The FRANCES Memory Board

Adding 32-bit RAM to the LUCAS board

By Brad Fowles and Eric Haberfellner

Back in Volume 1, issue 3 *Transactor for the Amiga*, I wrote an article describing a 68020/68881 accelerator board of asynchronous design called LUCAS (Little Ugly Cheap Accelerator System). Now I'd like to introduce you to FRANCES (Fast Ram At Nominal Cost, for Extended Storage).

FRANCES is a 32-bit wide memory board that fits onto the expansion connector of the LUCAS board. You should read the original LUCAS article for a more complete understanding of the project as a whole.

The raw computing power of the LUCAS/FRANCES combination is pretty impressive. It will turn your Amiga 1000 into a full 32-bit platform running at clock speeds from 16 to 20 MHz (MegaHertz). Like the LUCAS board, I will make available a bare board, 2 PALs, hardcopy schematic and documentation disk for $75.00 US. (See the end of the article for more information.) If you've got the urge to hack or, like me, just can't afford a full blown A2500, this project will tweek a wee bit more life out of the A1000.

I should state my bias right up front. My real interest in the Amiga is as an experimental platform for 3D character animation. My needs, therefore, are for raw computing power and the project shows this bias.

Before we get into details, let me thank all of you who participated in an experiment in public domain hardware. To date I have sent out 550 LUCAS boards. Some of you had problems with the project, and a few had no luck at all; but judging by the mail I receive, most of you seemed to get the project up and running, and were able to improve the performance of the A1000. Now we are going to see if we can make the grand ol' lady fly.

Eric Haberfellner, who is co-authoring this article, not only wrote the *AFM* memory utility and all the diagnostic software to help me design the system, he stayed up nights helping to get it to work . I thank him for all his past work and all the answers he'll be giving to you folks when the time comes. My only complaint is that he and his brother slowed down the development of the first running prototype by booting up with the *FA/18* flight simulator whenever my back was turned. By the way, at 20 MHz the FA/18 demo runs in a little under 3 minutes, compared to 5 minutes, 47 seconds on a stock Amiga.

The thing that really kept the project going was the support you all gave to each other. I was constantly heartened by calls from local user group hardware gurus who were helping their friends get their systems up. LUCAS is by no means perfect. There were problems with some peripherals, bus noise problems and the now famous selection of U9. But with true hacker spirit, most of you found a way to sort out the problems, and hopefully learned a little along the way.

In the original article I said that the main benefit of having an '020 in your machine was that it provided a 32-bit wide upgrade path. However, until FRANCES entered the picture, the only 32-bit data transfers were those that were happening between the 68881 math coprocessor and the '020.

This was great for floating point math but really did little for general purpose computing. All memory transfers were still only 16 bits wide. What we really needed was a 32-bit wide memory system to support the '020.

Design criteria

The first decision to be made was: "How much memory do we need?" (All we can get, right?) I wanted to keep the board to a form factor that would be able to sit in the A1000 and still allow the case to be put on. I didn't want to use those wonderful little SIMMs (Single Inline Memory Modules) because they can be hard to find, and can be quite expensive in small quantities. I looked at the space available and decided that I could put in 4 Mb (megabytes) of standard 20-pin DIP (dual inline package) memory chips without too many problems.

Next: "How fast?" I priced out static memory fast enough to operate with no wait states – back in February, '89 the price would have been $1200.00 per megabyte.

So, OK, we'll forget about that, what about DRAMs (dynamic random access memory)? At $25.00 per chip, that's $200.00 per megabyte (at the time of writing, prices have dropped to $19.00 U.S. per chip). That seemed a little more reasonable.

One-megabit by 1-bit chips are slightly cheaper than 256-Kbit by 4-bit chips, but with the former you would need to buy 32 chips right off. If I used 256-Kbit by 4-bit chips you could start out with eight chips (for one megabyte) and build up in one megabyte increments. Yeah! Better to do it that way.

Keeping price in mind, how fast should the memory be? It turns out that 100-ns. (nanosecond) DRAMs will operate at 16 MHz with only one wait state and at 20 MHz with two. Even if we were to use 70-ns. DRAMs, the same number of wait states would be required. Keeping these trade-offs in mind, I decided that 256-Kbit by 4-bit, 100-ns DRAMs were the best choice; but check the price of 70-ns. parts – the difference may be negligible.

How does the change from 16-MHz to 20-MHz clock speed affect performance? The '020 has a three-cycle access. At 16-MHz, three times 62.5 ns., plus one wait state of 62.5 ns., equals 250 ns. total. At 20 MHz, it's three times 50 ns., plus two wait states of 50 ns., which equals, let's see, uh... 250 ns. Hmmmmmmm! At either 16 MHz or 20 MHz, the effective memory speed will be equal. Of course, at 20 MHz the transfers between the '881 and the '020 will be 1.25 times faster, as will all the internal register stuff in the '020 itself. It is possible that running at 18 MHz with one wait state, using 70-ns. DRAMs, would yield the best performance. In a simple test run at presstime, this does appear to be the case.

