<<<<< **GVP Combo boards “G-Force” `030 (Combo 325/340/350)** >>>>>

(Author's note: this was an article I originally written for the French magazine AmigaNews, and printed in March 1996 issue; given my command of English language, I did my *best* to translate it!)

[Uploader's note 1: I reformatted this file and made a very few minor modifications to it to make it easier to read in an 80-column window. Carl Mueller, 6/7/96]

[Uploader's note 2: I took the information that Carl uploaded, made minor editing changes, and created a PDF file. I tried to create a page size that would print properly on either A4 or Letter size paper. Sanford Hersh, 9-Jan-2000]

Among the many accelerator boards available for our good old A2000s, GVP boards have always ranked among the best, thanks to their performance: high speed 32-bits DMA memory (Direct Memory Access for highest possible data exchange rates, whose design "relieves" the CPU's job), IDE or SCSI hard-disk controller (DMA too), extra parallel port etc.

Most of the competitors' boards lack many of these functions; a missing DMA, for example, forces the CPU to spend all its time exchanging data by itself, making multitasking drastically slow down during hard-disk transfers for example.

But no one's perfect: while G-Force '030 boards studied hereafter (also called "Combo" for they mix CPU, FPU, 32-bits memory and SCSI hard-disk controller altogether on a same board) are this powerful, they require a specific kind of memory that costs nearly as much as the board itself.

In the following article, I will do my best to show you clearly how to make your GVP "love" a classic, hence much cheaper, memory for PeeCees.

***** So-called "32 bits" memory

Nearly each accelerator board has its own memory, expandable (up to a certain amount) or not, to which the board has a priority access, much faster than any other CHIP or FAST memory. This extra fast memory is called 32-bits because the CPUs run by such accelerator boards, 68020/030/040, all have a 32-bits wide data bus.

Another slowing factor, when accessing CHIP or FAST 16-bits memory, is that this r/w accesses must be done thru the Zorro-II bus, while accesses to the 32-bits memory of the board is straightforward with no 'bottleneck'.

As Ralph Babel explains in his Guru ROM manual, some boards, like the GVP Combo itself, configure part or all of their 32-bits memory within the Zorro-II address space (16MB) corresponding to the "basic" 68000, making this memory accessible thru DMA to Amiga's custom chips (Agnus etc.) and other Zorro-II boards. High-speed DMA transfers are driven by the on-board electronic logic, in either 16 or 32-bits wide accesses, transparently for both the CPU (32 bits) and the "external" devices thru the Zorro-II bus (16 bits).

As a conclusion, we can consider 32-bits memory as some sort of FAST memory compared to CHIP memory: in fact, "FAST" memory has a preferred, notably faster access to CPU 68000 than "CHIP" memory, which is shared by nature between 68000 and Amiga's custom chips, leading to slowdown. All FAST memory expansion board's owners will tell you.

We can now see that it is our interest to have as much 32-bits memory as possible on the accelerator board itself. An easy speed test would be GvpMemTest (supplied with all GVP boards): 32-bits memory is read and written at blasting speed compared to FAST or CHIP.

And the faster the memory accesses are, the faster all programs run! It's rabbit versus tortoise.

***** What about bytes versus 32 bits ??

Don't let expressions like "1MB of 32-bits RAM" mislead you. How come bytes (== 8 bits) and 32 bits can be mixed together ??

There *is* an historical reason to this: first microprocessors (CPU) used to work on 8 bits wide data, then people designed memory in blocks of 8 bits, called bytes. Today, most of the microprocessors work on 32 bits wide data, but the old way of calling memory size still remains.

We call a "word" the data size that each CPU is able to process at a time: in general, from 8 to 64 bits. Then each word "weights", from CPU's point of view, 1 or more bytes: for example, a CPU 68030 works on 32-bits words, hence 4 bytes at a time. Therefore a Combo with 1MB of 32-bits RAM means: 1MB of RAM (this is 1 mega-bytes, 1 mega x 8 bits) 32-bits wide, thus 1 mega x 8 / 32 = 256 kilo-words of 32 bits. "4MB of 32 bits RAM" means, the same way, "1 mega-words of 32 bits". Et cetera.

Be aware that, under Workbench, the Amiga only displays the total amount of memory available in bytes, might it be 32 bits, CHIP or FAST. Other softwares like Sysinfo, can tell one from another, and will display the exact type of memory.

***** Special 32-bits memory for GVP Combo

To take full advantage of an accelerator board, the main point would be to increase its 32-bits on-board memory. Some old memory boards, like C= A2058 (Zorro-II), made use of "discrete" chips, to add by blocks of 8 (x4) or 32 (x1) to always get these 32-bits wide data, without which it would never work.

Other boards make use of SIMMs (Single In-Line Memory Module) 30 pins: old models "short-sized" of 8 bits, by 4 (8x4=32 bits again and again); or 72 pins, such as the new "long-sized" SIMMs of 32 bits (36 bits with parity) found in any recent PeeCee.

Some other boards even use specially designed memory, like SIMMs Chipack 3240 of 4MB for GVP boards (not only for the Combo models). And here comes the well-known money-making technique "made in GVP": those Chipack 3240 cost actually big bucks. Ouch.

