwas, paßt aber sonst ausgezeichnet.

Probierstunde

Bei richtig gesetzten Jumpers und passenden GALs (U6_30 und U12_31) sollte die PAK ohne mühsame Try-and-Error-Sessions auf Anhieb funktionieren. Bleibt noch, den Coprozessor zur Mitarbeit zu bewegen. Beim Mac ist das relativ einfach: Da hier sämtliche Fließkomma-Berechnungen grundsätzlich durch die SANE-Engine geschehen, haben von einer Beschleunigung dieser Routinen *alle* Programme etwas, nicht nur die, die explizit eine Fließkomma-Unterstützung bieten. Als sehr empfehlenswerter SANE-Ersatz (etwa fünfmal schneller als die 881-SANE des Mac II und zwanzigmal schneller als 68000-Code) bietet sich das in [4] vorgestellte Radius-MATH-Init des gleichnamigen Hardware-Herstellers an, das die trägen SANE-Routinen auf eigene 68881-Aufrufe umleitet. In gepatchter Form (Bytefolge \$67000110 suchen und durch \$4E714E71 ersetzen) eignet es sich grundsätzlich für alle Macs, die mit einem 68881- oder 68882-Coprozessor ausgerüstet sind.

Bei Null begonnen

Der Macintosh verdient bezüglich seines etwas eigenen Reset-Ablaufs eine besondere Beachtung. Nach einem Kaltstart blendet der Apple-Custom-Chip namens BBU die System-ROMs zunächst auf Adresse \$0 ein. In den ersten Bytes des ROMs findet der Prozessor seinen Reset-Vektor und einen 'Dummy'-Supervisor-Stackpointer (dazu später) und springt dann in Initialisierungsroutinen im System-ROM-Bereich (beim Mac SE: \$40 002A). Normalerweise nimmt die BBU beim ersten Zugriff auf eine Adresse zwischen \$40 0000 und \$4F FFFF die ROMs aus dem unteren RAM-Bereich wieder heraus. Da die Hauptplatine und damit die BBU bei aktiviertem PAK-ROM Zugriffe auf ihren eigenen ROM-Bereich nicht mehr mitbekommt, bleibt das nach unten 'gebeamte' Hauptplatinen-ROM so lange eingeschaltet, bis der Mac aufgrund des dort überdeckten RAMs abstürzt und einen Griff zum Netzschalter provoziert. Es gilt also, einen Zugriff außerhalb des PAK-ROMs, aber innerhalb des Bereiches \$44 0000 bis \$4F FFFF auszulösen, und das möglichst früh, noch bevor das Selbsttestprogramm fehlendes RAM im unteren Speicherbereich detektiert hat. Üblicherweise patcht man deshalb einen Befehl wie `move.w $48 0000,D0` in die Reset-Sequenz, und zwar dort, wo er wenig stört. Bei allen mir bekannten Macs ist am Ende des ROM-Bereichs das System-Erstellungsdatum in ASCII-Klartext abgelegt, was sich als Spielwiese für Patch-Versuche förmlich aufdrängt.



Speicherbelegung beim Kompakt-Mac. Er gilt sowohl für den klassischen Mac Plus als auch für seine kompakten Nachfolger, auch wenn der ROM-Bereich des Plus (128 KByte) doppelt so oft gespiegelt ist wie der des SE.

Patch me!

Vor dem erfolgreichen Verbiegen irgendwelcher Jumps im ROM haben dessen Väter allerdings eine hartnäckige Checksum-Routine gesetzt, die jegliche Änderungen mit einem `Sad Mac` und Fehlercode 1 (ROM-Check-Error) beantwortet. Übrigens erwartet die Routine die Prüfsumme im ersten Langwort des System-ROMs; der vermeintliche Supervisor-Stackpointer ist also in Wirklichkeit die ROM-Prüfsumme.

Ein kleiner, aber gemeiner Trick hilft weiter: Man leitet die Checksummen-Routine (sie beginnt in einem Mac SE bei \$40 26C8) auf die *gespiegelten* Original-ROMs, beispielsweise oberhalb \$48 0000 um. Erstens stört sie sich so an den gepatchten PAK-ROMs nicht, zweitens löst sie ohne weiteres Zutun den Umschaltvorgang durch die BBU aus, und drittens beschränkt sich der Patch auf das Ändern eines einzigen Bytes (genauer gesagt auf nur eines von zwei Millionen Bits - von \$40 auf \$48). Das vorgestellte ROMSplit-Programm erledigt diese Aufgabe quasi nebenbei. Es liefert vier fix und fertige Files, mit denen Sie ebenso viele 27C512er EPROMs füttern müssen (ggf. Kollegen mit PC und EPROM-Brenner fragen). Es entspricht übrigens nicht dem alten ROMSplit-Programm auf der eMedia-Macintosh-Sammeldisk 1, das keine Patch-Routine enthält.

STolperSTEin

Auch für den Atari sind speziell angepaßte GALs (U6_22 und U12_20/21/24) erhältlich, die neben dem normalen ST-ROM (\$FC 0000 bis \$FE FFFF) auch das neuere STE- beziehungsweise TT-ROM (\$E0 0000 bis \$EF FFFF) unterstützen. Beim Atari sind leider recht massive Eingriffe in das TOS-ROM nötig [3], weil deren Entwickler nicht auf eine vorausschauende Kompatibilität zum 68020 geachtet haben. Von eMedia erhalten Sie daher in Absprache mit Atari ein fertig angepaßtes ST-TOS 1.4 in fertig `konfektionierten` EPROMs, die Sie nur noch auf die PAK zu stecken brauchen. Das ältere PAK-TOS, das für die alte PAK [1] angeboten wurde, eignet sich ebenfalls, wenn es auf vier 27512-EPROMs umkopiert wird. Zu diesem Zweck - und *nur* zu diesem - eignet sich das ROMSplit-Programm für den Atari, das Sie auf den folgenden Seiten finden. Um es nochmals in aller Deutlichkeit zu sagen: Das originale ST/STE-TOS läuft definitiv nicht auf einem 68020-Prozessor. Ferner verwendet Atari im STE die PLCC-Version des 68000, die den Einbau der PAK vereitelt.

Der Einsatz grundverschiedener TOS-Versionen auf Hauptplatine und PAK wird übrigens für den Blitter leicht zur Falle: Da der Hilfsarbeiter an die PAK-ROMs nicht herankommt, schreibt er unter Umständen nur Müll auf den Bildschirm. Abhilfe: Blitter ausschalten (der wirkt in einem 68020-System ohnehin nur als Bremse) oder Hauptplatinen-ROMs ebenfalls updaten.

TT-TOS auf der PAK68/2

Bei neueren Modellen (STE und TT) hat Atari das ROM mit dem Betriebssystem von \$FC 0000 nach \$E0 0000 verschoben und so die Beschränkung auf 192 KByte für das TOS umgangen. Dies macht eine Auffrischungskur durch einfachen EPROM-Austausch im ST allerdings unmöglich. Nicht so bei der neuen PAK: Konfiguriert man das PAK-ROM für den Bereich \$E0 0000-\$E7 FFFF und beläßt den normalen Systembereich von \$FC 0000-\$FE FFFF auf der Hauptplatine des Rechners, lassen sich die alte Version und das TOS eines der Nachfolgerechner alternativ verwenden, je nachdem, welches man anstartet.

Das Objekt der Begierde ist natürlich das TOS 3.1 des TT. Zunächst bestand die vage Hoffnung, daß es weitgehend unabhängig von der Hardware geschrieben sei und man außer beim Startup nichts oder nur wenig ändern müsse. Also PAK mit RAM bestückt, zwei kleine Programme zum Laden und Starten des neuen TOS erstellt und einfach mal ausprobiert. Leider hat es - wie sollte es anders sein - nicht funktioniert. Da das System den Rechner neu initialisiert, kann man nicht einmal mit einem Debugger nachbohren, wo es denn hapert.

So mußte ich den ersten Teil der Initialisierung von Hand durcharbeiten und entsprechend ändern. Dazu zählen die Initialisierung der MMU im TT-68030, des DMA-Sounds, des Video-Syncmodes sowie der Shadow-Test, der die RAM-Bestückung bestimmt. Dieser funktioniert beim TT ebenso wie beim ST, prüft jedoch andere Adressen. Nachdem das System bis zur Initialisierung der Systemvektoren (Bus-Error etc.) lief, konnte ich die Routine, die dem System nach einem Bombenhagel eigentlich wieder auf die Beine helfen soll, durch eine Endlosschleife ersetzen. So gelang es, nach einem Reset vom alten TOS aus den Post-Mortem-Dump bei Adresse \$380 auszuwerten, um die Ursache für den Absturz zu ermitteln.

Modi-Erscheinung

Der TT hat mehr Video-Modi als der ST. Zu deren Steuerung benötigt er zwei Registersätze, einen ST-kompatiblen, über den auch nur die ST-Modi zugänglich sind, und den TT-Satz selbst:

ST: Palette bei \$FF8240 Shift Mode bei \$FF8260

TT (zusätzlich): Palette bei \$FF8400 Shift Mode bei \$FF8262

Das TT-TOS benutzt nur das Shift-Mode-Register bei \$FF8262. Alle Zugriffe darauf habe ich auf \$FF8260 umgelenkt und die Abfrage der gelesenen beziehungsweise die zu schreibenden Werte entsprechend angepaßt. Außerdem mußte die Definition des Monochrom-Monitors geändert werden, der ST-monochrome-Mode stellt beim TT eine Farbbetriebsart dar. Für die Video-Base hat der TT ein zusätzliches Register bei \$FF820D (Low-Byte). Beim ST läßt ein Zugriff darauf zwar keine Bomben fallen, er liefert jedoch beim Auslesen den Wert \$FF statt \$00, wie es zur Kompatibilität nötig wäre.

NOP (A0)

Ebenfalls eine Erweiterung stellt beim TT die zweite MFP dar. Er verwendet einen Timer dieses Chips für kurze Zeitschleifen, wie jene berühmte, die beim Zugriff auf den Floppycontroller verzögernd wirkt. Diese Routine muß leider wieder durch eine Software-Schleife ersetzt werden, die dann natürlich vom Takt der CPU abhängig ist. Der NOP-Befehl in der Schleife sorgt allerdings für einigermaßen vorhersagbare Laufzeiten, da der 68020 vor dessen Ausführung erst die Instruction-Pipe leert. Über diese Änderungen hinaus reicht es, die Initialisierungsroutine für den MFP2 mit NOPs zu überschreiben.

Seit dem STE gibt es sogenannte Cookies, über die man Software installieren kann. Das TT-TOS signalisiert über die System-Cookies CPU, VDO, MCH, SWI, SND und FPU das Vorhandensein bestimmter Erweiterungen der Hardware. Einen Teil davon setzt das Betriebssystem einfach blind, was Anpassungen erforderlich macht, andere werden an die aktuelle Hardware angepaßt. So setzt das TT-TOS den CPU-Cookie automatisch auf 68020 und erkennt auch die FPU ohne Patch selbständig.

Dagegen entgeht ihm das Fehlen der internen SCSI-Platte des TT. Damit das Booten von einer ACSI-Platte überhaupt funktioniert, war es nötig, die SCSI-Routinen durch einen Fehlerausgang zu ersetzen. Die erweiterten XBIOS-Routinen, die auf die TT-Hardware zugreifen, habe ich einfach durch Reduzieren der Konstanten am Anfang der XBIOS-Sprungtabelle unzugänglich gemacht, die die Anzahl der eingetragenen Funktionen angibt.

Das wär's

Mehr Patches sind nicht notwendig, da das Betriebssystem im weiteren entweder die Cookies auswertet oder das Vorhandensein der TT-Hardware durch sauberes Abfangen des Bus-Fehlers abfragt (z. B. beim SCC 8530). Der Desktop testet zum Beispiel den VDO-Cookie und bietet auf dem ST dann in der Dialogbox für die Auswahl der Bildschirm-Modi nur die entsprechenden an. Leider wird dabei auch der letzte Eintrag im Extras-Menü von Cache auf Blitter abgeändert, so daß man dieses Feature des 68020 nicht über den Desktop steuern kann. Dafür unterstützt das TT-TOS den ST-Blitter vollständig, obwohl dieser im Atari-Flaggschiff überhaupt nicht eingebaut werden kann!

Um dennoch das Cache des 68020 unter Kontrolle zu bringen, kann man sich des neuen TT-Kontrollfelds bedienen, das sich durch CPX-Module erweitern läßt. Eines dieser Module (GENERAL.CPX) erlaubt unter anderem das An- und Abschalten des 68030-Cache. Nach zwei kleinen Patches steuert es auch den Code-Cache des 68020 auf der PAK.

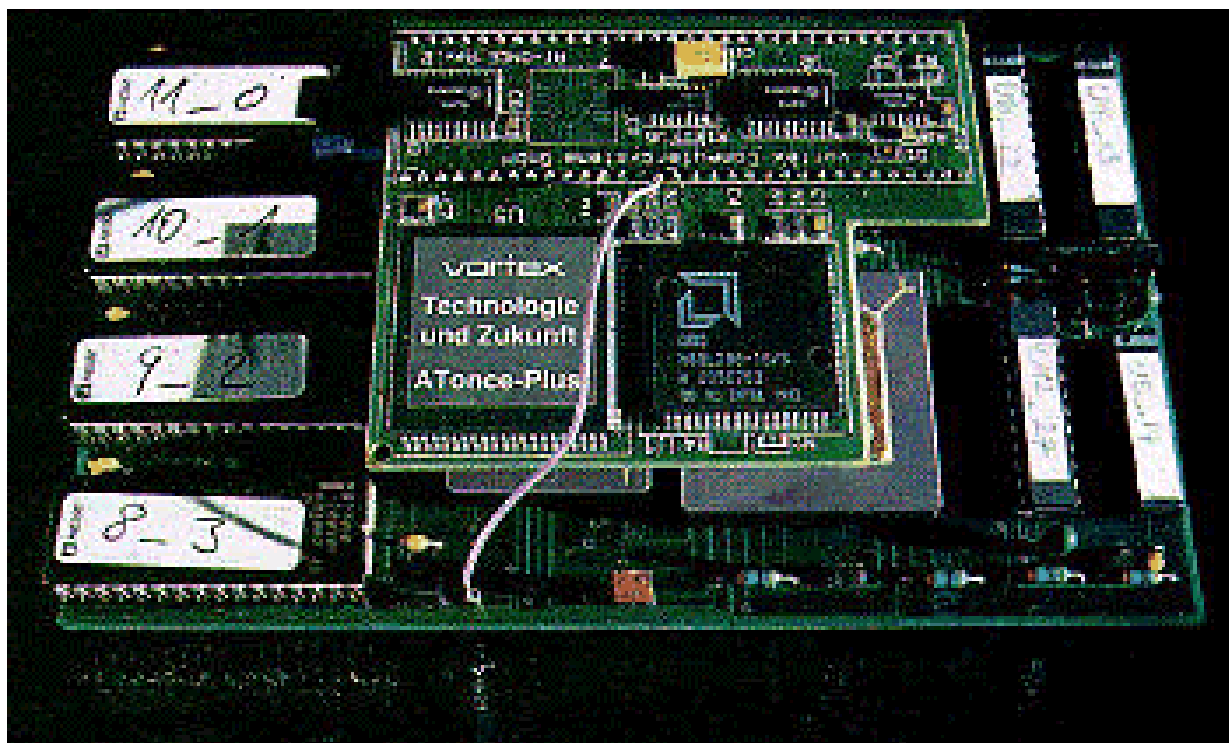
Mimose

Ein kleines Problem überdauert alle Patches: bei meinem ST klappt die Initialisierung des Shifters nicht immer, was sich beim Monochrom-Monitor durch ein nach links versetztes Bild oder durch senkrechte schwarze Streifen bemerkbar macht. Abhilfe schafft wiederholtes Neustarten über die Tasten Ctrl-Alt-Del, bis der Bildschirm in Ordnung ist. Der TT-Shifter scheint eben etwas ausgereifter zu sein als der mimosenhafte im ST.