Now we need to select a DRAM controller. I remember, just a few short years ago, designing DRAM control circuitry with a plethora of discrete components and delay lines. These days there is a wide variety of highly integrated controllers from which to choose. I decided on a controller which could be soft-programmed to accommodate various clock speeds and DRAM configurations, and chips made by different manufacturers. I also wanted a controller which could support memory interleaving – more on that later. The best choice, I believe, is the National Semiconductor 8421A-20 or -25. So, with the controller and memory chosen, and a bit of glue and a few buffers added, we're almost there.

In order to squeeze all the performance we can out of the memory board, it would be desirable to get all those routines that are in the KickStart area of memory out of 16-bit wide space and into 32-bit space. Since we don't have an MMU to do the address translations for us, we're going to have to do it some other way. To describe how this is done, it's time to go a little deeper into the technical details. If you know nothing of hardware, hang in there and I'll try to give enough background so that it should make sense.

Practising 'Not Being Seen'

I assume that you have read the article on the LUCAS board. When the '020 is accessing the '881, it does so at full speed and the data path is 32-bits wide. If the circuitry of the Amiga sees this access, it will become quite confused and respond with a *DTACK (Data Transfer Acknowledge) long after the original cycle has gone by. To prevent this, we simply don't tell Amy that this is going on. This is accomplished by not asserting *AS (Address Strobe) or either of the two data strobes, *UDS or *LDS (Upper Data Strobe, Lower Data Strobe). This is done in PAL U7 on the LUCAS board by conditioning the term for these signals with the *CPCS signal (Coprocessor Chip Select) nonasserted state. In this way, the '881 practises Not Being Seen by the Amiga system underneath it.

The FRANCES memory board has the same type of restrictions: because it is much faster than the normal Amiga memory, we don't want Amy to try and respond to these faster accesses, so, same answer, we don't let her know about them.

In order to effect these clandestine accesses, we take the *CS (Chip Select) signal from the FRANCES decode PAL and run it back to U7. It just so happens that the U7 PAL on the LUCAS board receives two signals from the LUCAS expansion connector: *SRDSACK0 and *SRDSACK1 (Static RAM Data Size Acknowledge). I had put these lines in before I realized the costs of high speed static RAM. So I redefined them. The *SRDSACK0 line became the *FRANCYC (FRANCES Cycle) line and the *SRDSACK1 line became *FDSACK (FRANCES DSACK). Only one *DSACK line is needed because the FRANCES PAL takes care of the byte addressability. We can therefore assume that all memory accesses on the FRANCES board are 32-bits wide. We condition the terms for *AS, *UDS and *LDS with the *FRANCYC line in the same way we did for the coprocessor. Thus, the FRANCES memory practises Not Being Seen and there are no conflicts with the Amiga's decode circuitry.

I started this section by talking about remapping KickStart into 32-bit memory space. To do this, the first thing we must determine is when an access to KickStart memory is about to take place. The '020 initiates a memory cycle by putting the address on the bus and then asserting *AS to tell the system that the address is valid. At 16 MHz the address is valid 15 ns. before *AS is asserted. This is the amount of time we have to do our remapping. (For a little perspective, 15 ns. is the time it takes for light to travel approximately 15 feet.) KickStart resides at the addresses $FC0000 to $FFFFFF (it is also at $F80000 to $FBFFFF). So if we decode an address in which the top five bits are ones, we know that this will be a KickStart access.

If you look at the schematic you'll see that I used a 74F30, eight input NAND gate to decode the address. If the top five bits of the address are ones, the 74F30 will output a low true signal, which I call *REMAPK. If we split this signal into two and invert one of them, one will be low true when there is a KickStart access, and one will be low true when there is a non-KickStart access.

I added two address buffers between the '020 and the DRAM controller on the upper six bits of the address bus. If the '020 puts a KickStart address on the bus, one buffer will substitute the top six bits with the contents of the jumper block that selects the 32-bit KickStart copy address. If the '020 puts a non-KickStart address on the bus, the other buffer passes it through.

The final requirement is that we copy KickStart to the area defined by the jumper block and then tell the circuitry that it is okay to begin remapping. We do this by setting a flip-flop that enables the 74F30 to begin decoding. Again, there is no contention with the Amiga's circuitry because any access to the FRANCES memory will assert the *FRANCYC signal and it will practise Not Being Seen. Nifty, huh?!

Addressing scheme of FRANCES

While designing the board, I did a very non-Amiga thing. The FRANCES memory doesn't autoconfigure. Before you send somebody for a rope, listen to my reasons and the ways I've solved various problems... and then send somebody for the rope.