Moreover, they are special in several points:

- their access time is 60ns, faster than most of the memory available at computer stores (usually 70ns or more); this speed is required to achieve the fastest possible memory accesses, with as less wait states as possible (which slow down the CPU); good job indeed GVP!!

- their connector width is 64 pins, and their pinout is GVP-specific, making them totally incompatible with "common" SIMMs 72 pins;

- the organization of the memory chips on the Chipack module itself is GVP-specific.

This means that the proud owner of a GVP board is *forced* to purchase these Chipack 3240 to increase the board's memory, because no one but GVP sells them. And no specific product sells for cheap: in France, one of these 4MB modules might cost up to 1990F (around US $400). No comment! Paying this price these days is suicidal: 16MB memory costs less!! But since it is impossible to use non-GVP memory, there is no alternative.

Impossible ?? Well, I'm going to prove you the contrary!

***** Needful things

!!! WARNING: The experiment may take quite long but isn't too difficult to do. However, you must be familiar with the use of a soldering iron and know how to make precise solderings, since the SIMM module itself will need to be hacked. Money-wise, it's very worth it, but YOU are responsible for any possible damage!

For hardware, you will need:

- a SIMM module 4MB x32 (c.f. previously) 72 pins, speed 60ns as specified by GVP; such 60ns SIMMs are still not very common, be persistent!

- a 72-pins SIMM socket; rather than soldering wires directly on the SIMM, the socket will come handy;

- a low power soldering iron (30W) with a thin tip (1mm), and another one more powerful (40W or more) with a tip as thin as possible (2mm);

- some wire, such as the "grey" one used to make IDE, SCSI, floppy cables.

- some wire a bit thicker, of 2 different colours, to drain power signals VCC and VSS;

- bits of wrapping wire (extra thin);

- soldering, plyers, X-acto knife etc. and also a "beeper" (continuity tester; any multimeter, even the simplest possible, has this function).

TO AVOID LIKE THE PLAGUE:

* SIMMs of 70ns or above (I did *not* want to dare the devil and hack a SIMM slower than 60ns, to realize in the end that it didn't work!);

* old SIMM 30 pins, 8 or 9 bits wide (there are 256KB, 1MB and even 4MB); 4 would be needed to get 32 bits wide memory;

* SIMMs bigger or smaller than 4MB;

* big soldering and soldering irons like a blowtorch.

***** About SIMM 4MB 32 bits:

Take care to purchase a SIMM 32 bits with NO parity, since it is useless in our case; a SIMM 36 bits is seldom found and costs more, as it is maily used by old PeeCees, like IBM PS/1.

Also, prefer a SIMM 32 bits but on which there still are the EMPTY pads ready for the extra parity chips: we will take advantage of such pads and tracks, left open, later. Ask for a friend's help: we literally "see" that there is some empty place left on the module. It is very commonly found and makes memory makers' life easy, to manufacture 36 bits modules from 32 bits modules with the same PCB.

From now on, I will explain the full experiment in every little detail, which might be tedious to follow but MUST be followed, so that you can avoid making mistakes.

***** "Common" SIMM's ways of working

Any dynamic RAM (DRAM), found in every computer on earth, bears 4 command signals:

/RAS (Row Address Strobe)

/CAS (Column Address Strobe)

/WE (Write Enable)

/OE (Output Enable)

These signals are sent to the memory chip in a specific order (I won't explain any deeper), and then the data bus either writes the data into the chip, or reads the data from the chip. Addresses are multiplexed, by rows and columns, like a 10 bits matrix A0...9.

On a 32 bits SIMM 4MB, there are 32 data bits split into 4 blocks of 8 bits each (4x8=32) hence 4 blocks of 1MB. Each 1MB block is made of 2 memory chips of 1Mbitx4. This makes 2 chips x 4 bits each x 4 blocks for a total of 32 bits, this is it!! Let's call the 1st 8 bits data block (bits D0...7) "0", the 2nd (bits D8...15) "1", the 3rd (bits D16...23) "2" and the 4th (bits D24-31) "3".

There are 2 signals /RAS0 /RAS2 (1 per 2 blocks, hence per 4 chips) and 4 signals /CAS0 /CAS1 /CAS2 /CAS3 (1 per block, hence per 2 chips). Address signals A0...9, plus /WE, /OE and power supplies (VSS and VCC) are common to all chips. Signal /OE is always wired to the ground, so it doesn't show up on the connector. The 32 data bits D0...31 are wired by 8 (1MB block) from each chip to the connector.

***** Chipack 3240 GVP's way of working

As no drawing was given with my Combo board, I took time to track down all signals, thanks to a simple "beeper". This way, blocks "0" to "3" appear (each made of 2 chips 1Mbitx4) but it's like GVP had fun and put a big mess on the connector on purpose. Look:

* /RAS1357 to the 1st chip (odd #) of each block

* /RAS2468 to the 2nd chip (even #) of each block

* /CAS13 to each 1st chip of the first 2 blocks

* /CAS24 to each 2nd chip of the first 2 blocks

* /CAS57 to each 1st chip of the last 2 blocks

* /CAS68 to each 2nd chip of the last 2 blocks

* /WE12 to block "0", /WE34 to block "1", /WE56 to block "2", /WE78 to block "3"

* /OE exists and goes to each memory chip

The other signals A0...9, VCC et VSS, and data bits are wired like a "regular" SIMM.