Um das neue Betriebssystem auf der PAK zu starten, gibt es zwei Möglichkeiten: entweder durch endgültiges Ändern des Reset-Vektors in den ROMS der Hauptplatine auf \$E0 0000 (an Adresse \$4) oder durch das abgedruckte Programm, das einem die Wahl zwischen altem und neuem TOS läßt. Es gehört an die erste Stelle im AUTO-Ordner. Zur Vermeidung einer endlosen Reset-Schleife fragt es zunächst ab, ob das TT-TOS bereits läuft. Falls nicht, überprüft es, ob sich bei \$E0 0000 überhaupt ROM beziehungsweise ein gültiger TOS-Anfang befindet. Ist dies der Fall und die linke Shift-Taste nicht gedrückt, werden die Interrupts gesperrt und nach \$E0 0030 gesprungen. Übrigens erwartet das TT-TOS beim ersten Start einen Tastendruck, ehe es versucht, von einer Festplatte zu booten. Leider ist das TT-TOS etwas größer als 256 KByte, so daß Sie für das PAK-ROM die (nicht ganz billigen) 27C010-120 verwenden müssen.

Andere Welt

Daß die neue PAK sogar mit den recht kritischen PC-Emulatoren klarkommt, spricht eindeutig für das neue Konzept. Mit dem gängigen ATSpeed von Sack Elektronik gab es ebensowenig Probleme wie mit dem schnellen 16-MHz-ATOnce von Vortex. Bei letzterem empfiehlt es sich, zur Schonung der überempfindlichen Atari-Custom-Chips das 16-MHz-Signal nicht von Pin 39 des Video-Shifters abzugreifen, sondern direkt von der PAK zu holen (siehe Detailfoto). Der 16-MHz-Takt liegt 'verkehrsgünstig' beispielsweise an Pin 8 der (nicht bestückten) Oszillatorfassung an, wenn man Jumper J5 und J6 steckt.



PC-Emulatoren lassen sich über zwei aufgelötete SIL-Streifen auf der PAK montieren. Der Vortex ATOnce Plus verarbeitet auch den 16-MHz-Takt, den die PAK selbst erzeugt.

Schnelle Freundin

Der Amiga in seinen unzähligen Permutationen konnte sich bislang nur unter Einsatz zusätzlicher Hardware mit der alten PAK anfreunden, weil er bezüglich des /AS-Timings als besonders pingelig galt. Mit dem neuen Design entfällt jegliche Trial-and-Error-Basterei, lediglich zwei passende GALs sind einzusetzen (U6_40 und ggf. U12_41), die den Amiga-ROM-Bereich von \$F8 0000 bis \$FF FFFF betreuen. Das abgedruckte GFA-BASIC-Programm ROMSplit gilt sowohl für Atari (mit den oben angeführten Einschränkungen) als auch für Amiga, lediglich die Kommentare sind je nach Rechner einzuarbeiten. Atari und Amiga bieten leider keine generelle Copro-Unterstützung; hier ist man auf 881-fähige Applikationen oder Compiler angewiesen. (ad/cm)

Literatur

- [1] Johannes Assenbaum: Mehr CPU, PAK-68 - Die Prozessor-Austausch-Karte für 68000-Rechner, c't 7/87, S. 68
- [2] Carsten Meyer: PAK im Mac, 68020 und 68881 im kleinen Macintosh, c't 12/88, S. 126
- [3] Detlef Wulf: Die 32-Bit-Kur für TOS und GEM, c't 3/88, S. 108
- [4] Leo Drisis: Der Real-Renner, 68881-Einbindung für die PAK im Mac, c't 5/90, S. 190

Insgesamt sieben verschiedene GAL-Sätze sind nötig, um alle 68000-Rechnerfamilien abzudecken.

Rechner und passende GALs

Generic

ohne ROM-Unterstützung

U4_13, U5_14, U6_15

Atari ST

Normal, TOS bei \$FC 0000

U4_13, U5_14, U6_22, U12_21

RAM auf PAK bei \$E0 0000

U4_13, U5_14, U6_22, U12_20

TT-TOS auf PAK bei \$E0 0000

U4_13, U5_14, U6_22, U12_24

Amiga

ROM von \$F8 0000 bis \$FF FFFF

U4_13, U5_14, U6_40, U12_41

Macintosh

ROM von \$40 0000 bis \$47 FFFF

68000 ausgelötet

U4_13, U5_14, U6_30, U12_31

68000 nicht ausgelötet

U4_13, U5_16, U6_30, U12_31

Kasten 2

Patches für TT-TOS

Video Sync Mode von 1 auf 0 ändern

	alt	neu
\$E000EF	01	00

Speicherbestückung ST-RAM (Shadow liegt an anderen Adressen)

\$E0013C	10	04
\$E00140	10	04
\$E0014E	10	04
\$E0015E	10	04

Initialisierung der SCU (durch NOPs ersetzen)

\$E0024C	08	4E
	F8	71
	00	4E
	00	71
	9E	4E
	09	71

Das Video-RAM ist nur 32 statt 150 KByte groß

\$E00277	02	00
\$E00278	59	80
\$E0027C	25	07
\$E0027D	8F	FF

_VDO und _MCH Cookies auf 0

\$E003BF	02	00
\$E003CB	02	00

_SND Cookie auf 0

\$E003EA	30	70
	38	00
	80	4E
	00	71

Monochrom Mode

\$E00567 06 02

2. TT-Video-Mode-Register (\$FF8262) für ST auf \$FF8260

\$E0056B 62 60

SCU

\$E0080C	10	4E
	38	71
	8E	70
	09	FF

Monitor-Change-Abfrage (Zugriff auf Sound-DMA-Controlregister durch NOPs ersetzen)

\$E00998	10	4E
	38	71
	89	4E
	01	71
\$E009B4	B0	4E
	38	71
	89	4E
	01	71
	66	70
	DE	00

Video-Mode-Erkennung bei Monitor-Change

	alt	neu
\$E009C7	62	60
\$E009CB	07	03
\$E009CF	06	02
\$E009DF	06	02
\$E009EF	06	02
\$E009FB	62	60
\$E00A05	62	60

Anzahl XBIOS-Funktionen reduzieren

\$E00B19 60 41

Low-Byte der Video-RAM-Startadresse im ST nicht setzbar

\$E00E3E	10	4E
	38	71
	82	4E
	0D	71

Nochmal Video-Mode-Gemauschel

\$E00E4F	62	60
\$E00E53	07	03
\$E00E8D	62	60
\$E00E99	62	60
\$E00F4B	62	60

Low-Byte der Startadresse des Video-RAM immer als 0 lesen

\$E00F61	38	3C
	82	00
0D	00	

SCSI-I/O muß immer Fehler -12 zurückgeben

\$E0169E	2F	08
	70	F4
\$E016A0	02	40
	4E	75
\$E016DA	02	70
	40	F4
	00	4E
	07	75
\$E01834	11	FC
	4E	75

TT-Palette

\$E06253 56 75

Cache-Abfrage auf Code-Cache beschränken

\$E2ECE2	B0	C0
\$E2ECE6	31	00
\$E2ECE7	11	01
\$E2ECE8	66	67

Patches für GENERAL.CPX mit Dateilänge 11238

\$1563	14	1E	;CPU-Typ in Cookie ;Cache-Abfrage
\$159E	C0	B0	
\$15A2	00	31	
\$15A3	01	11	
\$15A4	67	66	

Kasten 3

In den rund 300 KByte des TT-TOS kommen doch einige Änderungen zusammen. Die jeweils erste Zeile beinhaltet die alten Werte, die darunterliegende die neuen Patches.

Patches für TT-TOS V3.1

Initialisierung 68030 MMU nach Reset (durch NOPs ersetzen)

[illegible]

Initialisierung des DMA Sounds (durch NOPs ersetzen)

[illegible]

Suche nach TT-Fast-RAM (durch NOPs ersetzen)

[illegible]

Allgemeine Delay-Routine (benutzt im TT den MFP2, s. Listing)

\$E01308	61	0A														
	60	0C														
\$E01314	48	A7	C0	00	40	E7	00	7C	07	-	12	38	FA	9D	C2	3C
	4E	75	48	E7	C0	00	72	00	12	-	E0	48	02	40	00	07
\$E01324	00	0F	11	C1	FA	9D	08	B8	00	05	FA	89	11	FC	-	DF
	10	3B	00	10	C2	C0	4E	71	51	C9	FF	FC	4C	DF	-	03
\$E01334	FA	8D	08	B8	00	05	FA	95	08	F8	-	05	FA			
	4E	75	01	04	0A	10	32	40	64	C8	-	C8	00			

Noch ein MMU-Init

[illegible]

MFP-Init (MFP2 fehlt beim ST)

\$E027E6	41	F8	FA	81	70	00	01	C8	00	00	01	C8	00	08	01	C8
	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71
\$E027F6	00	10	11	7C	00	58	00	16								
	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71
\$E02844	41	F8	FA	81	20	3C	00	88	01	05	01	C8	00	26	11	FC
	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71				
\$E02854	00	01	FA	9D	11	FC	00	02	FA	A5						
	4E	71	4E	71	4E	71	4E	71	4E	71	4E	71				
\$E02986	00	38	00	1E	FA	87	00	38	00	1E	FA	93				
	4E	71	4E	71	4E	71	4E	71	4E	71						

Nochmal MMU

\$E03A5A	F0	39	40	00	00	E3	0F	AE	F0	39	08	00	00	E3	0F	AE
	4E	71	4E	71	4E	71	4E	71								
	4E	71	4E	71	4E	71	4E	71								
\$E03A6A	F0	39	0C	00	00	E3	0F	AE								
	4E	71	4E	71	4E	71	4E	71								

Power-On-Abfrage über SCU-Control-Register(durch NOPs ersetzen)

\$E00112	08	38	00	00	8E	09	67	14
	4E	71						

Kasten 4

Die HL-Benchmarks wurden um den Test `Komplex` erweitert, um aufwendige Betriebssystemoperationen besser beurteilen zu können. `Komplex` zeichnet eine sehr reichhaltige Dialogbox zehnmal auf den Mac-Schirm. Als Vergleichsgerät diente uns ein SE im Originalzustand und ein SE mit einer kommerziellen 68030-Beschleunigerkarte, allerdings ohne Coprozessor.

Benchmarks Macintosh

Rechner	SE	SE/PAK	SE/PAK	SE tn
	original	ohne ROM	mit ROM	68030-Board
Integer	0,32	0,08	0,08	0,07
Real	32,22	8,22	7,30	23,53
TrigLog	66,23	1,08	0,97	51,90
Text	9,27	5,88	4,78	5,30
Graph	6,80	6,72	4,73	6,05
Komplex	12,07	9,58	7,32	8,65
Store	2,5	2,62	2,33	2,5

(Werte in Sekunden)

Kasten 5

Durch den Befehls-Cache erreicht auch der Atari mit der PAK eine überzeugende Performance, hier ermittelt mit QuickIndex 2.1.

Benchmarks Atari

Rechner Mega-ST Mega-ST/PAK Mega-ST/PAK

Test	original	ohne ROM	mit ROM
CPU Mem	100	242	244
CPU Reg	100	402	406
CPU Div	100	502	504
CPU Shift	100	1708	1737
DMA Read	161	161	161
GEMDOS	100	100	100
TOS Text	100	124	182
TOS String	100	109	169
TOS Scroll	100	139	153
GEM Dialog	100	131	174

(Werte in Prozent)

Kasten 6

Das ROMSplit-Programm liest die System-ROMs aus und verteilt deren Inhalt mit Offset auf vier Binärfiles. Die Mac-Version nimmt zusätzlich einen kleinen Patch vor.

```
0  ' ROM-Split, bb 1991, GFA-BASIC 3.5 Atari/Amiga
1  '
2  ' AMIGA: RESERVE 300000 bzw. 540000
3  '
4  bs_len%=&H40000                                ! Größe des Betriebssystems
5  ' $30000 für ST
6  ' $40000 für TT/M-STE und Amiga-BS <1.3
7  ' $80000 für Amiga-BS 2.0
8  dst_pfad$="F:\zbv\"                               ! Destination-Pfad für 4 Split-Files
9  '
10 DIM buffer|(bs_len%)                             ! Puffer für BS
11 buff_adr%=VARPTR(buffer|(0))                     ! Puffer-Adresse
12 rom_base%=LPEEK(LPEEK(&H4F2)+8)                  ! BS-Anfang, gültig bei allen Ataris
13 ' AMIGA: rom_base%=$FC0000 bzw. $F80000 (2.0)
14 '
15 REPEAT
16   INPUT "Datei (D) oder ROM (R) splitten ",f$
17   f$=UPPER$(f$)
18 UNTIL f$="D" OR f$="R"
19 '
20 IF f$="D"
21   FILESELECT #"TOS-IMG-File wählen","\*.IMG","",wfile$
22   ' AMIGA: FILESELECT "Kickstart-File wählen","Load","",wfile$
23   IF wfile$=""
24     END
25   ENDIF
26   BLOAD wfile$,buff_adr%                          ! File in den Puffer laden
27 ELSE
28   BMOVE rom_base%,buff_adr%,bs_len%               ! aus dem ROM in den Puffer
29 ENDIF
30 '
31 OPEN "o",#1,dst_pfad$+"U11_ROM0.img"
32 OPEN "o",#2,dst_pfad$+"U10_ROM1.img"
33 OPEN "o",#3,dst_pfad$+"U09_ROM2.img"
34 OPEN "o",#4,dst_pfad$+"U08_ROM3.img"
35 '
36 FOR i%=buff_adr% TO buff_adr%+bs_len%-4 STEP 4
```

```

37 BPUT #1,i%,1
38 BPUT #2,i%+1,1
39 BPUT #3,i%+2,1
40 BPUT #4,i%+3,1
41 NEXT i%
42 CLOSE
! alle Files schließen