First: to keep the design as flexible as possible, I used a highly programmable DRAM controller. But autoconfiguring does not allow it to be programmed.

Second: we have this nifty KickStart remapping scheme, but games that take over the system won't be able to use it because they are not able to initialize the KickStart remapping circuitry.

We have found a way that these games can also take advantage of KickStart in 32-bit space. My good buddy Eric will describe how this is done in much greater detail a little later, which is only fair, since he came up with the solutions to all the problems of not autoconfiguring.

While we are waiting for the guy with the rope to get back, let me give you the bad news. There is no provision on the FRANCES board to allow DMA (Direct Memory Access) transfers. There are three reasons for this:

Cop-Out #1: It's a very knotty problem. It necessitates that the memory look like 32-bit wide memory to the '020, but like 16-bit wide memory to the expansion connector. I had a paper design that grew in complexity to the point where I realized that this beast was going to be very difficult to layout on the PCB (Printed Circuit Board).

Cop-Out #2: There are very few DMA devices available for the A1000. I couldn't see adding six more months delay to the project for a feature that very few people will use.

Cop-Out #3: My dog ate the original paper design... honest, teacher!

FRANCES can be run in or out of the normal Amiga 24-bit address space. Most people have 2 Mb of FAST ram, so if you're running in Amiga space, the board will reside between $400000 and $7FFFFFF. This has the added feature of keeping the decode circuitry as simple as possible.

If you want to run the board outside of Amiga space, perhaps because you have more than 2 Mb of FAST RAM, you can run it from $40400000 to $407FFFFF.

I also used two other high address spaces: $80800000 is the address that controls the flip-flop to enable the KickStart remapping, called ERKS (Enable Ram KickStart); and $80400000 is the address to strobe the programming pin of the DRAM controller. The DRAM controller doesn't have any data lines going to it so it must be programmed using the address bus. This is done by strobing a pin on the controller called *ML (ModeLoad) and placing the value to be put in the 23-bit programming register on the address bus.

The programming details of the DRAM controller are beyond the scope of this article. They are described both on the documentation disk that comes with the FRANCES board and in the 'spec' sheets that are available from National Semiconductor.

Interleaving

There is another way that we can squeeze a little more performance out of the memory system. The memory array is in four banks of one megabyte each. If you have a full four megabytes installed, it is possible to *interleave* the memory accesses to shorten the access time. Under normal circumstances, the top two address bits (in this case, lines A20 and A21) select which bank is active. If, however, we use the bottom two address lines, A2 and A3, then the long word sequential accesses that are the most common type of memory access will happen on separate banks. This allows the DRAM controller to reduce the *RAS (Row Address Strobe) precharge time on the second access. (A0 and A1, and SIZ0 and SIZ1, are not used for memory addressing directly; they are used by the PAL on the FRANCES board to allow byte addressability.)

Interleaving provides a relatively small performance improvement that is highly application specific; but it was easy to do and didn't cost anything. If you only have one, two or three megabytes installed then you cannot interleave the memory. There are jumpers on the FRANCES board to select the interleaving option.

The FRANCES board will take either a right angle or straight 96-pin DIN connector to mate with the LUCAS board. The right angle connector is preferable to the straight one, and is the one I intended. However, I forgot to specify this in the original LUCAS docs, so I made the FRANCES board compatible with both. However, the form factor of the board is slightly different for each of the two connectors. If you are ordering a FRANCES board, make sure you indicate which type of connector you used on the LUCAS board.

The FRANCES board is L-shaped and takes up the rest of the 'second level' inside the A1000. Installing the board requires that you first unplug the power connector that goes from the Amiga's power supply to the motherboard. Once the FRANCES board is in, this power connector plugs into a matching connector on the FRANCES board. Then you take another cable and plug one end into the second connector on the FRANCES board and the other end into the power connector on the motherboard.

AFM: the FRANCES memory configuration utility

Since the FRANCES memory does not autoconfigure, we were faced with the task of making it available to AmigaDOS.

The program usually used to add non-autoconfig RAM is the *Addmem* program written by Commodore-Amiga. There are, however, some problems with this program that make it a less than perfect utility for our purposes:

1. Addmem clears the memory that it adds to AmigaDOS. This makes it unsuitable for use with rebootable ram disks like ASDG's VD0: or AmigaDOS's RAD:, since it clobbers them.

2. Addmem is usually invoked by the s:startup-sequence script file that gets executed as AmigaDOS is booting. By the time it gets executed, some things have been loaded into CHIP RAM that could have been loaded into the FRANCES memory – the trackdisk.device for example. Since CHIP RAM is probably the most precious resource available on an Amiga 1000, and since programs may run slower there than in FAST RAM (depending on system DMA load), we decided that it would be desirable to come up with an alternative approach that would permit us to configure the FRANCES memory as soon as possible in the boot. This would insure that as much executable code as possible would get loaded into the FRANCES memory (and really fly), and that as much CHIP RAM as possible would remain available.