Sidenote: There is a rumor that says these Chipack 3240 have a cheaper Macintosh equivalent: SIMMs for Mac II FX. After investigation, I found out that these Apple SIMMs had the same 64 pins connector indeed, but their size is 1MB 8 bits each only, instead of 4MB 32 bits for a GVP. Therefore they are useless.

***** GVP Combo 64 pins connector

Considering the big mess I just told you about, I was expecting for the worst. Thankfully, many signals got connected together:

* an only /RAS connects /RAS1357 and /RAS2468 together

    -> one /RAS for all

* /CAS1 connects /CAS13 and /CAS24

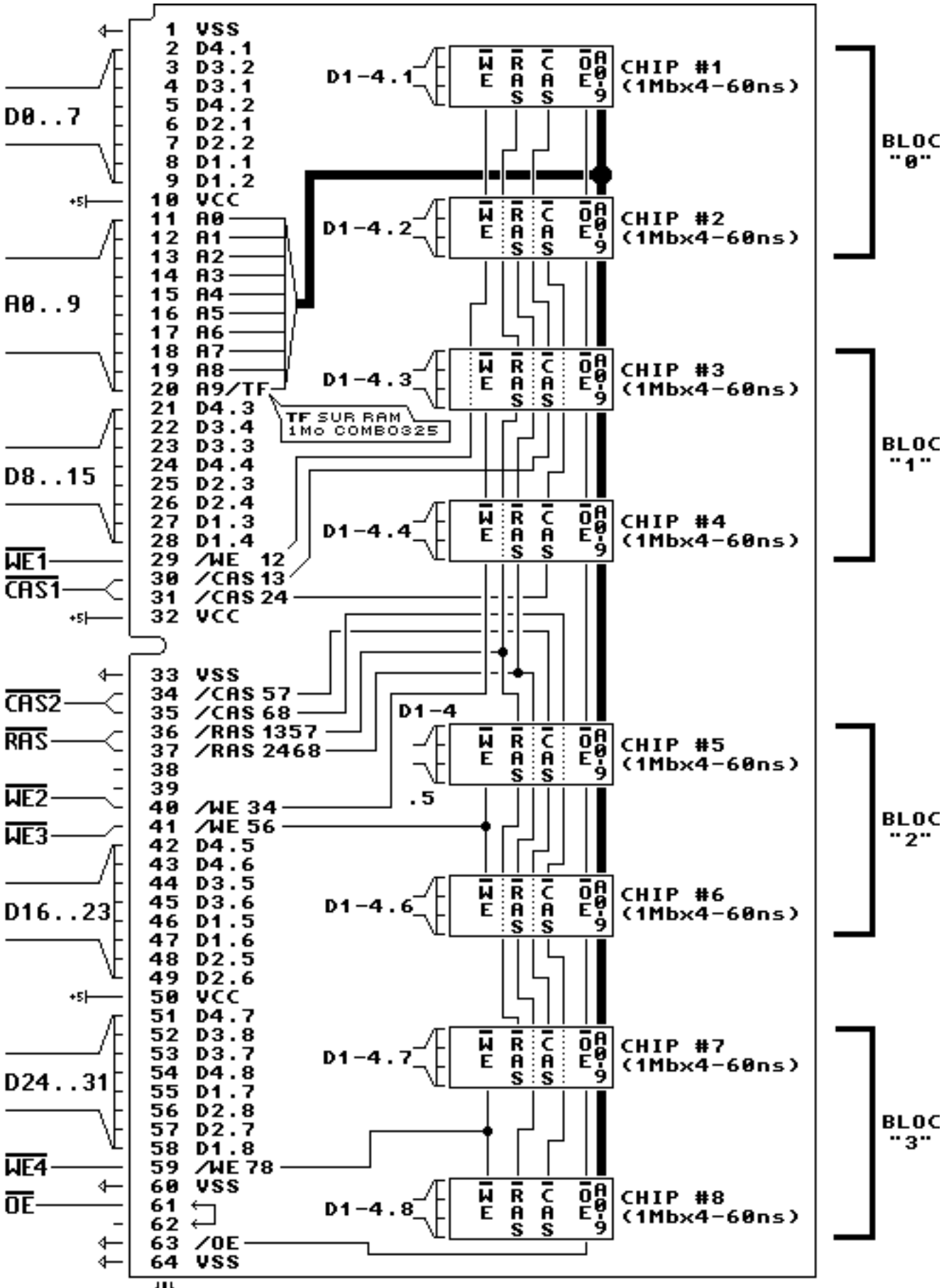
    -> one /CAS for the first 2 blocks

* /CAS2 connects /CAS57 and /CAS68

    -> one /CAS for the last 2 blocks

* /OE grounded

The connection Combo -> Chipack 3240 then looks like this:



GVP
Combo

GVP
Chipak 3240
4Mo

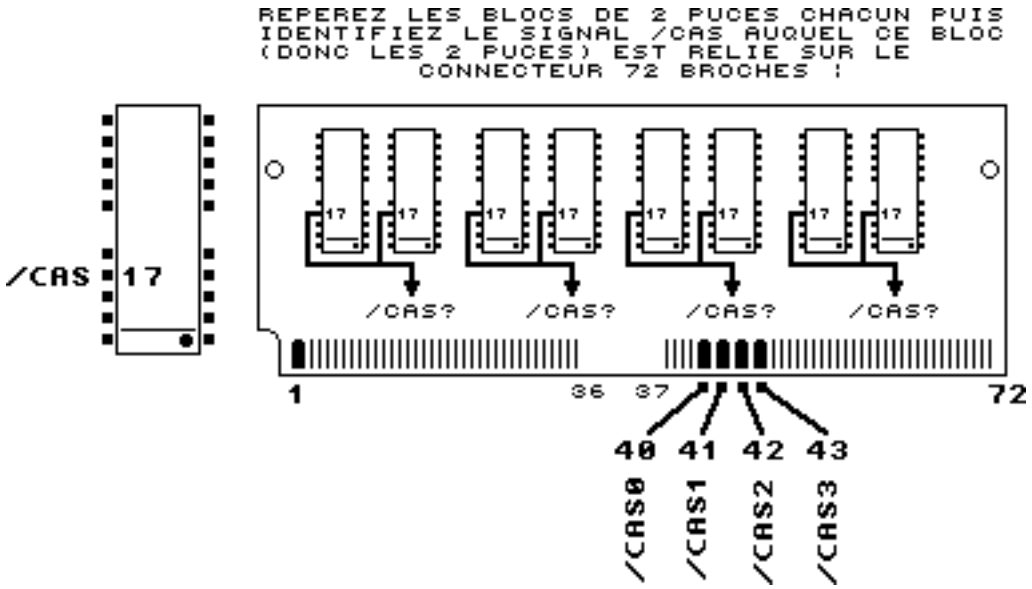
On the other hand, the 4 separate write signals /WE1, /WE2, /WE3, /WE4 still remain, although there is only one common /WE on a "regular" SIMM. This is why the SIMM 72 pins needs to be hacked.

***** 1st step: hacking the SIMM

This one takes the shortest time but is the most delicate to make. You will have to follow the checklist below closely:

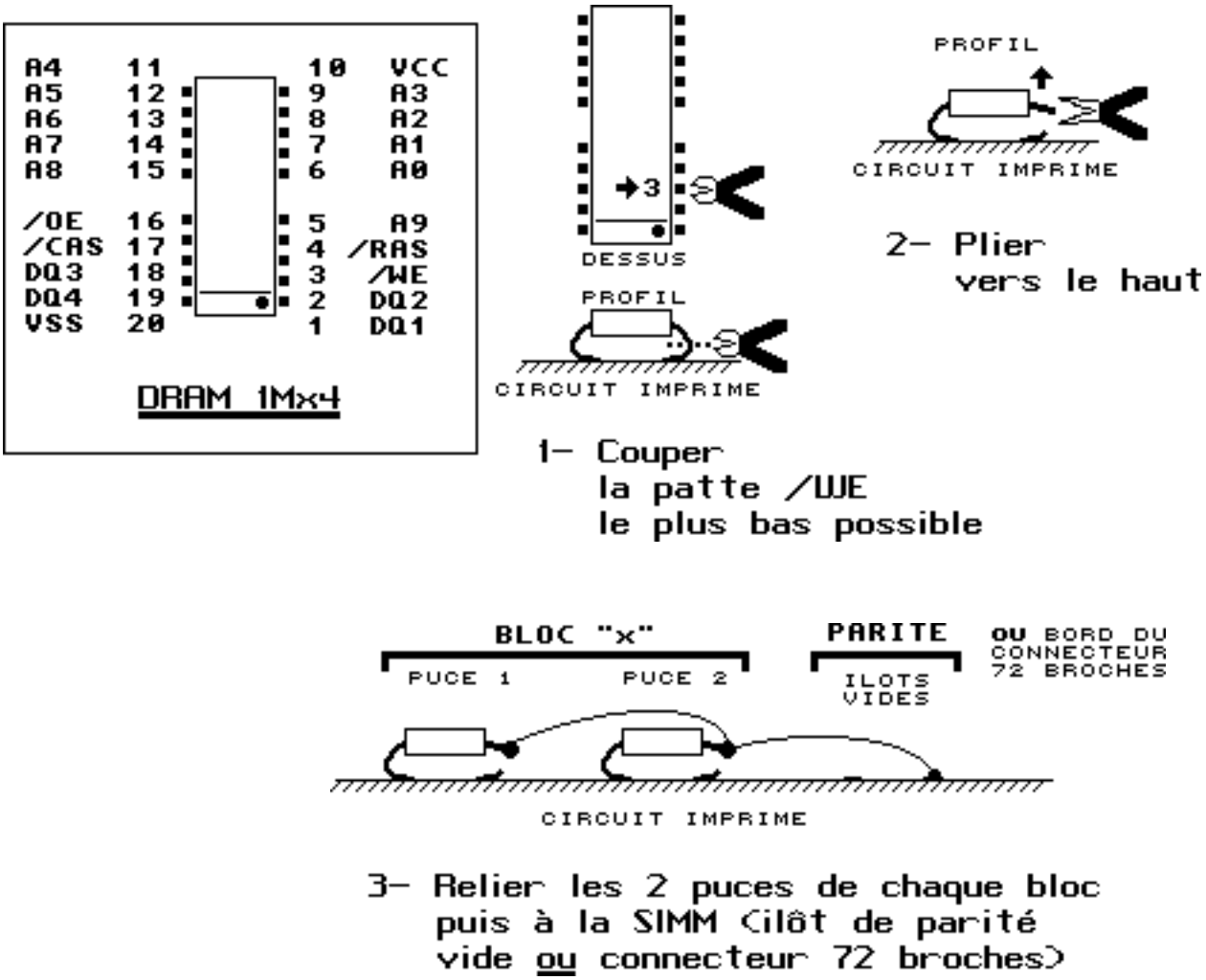
- Insert the SIMM into its socket, to protect it (we are going to SOLDER on it); be careful to insert it the right way, there is a little notch near pin 1 (see next drawing) to avoid inserting it the opposite/wrong way;