```

```

0 program ROMSplit;
1
2 {Schreibt ROM.img-Inhalt auf 4 Files verteilt auf Platte,}
3 {mit Langwort-Offset für 68020-PAK-MEM, THINK Pascal, Mac}
4 {C.Meyer 16.09.1991}
5
6
7 var
8     Befehl: Char;
9     vRefNum, refNum, errCode: Integer;
10    byteCount, MemBase, FileLen: LongInt;
11    BuffPtr: Ptr;
12    Buffer: Handle;
13    FileName: Str255;
14    searchCode, patchCode, patchOffset: LongInt;
15
16 const
17     Offset = 4;
18     ROMBase = $400000;
19     ROMLen = $40000;
20     search = $41F90040;
21     patch = $41F90048;
22
23 procedure writeOffset;
24     var
25         n, m: LongInt;
26 begin
27     errCode := Create(FileName, 0, '????', '????');
28     errCode := FSOpen(FileName, 0, refNum);
29     byteCount := 1;
30     m := round(FileLen / 4);
31     for n := 1 to m do
32         begin
33             errCode := FSWrite(refNum, byteCount, BuffPtr);
34             BuffPtr := Pointer(Ord(BuffPtr) + Offset);
35         end;
36     errCode := FSClose(refNum);
37     Writeln(FileName, ' geschrieben ');
38 end;
39
40 begin
41     repeat
42         showtext;
43     { Die Benutzerführung ist nicht ganz Mac-like... }
44     write('Datei ROM.img aus ROM schreiben, von Disk lesen und teilen,
45     Ende (s/l/e)? ');
46     readln(Befehl);
47     case Befehl of
48         's', 'S':
49             begin
50 { File mit ROM-Image schreiben, für evt. eigene Versuche }
51                 Writeln('ROM.img wird geschrieben, Base ', ROMBase,
52                 byteCount := ROMLen;
53                 BuffPtr := Pointer(ROMBase);
54                 errCode := Create('ROM.img', 0, '????', '????');
55                 errCode := FSOpen('ROM.img', 0, refNum);
56                 errCode := FSWrite(refNum, byteCount, BuffPtr);
57                 errCode := FSClose(refNum);
58             end;
59         'l', 'L':
60             begin
61 { richtet Buffer je nach Dateilänge ein, liest komplette Datei 'ROM.img' }
62                 if FSOpen('ROM.img', 0, refNum) = noErr then
63                     ;
64                 errCode := GetEOF(refNum, byteCount);
65                 FileLen := byteCount;
66                 writeln('ROM.img wird gelesen, Länge: ', FileLen);
67                 Buffer := NewHandle(byteCount + 16);
68                 HLock(Buffer);
69                 if Buffer = nil then
70                     begin
71                         errCode := FSClose(refNum);

```

```

71                                     writeln('ROM.img nicht gefunden oder
    Speicher unzureichend');
72                                     SysBeep(0)
73                                     end
74                                     else
75                                     begin
76                                     errCode := FSRead(refNum, byteCount,
77                                     errCode := FSClose(refNum);
78
79 { System-ROM patchen: CRC-Routine auf gespiegeltes altes ROM umlenken }
80 { Damit wird zum einen die CRC-Routine überlistet, zum anderen ein Zugriff }
81 { auf den Bereich oberhalb des PAK-ROM ausgelöst, was das System-ROM }
82 { aus unterem RAM-Bereich wieder entfernt. Die BBU-Logik schaltet nach }
83 { einem Reset bei allen Zugriffen zwischen $400000 und $4FFFFFF das ROM um.}
84
85                                     searchCode := search;
86                                     patchCode := patch;
87                                     if Munger(Buffer, 0, @searchCode, 4,
88                                     writeln('WARNUNG: Patch-Code
89
90                                     writeln('Bitte nun etwas Geduld...');
91                                     BuffPtr := Buffer^;
92                                     FileName := 'U11_ROM0.img';
93                                     writeOffset;
94
95                                     BuffPtr := Pointer(Ord(Buffer^) + 1);
96                                     FileName := 'U10_ROM1.img';
97                                     writeOffset;
98
99                                     BuffPtr := Pointer(Ord(Buffer^) + 2);
100                                    FileName := 'U09_ROM2.img';
101                                    writeOffset;
102
103                                    BuffPtr := Pointer(Ord(Buffer^) + 3);
104                                    FileName := 'U08_ROM3.img';
105                                    writeOffset;
106                                    end;
107
108                                    HUnLock(Buffer);
109                                    if Buffer <> nil then
110                                        DisposHandle(Buffer);
111                                    errCode := FlushVol(nil, 0);
112                                    end;
113                                end;
114    until Befehl = 'e';
115 end.

```

Breit-bereit

Schnelles 32-Bit-RAM für die PAK68/2 im Amiga

Holger Klamp

Der Betrieb des Amiga mit schnellem Speicher auf der PAK68/2 war bisher mit unserem GAL-Fundus nicht vorgesehen. Mit einigen Änderungen läßt sich aber auch Commodores Animationsprofi auf 32-Bit-RAM umrüsten.

Bevor wir mit Logikanalysator und Speicheroszilloskop den letzten Geheimnissen einiger widerspenstiger Amigas auf die Spur kommen, damit auch dort endlich der Betrieb mit 2er-Kickstart in den PAK-ROMs möglich wird, liefern wir Ihnen hier eine interessante Hybridlösung: 32-Bit-RAM statt ROM, wie einst beim Atari, ermöglicht die Einbindung schnellen Arbeitsspeichers.

Neue GALs

Die Änderungen beschränken sich lediglich auf die Neuprogrammierung zweier GALs (U12 und U6). U12 muß, wie beim Atari auch, durch die größere Type 22V10 ersetzt werden, weil hier mehr Logik-Gleichungen zu berücksichtigen sind. Diese beiden GALs sind, wie bereits bekannt, unter anderem für die Speicherunterstützung zuständig. Im Amiga muß nun U12 dazu gebracht werden, das 32-Bit-RAM als solches zu behandeln - also auch Schreibzugriffe zu delegieren - und es an eine vorbestimmte Adresse zu legen.

Der Adreßbereich des 32-Bit-RAMs wurde von mir aus Kompatibilitätsgründen auf \$880000 bis \$8FFFFFF festgelegt. Solange man nicht mehr als 6 MByte Fast-RAM hat, dürfte es damit keine Probleme geben. Trifft dies jedoch zu, dann könnte man eventuell den Adreßbereich ab \$100000 bis \$1FFFFFF nutzen. Hier muß man aber darauf achten, daß weniger als 2 MByte Chip-Memory vorhanden sind.

Alles im Griff

Hauptproblem ist die korrekte Ansteuerung der RAM-Chip-Selects, abhängig von den vom 68020 angelegten Zugriffs-SIZn und der Adresse. Eine kleine Tabelle verdeutlicht die Funktionen von SIZn:

SIZ1 SIZ0 RAM Zugriff

0 0 Langwort
0 1 Byte
1 0 Wort
1 1 3 Byte

Der Adreßoffset von der mittels A2...A31 eingestellten Basisadresse wird über A0 und A1 bestimmt:

A1 A0 Offset

0 0 + 0 Byte
0 1 + 1 Byte
1 0 + 2 Bytes
1 1 + 3 Bytes

Aus diesen Tabellen ermittelt U12, welche Chip-Selects bei welchem Speicherzugriff aktiv werden sollen. Weitere Informationen erhalten Sie in [2].

Im GAL-Listing für U6 wird jetzt nur noch der String *addmem* ausgewertet. Er wird aktiv, wenn das 32-Bit-RAM angesprochen wird. Es werden dabei nur die Adressen A23 bis A19 benutzt. Um die häufig aufgetretenen Timingprobleme bei manchen Amigas zu unterbinden, wird der String *sysrom* auf GND gelegt. Damit ist der von acht auf fünf Zyklen verkürzte ROM-Zugriff auf die Hauptplatine nicht mehr möglich.

Springer-Problem

Für den Betrieb der RAM-Erweiterung ist folgende Jumperstellung zu wählen:

J2: offen

J3: offen

J4: gesteckt

J7: gesteckt auf 2/3 und 4/5

J8 bis J11: gesteckt auf 1/2

J12: gesteckt auf 1/2

J13: gesteckt

Als PAK-RAMs finden, wie beim Atari auch, die Typen 431000 von NEC oder dazu pinkompatible Bausteine Verwendung. Achten Sie beim Einkauf auf eine maximale Zugriffszeit von 120 ns. Um die 512 KByte RAM nutzen zu können, muß man dem System mitteilen, wo der Speicherbereich liegt und wie groß er ist. Dazu fügt man in die Startup-Sequenz ganz zu Anfang folgende Zeilen ein:

```
addmem 880000 8FFFFFF >NIL:
fastmemfirst ; nur bei Kickstart < 2.0
```

Hier noch ein Tip für alle, die sich die GALs selber brennen wollen: Bei der Wahl des Adreßraumes sollte man bei schon vorhandenem FastMem darauf achten, die Anfangsadresse immer an volle MByte-Adressen plus 512 KByte zu legen. Es kann sonst passieren, daß das Autoconfig-Programm der vorhandenen Speicherkarte das PAK-RAM automatisch anmeldet. Mein Multi-Evolution-Controller wollte zum Beispiel unbedingt aus den 512 KByte volle 2 MByte Speicher machen. Solange nur die ersten 512 KByte benutzt werden, geht alles gut, doch dann schlägt der GURU unweigerlich zu. (cm)

Literatur

[1] Manfred Völkel, Carsten Meyer: Tausch mit Turbo, PAK-68/2 - mehr Dampf für 68000-Rechner, c't 10/91, S. 178

[2] MC68020 32-Bit Microprocessor User's Manual, Motorola/Prentice Hall 1985

[3] Hilf, Nausch: M68000-Familie, te-wi-Verlag 1984

Kasten 1

Da nicht jeder Amiga-Besitzer die Möglichkeit hat, Atari- oder Mac-Disketten zu lesen, finden Sie hier die beiden GAL-Listings für U6 (P20_42) und U12 (P20_43).

```
0 ;PALASM Design Description
1 ;
2 ;          AMIGA  500, 1000, 2000
3 ;
4 ;  Address Decoder für FPU, normaler ROM Zugriff, nur RAM Unterstützung
5 ;  Busererror, CPUSPACE, ...
6 ;
7 ;  Jumper 2 :   ROM auf PAK (jetzt ohne Bedeutung)
8 ;           3 :   deaktiviert PAK Mem , erlaubt die Verwendung von
9 ;               A24 - A31 als TAG Bits in mancher Software
10 ;           4 :   deaktiviert FPU d.h. erzeugt Busererror
11 ;
12 ;----- Declaration Segment -----
```

```

13 TITLE      PAK 68020-16
14 PATTERN    p20_42_U6
15 REVISION    1
16 AUTHOR      geaendert von HOLGER KLAMP
17 COMPANY     -----
18 DATE        02.07.92
19
20 CHIP        _p20_42 PALCE20V8
21
22 ;----- PIN Declarations -----
23 PIN 17      /csfpu      COMBINATORIAL ; OUTPUT
24 PIN 18      /cpuspace   COMBINATORIAL ; OUTPUT
25 PIN 19      /cycle_00   COMBINATORIAL ; OUTPUT
26 PIN 20      /romsel     COMBINATORIAL ; OUTPUT
27 PIN 21      /berr_20    COMBINATORIAL ; OUTPUT
28 PIN 22      /berr_00    ; INPUT
29 PIN 23      /as_20      ; INPUT
30 PIN 11      /jp2        ; INPUT
31 PIN 15      /jp3        ; INPUT
32 PIN 16      /jp4        ; INPUT
33 PIN 1..8    a[16..23]   ; INPUT
34 PIN 9       a29         ; INPUT
35 PIN 10      a30         ; INPUT
36 PIN 13      fc0         ; INPUT
37 PIN 14      fc1         ; INPUT
38
39 ; ROM Adressen :
40 ; sysrom = Systemrom  AMIGA F80000 - FFFFFF (hier nicht verwendet)
41 ; addmem = PAK-RAM 32-bit 512kB ab 880000 - 8FFFFFF an diesen Adressen
42 ;          liegt das RAM; ACHTUNG Bei mehr als 6MB FastMem kann es zu
43 ;          Überschneidungen des RAMs kommen
44
45 STRING sysrom '( GND )'
46 STRING addmem '( a[23] * /a[22] * /a[21] * /a[20] * a[19] )' ;$88
47
48
49 ;----- Boolean Equation Segment -----
50 EQUATIONS
51 MINIMIZE_ON
52
53 cycle_00 =/( a30 * /a29 * /JP3
54             + sysrom * jp2
55             + addmem
56             + fc0 * fc1 * /a[16] * a[17] * /a[18] * /a[19] ; fpu
57             )
58 romsel = /JP2 * sysrom
59 csfpu = fc0 * fc1 * /a[19] * /a[18] * a[17] * /a[16] * /JP4 ;
60 cpuspace = fc0 * fc1 ;
61
62 ;-----
63 ; Buserror wird erzeugt bei |
64 ; |
65 ; Buserror vom 68000 System |
66 ; FPU Select wenn JP4 gesteckt d.h. keine FPU im System |
67 ; (erlaubt Software Emulation der Line F Befehle) |
68 ; Breakpoint ACK cycle ( braucht 68851 PMMU ) |
69 ; Access Level fetch " " " |
70 ;-----
71
72
73 berr_20 = as_20 * ( berr_00
74             + fc0 * fc1 * /a[19] * /a[18] * a[17] * /a[16] * JP4
75             + fc0 * fc1 * /a[19] * /a[18] * /a[17] * /a[16]
76             + fc0 * fc1 * /a[19] * /a[18] * /a[17] * a[16]
77             );
78
79 ;----- Simulation Segment -----
80 SIMULATION
81
82 ;-----
83 0 ;PALASM Design Description
84 1 ;
85 2 ; Address Decoder für On Board Speicher RAM
86 3 ;
87 4 ; Für AMIGA 880000-8FFFFFF;
88 5 ;----- Declaration Segment -----
89 6
90 7 TITLE 68020-16 PAK
91 8 PATTERN P20-43-U12

```

```

9 REVISION 1
10 AUTHOR    geaendert von HOLGER KLAMP
11 COMPANY   -----
12 DATE      03.07.92
13
14 CHIP      _p20_43  PALCE22V10
15
16 ;----- PIN Declarations -----
17
18
19 PIN 1      /as_20      COMBINATORIAL ; INPUT
20 PIN 2      /cpuspace   COMBINATORIAL ; INPUT
21 PIN 3      siz0        COMBINATORIAL ; INPUT
22 PIN 4      siz1        COMBINATORIAL ; INPUT
23 PIN 5..10  a[18..23]   COMBINATORIAL ; INPUT
24 PIN 11     a[29]       COMBINATORIAL ; INPUT
25 PIN 12     GND         ; INPUT
26 PIN 13     a[30]       COMBINATORIAL ; INPUT
27 PIN 14     a1          COMBINATORIAL ; INPUT
28 PIN 15     a0          COMBINATORIAL ; INPUT
29 PIN 16     a[16]       COMBINATORIAL ; INPUT
30 PIN 17     /cs11       COMBINATORIAL ; OUTPUT
31 PIN 18     /cs10       COMBINATORIAL ; OUTPUT
32 PIN 19     /cs9        COMBINATORIAL ; OUTPUT
33 PIN 20     /cs8        COMBINATORIAL ; OUTPUT
34 PIN 21     /dsack1     COMBINATORIAL ; OUTPUT
35 PIN 22     /dsack0     COMBINATORIAL ; OUTPUT
36 PIN 23     a[17]       COMBINATORIAL ; INPUT
37 PIN 24     VCC         ;
38
39 string uucs '( /a0 * /a1 ) '
40 string umcs '( /a1*siz1 + a0*/a1 + /a1*/siz0 ) '
41 string lmcs '(a0*/a1*/siz0 + /a1*siz0*siz1 + /a1*/siz0*/siz1 + /a0*a1)'
42 string llcs '(a1*siz1 + a0*a1 + /siz0*/siz1 + a0*siz0*siz1)'
43
44 ;----- Boolean Equation Segment -----
45 EQUATIONS
46
47 MINIMIZE_ON
48
49 case ( a[23..16] ) begin
50   #h88..#h8F:
51     begin
52       cs11 = as_20 * /cpuspace * uucs
53       cs10 = as_20 * /cpuspace * umcs
54       cs9  = as_20 * /cpuspace * lmcs
55       cs8  = as_20 * /cpuspace * llcs
56     end
57   otherwise:
58     begin
59       cs11 = gnd
60       cs10 = gnd
61       cs9  = gnd
62       cs8  = gnd
63     end
64   end
65
66 dsack0.trst = /cpuspace * a[23] * /a[22] * /a[21] * /a[20] * a[19] ;$88
67 dsack1.trst = /cpuspace * a[23] * /a[22] * /a[21] * /a[20] * a[19] ;$88
68
69 dsack0 = as_20 ;
70 dsack1 = as_20 ;
71
72 ;----- Simulation Segment -----
73 SIMULATION
74
75 ;-----

```

Doppel-PAK

PAK-68/3 mit Cache und 68020/030 für 68000-Rechner

Holger Zimmermann

Ganz im Zeichen der fünften Zweierpotenz präsentiert sich die nunmehr dritte Auflage unseres Dauerbrenners PAK-68: 32 MHz, 32 Bit und 32 KByte Cache sind die gewichtigen Gründe, den optimalen 68000-Beschleuniger zum Selbstbau vorzustellen.