3. Addmem must be run every time that the system is booted or the memory will not be visible to AmigaDOS. This means that any memory made available to the system using the Addmem utility will not be used by applications that require rebooting the system, and that do not have easily alterable startup-sequence files. This includes many copy protected applications, games particularly.

4. Addmem has no mechanism for setting the priority of the memory that it adds. In the regular course of events, CHIP RAM is assigned a priority of -10, and autoconfigured FAST RAM is assigned a priority of zero. This insures that the FAST RAM will get allocated before CHIP RAM, unless CHIP RAM is explicitly indicated either in the hunks which the AmigaDOS loader reads, or in a program's memory allocation request.

Addmem assigns a priority of zero, the same as FAST RAM, to the memory it adds. This is reasonable in most cases: you can't Addmem CHIP RAM to the system and, because the data bus on the 68000 is only 16 bits wide, any memory added to the system is assumed to be 16-bit FAST RAM. The latter, of course, is *not* true of the LUCAS/FRANCES combination. The 32-bit wide FRANCES memory is clearly the fastest RAM in the system. We wanted a method to specify that the FRANCES memory have a higher priority than 16-bit FAST RAM.

5. Addmem also requires specifying the start and end addresses of the RAM being made available. This was awkward for our purposes since most people who build this memory board will probably not want to populate it with four megabytes of memory right away. It would be quite a nuisance for you to have to edit the startup-sequence files on all 42 of your boot disks everytime you wanted to add more RAM.

If you are running KickSsnetart out of FRANCES RAM, you would also have to correctly specify the range of RAM available as 256 Kb smaller than the full range of memory. This is not a big deal, but it would mean changing two places in the startup-sequence file instead of only one: the Addmem command, and the load KickStart into FRANCES RAM utility.

Since we were already going to have to write our own utility to configure the programable 8421A DRAM controller, copy KickStart to RAM and enable it, we decided to have it also add the memory and address the concerns listed above. Thus was born *AFM*, the FRANCES memory configuration utility. The name AFM stands for Add FRANCES Memory. This simple-to-use program is executed from your startup-sequence and deals with all of the situations listed above as best as possible for non-autoconfiguring memory.

The syntax for AFM is as follows:

```
AFM -mxxxxxx [-k] [-d] [-r] [-b]
```

Switches in square brackets are optional.

The **-m** is the ModeLoad switch and it is the only required switch. It is followed by a six digit hexadecimal number (24 bits) that specifies the programming of the DRAM controller. Don't panic – the FRANCES board documentation comes with the standard values for 16-, 18- and 20-MHz operation for all possible memory configurations. You will only have to pick the one that is right for your system. For brave souls who want to experiment with programming the DRAM controller, the description of what these bits do is provided in the FRANCES documentation.

The **-k** is the optional 32-bit KickStart switch. If it is specified, the top 256 Kb of FRANCES memory will not be made available to AmigaDOS. Instead, AFM will copy KickStart from the Amiga 1000 WCS (Writable Control Store) RAM to this area, and then enable the RAM KickStart feature of the FRANCES board by asserting ERKS. Once this is done, KickStart will run from 32-bit wide FAST RAM, which can make a whale of a difference for many applications.

The **-d** is the optional debug switch. It can be used to display information about AFM's activities if you are having any problems or if you are just interested. It will report where it found the the FRANCES memory and how much of it, and so on.

The **-r** is the optional Resident switch, the key to one of the most powerful features of the AFM program. If it is specified, AFM spawns a little program of less than 300 bytes that is

hooked into the AmigaDOS resident module list. It is then executed by AmigaDOS at warm boot time, even before autoconfiguration of peripherals occurs.

This resident module remembers what parameters AFM was run with, and will subsequently configure the FRANCES board the same way every time the Amiga is booted, until the Amiga is powered off. In other words, the FRANCES board will act as if it were an autoconfigure memory board until power is removed. Once the resident module is added, you can boot your favourite copy-protected version of Raster-Blaster, and if it can use fast RAM, it will use the 32-bit FRANCES RAM.

Whenever AFM is run, it checks to see if the resident module is already present. If it is, AFM knows that the memory has already been configured, and it exits without further ado. Someone once said about the RAMB0: RAM disk that it "hangs on like grim death" until the system is powered down. AFM is like RAMB0: in this respect. If you want to experiment with different programming values for the DRAM controller, don't use this switch.