- Find the blocks "0" to "3" thanks to their respective signals /CASx (x=0...3) with a "beeper", following the 72 pins connector's drawing closely. /CAS0 will go to block "0"*and this one only*, /CAS1 will go to block "1" etc.



In general, chips are physically aligned on the PCB, therefore blocks should do the same, but it's not sure! On the SIMM 4MB I used, the chips were soldered on both sides of the PCB, by pairs, so I easily found out the blocks #, but any kind of connection is possible a priori.

- Now that you have "split" the memory module into blocks "0" to "3", next you need, on each chip, to CUT delicately, as close to the PCB as possible, each /WE leg, then lift it upwards to bend it horizontally (if the leg is hard to cut, use a small screwdriver and use it as a lever on each edge of the chip) like this:



!!! WARNING: the previous handling requires lots of precision and EXTRA THIN tools. One bad cut and it breaks the leg apart ==> the SIMM is destroyed and trashed!!!

- You should now have 8 /WEx legs "bent up". Mate them by pairs (c.f. schematic).

  Pin #47 of connector 72 pins, called /WE, is now left open, since all /WEx legs have been individually disconnected. Check them all with the beeper.

- If you chose (as I suggested) a SIMM without parity but with "empty" spaces, you can take advantage of these copper pads to solder the /WEx wires coming from each block. This solution is the easiest because these pads, used for the parity bits, are already wired to the connector. No need then to solder /WEx wires directly to the edge of the connector, there isn't much space left there and you risk to damage the SIMM socket!

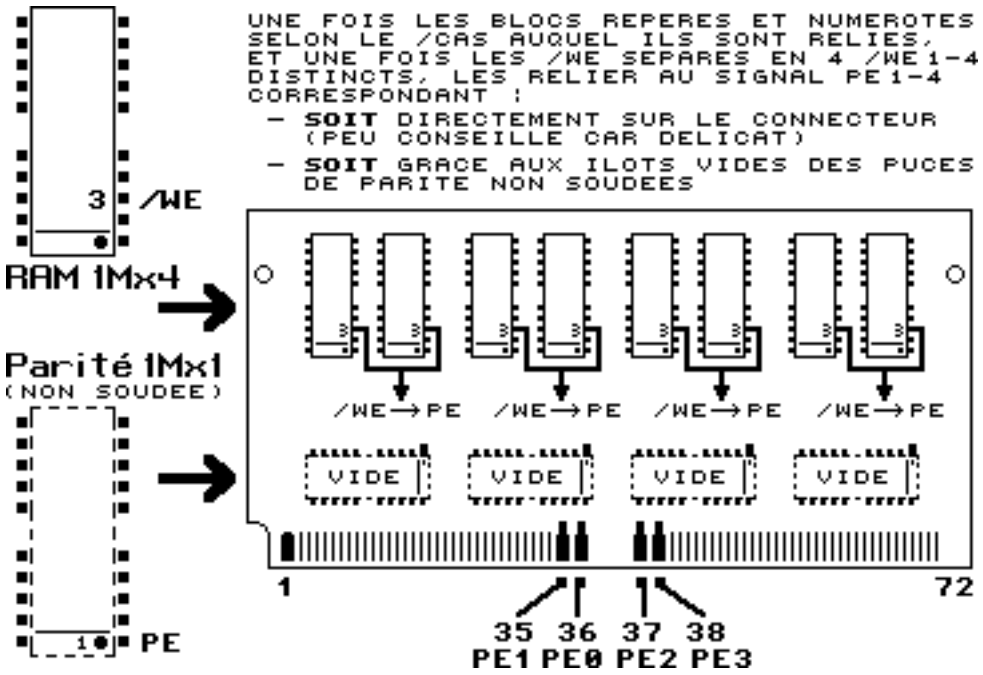
Following my advice, find (using the beeper) where the parity pins PE0 to PE3 go from the connector 72 pins: in principle, on 1 (or 2) copper pads of the "empty" chips (pin 1, or possibly 19, see drawing hereafter).