Die PAK-68 hat seit ihrer (nicht ganz ohne Komplikationen verlaufenen) Geburt im Juli 1987 viele Freunde gefunden. Anfangs [1] noch mit gemächlichen 8 MHz, später mit 16 MHz [2] und nun mit Taktfrequenzen fast nach Belieben getaktet, bleibt die Prozessor-Austauschkarte der konsequenteste Weg, einem vorhandenen Rechner zeitgemäße Leistung einzuhauchen.

Und die ist nicht von Pappe: Neben der eklatanten Beschleunigung kommen Atari-ST-Freunde in den Genuß der modernen TOS-Versionen 2.06 und (mit kleinen Patches) 3.06 im 32-Bit-Zugriff; mit der 68030-Version gibt's zusätzlich eine PMMU fürs Multi-TOS. Kompakt-Macs profitieren unter dem ressourcenfressenden System 7 besonders von der - je nach Ausstattung - Verdrei- bis Versechsfachung der Arbeitsgeschwindigkeit. Dabei blieb es unser Bestreben, so 68000-kompatibel wie möglich zu bleiben, damit auch kritische Rechner (Atari mit IMP-Chipsätzen, Amiga) nicht außen vor bleiben - obwohl es sicher immer noch Applikationen und Erweiterungen geben wird, die nicht 68020/030-konform aufgebaut sind. Erkundigen Sie sich also vor der PAK-Anschaffung, ob Ihr Lieblings-Editor auch auf größeren Maschinen (Atari TT, Falcon, Mac II) `spielt`.

Umweltfreundlich

Ein 68020 mit 8 oder 16 MHz ist aber mittlerweile auch nicht mehr ganz `state of the art`. Ganz oben auf der Wunschliste stand daher ein 68030, gefolgt von höherer Taktfrequenz und einem Second-Level-Cache. Zum Glück gibt es heute preiswerte Bauteile, die eine Implementation derlei Begierden ohne nennenswerte Hardwareverrenkungen und ohne Kontoaustrocknung ermöglichen. Zum alten Eisen respektive Plastik gehört Ihr ST oder Schuhkarton-Mac also noch lange nicht. Hier ist sie also, die PAK-68/3. Bei nahezu gleichen Abmessungen wie die PAK/2 hat sich `unter der Haube` einiges getan: Wahlweise eine 68020- oder 68030-CPU mit 16 bis 33 MHz Taktfrequenz bei 8 MHz Systemtakt, eine 68881- oder 68882-FPU, 256 oder 512 KByte ROM sowie ein 32 Bit breiter und 32 KByte großer Second-Level-Cache. Beim Nachbau der PAK/3 steht eine schwerwiegende Entscheidung gleich zu Anfang an: Soll es ein 68020 oder ein 68030 sein? Da es für beide Prozessoren jeweils eine eigene Platinenversion gibt, ist kein nachträglicher Wechsel möglich. Gewissermaßen als Ausgleich dazu können aber alle anderen Features auch nachträglich durch Austauschen der GALs hinzugefügt oder weggelassen werden. In Anbetracht der mitunter sehr preiswert angebotenen Gebrauchtprozessoren (siehe Kleinanzeigen) empfehlen wir bei nicht schon vorhandener alter PAK-68 die 030er-Version, die gegenüber der 020er bei gleicher Taktfrequenz einen Geschwindigkeitsvorteil von rund 15 Prozent bringt. Ansonsten können Sie die meisten Bauteile der alten PAK weiterverwenden, die GALs allerdings nur, wenn Sie eine geeignete Programmiereinrichtung besitzen. Natürlich muß der Prozessor die gewünschte Taktfrequenz verkraften.

Große Pläne

Der Schaltplan der 030er-Version der PAK/3 deckt im Prinzip die 020er mit ab. Da beim 68020 die Anschlüsse /STERM, /CIIN, /CIOUT, /MMUDIS, /CBREQ und /CBACK nicht vorhanden sind, entfallen in diesem Fall die entsprechenden Leitungen mitsamt zugehörigen Pullups und Jumpers. Die Pinbelegung beider CPUs ist leider deutlich unterschiedlich, was sich im Schaltplan nur im Kleingedruckten bemerkbar macht, beim Platinenlayout aber im Endeffekt zu den beiden unterschiedlichen Platinenversionen geführt hat. Bis auf die GALs, R1, R2 und eben die CPU sind beide Platinenversionen absolut gleich bestückt.

Das State-Machine-Konzept der PAK/2 hat sich in der Praxis bewährt, so daß es im Kern auch für die PAK/3 beibehalten werden konnte. CPU, FPU und die ROMs sind weitgehend wie beim Vorgänger verschaltet. Die synchrone Takterzeugung mit einem 74F86 als Verdoppler für den 8-MHz-Systemtakt ist ebenfalls identisch. Auch der Quarzoszillator war schon auf der PAK/2 zu finden, dort aber lediglich für die FPU zuständig. Bei entsprechender GAL-Bestückung kann die PAK nun auch asynchron zum Systemtakt (getestet bis 33 MHz) betrieben werden; mehr dazu in der nächsten c't. Die GALs U4, U5 und U6 haben ihre Funktion (und ihre Namen) weitgehend beibehalten. Bei den Gleichungen für U5 wird man noch einiges wiedererkennen, während bei U4 kaum ein Stein auf dem anderen geblieben ist. Gänzlich neu sind GAL U1, das die Kontrolle über die lokalen Buszyklen übernimmt (also bei ROM-Read und bei Cache-Hit) und GAL U2, das den Cache koordiniert.

Der 68020 hatte bereits einen Cache für Befehle in der CPU integriert, beim 68030 ist noch ein Datencache dazugekommen. Leider sind die internen Caches mit 256 Bytes nicht besonders üppig bemessen. Benchmarks benutzen häufig kurze Schleifen zum Testen der Performance, hier liegt der CPU-Cache noch gut im Rennen; in der Praxis bleibt davon jedoch nicht allzuviel übrig. Die PAK/3 hat deshalb einen 32 KByte großen Second-Level-Cache spendiert bekommen, und wer ihn mal probeweise ausschaltet, merkt, wie richtig diese Entscheidung war. Typische Anwendungen laufen bei Maximalausbau (32 MHz plus L2-Cache) in wenigstens sechsfacher 68000-Geschwindigkeit, einige Benchmarks zeigen gar zweistellige Performance-Gewinne. Im Vergleich zur alten PAK, die rund 250 Prozent der Leistung eines 68000 brachte, ein nennenswerter Fortschritt.

Hucke-PAK

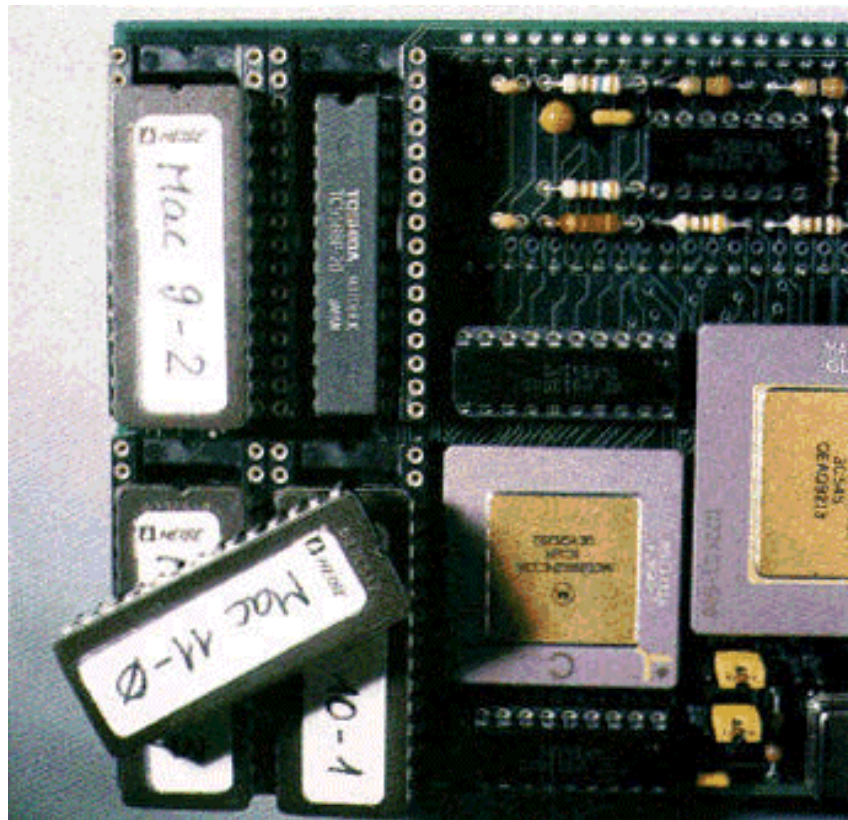
Für den L2-Cache sind zwei Tag-RAMs, vier schnelle SRAMs und zwei Datenbustreiber hinzugekommen. Die SRAMs (schmale Bauform, 300 mil) benötigen weitgehend dieselben Signale wie die ROMs (Breite 600 mil), so daß sich hier eine zweigeschossige Bauweise anbot. Nur dadurch war es auch möglich, die Abmessungen der PAK/2 beizubehalten. Nebenbei konnten wir auch einen kleinen Schönheitsfehler beseitigen: Die PAK überdeckt nun nicht mehr den PDS-Steckplatz im Mac SE.

Die vier SRAMs (je 8 KBit $\times$ 8, also 8 KByte) bilden zusammen einen 32 KByte großen Cache-Speicher, auf den die CPU sehr schnell und in voller Breite zugreifen kann. GAL U2 sorgt dafür, daß die aus dem Hauptspeicher geholten Daten sozusagen nebenbei im Cache abgelegt werden. Daten beispielsweise von der Hauptspeicheradresse 1 landen so auch im Cache unter Adresse 1. Da der Cache aber nur 32 768 Byte umfaßt, landen auch Daten von der Hauptspeicheradresse 32768+1 im Cache unter Adresse 1, wobei der vorherige Inhalt des Cache natürlich überschrieben wird. Wie kann das gutgehen?

Schildchen-Speicher

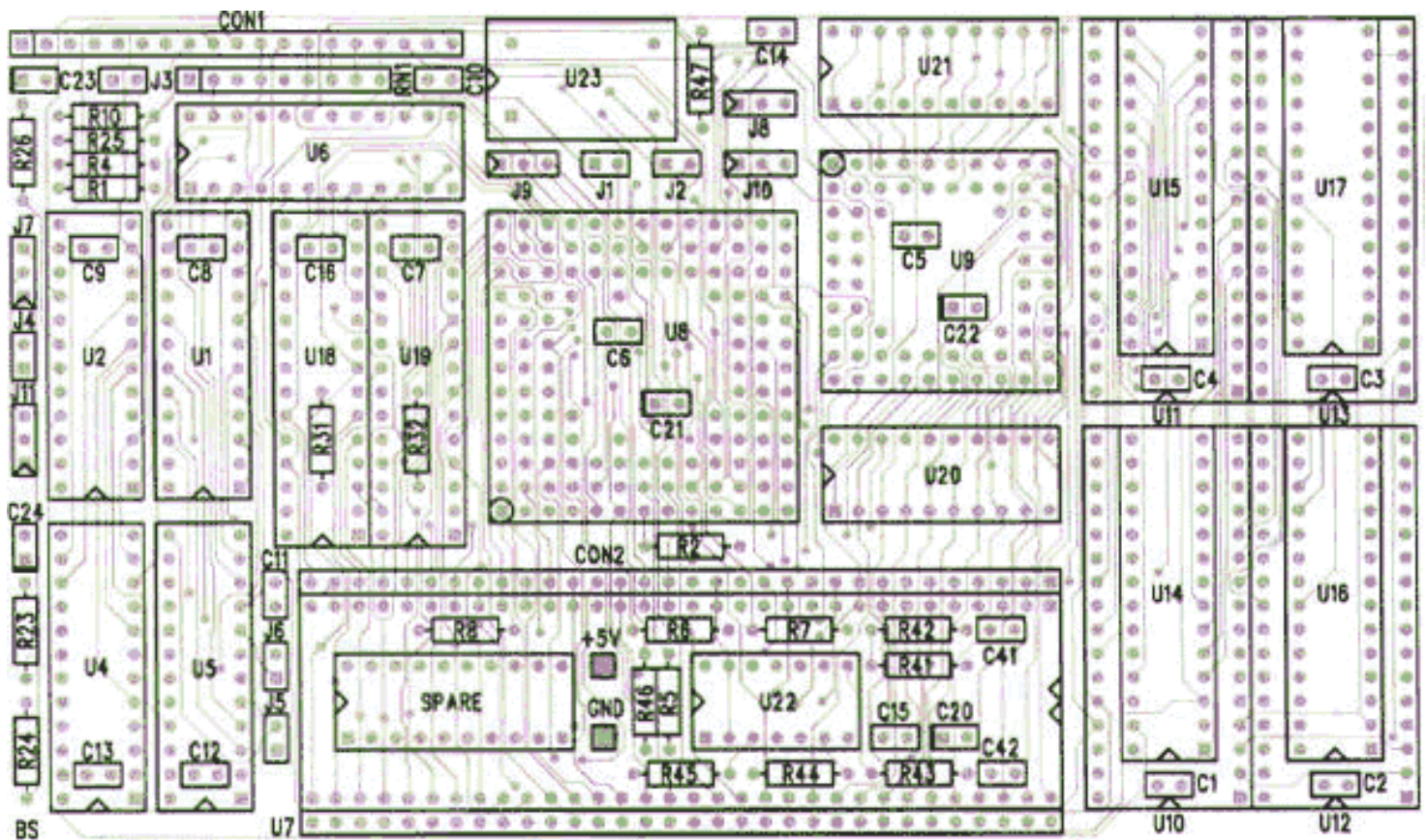
An dieser Stelle kommen die Tag-RAMs ins Spiel, das sind zunächst einmal ebenfalls statische RAMs mit 8K $\times$ 8, deren Datenleitungen aber an die CPU-Adreßleitungen A15 bis A21 angeschlossen sind. Jedesmal, wenn Daten in den Cache übernommen werden, kommen die dazugehörigen oberen Adreßbits im Tag-RAM zu liegen. Zu jedem einzelnen Datenwort im Cache steht hier also die Information, zu welcher 32-KByte-Page des Hauptspeichers der Inhalt des Cache gehört.