The **-b** is the optional boot switch. Using this switch only makes sense in conjunction with the **-r** switch. If **-b** is specified, AFM will reboot the Amiga as soon as the resident module is spawned. The reason: the first time that you boot, the resident module will not be present, which will result in some stuff being loaded into CHIP RAM or 16-bit FAST RAM that should have gone into FRANCES RAM. These things will be loaded into FRANCES RAM the next time that you warm boot because the resident module will have already run. The **-b** switch will cause this second boot to happen immediately. This only takes a few seconds, and insures that you are running with as much stuff as possible loaded into FRANCES RAM right away. If you are using this switch, you really should have the AFM program as the first line of your startup-sequence file since anything you have done during the boot-up to this point will be lost.

The first time we saw this complete re-boot from software, we were quite startled; you have to experience it to understand.

The AFM program first programs the DRAM controller. It then checks location $40400000 and $00400000 for memory, in that order. If it does not find memory in either of those locations, it announces that no FRANCES memory was found and exits. Once the FRANCES memory has been found, AFM nondestructively checks the first byte of each megabyte boundary up to four megabytes to determine the amount of FRANCES memory available. If the RAM KickStart switch was selected, the size of the available memory is reduced accordingly, and KickStart is copied up to FRANCES RAM and enabled. AFM adds the available FRANCES memory to the AmigaDOS memory list with a priority of 10. If the resident module was requested, it is spawned and added to the resident list awaiting the next reboot. To the resident module is passed the location and extent of the FRANCES RAM, and whether KickStart is to be run from FRANCES memory. Finally, if the automatic reboot was requested, the Amiga is now rebooted; otherwise, AFM exits cleanly.

AFM has been written so that it will run whether or not there is an '020 or FRANCES memory in the system. If AFM finds that there is no '020, it exits with a return value of 5. This value can be checked in the startup-sequence script and conditional action taken. This is useful for those of you who have done Evan Sidorak's modification to the LUCAS board that allows you to switch between the 68020 and the 68000 (with a reboot, of course). The 32-bit FRANCES memory will not be available when using the 68000; by checking the return code of AFM, it is possible to write a startup-sequence script that will work for both 68000 and 68020 boots.

If AFM does find an '020 in the system, but cannot find the FRANCES memory, it exits with a return value of 10. This is taken to be the more serious failure because it indicates that the FRANCES memory may have failed.

We believe that this scheme gives the startup-sequence programmer sufficient flexibility to deal with any LUCAS/FRANCES situation.

Benchmarks

The following benchmarks give an indication of the raw computing performance of a stock A1000, an A2500 with A2626 card running KickStart in 32-bit space, and the LUCAS-FRANCES combination running at 16 and 20 MHz. These are the same bench marks that I used in the LUCAS article. The results will vary slightly each time you run them.

You should bear in mind that these benchmarks do not take into account hard disk transfer speed. To give you an idea of how this affects the system, Eric compiled his *Handshake* program once the LUCAS & FRANCES was installed. The hard disk light came on and just stayed on until the compile and link was complete. Guess where the bottleneck is now?

Where raw computing power really helps is with applications like ray-tracing. I find that, with the LUCAS-FRANCES installed, I am seeing my images rendered 2 1/3 times faster than with just the LUCAS board.

Benchmarks

	Whetstone	Savage	Calcpi	Float
Stock A1000	24	23842	4.87	286.1
A2500/2620 (K32)	247	444	23.75	58.1
LUCAS-FRANCES (16 MHz, K32)	277	402	26.18	54.8
LUCAS-FRANCES (20 MHz, K32)	295	352	29.46	48.0
UNITS	KWhets/sec	50*sec	Kflops/sec.	seconds, 256000 i

(K32 means KickStart running in 32-bit space)

Conclusion

There is much more information available on the disk that comes with the bare FRANCES board. If you have questions, Eric can be contacted on BIX as **ehaberfellner** or through USENET at **becker!haberfellner!eric**; and I can be reached on BIX as **anakin.1** or via USENET at **utgpu!anakin**.

If you're interested in acquiring either a bare LUCAS or FRANCES board and PALs, send a cheque or international money order to:

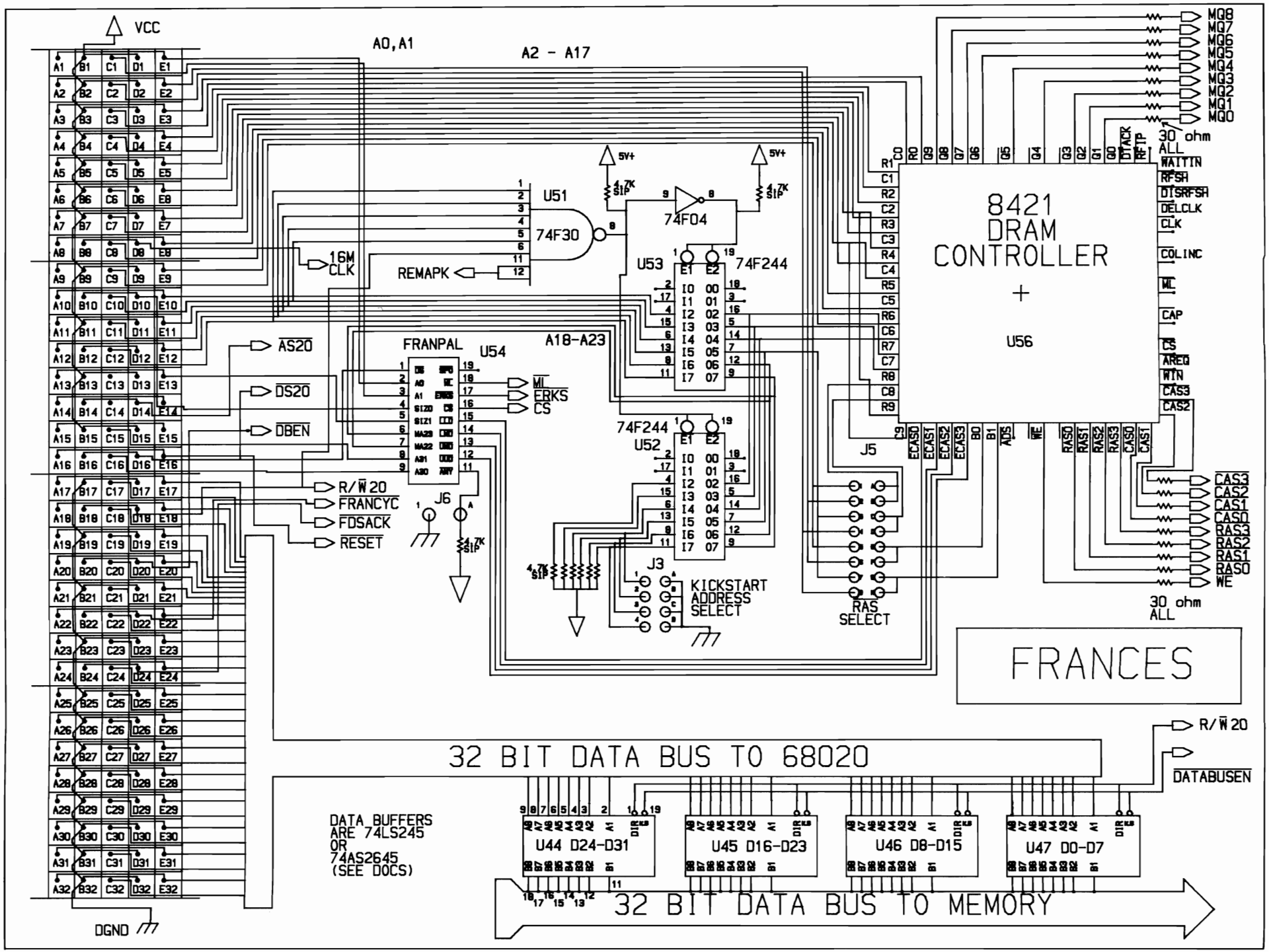
The LUCAS Project
c/o Brad Fowles
RR #5, Caledon East
Ontario, Canada.
L0N 1E0