Use the thin wrapping wire to connect /WEx leg from each block (hence the /WE of each of the 2 chips) to the copper pad PEx (either an empty pad or on the connector 72 pins, it's up to you):

* block "0" : /WE 1st chip + /WE 2nd chip linked together to pad PE0

* block "1" : /WE 1st chip + /WE 2nd chip linked together to pad PE1

* same for blocks "2" et "3" with PE2 and PE3

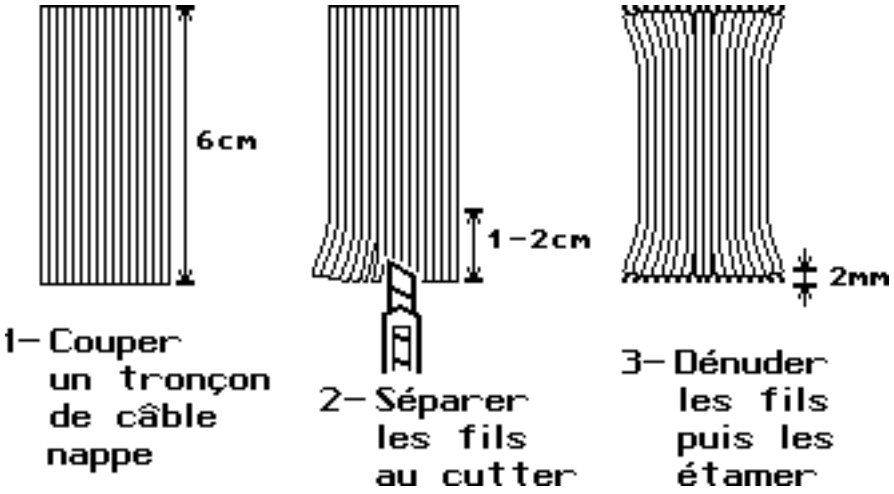


- It's all set! The SIMM module must now have 4 short wires connecting the /WE legs of each chip, pair by pair, to a copper pad, linked to (or, depending on what you chose to do, directly onto) the connector 72 pins. Go and have some champagne, you did a great job!!

***** Get wired

It's the longest and most tedious part of the job, but thankfully less tricky than it seems. To get everything reading for the wiring, cut the grey cable into several 6 cm (2") long sections. Separate the small wires from each other with an X-akto knife, into 4 chunks of 8 wires each, plus 1 chunk of 10 wires. OK OK, I know it's quite hard to describe, please have a look at the next drawing with the cable handy and you will understand.

Then, at each wire's end, separate it from its neighbors on a small length (2cm or 1" max) until you get some sort of "double sided comb", with wires still attached together by the center, like this:



Then, prepare, with what is left of the cable, some longer separate wires (10cm or 4"), for the several remaining command signals.

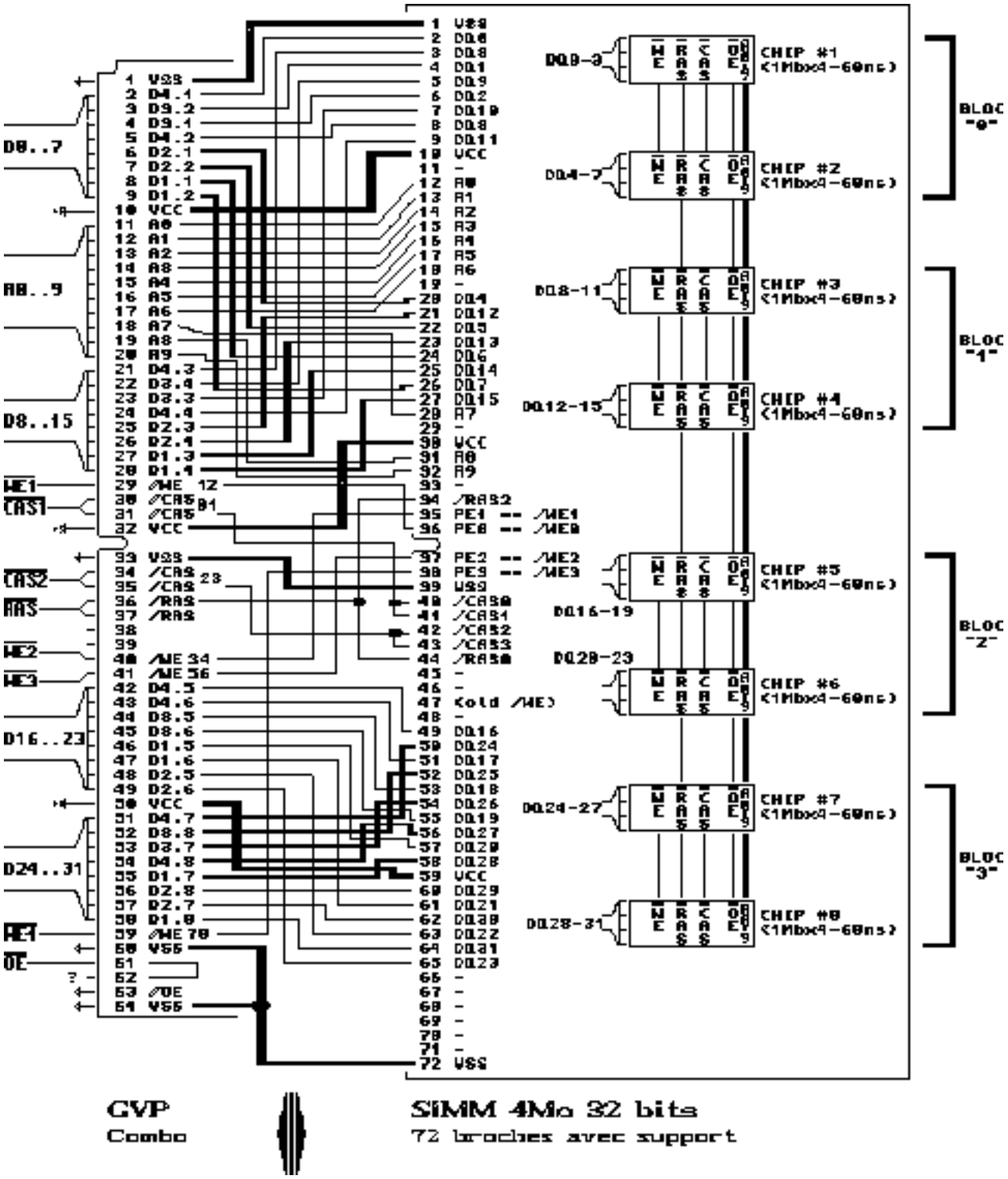
In the end, take the slightly thicker wire, and cut it into as many wires (7cm or 3") as there are VCC and VSS pins on the 72-pins connector. I advise you to use 2 different colours for VCC and VSS, so that there won't be any short circuit to blow up everything.