Wenn die CPU einen Lesezugriff initiiert, wird das Tag-RAM jetzt nicht ausgelesen, sondern im sogenannten Compare-Modus betrieben, wobei es die von außen angelegten Adreßbits A15 bis A21 mit den gespeicherten Adreßbits vergleicht. Der dazu notwendige Komparator ist im Tag-RAM integriert und meldet das Ergebnis über den Anschluß MATCH nach außen, im Falle der Übereinstimmung erhält man hier *high*-Pegel. Das ist das Zeichen für die CPU und ihre Read-Logik, daß sich die gesuchten Daten bereits im Cache befinden. Meldet das Tag-RAM keinen Treffer (MATCH ist *low*), dann gehört das Datenwort im Cache zu einer anderen Page, woraufhin die CPU jetzt doch im Hauptspeicher nachsehen muß. In der Hoffnung, das gerade geholte Datenwort werde in naher Zukunft nochmals gebraucht, legt es die Steuerlogik dann auch gleich im Cache ab.



Die Cache-SRAMs liegen unter den EPROM-Fassungen. Probieren Sie vor dem Einsatz von SRAM-Sockeln aus, ob die 'Türmchen' in Ihren Rechner passen. Vor allem im Kompakt-Mac geht es recht eng zu.

Wegen des 16 Bit breiten Sytembusses muß der Cache wortweise verwaltet werden, für den 32 Bit breiten Cache der PAK werden also zwei Tag-RAMs benötigt. Die sieben Adreßbits, die das Tag-RAM auswertet, decken 128 Pages, also den Adreßraum von 0 bis 4 MByte ab. Im Bereich oberhalb \$40 0000 ist der Second-Level-Cache unwirksam, was bei den in Frage kommenden Macs und Ataris keine Einschränkung bedeutet; die CPU hat das ROM auf der PAK ja ohnehin im direkten Zugriff.



Gleicher Plan für zwei Versionen: Bis auf die CPU sind beide neuen PAKs identisch. Der 64polige Sockel zur Verbindung mit der Hauptplatine gehört auf die Lötseite in die DIL-Reihen, die zur Platinenmitte weisen (siehe auch Detailfoto). R1 und R2 werden nur bei der 030-Version bestückt.

Das achte Bit der Tag-RAMs bildet das Valid-Bit und liegt daher an V_CC . Um alle Daten im Cache für ungültig zu erklären (z. B. bei DMA-Zugriffen oder Reset), werden die Tag-RAMs über den Anschluß /TCLR gelöscht und damit alle Bits auf *low* gesetzt. Damit erkennt der Komparator beim nächsten Lesezugriff in jedem Fall ein Cache Miss (kein Treffer) und lädt daraufhin den Cache neu.

UngeBursted

Im Ablauf eines Buszyklus gibt es auf der PAK/3 keine Unterschiede zwischen 68020 und 68030. Der synchrone Busmodus des 68030, der einst den Mac IIci so beschleunigte (Burst-Mode mit /STERM), darf hier leider nicht genutzt werden; zum einen sind die ROMs und Tag-RAMs bei 32 MHz zu langsam, zum anderen gäbe es ein Problem, wenn nach einem Lesezugriff auf den Hauptspeicher ein lokaler Lesezugriff auf der PAK/3 folgte, weil die Datenbustreiber des Mainboards (im Mac genauso wie im Atari) noch vom vorherigen Zugriff her eingeschaltet wären und somit dazwischenfunken würden.

Zur Veranschaulichung der Abläufe soll zunächst eine PAK/3 mit 16 MHz Taktfrequenz herhalten. Ein Buszyklus beginnt, indem die CPU die Adresse anlegt und, nachdem die Suche im CPU-internen Cache erfolglos verlaufen ist, /AS_20 mit fallender Flanke von CLK16 aktiviert. Falls dies ein ROM-Zugriff ist (/ROM von U6 aktiv) und sich ROMs auf der PAK befinden (J7 auf 1-2), schaltet U1 die ROMs ein und bestätigt sofort mit /DSACK0 und /DSACK1. Mit der nächsten fallenden Flanke von CLK16 wird dies von der CPU erkannt, worauf sie den Buszyklus genau einen CLK16-Takt später durch Desaktivieren von /AS_20 beendet. U1 hält /CYC_00 die ganze Zeit auf *high*, so daß U4 keinen Zugriff auf das Mainboard startet.

Angenommen, es findet ein Lesezugriff auf den Bereich \$0 bis \$40 0000 statt (/DRAM von U6 aktiv), der L2-Cache ist aktiviert (/TCLR auf *high*), aber mindestens eines der beiden Tag-RAMs meldet keinen Treffer (MAT0 oder MAT2 auf *low*): In diesem Fall geht /CYC_00 nach kurzer Wartezeit auf *low* und startet damit in U4 einen Zugriff auf das Mainboard, der im Prinzip wie bei der PAK/2 abläuft. Hier sorgt jetzt aber U2 dafür, daß das gelesene Datenwort auch in den SRAMs abgelegt und gleichzeitig die zugehörige Adresse im entsprechenden Tag-RAM festgehalten wird.

sich übrigens aus PGA-Typen mit Kelchfederkontakten (‘Präzisionsfassungen’), wenn überhaupt, nur mit roher Gewalt entfernen. Besser sind hier normale Ausführungen, wie sie etwa AMP herstellt. Sicherheitshalber sollten Sie auch den Tag-RAMs Sockel spendieren; die Stege der darüberliegenden EPROM-Fassungen müssen vorsichtig entfernt werden.

Wie bei der alten PAK stellt ein lötlseitiger 64poliger DIL-Sockel mit eingestecktem beidseitig ‘piekendem’ Adaptersockel oder ebensolchem SIL-Streifen die Verbindung zum Mainboard her. Aus Stabilitätsgründen ist diesmal das operative Entfernen der alten 68000-CPU - sofern nicht schon geschehen - unumgänglich, eine 64polige DIL-Fassung tritt an ihre Stelle. Ein GAL-Satz, der den Betrieb bei auf dem Mainboard verbleibender CPU gestattet, ist im Gegensatz zur PAK-68/2 nicht vorgesehen. Bewährt hat sich die Amputation sämtlicher 68000-Beinchen mit einem kleinen Seitenschneider. Einzeln lassen sich die Pins dann leicht aus der Platine ziehen. Wem bei dieser Prozedur ob der leidenden CPU die Tränen kommen, der kann dem beinlosen 68000 später eine Prothese in Form einer DIL-Fassung unterlöten und ihn für Testzwecke weiterverwenden.

Optional ist übrigens der Betrieb der 68000 umschaltbar neben dem 68020/030 möglich, wenn man einige Leitungen zur bislang leeren Spare-Fassung fädelt und ein entsprechend programmiertes GAL einsetzt (Tips dazu folgen). Leider waren die Verbindungen beim besten Willen nicht mehr auf der Platine unterzubringen, und da nicht jeder eine ‘Vorher/Nachher’-Option braucht, haben wir für diese nur eine freiverdrahtbare Leerfassung vorgesehen. Die auf der Platine vorhandene 68000-Fassung auf der Bestückungsseite kann alternativ auch zur Aufnahme von MSDOS-Emulatoren und ähnlichen Erweiterungen dienen, die sich direkt der Prozessorsignale bedienen.

Der Einsatz mittelschneller CPUs (20 und 25 MHz) bei asynchroner Taktung, das heißt, bei Verwendung des 32-MHz-GAL-Satzes und entsprechend ausgelegtem Quarzoszillator, führt in der Regel zu Problemen; auch ist eine asynchron laufende 20-MHz-CPU bedingt durch unvermeidliche Wartezyklen kaum schneller als die synchron getaktete 16-MHz-Version.

Was es bei der Anpassung des TOS 3.06 zu beachten gilt, wie die asynchronen Zugriffe der 32-MHz-Version funktionieren und wie man den Mac zur Nutzung der FPU und des Datencache überredet, erfahren Sie in der nächsten c’t. (cm)

Literatur

- [1] Manfred Völkel, Carsten Meyer, Tausch mit Turbo, PAK-68/2 - mehr Dampf für 68000-Rechner, c’t 10/91, S. 178
- [2] Manfred Völkel, Carsten Meyer, Tausch mit Turbo, PAK-68/2 in Mac, Atari und Amiga, c’t 11/91, S. 314
- [3] Manfred Völkel, Tausch mit Turbo, Tips zur PAK-68/2 im ST, c’t 12/91, S. 222
- [4] Manfred Völkel, Turbo für den Austauschmotor, Schneller Atari mit modifizierter PAK-68, c’t 3/91, S. 282
- [5] Carsten Meyer, Schnelle Freundin, Amiga mit PAK und 14 MHz, c’t 9/90, S. 338
- [6] Roland Beck, Hilfskraft: 68000-Prozessor trotz PAK-68/2, [c’t 7/92, S. 196](#)
- [7] Leo Drisis, Real-Renner, 68881-Einbindung für die PAK im Mac, c’t 5/90, S. 190
- [8] Carsten Meyer, Fast-PAK, 68000 und PAK-68 mit 16 MHz im Macintosh, c’t 1/90, S. 194
- [9] Johannes Assenbaum, Mehr CPU, PAK-68 - die Prozessor-Austausch-Karte für 68000-Rechner, c’t 8/87, S. 68

PAK-68/3-Jumper

Jumper Name	Funktion
J1 MMU disable	gesteckt MMU disabled offen enabled
J2 CPU Cache disable	gesteckt CPU-interner Cache disabled offen enabled
J3 FPU enable	gesteckt FPU aktiviert offen deaktiviert oder nicht vorhanden
J4 L2 Cache disable	gesteckt Second-Level-Cache ausgeschaltet offen eingeschaltet
J5 PAK enable	gesteckt PAK ist am 68000 Bus aktiv offen alle Ausgänge hochohmig
J6 BR	gesteckt Normalbetrieb offen Betrieb mit 68000 auf der PAK (extra Verdrahtung notwendig)
J7 ROM	1-2 ROM auf der PAK aktiviert 2-3 deaktiviert oder nicht vorhanden
J8 ROM_CS	1-2 /ROM_CS auf GND für langsame EPROMs 2-3 mit /ROM_OE verbunden spart Strom
J9 CPUCLK	1-2 CPU-Takt 32 MHz (asynchron) 2-3 16 MHz (synchron)
J9 FPUCLK	1-2 FPU-Takt 32 MHz (asynchron) 2-3 16 MHz (synchron) (die FPU sollte nicht langsamer laufen als die CPU)
J11 Cache Control	1-2 L2-Cache wird über die MMU kontrolliert 2-3 Umschaltung über Schreibzugriff auf das ROM offen L2-Cache immer an

Kasten 2

GAL-Bestückung

U1 P12_16 68020 mit 16 MHz
P12_32 68020 mit 32 MHz
P13_16 68030 mit 16 MHz
P13_32 68030 mit 32 MHz
U2 P2_JP für Mac, L2-Cache wird über Jumper geschaltet
bleibt im Normalfall immer eingeschaltet
P2_RST für Atari ST, L2-Cache bleibt nach Reset bis
zum ersten DMA-Zugriff ausgeschaltet
U4 P4_16 für 16 MHz
P4_32 für 32 MHz
U5 P5_16 für 16 MHz
P5_32 für 32 MHz
U6 P6_MAC für Mac
P6_ST für Atari

Beispiel: Sie wollen im Atari eine 030-PAK mit 16 MHz einsetzen. Benötigt werden also P13_16, P2_RST, P4_16, P5_16 und P6_ST.

Kasten 3

Bei der Bestückung sollten Sie in der Reihenfolge der Optionen vorgehen und jede Variante gründlich testen.

Stückliste

Halbleiter

U8	MC68020/68030, Taktfrequenz ab 16 MHz
U22	74F86
U1, U2, U4, U5, U6	GAL 20V8-15 Prog.
U23	Quarz-Osz. 32 MHz (Option 32 MHz)
U9	MC68882 (Option FPU)
U10, U11, U12, U13	27C010-120 (Option ROM) oder 27C512-120
U14, U15, U16, U17	Cypress CY7C185-25, (Option Cache) oder Toshiba TC5588-25
U18, U19	SGS MK48S74-25, (Option Cache) oder SGS MK48S80-25
U20, U21	74F541

Widerstände

RN1	Array 4k7 × 8
R1, R4, R7, R10, R23...R27	4k7
R8	2k2
R2, R5, R6	1k
R31, R32	330R
R41, R43	560R
R44...R47	33R

Kondensatoren

C1...C15	100n ker. RM2.5
C20...C24	Ta 10uF/16V

Sonstiges

1	IC-Fassung 128pol. PGA (evt. 124pol.)
1	IC-Fassung 68pol. PGA
1	Adapterfassung 64pol. DIL (oder SIL-Stecker o.ä.)
8	Fassung 16pol. SIL für ROMs oder
4	IC-Fassungen 32pol.
6	IC-Fassung 28pol. DIL schmal (oder 2 × 14er)
5	IC-Fassung 24pol. DIL schmal
2	IC-Fassung 20pol. DIL
2	IC-Fassung 14pol. DIL
1	Platine PAK-68/3-20 oder PAK-68/3-30
Pfostenleisten einreihig, Jumper nach Bedarf	

Kasten 4

Diese sieben GAL-Listings gelten für die 16-MHz-Version mit 68020 im Atari und Mac.

```
0 PAK 68/3, GAL 1: CPU-Clk State Machine für 68020 mit 16 MHz
1 26-08-93 V12_16a Holger Zimmermann @ PE
3 P12_16
4 %TYP
5 GAL20V8A
6 %PINS
7 CLK mat2 16m a1 !as_00 !tclr !wr !as_20 !prom !csp19 glp11
8 !OE !dram !roe !as_21 !doe !as_22 !sterm !cyc_00 !dsack1 !dsack0 mat0
9 %LOGIC
10
11 sterm.OE = GND;
12
13 as_21.OE = VCC;
14 as_21 = as_20 * 16m
15 + as_21 * !16m
16 + as_20 * as_21;
17
18 as_22 = as_20 * as_21 * !16m
19 + as_22 * !16m
20 + as_20 * as_22;
21
22 doe = as_20 * !as_00 * dram * !tclr * !wr * mat0 * mat2 * as_22
23 + as_20 * !as_00 * dram * !tclr * !wr * a1 * mat2 * as_22;
24
25 !cyc_00 = !as_21
26 + csp19
27 + !as_00 * dram * !tclr * !wr * mat0 * mat2
28 + !as_00 * dram * !tclr * !wr * a1 * mat2
29 + prom * !wr;
30
31 dsack0.OE = as_21 * !cyc_00 * !csp19;
32 dsack0 = as_20 * dram * !tclr * !wr * mat0 * mat2 * 16m
33 + as_20 * dram * !tclr * !wr * a1 * mat2 * 16m
34 + as_20 * dram * !tclr * !wr * mat0 * mat2 * dsack1
35 + as_20 * dram * !tclr * !wr * a1 * mat2 * dsack1
36 + as_20 * prom * !wr * 16m
37 + as_20 * prom * !wr * dsack1;
38
39 dsack1.OE = as_21 * !cyc_00 * !csp19;
40 dsack1 = as_20 * dram * !tclr * !wr * mat0 * mat2 * 16m
41 + as_20 * dram * !tclr * !wr * a1 * mat2 * 16m
42 + as_20 * dram * !tclr * !wr * mat0 * mat2 * dsack1
43 + as_20 * dram * !tclr * !wr * a1 * mat2 * dsack1
44 + as_20 * prom * !wr * 16m
45 + as_20 * prom * !wr * dsack1;
46
47 roe = as_20 * prom * !wr * !OE;
48
49 %END
50
```