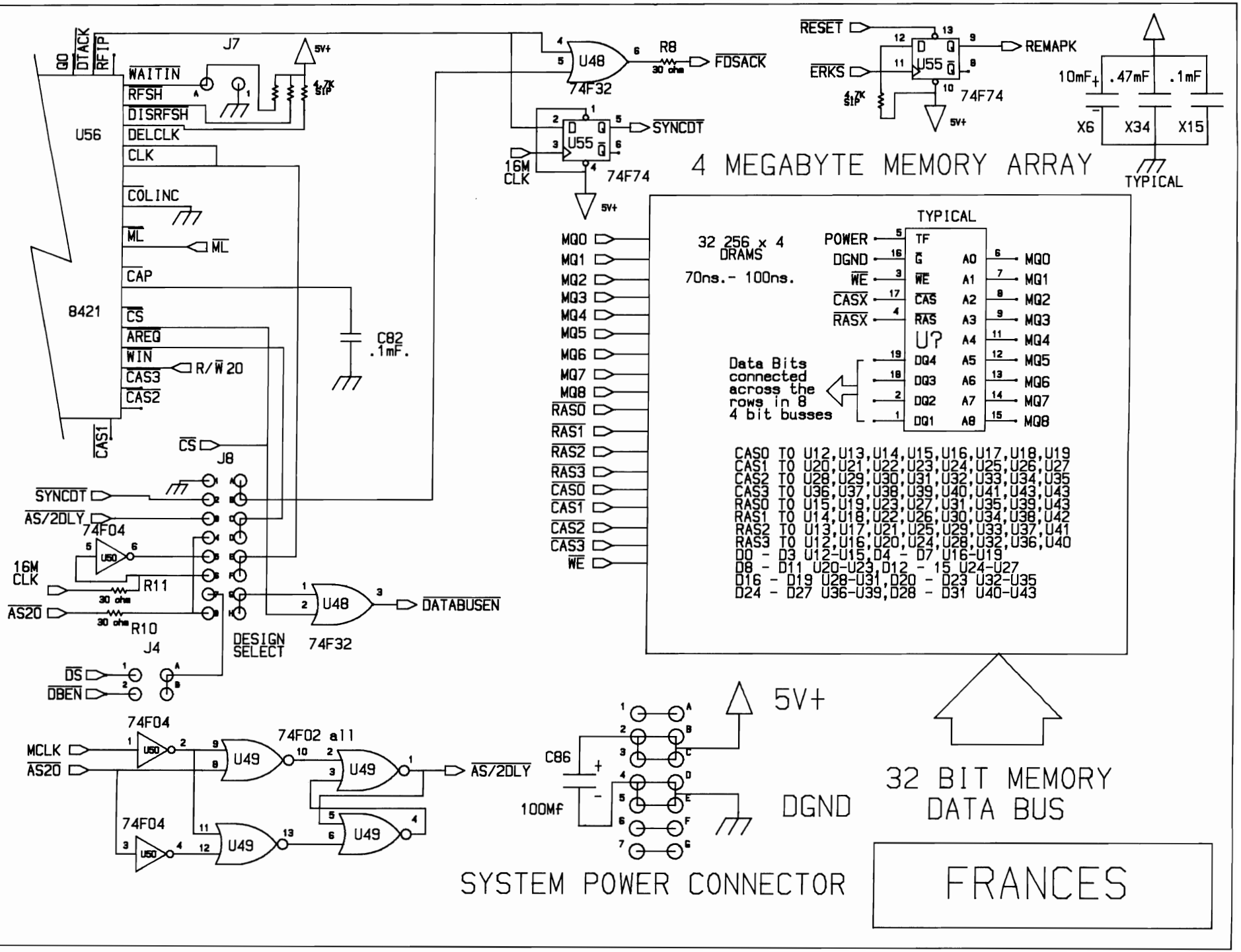
LUCAS, 4 pals, and documentation disk: $75.00 US
FRANCES, 2 pals, and documentation disk: $75.00 US

Don't forget to specify the type of 96-pin DIN connector you used.

PAL equations:

```
PARTNO      U54 ;
NAME        FRANCES3 ;
REV         03 ;
DATE        March 3rd 1989;
DESIGNER    Brad Fowles ;
COMPANY     Anakin  ;
ASSEMBLY    Frances ;
LOCATION    U54 ;

/* PAL1618B2 */
/* PAL DESIGN SPECIFICATION */
/* 68020-68881 /68000 AMIGA INTERFACE */

PIN 1 = DS ;
PIN 2 = A0 ;
PIN 3 = A1 ;
PIN 4 = SIZ0 ;
PIN 5 = SIZ1 ;
PIN 6 = MA23 ;
PIN 7 = MA22 ;
PIN 8 = A31 ;
PIN 9 = A30 ;
PIN 11 = AMY ;
PIN 12 = UUD ;
PIN 13 = UMD ;
PIN 14 = LMD ;
PIN 15 = LLD ;
PIN 16 = CS ;
PIN 17 = ERKS ;
PIN 18 = ML ;
PIN 19 = SPO1 ;

!UUD =   !A0   & !A1   & !DS;

!UMD =   !SIZ0 & !A1   & !DS
       # A0    & !A1   & !DS
       # SIZ1  & !A1   & !DS;

!LMD =   !A0   & A1    & !DS
       # !A1   & !SIZ0 & !SIZ1 & !DS
       # SIZ0  & SIZ1  & !A1   & !DS
       # !SIZ0 & !A1   & A0    & !DS;

!LLD =   A0    & SIZ0  & SIZ1  & !DS
       # !SIZ0 & !SIZ1 & !DS
       # A0    & A1    & !DS
       # A1    & SIZ1  & !DS;

!CS  =  !MA23 & MA22  & !A31  & !A30  & AMY
     #  !MA23 & MA22  & !A31  & A30   & !AMY;

!ML  =  A31  & !A30  & MA23 & !MA22 & !DS ;

!ERKS =   A31 & !A30 & !MA23 & MA22 ;

/*
DESCRIPTION: BYTE WRITE DECODE FOR *CAS GENERATION AND SYSTEM DECODE
*/


PARTNO      U7 ;
NAME        NEWU7 ;
REV         01 ;
DATE        MARCH 3RD, 1989 ;
DESIGNER    Brad Fowles ;
COMPANY     Anakin  ;
ASSEMBLY    Lucas & Frances ;
LOCATION    U7 ;

/* PAL1618B2 */
/* PAL DESIGN SPECIFICATION */
/* 68020-68881 /68000 AMIGA INTERFACE */

PIN 1 = HIGHZ ;
PIN 2 = DS20DLY ;
PIN 3 = A0 ;
PIN 4 = SIZ0 ;
PIN 5 = SIZ1 ;
PIN 6 = AS20DLY ;
PIN 7 = CPCS ;
PIN 8 = CPDSACK0 ;
PIN 9 = FDSACK ;
PIN 11 = SYSDSACK1 ;
PIN 12 = DSACK0  ;
PIN 13 = FRANCYC ;
PIN 14 = DSACK1 ;
PIN 15 = CPDSACK1 ;
PIN 16 = AS00BUF ;
PIN 17 = AS00 ;
PIN 18 = LDS ;
PIN 19 = UDS ;

AS00.OE = HIGHZ & FRANCYC ;
!AS00   = (CPCS )& (!AS20DLY );

AS00BUF.OE = HIGHZ &FRANCYC;
!AS00BUF   = (CPCS )& (!AS20DLY );

UDS.OE = HIGHZ & FRANCYC;
!UDS  = (!DS20DLY )& (!A0 ) & (CPCS) ;

LDS.OE = HIGHZ & FRANCYC ;
!LDS  = ( !DS20DLY ) & ( SIZ1 ) & (CPCS) #
        ( !DS20DLY ) & (!SIZ0 ) & (CPCS) #
        ( !DS20DLY ) & ( A0 ) & (CPCS) ;

!DSACK1 = (!FDSACK )& (!AS20DLY )#
          (!CPDSACK1 )& (!AS20DLY )#
          (!SYSDSACK1)& (!AS20DLY ) ;

!DSACK0 = (!FDSACK )& (!AS20DLY )#
          (!CPDSACK0 )& (!AS20DLY ) ;

/*
DESCRIPTION: ADDRESS STROBE, UPPER AND LOWER DATA STROBE AND FINAL DSACKX
             GENERATION
*/
```