All you have to do is to strip each wire over a few millimeters at BOTH ends, then put some soldering (using the low power iron) to make it more rigid.

You're now ready to do the wiring!

***** Connecting the 72-pins SIMM to the Combo

A drawing is self-explanatory, so here are the required connections in every detail:



This looks rather. confused, eh? but if you follow each track from one end to the other, there is no risk to mislead one wire for another. I voluntarily drew some tracks thicker, otherwise they would have been "drowned" under the rest of the wires, but there's no need to use thicker wires for them. Note that the thickest wires are for the power supplies VCC and VSS.

To help you, refer to this cross table:

Combo	Conn. 64 pins GVP	Conn. 72 pins SIMM 4Mo
D0...7	Dx.1 (4) et Dx.2 (4)	DQ0...7
D8...15	Dx.3 (4) et Dx.4 (4)	DQ8...15
D16...23	Dx.5 (4) et Dx.6 (4)	DQ16...23
D24...31	Dx.7 (4) et Dx.8 (4)	DQ24...31
/WE1	/WE12	/WE0 (PE0)
/WE2	/WE34	/WE1 (PE1)
/WE3	/WE56	/WE2 (PE2)
/WE4	/WE78	/WE3 (PE3)
/CAS1	/CAS13 or /CAS24	/CAS0 + /CAS1 together
/CAS2	/CAS57 or /CAS68	/CAS2 + /CAS3 together
/RAS	/RAS1357 or /RAS2468	/RAS0 + /RAS2 together
/OE	/OE + pin 62 together	not connected
VSS	VSS	VSS
VCC	VCC	VCC

"Together" = solder the 2 pins TOGETHER!

/OE must return back to the Combo thru pin 62, thanks to a small wire that you will solder directly on the Combo's PCB.

"Or" = signal coming from Combo goes to the 2 pins, so that SIMM 72 can be connected to ANY of the two (it's up to you, on the schematic I chose for example one of the two /CAS but you freely use the other one).

***** Soldering

I advise you to solder, in this order:

* D0...7, D8...15, D16...23, D24...31

* A0...9

* /RAS /CAS1 /CAS2 /WE1...4 /OE, bridge on /OE

* VSS VCC

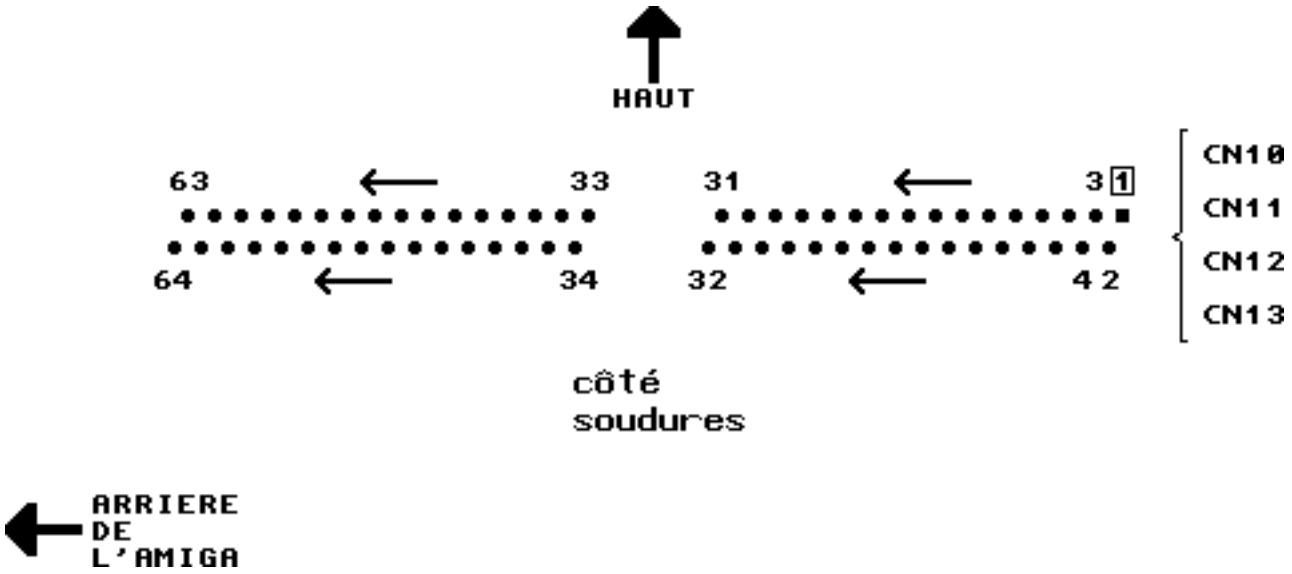
The Combo connector to which the SIMM 72 is to be soldered can be CN11, CN12 or CN13, but some basic rules are to be followed:

* if your Combo has some RAM originally soldered on its PCB (and this should be the case for the first 1MB or 4MB), don't use CN10 pads!

* solder on soldering side, not on components' side (SIMM sockets won't let you do that!) and pick up the NEXT FREE connector: if you have no Chipack installed, use CN11, otherwise use CN12 (Chipack in CN11) or even CN13 (2 Chipack in CN11 and CN12)

* pin 1 on the 64-pins connector has a square-shaped pad, while the other pins are round-shaped; starting from the pin 1, pads on the same side are the odd ones (1...63) and the opposite ones are the even ones (2...64).

The spacing in the center of the 64-pins connector "splits" it into 2 parts: pins 1 to 32 on the left (facing the Combo's components' side, Zorro connector on the bottom) and pins 33 to 64 on the right. Same for the SIMM 72-pins (pins 1...36 then 37...72). As expected, on soldering side, the order is reversed:



* ALL VCC and VSS pins must be connected to the Combo, to make sure that the SIMM gets "clean" power supplies (draining enough current, that a wire alone can't do) .

  To the contrary, on the Combo, these VCC and VSS pads are connected to the power planes, the innermost copper layers in the thickness of the PCB. Therefore, these layers will dissipate more heat and a more powerful iron is required to solder these wires only.

Check carefully your connections with the beeper, between the 72-pins SIMM connector and the 64-pins Combo connector, and even between this one and the memory chips on the SIMM if you have any doubt.

And of course... FORGET NO WIRE!!

***** Configuration of the Combo

The important jumper not to forget is J12. Now that you have more than the original 1MB or 4MB, you must avoid any possible conflict between the extra RAM you just added and the FAST RAM (if any) already installed in your Amiga on a separate Zorro expansion board (such as a GVP HCD8+). Your Combo's manual will tell how to set J12 accordingly.

Remember: if J12 is closed (installed), there can be a maximum of 8MB of FAST RAM (or 6MB if you own a BridgeBoard XT/AT) installed in the system at the same time (this is the A500/2000 Zorro II limitation), but there is no limit to the amount of "32-bits" RAM installed (on the Combo only, with J12 open).

***** Turning it on

Hold your breath...

If the normal dark grey / light grey / white screens show up, if your Amiga boots up, if the Workbench is back as always, you have almost won!

You must make sure, though, that the extra 4MB have duly been found and added to the system memory (by running "avail", "sysinfo" or else). If not, you must have forgotten one of the many wires, or the bridge on /OE, which seems to detect the SIMM (I didn't really check, but it exists on the Chipack 3240 so it MUST be of some use!).

If the dark grey screen shows up but soon after the "POWER" LED starts blinking and all you get is a yellow screen, this also happened to me: the SIMM is found but the command signals are not connected the right way. Did you connect all of the /RAS, /CAS and /WEx ? Is the SIMM itself well inserted in its socket ? None of the wire got broken when re-installing the Combo ?

In my case, I had left one of the /RAS open, then I had tried with a normal SIMM (not "hacked") but no luck, yellow screen was back! Once these 2 mistakes corrected, miracle, my Workbench was back on screen!

***** Footnote

I hope that reading the description above wasn't too boring. Personally, it mainly took me 1 hour to hack the SIMM then 2+ hours to do the wiring, LITTLE BY LITTLE. But then I duly appreciated the extra 4MB, trust me on this! I did this hack during December 1995, and it's still working flawlessly.

Once it works for you and you are sure that you did good solderings, do not hesitate to "drown" all the wires into glue (like Araldite) to avoid tearing the wires away. You can also tie up the SIMM to the Combo, using some thread (isolated of course, such as a piece of fishing nylon thread) the way you want. No need to say that putting a broken wire back in place isn't this easy (I had to do it).

There is a slight chance for me to design a dedicated PCB for it, to replace these many wires... one day (bye bye spare time these days). Of course, I will let you know, just email me some time to time!

Now a question for you: there IS some nasty trick I found and used, about the pin-out of the SIMM, that I didn't tell you to avoid confusing you any further! Can you see what it is ? The first one to tell me will win a free PCB, when it will be done, that is ;->

Look out for my next articles in French magazine *AmigaNews*! Soon to come: how to speed up Combos, add FPU, add Guru ROM, configure SCSI etc.

Pascal "P-chan" Janin.
EMAIL: p-chan@akane.swb.de