Kasten 5

```

0 PAK 68/3, GAL U2:  Cache Control Logic, L2-Cache über Jumper schaltbar
1 08-09-93   V2_JPa      Holger Zimmermann @ PE
2 %ID
3      P2_JP
4 %TYP
5      GAL20V8A
6 %PINS
7 !rom mat2 a0 a1 !as_00 !bgack_20 !wr !as_20      !dsack !reset siz1
8 siz0 !dram !twe2 !dwe3 !dwe1 !c2off !dwe2 !tclr
   !dwe0 !twe0 mat0
9 %LOGIC
10
11 dwe0.OE = VCC;
12 dwe0 = dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * !a0
13       + dram * !tclr * as_20 * as_00 * !wr * !mat0 * !a1 * dsack
14       + dram * !tclr * as_20 * as_00 * !wr * dwe0 * !a1 * dsack;
15
16 dwe1 = dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * siz1
17       + dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * a0
18       + dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * !siz0
19       + dram * !tclr * as_20 * as_00 * !wr * !mat0 * !a1 * dsack
20       + dram * !tclr * as_20 * as_00 * !wr * dwe1 * !a1 * dsack;
21
22 dwe2 = dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * !a0
23       + dram * !tclr * as_20 * as_00 * !wr * !mat2 * a1 * dsack
24       + dram * !tclr * as_20 * as_00 * !wr * dwe2 * a1 * dsack;
25
26 dwe3 = dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * siz1
27       + dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * a0
28       + dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * !siz0
29       + dram * !tclr * as_20 * as_00 * !wr * !mat2 * a1 * dsack
30       + dram * !tclr * as_20 * as_00 * !wr * dwe3 * a1 * dsack;
32 twe0 = dram * !tclr * as_20 * as_00 * !wr * dwe0 * !a1 * dsack
33       + dram * !tclr * as_20 * as_00 * !wr * dwe1 * !a1 * dsack;
34
35 twe2 = dram * !tclr * as_20 * as_00 * !wr * dwe2 * a1 * dsack
36       + dram * !tclr * as_20 * as_00 * !wr * dwe3 * a1 * dsack;
37
38 tclr = as_20 * tclr
39       + !as_20 * bgack_20
40       + !as_20 * reset
41       + !as_20 * c2off;
42
43 c2off.OE = GND;
44
45 %END
46

```

Kasten 6

```

0 PAK 68/3, GAL U6: Adressdekoder, Atari ST
1 16-09-93 V6_STc1 Holger Zimmermann @ PE
2 %ID
3 P6_ST
4 %TYP
5 GAL20V8A
6 %PINS
7 !vpa fc0 fc1 !as_20 a21 a20 a17 a18 a16 a19 a22
8 !berr_00 a23 !avec !dram !ciin !word !fpucs !rom !csp19
9 !berr_20 !jp3
10 %LOGIC
11
12 berr_20.OE = VCC;
13 berr_20 = as_20 * fc1 * fc0 * !a19 * !a17 'PMMU, BKPT
14 + as_20 * fc1 * fc0 * !a19 * a17 * !jp3 'FPU disabled
15 + as_20 * berr_00;
16
17 fpucs = fc1 * fc0 * !a19 * !a18 * a17 * !a16 * jp3; 'FPU enabled
19 avec = fc1 * fc0 * vpa;
20
21 csp19 = fc1 * fc0 * !a19;
22
23 dram = !fc0 * !a23 * !a22
24 + !fc1 * !a23 * !a22;
25
26 rom = a23 * a22 * a21 * a20 * a19 * a18 * !a17 'TOS 1.04
27 + a23 * a22 * a21 * a20 * a19 * a18 * !a16 'TOS 1.04
28 + a23 * a22 * a21 * !a20 * !a19; 'TOS x.06
29
30 word = a23 * a22 * a21 * a20 * a19 * a18 * !a17
31 + a23 * a22 * a21 * a20 * a19 * a18 * !a16
32 + a23 * a22 * a21 * !a20 * !a19
33 + !fc0 * !a23 * !a22
34
35
36 !ciin = a23 * a22 * a21 * a20 * a19 * a18 * !a17 'ROM
37 + a23 * a22 * a21 * a20 * a19 * a18 * !a16 'ROM
38 + a23 * a22 * a21 * !a20 * !a19 'ROM
39 + !fc0 * !a23 * !a22 'RAM
40 + !fc1 * !a23 * !a22; 'RAM
41 %END
42
Kasten 7

```

```

0 PAK 68/3, GAL U2:  Cache Control Logic, L2-Cache nach Reset aus bis 1.      DMA
1 18-09-93   V2_RSTb   Holger Zimmermann @ PE
2 %ID
3      P2_RST
4 %TYP
5      GAL20V8A
6 %PINS
7 !ciout mat2  a0  a1  !as_00  !bgack_20  !wr  !as_20      !dsack  !reset  siz1
8 siz0  !dram  !twe2  !dwe3  !dwe1  !c2off  !dwe2  !tclr      !dwe0  !twe0  mat0
9 %LOGIC
10
11 dwe0.OE = VCC;
12 dwe0  = dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * !a0
13      + dram * !tclr * as_20 * as_00 * !wr * !mat0 * !a1 * dsack *      !ciout
14      + dram * !tclr * as_20 * as_00 * !wr * dwe0 * !a1 * dsack *      !ciout;
15
16 dwe1  = dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * siz1
17      + dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * a0
18      + dram * !tclr * as_20 * as_00 * wr * mat0 * !a1 * !siz0
19      + dram * !tclr * as_20 * as_00 * !wr * !mat0 * !a1 * dsack *      !ciout
20      + dram * !tclr * as_20 * as_00 * !wr * dwe1 * !a1 * dsack *      !ciout;
21
22 dwe2  = dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * !a0
23      + dram * !tclr * as_20 * as_00 * !wr * !mat2 * a1 * dsack *      !ciout
24      + dram * !tclr * as_20 * as_00 * !wr * dwe2 * a1 * dsack *      !ciout;
25
26 dwe3  = dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * siz1
27      + dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * a0
28      + dram * !tclr * as_20 * as_00 * wr * mat2 * a1 * !siz0
29      + dram * !tclr * as_20 * as_00 * !wr * !mat2 * a1 * dsack *      !ciout
30      + dram * !tclr * as_20 * as_00 * !wr * dwe3 * a1 * dsack *      !ciout;
32 twe0  = dram * !tclr * as_20 * as_00 * !wr * dwe0 * !a1 * dsack *      !ciout
33      + dram * !tclr * as_20 * as_00 * !wr * dwe1 * !a1 * dsack *      !ciout;
34
35 twe2  = dram * !tclr * as_20 * as_00 * !wr * dwe2 * a1 * dsack *      !ciout
36      + dram * !tclr * as_20 * as_00 * !wr * dwe3 * a1 * dsack *      !ciout;
37
38 tclr  = as_20 * tclr
39      + !as_20 * bgack_20
40      + !as_20 * reset
41      + !as_20 * c2off;
42
43 c2off.OE = !bgack_20;
44 c2off   = reset      'SLC off nach Reset
45      + c2off * !bgack_20;      'notwendig für TOS 2.06 Speichertest
46
47 %END
Kasten 8

```

```

0 PAK 68/3, GAL U4: State Machine 1 für 68000 Businterface, 16 MHz
1 26-08-93 V4_16b Holger Zimmermann @ PE
3 P4_16
4 %TYP
5 GAL20V8A
6 %PINS
7 CLK clk8 !from !cyc_00 rd e2 e1 siz1 !as_20 !word a0
8 !OE siz0 !dsack !dsack1 q0 !vma !as_00 q1 !lds !uds !dtack
9 %LOGIC
10
11 dsack.OE = VCC;
12 dsack = as_20 * dsack1;
13
14 dsack1 <- as_00 * !dsack1 * q0 * !q1 * as_20 * rd * !clk8 * dtack
15 + as_00 * !dsack1 * q0 * !q1 * as_20 * rd * !clk8 * vma * !e1 * e2
16 + as_00 * !dsack1 * q0 * q1
17 + as_00 * dsack1 * !q1;
18
19 q0 <- !as_00 * !dsack1 * !q0 * !q1
20 + !as_00 * !dsack1 * q0 * !q1 * !as_20
21 + !as_00 * !dsack1 * q0 * !q1 * as_20 * !clk8
22 + !as_00 * !dsack1 * q0 * !q1 * as_20 * !cyc_00
23 + as_00 * !dsack1 * !q1 * as_20;
24
25 q1 <- as_00 * !dsack1 * q0 * !q1 * as_20 * !rd * !clk8 * dtack
26 + as_00 * !dsack1 * q0 * !q1 * as_20 * !rd * !clk8 * vma * !e1 * e2
27 + as_00 * dsack1 * q0 * !q1;
28
29 vma <- !clk8 * !e1 * !e2
30 + vma * !e2
31 + vma * !e1
32 + vma * clk8;
33
34 as_00 <- !as_00 * !dsack1 * q0 * !q1 * as_20 * clk8 * cyc_00
35 + as_00 * !dsack1 * !q0 * !q1 * as_20
36 + as_00 * q0 * !q1 * as_20
37 + as_00 * !dsack1 * q0 * as_20;
38
39 uds <- !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * word
40 + !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * !a0
41 + as_00 * !q1 * as_20 * rd * uds
42 + as_00 * !dsack1 * q0 * as_20 * !rd * !a0;
43
44 lds <- !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * word
45 + !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * a0
46 + !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * !siz0
47 + !as_00 * !dsack1 * q0 * !q1 * as_20 * rd * clk8 * cyc_00 * siz1
48 + as_00 * !q1 * as_20 * rd * lds
49 + as_00 * !dsack1 * q0 * as_20 * !rd * a0
50 + as_00 * !dsack1 * q0 * as_20 * !rd * !siz0
51 + as_00 * !dsack1 * q0 * as_20 * !rd * siz1;
52
53 %END
54
Kasten 9

```

```

0 PAK 68/3, GAL U5: State Machine 2 für 68000 Businterface, 16 MHz
1 21-08-93 V5_16a Holger Zimmermann @ PE
2 %ID
3 P5_16
4 %TYP
5 GAL20V8A
6 %PINS
7 CLK clk8 !cyc_00 !vpa !pak_en !bgack_00 !bg_20 !br_20 !as_20
8 fc0 fc1
9 !OE nc !bgx e0 !enab e1 e2 !bg_00 !bgack_20 e nc
10 %LOGIC
11
12 enab.OE = VCC;
13 enab = !bg_00 * !bgack_20 * pak_en * cyc_00
14 + !bg_00 * !bgack_20 * pak_en * enab * !clk8;
15
16 bgx <- clk8 * bg_20 * !as_20
17 + !clk8 * bgx;
18
19 bg_00 <- clk8 * bg_20 * bgx * br_20
20 + clk8 * bg_20 * bgx * bgack_00
21 + !clk8 * bg_00;
22
23 bgack_20 <- bgack_00
24 + !clk8 * bgack_20;
25
26 e <- e * !e1
27 + e * !e0
28 + e * clk8
29 + !e2 * e1 * e0 * !clk8
30 + !e2 * !e1 * !e0 * !clk8;
31
32 e0 <- e0 * clk8
33 + !e0 * !clk8;
34 + !fc1 * !a23 * !a22;
35 !e1 <- !e * e2 * !e1 * !e0
36 + e * !e1 * e0
37 + !e1 * clk8
38 + !e * e2 * e1 * e0 * !clk8
39 + e * !e2 * e0 * !clk8;
40
41 !e2 <- !e2 * e1 * !e0
42 + !e * !e2 * e1
43 + !e2 * clk8
44 + !e * !e1 * e0 * !clk8
45 + !e * e2 * !e1 * !clk8 * !fc0 * vpa
46 + !e * e2 * !e1 * !clk8 * !fc1 * vpa;
47
48 %END

```

Kasten 10

```

0 PAK 68/3, GAL U6:  Adressdekoeder, Mac SE und Mac Plus
1 01-09-93   V6_MCa   Holger Zimmermann @ PE
3   P6_MAC
4 %TYP
5   GAL20V8A
6 %PINS
7 !vpa fc0 fc1 !as_20 a21 a20 a17 a18 a16 a19 a22
8 !berr_00 a23 !avec !dram !ciin !word !fpucs !rom !csp19
9 !berr_20 !jp3
10 %LOGIC
11
12 berr_20.OE = VCC;
13 berr_20     = as_20 * fc1 * fc0 * !a19 * !a17           'PMMU, BKPT
14             + as_20 * fc1 * fc0 * !a19 * a17 * !jp3     'FPU disabled
15             + as_20 * berr_00;
16
17 fpucs       = fc1 * fc0 * !a19 * !a18 * a17 * !a16 * jp3;
18             'FPU enabled
19 avec        = fc1 * fc0 * vpa;
20
21 csp19       = fc1 * fc0 * !a19;
22
23 dram        = !fc0 * !a23 * !a22                        'RAM
24             + !fc1 * !a23 * !a22;
25
26 rom         = !a23 * a22 * !a21 * !a20 * !a19;         'ROM
27
28 word        = !a23 * a22 * !a21 * !a20                  'ROM
29             + !fc0 * !a23 * !a22                        'RAM
30             + !fc1 * !a23 * !a22;
31
32 !ciin       = !a23 * a22 * !a21 * !a20                  'ROM
33             + !fc0 * !a23 * !a22                        'RAM
34             + !fc1 * !a23 * !a22;
35
36 %END

```

Doppel-PAK

PAK-68/3 mit Cache und 68020/030 für 68000-Rechner, Teil 2

Holger Zimmermann

Nun wird es richtig spannend: nach der gut 800stelligen Lötorgie und den ersten Tests wartet die PAK auf Taktsignale im VHF-Bereich. Damit sich die CPU im Gestrüpp asynchroner Zugriffe und Wait-Zyklen nicht verheddert, ist neben dem 32-MHz-Quarz auch eine listige GAL-Logik vonnöten.

Im bislang besprochenen 16-MHz-Betrieb liefert CLK16 den Takt für CPU und GALs U1, U4 und U5, gewonnen durch Taktverdopplung aus dem 8-MHz-Systemtakt. Damit läuft die CPU synchron zum Systemtakt. Auch mit 12 MHz Systemtakt auf dem Atari-Mainboard [6] kommt die PAK/3 hervorragend zurecht, wobei die CPU dann, richtig geraten, mit 24 MHz läuft. Die einzig notwendige Änderung betrifft die Taktverdopplerschaltung. Für R41 und R43 sollten in diesem Fall 470 Ohm, für C41 und C42 15 pF vorgesehen werden (irrtümlicherweise fehlen die Kondensatoren C41 und C42 in der Stückliste; im Schaltplan sind sie jedoch richtig `bewertet`).