VCC
A0,A1
A2 - A17
8421
DRAM
CONTROLLER
+
U56
U51
74F30
74F04
U53
74F244
U52
74F244
U54
FRANPAL
U44 D24-D31
U45 D16-D23
U46 D8-D15
U47 D0-D7
J3
KICKSTART
ADDRESS
SELECT
J5
RAS
SELECT
J6
16M
CLK
REMAPK
AS20
DS20
DBEN
R/W20
FRANCYC
FDSACK
RESET
ML
ERKS
CS
A18-A23
5V+
4.7K
SIP
30 ohm
ALL
MQ8
MQ7
MQ6
MQ5
MQ4
MQ3
MQ2
MQ1
MQ0
WAITIN
RFSH
DISRFSH
DELCLK
CLK
COLINC
ML
CAP
CS
AREQ
WIN
CAS3
CAS2
CAS1
CAS0
RAS3
RAS2
RAS1
RAS0
WE
DATABUSEN
FRANCES
32 BIT DATA BUS TO 68020
32 BIT DATA BUS TO MEMORY
DATA BUFFERS
ARE 74LS245
OR
74AS2645
(SEE DOCS)
DGND


4 MEGABYTE MEMORY ARRAY
32 256 x 4 DRAMS
70ns.- 100ns.
TYPICAL
Data Bits connected across the rows in 8 4 bit busses
CAS0 TO U12,U13,U14,U15,U16,U17,U18,U19
CAS1 TO U20,U21,U22,U23,U24,U25,U26,U27
CAS2 TO U28,U29,U30,U31,U32,U33,U34,U35
CAS3 TO U36,U37,U38,U39,U40,U41,U43,U43
RAS0 TO U15,U19,U23,U27,U31,U35,U39,U43
RAS1 TO U14,U18,U22,U26,U30,U34,U38,U42
RAS2 TO U13,U17,U21,U25,U29,U33,U37,U41
RAS3 TO U12,U16,U20,U24,U28,U32,U36,U40
D0 - D3 U12-U15,D4 - D7 U16-U19
D8 - D11 U20-U23,D12 - 15 U24-U27
D16 - D19 U28-U31,D20 - D23 U32-U35
D24 - D27 U36-U39,D28 - D31 U40-U43
32 BIT MEMORY DATA BUS
SYSTEM POWER CONNECTOR
FRANCES
U56
8421
J7
J8
J4
U48
U55
74F32
74F74
74F04
74F02 all
U49
U50
R8
R10
R11
C82 .1mF.
C86
100MF
5V+
DGND
REMAPK
RESET
ERKS
FDSACK
SYNCDT
DATABUSEN
DESIGN SELECT
16M CLK
10mF .47mF .1mF
X6 X34 X15
TYPICAL