Wollte man die CPU auf diese Art bei 8 MHz Systemtakt mit 32 MHz betreiben (also mit vervierfachtem Takt), dann wäre eine aufwendigere Takterzeugung, etwa mit Hilfe einer PLL, notwendig. Bei der PAK-68/3 wird im 32-MHz-Betrieb ein anderer Weg beschritten: Ein Quarzoszillator versorgt CPU, FPU und GAL U1 mit eigenen 32 MHz, alles läuft also asynchron zum Systemtakt. Die GALs U4 und U5, zuständig für das Interface zum Mainboard, laufen hingegen weiterhin mit CLK16 und somit synchron zum Mainboard.

Im Gleichschritt ...

Alle lokalen Zugriffe auf der PAK-68/3 (Cache Hit, ROM und FPU) laufen immer synchron zu CPUCLK ab. Die Kommunikation zwischen CPU und FPU geschieht genauso wie bei 16 MHz, nur eben doppelt so schnell. Bei ROM-Zugriffen wird die Geschwindigkeit durch die Zugriffszeit der EPROMs begrenzt. Während 120er EPROMs bei 16 MHz für 0 Waitstates ausreichen, wären bei 32 MHz Zugriffszeiten kleiner 60 ns notwendig. Derartige EPROMs sind nicht nur kaum zu beschaffen, sondern auch unangenehm teuer. Auf der PAK-68/3 sorgt daher U1 mit den Signalen W1 und W2 dafür, daß bei ROM-Zugriffen vier Waitstates eingeschoben werden, wodurch EPROMs mit bis zu 150 ns (dann aber mit /CS auf GND) eingesetzt werden können.

Auch bei einem Cache Hit kommt man leider nicht um einen Waitstate herum, da zunächst das Tag-RAM befragt werden muß, ob die gewünschten Daten überhaupt im Cache vorliegen. Schnellere Tag-RAMs mit Zugriffszeiten unter 20 ns sind leider, bedingt durch ihre geringe Verbreitung, prohibitiv teuer. In PCs kommen nämlich für diese Aufgabe fast ausschließlich normale SRAMs zum Einsatz, die Komparatorlogik ist dort im LSI-Chipsatz mit integriert. Dagegen sind die auf der PAK/3 für den Cache verwendeten SRAMs die gleichen wie in der PC-Welt. Somit sind wenigstens beim SRAM Zugriffszeit, Beschaffbarkeit und Preis kein Problem.

... Marsch!

Solange keine Zugriffe auf das Mainboard stattfinden, bleibt der Ausgang CYC_00 von U1 inaktiv, und die State-Machine in U4 tritt sozusagen auf der Stelle, wobei die 68000-Bussignale ebenfalls inaktiv sind. Bei Schreibzugriffen, I/O-Zugriffen und Cache Miss aktiviert U1 CYC_00, worauf U4 einen Buszyklus startet. Mit dem weiteren Ablauf dieses Buszyklus hat U1 dann nichts mehr zu tun, die Ausgänge DSACK0 und DSACK1 von U1 sind abgeschaltet, und alles wartet auf DSACK1 aus U4, dessen Ausgang mit CYC_00 eingeschaltet wurde. Inzwischen hagelt es Waitstates.

Soweit gibt es keine Unterschiede zwischen dem 16-MHz- und dem 32-MHz-Betrieb. Da jedoch bei 32 MHz die Prozessorsignale sowie CYC_00 als Eingangssignale für U4 völlig asynchron kommen, ist damit zu rechnen, daß gelegentlich die Setup- und Hold-Zeiten für U4 verletzt werden. Dann hängt es vom Zufall ab, ob die Eingangssignale als *low* oder *high* erkannt werden, wobei sich die einzelnen Register des GALs durchaus unterschiedlich entscheiden können.

Um zu verdeutlichen, was passieren kann, stelle man sich zwei Registerausgänge mit exakt denselben Gleichungen vor. Man erwartet zunächst, daß beide Ausgänge immer übereinstimmen. Wenn aber ein Eingangssignal gerade zum `richtigen` Zeitpunkt wechselt, könnte das eine Register durchaus noch den alten Wert erkennen, während das andere bereits den neuen Wert sieht und daraufhin seinen Zustand ändert. Eine Taktperiode lang würden also beide Ausgänge unterschiedliche Ergebnisse liefern.

Verbotene Zone

Damit daraus keine unangenehmen Situationen entstehen, dürfen sich niemals zwei Zustandsvariablen, die von asynchronen Eingangssignalen abhängig sind, gleichzeitig ändern. Bei der PAK-68/3 ist sichergestellt, daß sich nur jeweils eine Zustandsvariable ändert, und somit keine `Bombenstimmung' aufkommt.

Der Abschluß eines Zugriffs auf das Mainboard sieht bei 32 MHz ebenfalls etwas anders aus. Zunächst wartet U4 einen CLK16-Takt länger als bei 16 MHz, bis DSACK1 aktiviert wird. Mit der nächsten fallenden Flanke des 32-MHz-Taktes erkennt die CPU, daß jetzt endlich Schluß ist mit der Wartezeit. Da beide Takte asynchron laufen, erfolgt die Datenübernahme beim Lesen nicht exakt mit der steigenden Flanke von AS_00, sondern mit einem Jitter von 30 ns. Die PAK/2 hat bewiesen, daß dies kein Problem darstellt, denn dort fand die Datenübernahme durch die CPU stets 30 ns vor AS_00 statt.

Prinzipiell ist man bei CPUCLK nicht auf 32 MHz festgelegt. Bei niedrigeren Frequenzen würde jedoch der Jitter zu groß (stets eine CPUCLK-Periode, bei 25 MHz also bereits 40 ns), was gutgehen kann, aber nicht muß. Zu höheren Taktfrequenzen hin gäbe es mit dem Jitter keine Probleme, wohl aber mit den Datenbustreibern auf dem Mainboard, die nach getaner Arbeit nicht schnell genug abgeschaltet werden und dadurch den folgenden Buszyklus stören würden. FPU-Zugriffe sind, da sie ohne Waitstates auskommen, am ehesten gefährdet.

Hilfskraft

Wie im ersten Teil des Beitrags bereits angesprochen, sind auf der Platine eine Leerfassung und ein 68000-Sockel vorgesehen, mit denen Sie recht einfach eine Umschaltmöglichkeit zwischen beiden Prozessoren (natürlich nicht im laufenden Betrieb) realisieren können.

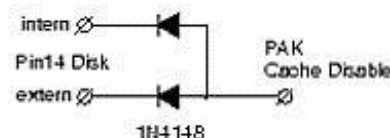
An dem 64poligen Sockel U7 auf der PAK-68/3 sind die Pins 11 (/BG), 13 (/BR) und 20 (E) nicht angeschlossen. Wer hier Erweiterungen aufstecken will, die diese Signale benötigen (eventuell Grafikerweiterungen oder Emulatoren im Atari), muß sie per Drahtbrücke nachreichen. Falls ein 68000 als Notantrieb eingesetzt werden soll, müssen diese sechs Leitungen (je drei von der PAK und vom 68000er) zu einem Umschalt-GAL (Listing PUK) geführt werden. Zusätzlich wird am GAL noch /BR_20 (von J6) und /PAK_EN (von J5) benötigt. Der Jumper J6 entfällt damit, die Umschaltung zwischen den Prozessoren erfolgt mit J5.

Atarisch

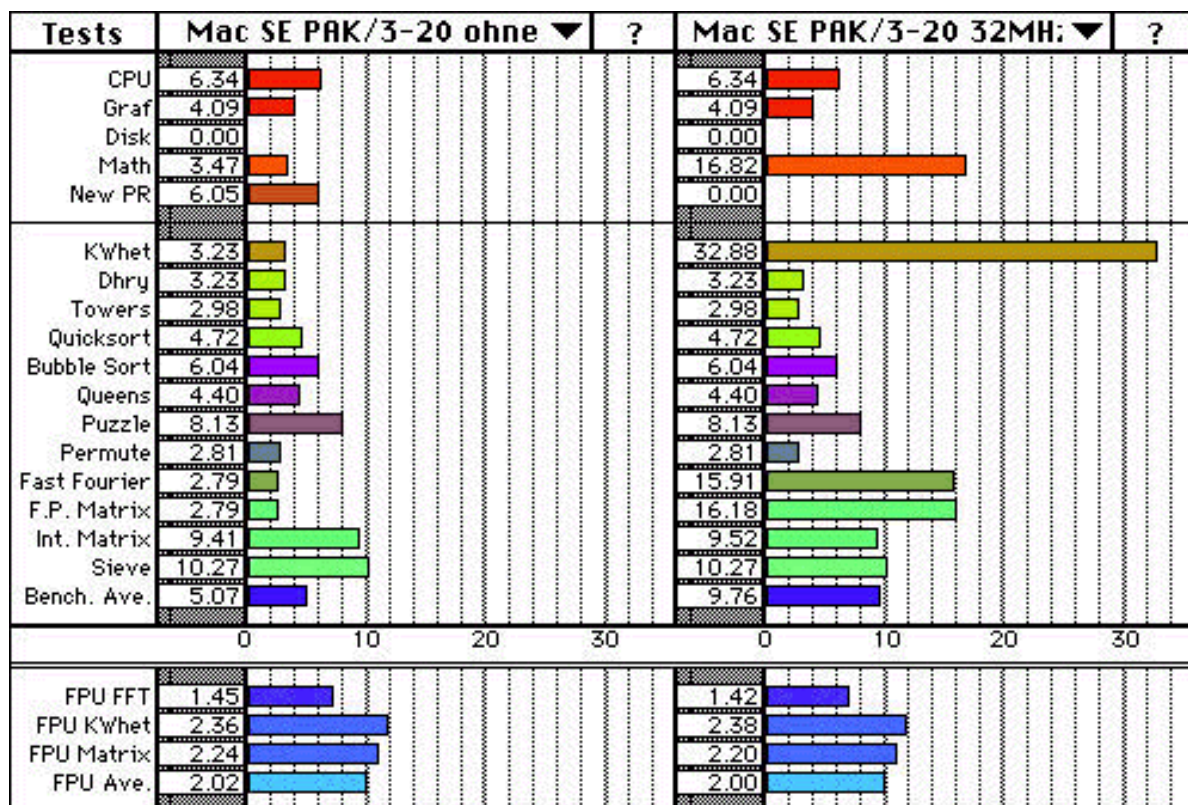
Die PAK/3-020 läuft unter TOS 2.06 problemlos, was im Prinzip auch für die 030er PAK gilt. Da TOS 2.06 jedoch nichts von der Existenz einer MMU und eines CPU-internen Datencache weiß, kann es in bestimmten Situationen Probleme geben, so zum Beispiel bei einigen Plattentreibern mit eingeschaltetem Datencache. Im TT fällt das nicht weiter auf, weil dort die MMU vom TOS 3.06 entsprechend programmiert wird. Daß es aber auch anders geht, zeigt AHD1, der auch auf einer PAK/3-030 unter TOS 2.06 ohne MMU-Unterstützung einwandfrei läuft.

Des weiteren gibt es noch einige Programme, etwa Treiber für virtuelles RAM, die die MMU verwenden, diese aber in genau dem Zustand vorfinden möchten, wie sie vom TOS 3.06 im TT initialisiert wird. Bei TOS 2.06 kann diese MMU-Initialisierung durch ein kleines Programm im Auto-Ordner nachgeholt werden, womit dann auch solche Programme laufen.

Leider handelt man sich damit ein weiteres (kleines) Problem ein: Ein über die Tastatur ausgelöster Kaltstart führt zum Aufhängen des Rechners, denn TOS 2.06 löscht dabei den gesamten Speicher und mithin auch die MMU-Tabelle. Die CPU trifft zwar in der Startup-Sequenz des TOS auf einen RESET-Befehl, womit die gesamte Hardware des Rechners (auch der L2-Cache) zurückgesetzt wird. Der interne Status der CPU bleibt davon jedoch unberührt, die MMU und die beiden CPU-Cache sind nach wie vor aktiv, was dann auch prompt in die Hose geht.



Ein ODER-verknüpftes Enable-Signal beider Laufwerke schaltet den Cache ab, wenn der Mac Schwierigkeiten mit Disketten haben sollte.



Mit und ohne: Die Ergebnisse (relativ zum Mac Classic) in der rechten Spalte wurden mit dem SANE-Beschleuniger GEMStart ermittelt; der Zuwachs an FPU-Performance im Benchmark-Mix ist deutlich. Die FPU-Benchmarks ermittelt Speedometer (Version 3.23) dagegen immer an SANE vorbei. Mit einer 68030-PAK liegen die Ergebnisse im Schnitt 10 Prozent höher.

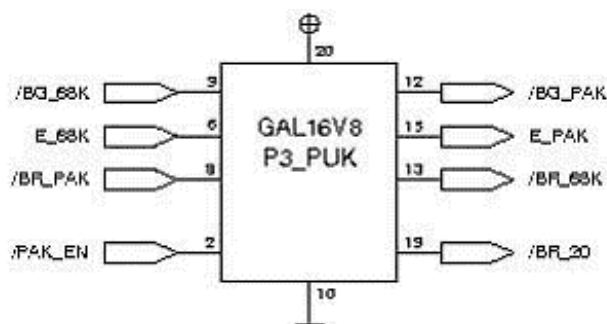
Die elegantere Alternative ist die Verwendung von TOS 3.06, das ja bereits für einen 68030 ausgelegt ist. Da zudem im Gegensatz zum TOS 2.06 an einigen Stellen 68020-Code verwendet wird, gibt es einen spürbaren Geschwindigkeitsgewinn.

Das originale TOS 3.06 des TT kann aufgrund unterschiedlicher Hardwarevoraussetzungen auf der PAK-68/3 leider nicht verwendet werden. Dies gilt gleichermaßen für den in der c't 9/92, S. 212 vorgestellten Patch, dem damals kurzerhand die gesamte MMU-Initialisierung zum Opfer fiel. Dort wurde außerdem ein Unterprogramm, in dem die CPU einigen ihrer Kollegen gelegentlich kurze Verschnaufpausen gönnt (im TT mit dem TT-MFP realisiert), einfach übersprungen. Was bei der PAK/2 wegen der geringeren Geschwindigkeit noch funktioniert haben mag, geht bei der PAK-68/3 garantiert schief. Daher wird es einen neuen Patch für TOS 3.06 geben, den wir aber erst nach einer ausgiebigen Testphase freigeben wollen. Einziges ungelöstes Problem ist zur Zeit noch die Memory-Protection von MiNT, die leider bisher auf der PAK/3 nicht zum Laufen zu kriegen ist. Da aber auch andere 68030-Boards dasselbe Problem haben, ist der Fehler wohl eher bei MiNT zu suchen.

Inside Mac

Grundsätzlich ist der Einbau in alle Macs mit 68000-8 möglich, einschließlich des Classic 1. Das Dumme ist nur, daß letzterer (wie auch der Atari STE) einen 68000 in PLCC-Ausführung beschäftigt - und ohne entsprechenden Adapter ist da nichts zu machen. Verschiedene Firmen, etwa MW Elektronik aus Königswinter, bieten derartiges an.

Auch beim Mac ist es (wie schon bei der PAK-68/2) möglich, das Rechner-ROM auf vier EPROMs zu verteilen und auf die PAK zu stecken. Auf der Mac-Sammeldiskette 7 ist das Split-Utility aus [3] neben allen GAL-Sourcen und -JEDECs mit enthalten. Da der Mac Classic von Haus aus mit 512 statt 256 KByte ROM ausgerüstet ist (warum, weiß allein Apple), ist hier das Split-Programm entsprechend anzupassen (Variable ROMLen); außerdem werden 27C010-EPROMs benötigt.



Das GAL P3_PUK wird mit etwas 'Freiluftverdrahtung' im Reservesockel der PAK-Platine untergebracht.

Bei manchen Maskenversionen des IWM beziehungsweise SWIM (Diskettencontroller des Mac, 'Integrated Woz Machine') tritt folgendes Problem auf: Der Mac erkennt die eingelegte Diskette nicht, formatiert sie fehlerhaft oder zeigt einen nicht vorhandenen Schreibschutz an. Sollte Ihr Exemplar so reagieren, müssen Sie den Prozessor- und eventuell auch den L2-Cache während aller Diskettenzugriffe lahmlegen. Sehr einfach geht das mittels zweier Dioden, die gemäß obenstehender Abbildung zu verschalten sind. Da der Mac ohnehin etwas pingelig bezüglich des Diskettenformatierens ist, kann man Disketten auch mit dem 'Diskcopy 4.2' formatieren: Einfach ein Image einer Leerdiskette anlegen und neue Disks damit überbraten. Vor allem, wenn eine Diskette schon einmal MSDOS-formatiert war, wirkt dies Wunder.

Empfehlenswert ist der Einsatz eines SANE-Beschleunigers (ein INIT oder Kontrollfeld, das die SANE-Aufrufe auf den Koprozessor umleitet), damit alle 'rechnenden' Applikationen etwas vom 68881/882 (so vorhanden) haben - der bleibt sonst nämlich außen vor, wenn ihn das laufende Programm nicht höchstpersönlich anspricht. [7] beschreibt den Patch des Radius-MATH-INITs, das eigentlich für Beschleunigerkarten der amerikanischen Firma gedacht ist. Wer Zugang zu CompuServe (Mac Systems Forum) oder AppleLink (Software Updates) hat, sollte sich das Kontrollfeld GEMStart (Version 2.2 oder 3.0) der Firma Total Systems herunterladen; es harmoniert auch ohne Patch ganz wunderbar mit der PAK - alt wie neu.

Output

Wichtig ist, wenn man einem bekannten deutschen Politiker glauben darf, vor allem, was hinten heraus kommt - siehe untenstehende Benchmark-Tabelle. Zum einen fällt auf, daß QuickIndex bereits der PAK/2 eine recht hohe Rechenleistung bescheinigte. Dies liegt an den dort verwendeten kurzen Programmschleifen, die weitgehend im CPU-internen Cache Platz finden; so nimmt sich der Schritt zur PAK/3 in den verschiedenen Ausbaustufen nicht mehr so groß aus. In der letzten Spalte fällt auf, daß eine 24-MHz-PAK in einem 12-MHz-Atari fast ebenso schnell ist wie die 32-MHz-Version bei 8 MHz Bustakt, weil Datenzugriffe auf das Mainboard immer nur zu bestimmten Zeitpunkten stattfinden können; zwischendurch ist der Videocontroller dran. Eine schnellere CPU vertrödelt somit bei 8 MHz einen guten Teil der gewonnenen Zeit wieder mit dem Warten auf das nächste 'Zugriffsfenster'. Auch beim GEM-Test wirken sich die vielen Schreibzugriffe auf den Bildschirmspeicher, die ja nicht über den L2-Cache laufen, etwas bremsend aus.

Der Dhrystone-Test und die Berechnung der Juliamenge zeigen ein anderes Bild, das sich auch mit den Erfahrungen aus der täglichen Praxis deckt, zum Beispiel bei Anwendungen wie TeX. Unter dem Strich erreicht die PAK/3 mit 32 MHz etwa den Faktor 3,5 bis 4 gegenüber einem normalen ST mit 8 MHz und ist etwa doppelt so schnell wie übliche 68000er Beschleuniger mit 16 MHz.

Zum Betrieb im Amiga liegen uns derzeit noch keine Erfahrungen vor; wir rechnen aber in Bälde mit positiven Rückmeldungen einiger 'Beta-Tester'. (cm)

Literatur

[1] Holger Zimmermann, Doppel-PAK, PAK-68/3 mit Cache und 68020/030 für 68000-Rechner, [c't 11/93, S. 222](#)

[2] Jankowski/Reschke/Rabich, Atari Profibuch ST-STE-TT, 10. Auflage 1991, Sybex-Verlag

[3] Manfred Völkel, Carsten Meyer, Tausch mit Turbo, PAK-68/2 in Mac, Atari und Amiga, c't 11/91, S. 314

[4] Jürgen Knufinke, Manfred Völkel, Stufe sechs, TOS x.06 für die PAK-68/2, [c't 9/92, S. 212](#)

[5] MausNet, Gruppen ATARI.HARD, ATARI.SOFT, ATARI-EXP 29. 11. 91 - 15. 10. 93

[6] Und es geht doch, Atari mit mehr als 8 MHz, ST Computer, [c't 9/92, S. 118](#)

[7] Leo Drisis, Real-Renner, 68881-Einbindung für die PAK im Mac, c't 5/90, S. 190

Atari-Benchmarks

System	1040 ST	Speeder	PAK/2	PAK/3	PAK/3	PAK/3	PAK/3
	68000	68000	68020	68020	68020	68030	68020
	8 MHz	16 MHz	16 MHz	16 MHz	32 MHz	32 MHz	24 MHz
		16 KB		32 KB	32 KB	32 KB	32 KB
		Cache		Cache	Cache	Cache	Cache
							System
							12 MHz
CPU Mem	100	164	245	330	374	396	495
Reg	100	204	402	406	811	811	614
Div	100	203	504	507	1014	1014	762
Shift	100	207	1708	1737	3534	3534	2628
TOS-T>ext	290	452	359	638	753	717	778
String	1075	1804	2406	2807	3062	3062	4218
Scroll	132	143	147	222	251	252	331
GEM Dialog	323	519	466	720	872	872	2159
Speedometer	104	188	209	335	479	485	465
Dhrystone	1059	1839	1282	2649	3407	3424	3944
Juliamenge	5622	3098	4040	2045	1429	1357	1350
Text	857	1460	1798	2452	3158	3203	4001
Linien	301	470	523	775	1055	1033	1099
Rechteck	411	573	605	736	916	941	1063
Polygone	201	365	399	640	1002	1019	984
Kreise	372	664	706	1158	1820	1813	1863
Raster	291	449	557	716	869	875	1076
Attribut	484	767	708	1134	1478	1432	1116
Auskunft	361	569	519	824	1052	1010	893
ESC	262	399	513	634	691	689	940
BIOS	201	282	234	417	495	468	578
GEMDOS	631	993	1316	1564	1704	1694	2332
AES	334	542	476	758	919	925	3150

Testkonfiguration: 1040 ST, 2 MByte RAM, kein Blitter, TOS 2.06, NVDI 2.03, 520 ST mit 12 MHz, 4 MB RAM, KAOS 1.4.2, NVDI 2.03

Testprogramme: Quick Index 2.2, Speedometer V 1.0, Dhrystone und Juliamenge in Modula programmiert, NVDI GEM-Test V 1.02

Höhere Zahlen bedeuten höhere Performance, bis auf `Juliamenge`

Neben den drei neuen 32-MHz-GALs finden Sie hier auch den Umschalter P3_PUK.

```
1
2 PAK 68/3, GAL U1: CPU-Clk State Machine, 68030, 32 MHz
3
4 18-09-93 V13_32c Holger Zimmermann @ PE
5 %ID
6 P13_32
7 %TYP
8 GAL20V8A
9 %PINS
10 CLK mat2 32m a1 !as_00 !tclr !wr !as_20 !prom !csp19 !ciout
11 !OE !dram !roe !as_21 !doe !w1 !w2 !cyc_00 !dsack1 !dsack0 mat0
12 %LOGIC
13
14 w1 <- as_20 * as_21 * !w2;
15
16 w2 <- as_20 * as_21 * w1
17 + as_20 * as_21 * w2;
18
19 as_21 = as_20 * 32m
20 + as_20 * as_21;
21
22 doe = as_20 * !as_00 * dram * !ciout * !tclr * !wr * mat0 * mat2
23 * as_21 * !32m
24 + as_20 * !as_00 * dram * !ciout * !tclr * !wr * mat0 * mat2 * as_21;
25
26 !cyc_00 = !as_20
27 + !as_21
28 + csp19 'FPU, PMMU, BKPT
29 + !as_00 * dram * !ciout * !tclr * !wr * mat0 * mat2
30 'Cache hit
31 + prom * !wr; 'ROM auf der PAK
32
33 dsack0.OE = as_21 * !cyc_00 * !csp19;
34 dsack0 = as_20 * dram * !ciout * !tclr * !wr * mat0 * mat2
35 * as_21 * !32m
36 + as_20 * dram * !ciout * !tclr * !wr * mat0 * mat2 * dsack0
37 + as_20 * prom * !wr * w2 * !w1 * !32m
38 + as_20 * prom * !wr * dsack0;
39
40 dsack1.OE = as_21 * !cyc_00 * !csp19;
41 dsack1 = as_20 * dram * !ciout * !tclr * !wr * mat0 * mat2
42 * as_21 * !32m
43 + as_20 * dram * !ciout * !tclr * !wr * mat0 * mat2 * dsack0
44 + as_20 * prom * !wr * w2 * !w1 * !32m
45 + as_20 * prom * !wr * dsack0;
46
47 roe = as_20 * prom * !wr;
48
49 %END
50 *****
51
52 PAK 68/3, GAL 4: Statemachine 1 für 68000 Businterface, 32 MHz
53
54 16-10-93 V4_32c3 Holger Zimmermann @ PE
55
56 %ID
57 P4_32
58 %TYP
59 GAL20V8A
60 %PINS
61 CLK clk8 !from !cyc_00 rd e2 e1 siz1 !as_20 !word a0
62 !OE siz0 !q1 !dsack1 q0 !vma !as_00 !dsack !lds !uds !dtack
63 %LOGIC
64
65 dsack1.OE = cyc_00;
66 dsack1 = as_20 * cyc_00 * dsack
67 + as_20 * cyc_00 * dsack1;
68
69 dsack <- as_00 * !dsack * q1 * q0 * as_20 * rd
```

```

66      + as_00 * !dsack * q1 * !q0 * as_20 * !rd;
67
68      q0 <- !as_00 * !dsack * !q1 * !q0 * as_20 * rd * cyc_00
69      + !as_00 * !dsack * !q1 * q0 * as_20 * cyc_00 * !clk8
70      + as_00 * !dsack * !q1 * as_20;
71
72      q1 <- as_00 * !dsack * !q1 * q0 * as_20 * !clk8 * dtack
73      + as_00 * !dsack * !q1 * q0 * as_20 * !clk8 * vma * !e1 * e2
74      + as_00 * !dsack * q1 * q0 * as_20
75      + as_00 * dsack * q1 * !q0;
76
77      vma <- !clk8 * !e1 * !e2
78      + vma * !e2
79      + vma * !e1
80      + vma * clk8;
81
82      as_00 <- !as_00 * !dsack * !q1 * !q0 * as_20 * !rd * cyc_00 * clk8
83      + !as_00 * !dsack * !q1 * q0 * as_20 * cyc_00 * clk8
84      + as_00 * !dsack * !q1 * !q0 * as_20
85      + as_00 * !dsack * q0 * as_20;
86
87      uds <- !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00 * clk8 * word
88      + !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00 * clk8 * !a0
89      + as_00 * !dsack * !q1 * !q0 * as_20 * rd * word
90      + as_00 * !dsack * !q1 * !q0 * as_20 * rd * !a0
91      + as_00 * !dsack * q0 * as_20 * rd * word
92      + as_00 * !dsack * q0 * as_20 * !a0;
93
94      lds <- !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00
95      * clk8 * word `2
96      + !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00 * clk8 * a0
97      + !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00 * clk8 * !siz0
98      + !as_00 * !dsack * !q1 * q0 * as_20 * rd * cyc_00 * clk8 * siz1
99      + as_00 * !dsack * as_20 * rd * lds
100     + as_00 * !dsack * q0 * as_20 * !rd * a0
101     + as_00 * !dsack * q0 * as_20 * !rd * !siz0
102     + as_00 * !dsack * q0 * as_20 * !rd * siz1;
103 %END
104 *****
105 PAK 68/3, GAL U5: State Machine 2 für 68000 Businterface, 32 MHz
106
107 26-08-93 V5_32a Holger Zimmermann @ PE
108
109 %ID
110 P5_32
111 %TYP
112 GAL20V8A
113 %PINS
114 CLK clk8 !cyc_00 !vpa !pak_en !bgack_00 !bg_20 !br_20 !as_20
115 fc0 fc1
116 !OE nc !bgx e0 !enab e1 e2 !bg_00 !bgack_20 e nc
117 %LOGIC
118
119 enab.OE = VCC;
120 enab = !bg_00 * !bgack_20 * pak_en;
121
122 bgx <- clk8 * bg_20 * !as_20
123 + !clk8 * bgx;
124
125 bg_00 <- clk8 * bg_20 * bgx * br_20
126 + clk8 * bg_20 * bgx * bgack_00
127 + !clk8 * bg_00;
128
129 bgack_20 <- bgack_00
130 + !clk8 * bgack_20;
131
132 e <- e * !e1
133 + e * !e0
134 + e * clk8
135 + !e2 * e1 * e0 * !clk8

```

```

136         +          !e2 * !e1 * !e0 * !clk8;
137
138     e0      <-          e0 *   clk8
139         +          !e0 * !clk8;
140
141     !e1     <- !e *   e2 * !e1 * !e0
142         +   e          * !e1 *   e0
143         +          !e1          *   clk8
144         + !e *   e2 *   e1 *   e0 * !clk8
145         +   e * !e2          *   e0 * !clk8;
146
147     !e2     <-          !e2 *   e1 * !e0
148         + !e *   !e2 *   e1
149         +          !e2          *   clk8
150         + !e          * !e1 *   e0 * !clk8
151         + !e *   e2 * !e1          * !clk8 * !fc0 * vpa
152         + !e *   e2 * !e1          * !clk8 * !fc1 * vpa;
153
154 %END
155 *****
156
157 PAK 68/3, GAL PUK: Umschaltung zwischen 68000-8 und PAK/3
158
159 08-09-93   P3_PUK   Holger Zimmermann @ PE
160
161
162 %ID
163     P3_PUK
164 %TYP
165     GAL16V8A
166 %PINS
167     nc !pak_en nc nc nc e_68k nc !br_pak !bg_68k
168     nc !bg_pak !br_68k nc e_pak nc nc nc !br_20
169 %LOGIC
170
171     e_pak.OE = !pak_en;
172     e_pak    = e_68k;
173
174     bg_pak.OE = !pak_en;
175     bg_pak    = bg_68k;
176
177     br_68k    = !pak_en * br_pak
178         +   pak_en;
179
180     br_20     =   pak_en * br_pak
181         + !pak_en;
182
183 %END

```